

Volume 21 Number 1 March 1997

ISSN 0350-5596

Informatica

**An International Journal of Computing
and Informatics**

Informational Supervenience

Maruyama's Case Study

Integrated Software Life-Cycle Tool

Hierarchical Classification Browsing

Generating Modula-2 from Prolog

Framework for Object-Oriented Programs

Informatica 21 (1997) Number 1, pp. 1-156



The Slovene Society Informatika, Ljubljana, Slovenia

Informatica

An International Journal of Computing and Informatics

Basic info about Informatica and back issues may be FTP'ed from `ftp.arnes.si` in `magazines/informatica` ID: anonymous PASSWORD: `<your mail address>`
FTP archive may be also accessed with WWW (worldwide web) clients with
URL: `http://www2.ijs.si/~mezi/informatica.html`

Subscription Information Informatica (ISSN 0350-5596) is published four times a year in Spring, Summer, Autumn, and Winter (4 issues per year) by the Slovene Society Informatika, Vožarski pot 12, 1000 Ljubljana, Slovenia.

The subscription rate for 1997 (Volume 21) is

- DEM 50 (US\$ 35) for institutions,
- DEM 25 (US\$ 17) for individuals, and
- DEM 10 (US\$ 7) for students

plus the mail charge DEM 10 (US\$ 7).

Claims for missing issues will be honored free of charge within six months after the publication date of the issue.

LaTeX Tech. Support: Borut Žnidar, DALCOM d.o.o., Stegne 27, 1000 Ljubljana, Slovenia.

Lectorship: Fergus F. Smith, AMIDAS d.o.o., Cankarjevo nabrežje 11, Ljubljana, Slovenia.

Printed by Biro M, d.o.o., Žibertova 1, 1000 Ljubljana, Slovenia.

Orders for subscription may be placed by telephone or fax using any major credit card. Please call Mr. R. Murn, Jožef Stefan Institute: Tel (+386) 61 1773 900, Fax (+386) 61 219 385, or use the bank account number 900-27620-5159/4 Ljubljanska banka d.d. Slovenia (LB 50101-678-51841 for domestic subscribers only).

According to the opinion of the Ministry for Informing (number 23/216-92 of March 27, 1992), the scientific journal Informatica is a product of informative matter (point 13 of the tariff number 3), for which the tax of traffic amounts to 5%.

Informatica is published in cooperation with the following societies (and contact persons):

Robotics Society of Slovenia (Jadran Lenarčič)

Slovene Society for Pattern Recognition (Franjo Pernuš)

Slovenian Artificial Intelligence Society (Matjaž Gams)

Slovenian Society of Mathematicians, Physicists and Astronomers (Bojan Mohar)

Automatic Control Society of Slovenia (Borut Župančič)

Slovenian Association of Technical and Natural Sciences (Janez Peklenik)

Informatica is surveyed by: AI and Robotic Abstracts, AI References, ACM Computing Surveys, Applied Science & Techn. Index, COMPENDEX*PLUS, Computer ASAP, Cur. Cont. & Comp. & Math. Sear., Engineering Index, INSPEC, Mathematical Reviews, Sociological Abstracts, Uncover, Zentralblatt für Mathematik, Linguistics and Language Behaviour Abstracts, Cybernetica Newsletter
--

The issuing of the Informatica journal is financially supported by the Ministry for Science and Technology, Slovenska 50, 1000 Ljubljana, Slovenia.

Post tax payed at post 1102. Slovenia tax Percue.

Informational Supervenience

By Anton P. Železnikar, the Editor

Today, some professional and scientific orientations have to be essentially reconsidered in an interdisciplinary sense. For instance, the Chalmers book¹ calls both the field of computer science and especially the field of artificial intelligence to look into their own conceptual and technological fundamentals and, simultaneously, to the newly emerging discipline of consciousness studies. These traditional disciplines (currently still mainstream doctrines) would intend to increase the performance and sophistication of the future computing machines, operating systems and application programs. But all this might require fundamentally new and innovative methodologies, and an interdisciplinary understanding and communication between researchers and engineers. The nowadays mainstream AI is already becoming a traditional and firm routine, so to say, a technical (engineering) field. On the other side, the study of the conscious realm, as a long-term project, might enter into the mainstream phase where the new interdisciplinary research and sciences become not only necessary but crucially mutually dependent and impacted in a fundamentally interdisciplinary way.

The epistemology of traditional sciences² might change in the direction of higher complexity, different understanding of causalism (nonreciprocal, morphogenetic, metaphysicalistic), less reductionism (artificial simplification), and a clear distinction of scientific supervenience, how it is funded on both a materialistic (physicalistic) and a phenomenal (consciousness-like, experiential) ground. The supervenience of sciences has to be additionally explained to the readers of *Informatica*. What could the informational supervenience mean at all and, how does it meet the phenomenologies (phenomenalisms) of sciences in particular?

Informatics as a wide-ranging field of research and scientific background (underground) has unconsciously and unexpectedly left the interdisciplinary horizons comprehensively open when the term was initially coined by the French *informatique* and the German *Informatik*. In Germany, it was definitely only an academic replacement for the Anglo-Saxon

Computer Science. When in 1977, *Informatica* appeared as a new professional journal in former Yugoslavia (and among the identically named professional journals as incontestably the first one), the difference between computer science and informatics was clearly distinguished quite in the beginning, by the subtitle of *Informatica*: a journal for computing technology and problems of informatics. Later, by its international issue, it became a journal for computing and informatics³. At that time, intuitively, the informational supervenience was pre-understood in the spirit, how the informational could supervene not only on the natural, the physical, the methodological (technological, abstract, mathematical), the psychological, the sociological, the managerial, the phenomenal, etc., but also on the informational itself. Recursiveness in computer programming was presumably the way to the concept of informational circularity in a spontaneous way (informational causality), as one of the most basic informational principles in general.

On this way of development, informatics as an autonomous research domain of disciplines slightly became also, so to say, a phenomenon of the informational second-order cybernetics⁴. The concepts of information as a mere flow of information particles (bits, semantic codes) (e.g., Shannon⁵, Dretske⁶) became essentially narrowed and reductionistic (simplified) disciplines, philosophically, and traditionally bounded forms of research and scientific *normalism* (Maruyama⁷), leaving the abyss of the problems of postmodernistic sciences open in the sense of their informational amalgamation.

I promised my explanation of the concept of supervenience (together with its grammatical deriva-

³Today, informatics unites the information theory (communication, librarianship), information technology (methodology) and informational phenomenality (in society, humanities, management, biology, mind processes, consciousness, and the like).

⁴BRIER, S. 1995. Cyber-semiotics: On autopoiesis, code-duality and sign games in bio-semiotics. *Cybernetics & Human Knowing* 3: 1: 3-14.

⁵SHANNON, C. 1948. The mathematical theory of communication. *Bell Systems Technical Journal* 27: 379-423, 623-656.

⁶DRETSKE, F.I. 1981. *Knowledge and the Flow of Information*. MIT Press. Cambridge, MA.

⁷MARUYAMA, M. 1980. Normalism and its costs. *Futurics* 4: 1: 83-84.

¹CHALMERS, J.W. 1996. *The Conscious Mind*. Oxford University Press. New York.

²MARUYAMA, M. 1980. Mindscapes and science theories. *Current Anthropology* 21: 589-600.

tives, indeed) since it brings to the surface the concept, which I call the *informational*⁸.

In the framework of the materialistic (physicalistic, traditional) view, the physical facts should be the most fundamental facts about the universe. The question is on which facts does the physical supervene on⁹?

Chalmers¹ (p. 33) gives a loose (unidirectional) definition of supervenience:

B-properties *supervene* on *A*-properties if no two possible situations are identical with respect to their *A*-properties while differing in their *B*-properties.

What does Chalmers mean by property, another loosely assigned possibility along with supervenience? Properties could represent relations within a class of objects but, more generally, they could be grasped also as relational, existing, or operational entities, impacting the domain of property classes, that is, being circularly closed to a certain property domain.

If the informational supervenes, for instance, on the physical, the managerial, the psychological, etc., then for each different situation (and attitude) in the informational there must exist a different situation (and attitude) in the physical, the managerial, the psychological, etc., respectively. However, for the informational supervenience one may request simultaneously the reverse definition, which protects the informational to become (scientifically, methodologically, traditionally, etc.) *reductionistic* (simplified, oversimplified) when being supervenient on something. That is:

B-properties *supervene* on *A*-properties if no two possible situations are identical with respect to their *B*-properties while differing in their *A*-properties.

The example, given by Chalmers¹ (p. 35), concerns the *logical* supervenience in the following form:

⁸A typical and solely verbal presentation of the concept was given in the author's paper *Principles of information, Cybernetica* : 99–122 (1988) or, more exhaustively, in the book *On the Way of Information*, The Slovene Society Informatika (1990) ISBN 86–901125–1–0.

⁹One of the answers to this question could be that the physical supervenes on the natural (a kind of nomic, empirical, or even experiential facts, that is, on physicists' consciousness). However, the reader already begins to feel that the discussion aims at a sort of the source common to the all. Chalmers says it could be a superbeing (Laplace demon) or God, or simply to say "It is logically so", etc.

B-properties supervene on *A*-properties if no two *logically possible* situations are identical with respect to their *A*-properties but distinct to their *B*-properties.

We see how this definition fits the basic supervenience template when the *possible* is replaced by the *logically possible*, and the word *differing* is replaced by the word *distinct*. The reader can observe that it is not anywhere said to which realms the *A*-properties and the *B*-properties might belong. In this respect, the logical supervenience may (probably not only in a unidirectional way) connect different fields of investigation, different sciences and, at last but not least, substantially different systems—different logical systems too. Within this view, supervenience functions as a sort of intersystemic *reversal implication*¹⁰ (in respect to the reading convention, *B*-properties supervene on *A*-properties), in one or another way. Thus, in metamathematics¹¹, it actually happens that the basic axioms are circularly structured¹² in the informational sense¹³.

The next most significant term concerning (the informational) supervenience is the phenomenal. According to Chalmers¹, the informational should supervene on the physical, and the phenomenal on the informational. The informational could be recognized as a necessary condition (e.g., scientifically senseful fact) existing between the physical and the phenomenal. But what to say, if the informational appears simultaneously not only as the phenomenal, but to some extent also as the physical?¹⁴ What does the phenomenal mean at all?

The phenomenal can be explained in many different ways, not only in the domain of the philo-

¹⁰I intentionally use the word *implication* to bring near the concept of supervenience as an intersystemic inferential (concluding) procedure on one side, and the mathematically (metamathematically) conceptualized logical inferential procedure (in fact, operation) within the unique system of logic.

¹¹HILBERT, D. UND P. BERNAYS. 1934. *Grundlagen der Mathematik*. Erster Band. *Grundlagen der mathematischen Wissenschaften in Einzeldarstellungen*, Band XL. Verlag von Julius Springer. Berlin.

¹²The critical reader could easily say, that these axioms are tautological in the best sense of the known traditional linguistic meaning.

¹³ŽELEZNIKAR, A.P. 1995. Elements of metamathematical and informational calculus. *Informatica* 19: 345–370.

¹⁴Informational processes, for instance, procedures of learning, can directly influence the formation of neurons and their circular synaptic interconnections. A bird-song is learned after the adequate neuronal circuit is being physically formed in the brain.

logical (the philosophical) but also in the domain of the physical (the scientific). One of the philosophical outcomes is *phenomenology* as conceptualized by Brentano, Husserl¹⁵, and Heidegger¹⁶. Phenomenology begins from a scrupulous inspection of one's own consciousness, about the external causes and consequences from which the internal causes and consequences have to be excluded. Later, the internal causes and consequences become equally important, for instance, concerning the Being of understanding and interpretation. Besides other views, the Chalmers's project¹ roots firmly in phenomenology as developed up to this time¹⁷.

On the other side, the informational can be understood as the phenomenal of information in the widest possible way. Through this view and comprehension, the informational comes close to the study of the physical and the phenomenal of consciousness, enabling a new philosophical and scientific discourse, and a new formalistic (mathematics-like) informational approach (phenomenalistic logical formalism, where informational formulas by themselves behave as phenomena and not as firm statements, equations, or formulas in mathematics).

Let us discuss in short the meaning and the use of the noun *supervenience*, the adjective *supervenient*, and the verb *supervene* [on].

The word 'supervenience' (an intimidating name for a philosophical concept) is rarely explained or listed in dictionaries. The meaningly closest noun is *supervention*, derived from the verb 'supervene'. Supervenience is favored in respect to the philosophical notion of identity or the mathematical notion of (reversal) implication ($B \leftarrow A$), where one set of facts determines (better, explains, identifies, implies, entails) another set of facts. The idea of supervenience was introduced by Moore¹⁸ and by Hare¹⁹. The first who applied the notion of super-

venience to the mind-body problem was Davidson²⁰. Later, a theory of supervenience was developed, for instance, by Kim²¹, Horgan²², Hellman & Thompson²³, and others.

The verb 'supervene' (super = above, beyond, in addition + come) is used consequently¹ with the preposition 'on', so the meaning of 'it supervenes on' for a newcomer would sound literally, for example, as *it comes beyond on* (in the form, something comes beyond on something). In a similar context the adjective 'supervenient' is used. This short explanation should relieve the understanding of the new terms to the nonnative English speaker.

More precisely, for a better understanding, the following correspondence schemata (\rightarrow) can be introduced:

supervenes on	\rightarrow is implied by
	\rightarrow is entailed by;
is supervened on by	\rightarrow implies;
is supervenient on	\rightarrow supervenes on

The kinds of supervenience in Chalmers¹ are manifold: global, local, logical, natural, etc. The reader can develop its own notions of possible types of supervenience, concerning any scientific and imaginable field: the physical, the phenomenal and, certainly, the informational. Informational supervenience could be an essential step in understanding of the realm of information dynamics.

Informatica, as a journal of informatics—and informatics, as an interdisciplinary field—is a place where different mindscape types (Maruyama)²⁴

²⁰DAVIDSON, D. 1970. Mental events. In L. Foster & J. Swanson, Eds. Experience and Theory. Duckwords. London.

²¹KIM, J. 1978. Supervenience and nomological incommensurables. American Philosophical Quarterly 15: 149–156.

KIM, J. 1984. Concepts of supervenience. Philosophy and Phenomenological Research 45: 153–176.

KIM, J. 1993. Supervenience and Mind. Cambridge University Press. Cambridge.

²²HORGAN, J. 1982. Supervenience and microphysics. Pacific Philosophical Quarterly 63: 29–43.

HORGAN, J. 1984. Supervenience and cosmic hermeneutics. Southern Journal of Philosophy, suppl. 22: 19–38.

HORGAN, J. 1993. From supervenience to superdupervenience: Meeting the demands of a material world. Mind 102: 555–586.

²³HELLMAN, G. & F. THOMPSON. 1975. Physicalism: Ontology, determination and reduction. Journal of Philosophy 72: 551–564.

²⁴MARUYAMA, M. 1993. Mindscapes, individuals, and cultures in management. Journal of Management Inquiry

¹⁵HUSSERL, E. 1900. Logische Untersuchungen. Erster Band. Prolegomena zur reinen Logik. 1901. Zweiter Band. Untersuchungen zur Phänomenologie und Theorie der Erkenntnis. I. Teil. 1901. Zweiter Band. Elemente einer phänomenologischen Aufklärung der Erkenntnis. II Teil. Max Niemeyer Verlag, Tübingen.

¹⁶HEIDEGGER, M. 1927. Sein und Zeit. 1986. Sechzehnte Auflage. Max Niemeyer Verlag, Tübingen.

¹⁷DREYFUS, H.L. 1991. Being-in-the-World. The MIT Press. Cambridge, MA.

¹⁸MOORE, G.E. 1922. Philosophical Studies. Routledge & Kegan Paul. London.

¹⁹HARE, R.M. 1952. The Language of Morals. Clarendon Press. Oxford.

meet and separate, simultaneously. Roughly, the technological (physical) and the phenomenal (informational) build up and reinstate the bridge upon the conceptual abyss between the physically possible and the phenomenally, that is, informationally possible. Thus, a new implementation of methodologies and organization of machines and conceptualism is on the way.

Another problem of *Informatica* which should be faced up to a broader understanding of its readers, authors, reviewers, and editors is the multidisciplinary of informatics. For instance, there exists the so-called Gesellschaft syndrome in the fields of sciences (Maruyama)²⁵. Interdisciplinarity is a contextual orientation of mind²⁶. Genuine interdisciplinarians ... became unable to function because of the resistance inside their own departments and in the mainstream journals. In spite of all budget cuts, there were specialists in each department as well as interdisciplinarians who could have been innovative, but they were under subsedure²⁷, i.e. they are superseded by the dominant group. ... The 21st century will be an

age of *interwoven* and *interactive* heterogeneity²⁸ in contrast to the past age of localized heterogeneity.

Informatica, as an interdisciplinary journal, is supervenient on various scientific disciplines and research experiences. A long time ago, after the acceptance by Professor Magoroh Maruyama to become an editor of *Informatica*, I asked him to deliver the data for his profile. I was informed from several sources about his international and publicistic activity (interdisciplinary reputation), as an educator and a researcher in the sense of the nontraditional (he would probably like to say, an outside-of-the-mainstreams) research and investigation. At that time he responded that instead of a profile he would like to submit something researching his own experience in his interdisciplinary academic and research work in different academic and social environments (Japan, USA, Alaska, Europe, etc.) I agreed. But until, a short time ago, I learned that he was enormously burdened by lecturing seven curses at his university in Tokyo and that he could not find time even for answering the letters.

Computing and informatics unite various scientific, research, methodology, and application mainstreams (traditional and firm fields of activity) being variously informationally structured and organized disciplines. Besides evident mainstreams of computer science, artificial intelligence, cognitive science, etc. some new nontraditional approaches come into the foreground. The most outstanding fields of research open in the domain of consciousness together with informational phenomenalism. The common characteristic of these offshoots is interdisciplinarity, nontraditional ways of scientific interpretation, in short, the evident phenomena of informational supervenience. And on this route, the *mainstreamers* have to loosen their disciplinary boundaries to enable the possible development of their own mainstream activity.

Staying within this context, I believe that readers of *Informatica* will enjoy in reading Maruyama's autobiographical case study entitled "A Situational Informatical Dynamics: The Case of Situation-contextual and Time-contextual Non-additive Influences" recognizing his enormously reach interdisciplinary research experience within various academic and social environments. ■

2: 2: 140-155. — Mindscape types are originally categorized epistemological types emerged in different national, race, tribe, managerial, academic, scientific, political, social, cultural environments, etc. Roughly, by the Editor's interpretation, the four epistemological types of Maruyama, could be signified also as (1) rationalistic (universalistic, hierarchical, classifying, Newtonian, Cartesian, Kantian, reductionistic, sequential, one-truth) types; (2) intuitivistic (individualistic, isolationistic, randomizing, artistic, nominalistic, haphazard, subjective) types; (3) cybernetic (mutualistic, interactive, contextual, circular, stabilizing, simultaneous, polyobjective, Husserlian, computeristic) types; and (4) second-order-cybernetic (informational, evolutionary, generative, spontaneous, parallelistic, polysubjective, Heideggerian) types. In the future, in the era of the globalization in an informational way (a global information society), it would be perhaps possible to introduce the so-called (5) informational epistemological (phenomenalistic, serial-parallelistic, parallel-serialistic, complexly causal-circularly organized, intrinsic, metaphysicalistic, possible worlds view, informational philosophy and the like) types. The last could reasonably unite (absorb) the experience of the previous four types, that is, bringing up the highly integrated human consciousness.

²⁵MARUYAMA, M. 1992. Anti-monopoly law to prevent dominance by one theory in academic departments. *Human Systems Management* 11: 219-220.

²⁶MARUYAMA, M. 1992. *Context and Complexity*. Springer-Verlag. New York.

²⁷MARUYAMA, M. 1991. Epistemological heterogeneity and subsedure: Individual and social processes. *Communication and Cognition* 24: 255-272.

²⁸MARUYAMA, M. 1973. Human futuristics and urban planning. 1973. *Journal of American Institute of Planners* 39: 246-357.

A Situational Informatical Dynamics: The Case of Situation-contextual and Time-contextual Non-additive Influences

Magoroh Maruyama

Aomori Koritsu Daigaku, Goushizawa Aza Yamazaki 153-4

Aomori City, 030-01 Japan

Fax: Japan+17764-1544

Keywords: causal loops, conjunction, disjunction, individual case study, informational dynamics, interlocking, motivation, non-additive concepts, parajunction, stress

Edited by: Anton P. Železnikar

Received: January 20, 1997

Revised: February 9, 1997

Accepted: March 14, 1997

Individual decision making and organizational processes do not follow rules or patterns which are a priori predictable or situation-independently and time-independently programmable. We must look into situation-contextual and time-contextual texture instead of theoretical skeletonization and formula-fitting. There are many types of situational informatical dynamics. In this article the concept of non-additive influences is used as an example. A case is analyzed in detail.

1 Introduction

When several types of job-related stresses occur simultaneously or sequentially, it is usually assumed that their effects are quantitatively additive in such a way that when their coefficient-weighted "sum" exceeds a certain "threshold", the individual takes an action such as quitting the job. However, some situational or temporal combinations may produce less-than-additive effects while other combinations may yield more-than-additive results, especially when there are causal loops. Moreover, non-loop non-additive conditions such as conjunction, disjunction and interlocking may also exist. Sometimes the process is more like chemistry than physics, i.e. the combined effect is dissimilar to any of the components. Consequently, in order to rate job satisfaction or dissatisfaction as a predictor of future performance or a retrospective explanation of past behavior, the additive concept may be highly misleading. Theoretically, this article presents a use of the concept of situation-contextual and time-contextual non-additive influences. Empirically, i.e. from the point of view of data, this article uses perceptual phenomenology instead of a prestructured data collection, because the individual's decision was based on his perception of

the situations.

2 Non-additive Relations

There are many non-additive relations. It is impossible to list them all not only because they are numerous but because each case requires a new set of relations. In this article, the following relations are used:

- Causal loops: There are basically two types with several subtypes. The first is change-counteracting. The second is change-amplifying. The first has two main subtypes: asymptotic stabilization and constant oscillation. But if there are time delays in the first type, asymptotic stabilization may change to an oscillation, and a constant oscillation may change to amplification of oscillation. In the second type, there are also several subtypes: steady increase; amplification of oscillation; breakdown due to excessive increase or excessive amplification. But if there are time delays in the second type, change-amplifying may become change-counteracting.
- Conjunction: Simultaneous presence of *A* and *B* is a necessary condition to produce *C*, which cannot be caused by *A* or *B* alone.

- Disjunction: *A* alone or *B* alone is a sufficient condition to produce *C*. Their joint presence does not increase the effect.
- Parajunction: Neither a sufficient nor a necessary condition, but an incentive or a disincentive which may be in a causal loop with another condition.
- Interlocking: Like a jigsaw puzzle, some pieces fit together while others do not. Usually a result of situational or temporal coincidence.

3 The Case of *P*

P is a professor who left a university and went to another. On the surface, his decision may be attributed to an accumulation of several types of stresses. However, the combination of these stresses was not additive. It is important to look into its non-additive nature. Let us examine what happened step by step.

3.1 Events preceding the decision

At the beginning of 1995, *P* was under a heavy psychological pressure primarily due to extra work needed to avoid anticipated attacks from other components against his unit at his university. He was in a position to be scapegoated or symbolically pinpointed in the attack because of his special ways of teaching. In the 120-year history of the university, his unit was only 15 years old. Other components practiced mass-production in teaching typical in Japan. Some classes had as many as 700 students. There were no teaching assistants. The grades were mainly based on one or two examinations. Most students did not show up until a few days before the examinations. The teacher hinted at questions likely to be asked in the exams. Very often the students did not have to know much to pass the exams. For example, a typical question would be: “Choose one from the following three topics and discuss it.” The three topics had already hinted at by the teacher a few days before the exam. It sufficed to study only one topic, not three topics. But *P*’s unit, the newest at the university, practiced teaching methods closer to those in the U.S.A. The faculty/student ratio in his unit was therefore much

higher than in other components. Consequently from its inception, his unit was a target of political attacks from other components which did not understand the fact that the faculty members of his unit worked several times harder than theirs. *P* joined the faculty in 1987 after having taught for three decades in the U.S.A., Sweden, France and Singapore. *P* introduced new teaching methods which he had developed in these countries. His courses were on international business. But he combined teaching and learning methods from anthropology, sociology and psychology, such as field work observation, analysis of nonverbal behavior, participant observation, interactive methods, intervention methods, and in-depth psychological analysis. These methods were neither understood nor appreciated by other components of the university.

One of his courses was a project-based course which required very intensive work on the part of both the students and the teacher. There were usually about 150 students in the class. They were divided into about 20 groups. Each group had to choose a business category, and compare 5 or 6 firms from various countries within that business category in terms of their strategy and management practices. Tokyo, where the university was located, was one of the few places in the world, along with Singapore, Hong Kong and London, where firms from many countries were present, and students could learn by directly talking with their managers, employees clients and competitors, psychologically analyzing their verbal and nonverbal reactions during the conversation, and anthropologically and sociologically observing their behavior and actual performance. He devised new project methods which he had begun while teaching in Singapore in 1983 and refined in Tokyo since 1987 [2, 3].

During the first semester, each group of students studied their chosen firms from the managers’ point of view. During the second semester, the students had to study the same firms from the points of view of their clients, employees and competitors, had to make on-site observations of the firms’ behavior and action, generate several future scenarios to determine the loser, devise strategy changes for the loser, and go back to the manager of the loser firm to discuss the proposed strategy. In interviews and other interactions with firms,

clients and competitors, the students had to analyze nonverbal cues and hidden levels of intention and meaning, not just verbal statements. *P* made no introduction of students to firms. The students had to initiate contacts, and learn how they were rejected or accepted, and how the managers evaded, deflected, circumvented or camouflaged some topics, or ignoring a question by pretending not to understand or notice it, or by purposely misunderstanding it. The students had to analyze nonverbal cues and hidden levels of intention.

When interviewing the employees, *P* advised the students to avoid going through the management. Some groups invented methods of their own. For example, in order to talk with employees, a subgroup used a walkie-talkie to describe employees coming out from the firm building, while another subgroup waited at the railway station to intercept and trail the described persons, follow them into the train, casually stand next to them in the crowded train, and begin unsuspectable conversations such as "I moved to this area and am looking for a job. How would you compare the working conditions in various firms?" Sometimes the students posed as customers or as someone who simply happened to be there. For example, a group which compared diaper firms borrowed a baby, went to a park where the firm employees took a break, tried to change the diaper with feigned clumsiness to see whether the employees would help.

Such ways of teaching requires a great deal of extra work for both students and teachers. The course in international business, which *P* turned into a project course, was a required course. At the beginning of the course the students complained for two reasons:

- (1) They had never had a project-based course and they did not see why they could not just sit in the library and study;
- (2) The project was too time-consuming and difficult.

But as their projects got under way, they became absorbed in their work, often sacrificing the homework of other courses. For this, *P* was resented by other teachers. At the same time, among the students he became known as a teacher who teaches in a very difficult way, and other courses of his which were not required

courses drew only few students who were ambitious enough to try something in which they could use their inventiveness. In a nationwide survey conducted by the prestigious Toyo Keizai Shimpo Sha in which 31,410 students in and graduates from various universities in Japan responded, *P* was one of the very few who received a special mention and recommendation by the publisher for his innovative ways of teaching [7].

Then in 1994 due to a reorganization, there were no longer required courses. The number of students in *P*'s project course fell, though the few who came were excellent. But the political attacks from other components of the university were based on the number of students regardless of their quality. In 1995 *P* would become a primary target of attacks as a prototype of a bad guy. The academic year in Japan begins in April. Therefore *P* devised campaigns during January and February to attract students, because March is a vacation time when no students would be around. The campus newspaper did not want to print articles on courses. Therefore *P* tried outside newspapers and weekly magazines which had nationwide circulation. He had tried them before and knew that they had much effect on students in the entire country including those at his university.

But two big incidents detracted the daily newspapers and weekly magazines: Kobe Earthquake in January, and Oum Sect terrorism in February. The earthquake was the worst in the Japanese history. Elevated highways collapsed. Debris of buildings blocked the streets and rescue vehicles were unable to enter the area. Victims were buried under collapsed buildings for weeks, and the count of the discovered dead persons was updated daily. The newspapers were filled with daily changing statistics of the destruction. Moreover, various measures to set up temporary housing, food supply and employment for survivors who lost not only their houses but also their firms were ineffective, and this problem filled the newspapers for several months. The number of deaths eventually totaled to 6310, but the most recent amendment dated December 10, 1996 (almost two years after the earthquake) added 51 more, making the total of 6361.

The Oum Sect incident was equally unprecedented. On a fine winter day, poison gas was

spread at several places in the subway system. Seven persons died and other 2475 persons became sick and were hospitalized. At the beginning, it was impossible to find who did it. Gradually the investigators linked the incident with several mysterious unsolved cases of murders and kidnappings which had occurred during a few preceding years. What gradually emerged was a shock to the world. It was not a political terrorism, but a large-scale organized mass murder by a religious sect which had secret factories to manufacture poisons, guns and other weapons. Even the American FBI was shocked. The discovery of the hidden factories, the methods of the manufacturing, the indoctrination of the sect members, and the cruel methods of their murders of not only outsiders but also disobedient insiders were continually reported in the newspapers and magazines as investigations proceeded.

Kobe Earthquake and Oum Sect Incident kept reporters and editors of newspapers and magazines busy more than full-time. In spite of *P*'s innumerable telephone calls, he got only one newspaper *Asahi Evening News* to print an article on his project course, which was to appear on April 24 [1], too late to have any appreciable effect of the course enrollment. At the beginning of April, *P* made another desperate attempt: to attract foreign students to his seminars. It cost him many days of legwork. Expecting that some of his courses might not attract a sufficient number of students and therefore might get canceled, he announced a total of seven courses including the project course, seminars and a few other courses. It turned out that his campaigns were too successful. All seven courses were filled. He could not cancel any of them.

Note that this teaching load was not imposed from above, but resulted from overcautious preventive measures which were too successful. It was due to overprecaution. But what were the causes which necessitated the overprecaution? The "triggering" cause was that there were no longer required courses due to restructuring. But behind the overprecaution were:

- (1) attacks from other components;
- (2) *P*'s special methods of teaching.

(1) or (2) alone was not a sufficient cause. Therefore these two were *conjunctive causes*. Further-

more, the overprecaution was prompted by the unavailability of media which were busy with two unprecedented incidents of Kobe Earthquake and Oum Sect terrorism. Either one of these two incidents was sufficient to block the media. Therefore these two incidents were *disjunctive causes* to the unavailability of media which was a *conjunctive cause* for the overprecaution.

The "effect" of the combination of these conjunctive and disjunctive causes was teaching overload. But this teaching overload was not a sufficient cause of *P*'s moving to another university. It took him other conjunctive causes to make his decision, as will be seen. Therefore the teaching overload was a *conjunctive cause*.

Then in April, another incident struck. Almost a year earlier, *P* had been invited to a conference in Cleveland, to take place at the beginning of May 1995. Knowing that it would coincide with the Golden Week in Japan—a week-long holiday when airplane reservations to foreign countries were almost impossible to obtain—, *P* made his reservations six months in advance at a substantial discount price. But the clerk at the make-shift conference office, who was unfamiliar with foreign travel, canceled *P*'s reservation believing that she could simplify the matter by asking a travel agency in her city to remake reservations of all conference participants. Of course she could not get a new reservation at that late date, and she could not reinstate my reservation which she had canceled. I had to make many phone calls and fax calls to the U.S.A., and many trips to airline offices and travel agencies in Tokyo. Because of the time zone difference, phone calls from Japan had to be made at night, and because of the crowded commuting trains, *P* had to take 5:00 am trains to airline offices and travel agencies, all these while teaching seven courses. He was exhausted. Finally the conference organizer ended up having to pay the first class tickets for *P*. But *P*'s psychological stress and fatigue were beyond tolerance.

Teaching seven courses left him no time to even read letters. It caused many inconveniences to his correspondents. One colleague who wanted to co-author a manuscript with him had to give it up. Several other colleagues' research was delayed because of lack of response or information from him.

Meanwhile, an unprecedented opportunity arose which he had to miss: the value of the U.S.A. Dollar went down by more than 25% due to a sudden dollar flowback from Mexico. At the end of 1994, the value of one USA Dollar had been more than 105 Japanese Yen. By April it went down to ¥80. *P* had saved more than ¥3,000,000 for his daughter's study in the U.S.A. At the rate of ¥105, it would be US\$28,571. At the rate of ¥80, it would be US\$37,500. The difference was US\$8,929, almost nine thousand U.S.A. dollars. *P* had no time to go to a bank. He had a strong urge to do so. But he kept saying to himself: "Money is not a priority. Other things have higher priorities. Even if I sink, I should not go to a bank now." The low value of the USA dollar seemed to last, and he got accustomed to thinking that it would last. Moreover, he saw a prediction by a British economist that the USA dollar would go down more toward the end of 1995. He usually did not trust economists' extrapolative predictions, but in the spring of 1995 it was like a consolation, a straw for a drowning man.

The Japanese spring semester lasted until July. In July he was still struggling with the huge backlog of his work. He was scheduled to leave on August 9 for a lecture tour in South America. Yet on the first days of August he could have gone to a bank. Instead, he shopped around for shoes as his shoes were getting worn out. During the first days of August the value of US\$ still stayed in the ¥80s. He hoped that the exchange rate would go down further as the British economist said.

When he returned from South America in September, the exchange rate was around ¥105. He hoped that it would go down again. He waited and waited. But the US\$ stayed strong. He gradually began to realize his mistake, first imperceptibly, and increasingly seriously.

3.2 Phases of Psychological States Which Followed His Realization of the Mistake

From the material point of view, the mistake was a missed opportunity, not a loss. However, from the psychological point of view, it had profound consequences which contributed to *P*'s decision to leave the university. But here again its effect was non-additive. It became entangled in causal loops as well as in interlocking. Therefore it is

interesting to examine subsequent developments step by step.

3.2.1 Phase I: shock and disbelief

It took several days for him to become aware of his mistake. When he returned from South America on September 9, he was expecting the exchange rate to have stayed in the ¥80s or possibly to have gone down to the ¥70s. His first reaction was disbelief. For six months, from February to August, the exchange rate stayed in the ¥80s. It could not change to ¥105 in one month. He hoped that it was just a bad dream, not real. He thought that the low rate would come back and should come back. In retrospect, his reaction was similar to what happened in his mind when John F. Kennedy was assassinated: he thought that Kennedy was not really killed and would come back. Psychologically, this is a reaction common to many people who suddenly lost someone close. When he gradually realized that the low exchange rate was irrevocably gone, he was shocked by his stupidity of having lost the golden opportunity. It was his "stupidity" that bothered him rather than the monetary loss. He almost came to doubt his intelligence. The "trauma" was not material: it was loss of confidence in himself.

3.2.2 Phase II: resignation

He realized that the boat has been missed, and the boat would not come back. His state of mind, when he searched in his memory, was similar to the time when he lost a beautiful girl friend when he was a student. She appeared unexpectedly and stayed within his reach for quite a while. In fact, a mutual friend between him and her told him that she would like to get to know him. But he was rather busy with his research too long, and one day she was gone, never to come back, even though he expected and hoped her return. Her memory haunted him for a long time. Was she real? Was she an apparition or a phantom? Was she just a dream? The exchange rate of ¥80 sure was a very beautiful and sweet girl who was gone forever. He became resentful of the work overload and resulting stress-causing situations which prevented him from going to a bank. The more he had an urge to go to a bank, the more he resisted the "temptation". There was a change-amplifying

causal loop between the urge and the resistance to the urge, and the tension between the two also amplified the psychological stress. Now his resentment toward the work overload was stronger than before. This strengthened resentment was in a way an attempt to counteract his loss of self-confidence: He was stupid but he was not really stupid. It was his fault but it was not really his fault, etc.

3.2.3 Phase III: puzzle about penny-saving mentality

Since *P* moved to Japan in 1987, his daily routine included going to grocery stores around their closing time when fresh goods got discounted down 30% or even 50%. Often he hanged around in stores for 20 or 30 minutes to wait until the prices came down: If he went too early, the prices were still high. If he went too late, other people had already bought what he wanted. People waited until shop attendants came out to put discount stickers on various fresh items.

Now that he had lost nine thousand USA dollars, why bother to save pennies? Why not pay more money to enjoy life? In fact at the beginning of August, he shopped around for shoes to save pennies instead of going to a bank. It was literally saving pennies to lose thousands of dollars.

But in September he kept going to grocery stores to save pennies. The whole thing was irrational, he thought. Then it gradually dawned on him that "saving pennies" could not be explained in terms of economics: it was more like a hobby or a game. In fact, he enjoyed hunting for discount prices. Some psychologists may say it is compulsiveness. If so, then he should have some guilt feeling behind the compulsiveness. But he simply enjoyed discount shopping. Often he bought too much at discount prices and thus wasted money. Csikszentmihályi might say it is "flow". In any case, his "saving pennies" cannot be explained in economic terms. This corroborates the view that it was his "stupidity" of having missed the opportunity that bothered him more than the monetary loss of nine thousand USA dollars.

In a way, his habit of overbuying and overstocking originated in the late 1940s in Japan during the post-war period, when everything was scarce and one had to buy whenever one could find something in a store or on the black market. Another

habit he developed during the post-war period was to send off portions of his manuscripts to friends as he wrote them instead of sending the entire manuscript after its completion. The life at that time, more than during the war, was filled with uncertainties and one did not know when one might die. Railway trains had been destroyed during the war, and the few trains that ran came very infrequently, overcrowded, worn out and could not meet safety standards. Moreover, extensive bombings during the war destroyed many if not all of the houses in Tokyo, and people had to commute to Tokyo from long distances away. Passengers rode on the roofs of the trains, hang outside the trains, and doors of the overpacked trains occasionally cracked open by the passengers' body pressure from inside and spilled out passengers while the trains were running. Busses were also dangerous. Tires were so worn out that plies were exposed. Some vehicles had brakes which did not work. In any case the commuting distance was much beyond the short bus routes, and people had to ride trains. Another consideration was that one's own house might burn down anytime because houses were aging without proper maintenance and repair. For example, electric insulators were worn out, and a short-circuit might cause a fire. Under such conditions, it was wise not to wait for the completion of a manuscript because one might die or the manuscript might burn before its completion. This habit lasted for many years. During his postgraduate years in Sweden, he continued the habit even though everything was safe in Sweden. Either his habits were formed, not in childhood but in his adolescence, or his life during the postwar years had a long-lasting traumatic effect on him. It should be noted that while shopping in Japan after 1987 or while studying in Sweden in late 1950s, he no longer had the anxiety regarding food shortage or sudden death, but he continued his habit without the anxiety which caused the habit formation. It should also be added that even during the period of the habit formation, money was not the primary consideration. In the case of overbuying, the cause was scarcity of foods. In the case of manuscript mailing, the motivation was preservation of the manuscript.

3.2.4 Phase IV: intellectualization

He became less grief-stricken or broken-hearted about having missed the exchange rate of ¥80, and began to intellectualize the mistake. "If penny-saving is not an economic activity, then how can you empirically or experimentally prove it?" He decided to observe himself in similar situations in the future.

3.2.5 Phase V: intentional neglect or intentional repression

During the spring when he had a strong urge to go to a bank, he kept saying to himself: "Money is not the top priority. Other things have higher priorities." He fought the urge. He tried to forget and repress the urge. He was already busy but kept himself busier to forget the urge. He realized this in Phase V, and thought that he was successful in repressing his urge in the spring. Well, the loss was his mistake, but due to his success in repressing his urge. Perhaps he was not stupid after all.

The time span from the beginning of Phase I to the end of Phase V was about five weeks. The changes were continuous, not neatly divided into distinct phases. Then another turn of events took place.

3.2.6 Phase VI: apartment building repair and its consequences

He lived on the fifth floor of an apartment building. In mid-October there was a notice posted at the bottom of the stairway. The exterior of the building would undergo a decennial overhaul. This meant that the building would be encased in scaffolds for the workers to walk around, and covered with a net to prevent falling objects from hitting the pedestrians below. There would be very little daylight into the apartment. Because the workers would walk around outside the windows, the curtains must be closed all the time. Anyone other than the workers, for example a thief, could come up the scaffolds any time of the day or night. Therefore the windows must be locked. He did his work, mostly writing and reading, in daylight because otherwise his eyes got tired easily. This would mean that he would have to do his writing at the university.

In September he had received a letter from an editor, suggesting him to write some books. During the spring semester which lasted from April to July, he taught seven courses. Most of the courses in Japan last for a year. But an MBA course which he taught lasted only for half a year. Therefore during the fall semester, he taught six courses, which was much, but better than seven courses. He decided to work on these books in any spare time and on weekends at the university. This would keep his mind off the "silly mistake". Nevertheless, he could not completely forget about the mistake which kept coming back to his mind. He made another rationalization: the only way to justify the mistake would be for something good to turn out, for example writing a good book, as a result of the mistake. It was a far-fetched rationalization because he would try to write a good book regardless of whether he had made the silly mistake. Another rationalization was that the silly mistake might help him to avoid missing another golden opportunity. It was an unrealistic wishful thinking because the exchange rate of ¥80 was very unlikely to return.

3.2.7 Phase VII: out of the blue sky

At the end of November, *P* received a phone call from an old colleague who had moved to northern Japan to create a new university. The colleague *G*, now the president of the new university, said he would come to Tokyo to talk with *P*. From the tone of his voice, *P* felt that *G* probably wanted *P* to join the new university. *P* considered northern Japan to be an inconvenient location for his activities, but decided to talk with *G*. *G* came, and to *P*'s surprise *G* offered *P* very good conditions: only four courses to teach, and research expenses of more than US\$10,000 per person per year. *P* was cautious and answered that he needed to see the place before he could make a decision. *G* agreed to pay for *P*'s trip. This conversation took place in a morning.

Immediately afterwards there was a faculty meeting at *P*'s university to plan for the academic year 1996. It was announced there that everyone must teach at least six courses. In the past, the maximum had been 5 courses, though the average was 3 courses plus a seminar. Prior to this faculty meeting, *P* had been quite resentful to the circumstances which had led him to teach 7 courses

in the spring of 1995, and he had sworn to himself that he would not have more than 5 courses in 1996. When it was announced in the faculty meeting that everybody must teach at least 6 courses, his resentment surfaced in full force. All his intellectualization and rationalization of the previous Phases vanished. It was clear to him that he should move to *G*'s university. He had no hesitation. The unrealistic wishful thinking of Phase VI suddenly came true in an unexpected form.

The escape from the situation at the University in Tokyo was not the only motivation for *P*'s decision. The escape by itself would have been a sufficient condition. But there was another motivation which favored the decision, even though it alone was not a *sufficient condition*. Actually that motivation was triggered by the decision to reinforce the decision in the manner of a *causal loop*. But it had been latent until the decision was taken. Its nature was something like: "If that is the case, then I could also do *X* which I have never thought of being able to do even though I wanted to do it someday." This type of "cause" is neither conjunctive nor disjunctive. We may call it *parajunctive*. The parajunctive motivation in this case was as follows:

At *G*'s university, *P* would be able to create an institute of a very new kind—an institute for export product adaptation to foreign users' habits—an idea which he had advocated for some years because there was no such institute anywhere in the world.

Technologically excellent products may fail if they do not match the users' habits. Users' habits cannot be discovered by interviews or questionnaires. They must be observed. It is often necessary to use anthropological behavior observation methods, sociological participant observation methods and interaction methods, and psychological analysis of the habits. For example in the early 1980s Japanese cars tended to stall on the streets of Beijing in China. The reason was that the drivers in Beijing stopped the engine while waiting for the red signal to change, and in many Japanese cars the air-conditioning kept running after the engine was stopped. The battery went dead and the engine could not start. The problem was in the way the switch operated. But from the point of view of the users, the entire car was worthless. The solution was simple: to modify the

ignition switch to stop the air-conditioning when the engine was stopped [4, 5].

The most elementary step of export product adaptation is to meet the legal and technical specifications. For example, automobile emission control is a legal specification. Electric appliances' line voltage is a technical specification. Legal and technical specifications are usually explicit and visible, and therefore easy to meet. The second step is cultural and social adaptation. For example, in Japan chocolate is considered to be for children and young women. Therefore it has to be sized, shaped and packaged for them. Cultural and social product adaptation is already practiced by many firms. The third step is product adaptation to foreign users' habits. This is still underdeveloped.

In some countries such as the U.S.A. and France where the main product outlet is the domestic market, the manufacturers consider export as a way to get rid of the surplus without product adaptation. When they cannot sell, they blame their failure on foreign users or governments. In countries where the domestic market is small, such as Sweden, Switzerland and Holland, some manufacturers were able to export their products successfully because of the technological excellence of their products. As competition from firms in other countries increased, some of them had to switch to market-oriented policies, for example ASEA of Sweden in the 1980s. From now on, firms which study foreign users' habits will be far ahead of others in foreign market penetration. However, an institute for export product adaptation to foreign users' habits still did not exist. *P* had suggested it in France and in the U.S.A., but no one had implemented it.

The basic industries of northern Japan are fishery and agriculture. An institute for adaptation of fishery and agricultural products to tastes of consumers in foreign countries is needed and will be useful to the local firms. Outdoor agricultural products of Japan have almost no hope for exportation simply because of their high price resulting from decades of compounded government subsidy and protectionism [6]. But fishery products would still be price-competitive because of the high price in the U.S.A. For example, some products which are expensive in the U.S.A. such as scallops, urchin and abalone are abundant in

northern Japan. Salmon in northern Japan is not only inexpensive but has a seasonal cycle different from Alaskan salmons. Until now they are processed for Japanese taste. But they can be prepared for foreign tastes. Aquaculture technology is most advanced in Norway where, for example, the yield of roe per urchin is increased by 500%. *P* had lived in Sweden and Denmark. It would be a pleasure for *P* to develop exchange programs or joint research with Norway. Even in agriculture, there is one technology which is exportable to foreign countries: the indoor agricultural technology. It is ironical that the high price of outdoor agricultural products in Japan made indoor agriculture price-competitive [6]. Several firms had developed indoor agriculture technology in the 1980s. Vegetables were grown on racks without soil. The roots were sprayed with fertilizing liquid. Artificial sunlight was used 24 hours a day. Temperature and humidity were computer-controlled and adjusted to the growth stages of the vegetables. Evaporation was recycled. There were no insects and therefore insecticides were not used. Absence of weeds eliminated the need for weed killers. This type of indoor agriculture had several advantages over outdoor agriculture:

- (1) independence from weather and season;
- (2) independence from water supply;
- (3) fast growth of vegetables due to use of artificial sunlight 24 hours a day;
- (4) absence of insecticides and herbicides which are harmful to humans;
- (5) It can be operated in urban areas, eliminating the transportation time to assure fresh supply, as well as reducing transportation costs;
- (6) It can be used in arid lands, polar regions and outer space communities, where agriculture was difficult until now.

In the 1980s the indoor agriculture was able to compete with the outdoor agriculture in Japan because of the high price of outdoor agricultural products due to compounded results of the governmental protectionism [6]. But in the early 1990s the exchange rate of the Japanese currency became very high, and many department chains

and supermarket chains in Japan developed import channels for all types of merchandises including agricultural products to make use of low foreign prices. Need for indoor agriculture decreased. Currently there is only one firm called QP (pronounced "kew Pee" which is a japanization of "cupid") which still continues indoor agriculture. But application of indoor agriculture technology in arid lands such as Saudi Arabia, in polar regions and in outer space communities will grow. Eventually some arid regions of the world may become new industrial and economic centers of the world when fuel energy becomes exhausted or expensive.

The concept of export product adaptation to foreign users' habits and tastes, and use of indoor agriculture technology in arid lands and outer space communities are two topics which are new and long-range future-oriented. An institute which will begin from a scratch can incorporate these topics. *P* had always loved exploratory and pioneering work, and the institute would open new frontiers. This was a strong *parajunctive consideration*.

There were two other *parajunctive considerations*. The first was that *P* was once an electronics engineer, and he always wanted to analyze some technological elements in cultural and social change. The other was that he had worked in Alaska in mind 1960s. At that time, some faculty members at the University of Alaska were thinking of creating an institute which would facilitate research exchange between USSR and USA. It was during the Cold War, and the institute did not materialize. *G*'s university in northern Japan, since its inception in 1992, already had exchange programs with Russia, more specifically with Vladivostok. For *P*, this had an effect of a revival of an old dream. These several *parajunctive considerations* seemed to fit together with one another and with other considerations like pieces in a jigsaw puzzle, making an *interlocking*.

Exploration and innovation had always been the main motivations in *P*'s work. In the past, when the official policy of his employers did not quite encourage them, *P* did it on his own anyway. But in *G*'s university, *P* would have an official support and encouragement.

At the university in Tokyo where *P* was, the Dean was in his own way innovative and initi-

ated many programs, including the first evening MBA program in Japan, and an internationally linked business policy computer game. But the Dean made all decisions. The faculty simply had to follow. The political wind against him from other components of the university was very strong, especially regarding the faculty/student ratio. Partly because of it as well, he kept adding new types of programs without reducing the faculty load elsewhere. The faculty members endured it in the Japanese style. If one overloads a truck, accidents are likely to happen even though each of what is put on the truck is marvelous. In fact accidents happened in *P*'s case: his teaching load caused delays in his colleagues' research in many countries. Not only he wanted to avoid further accidents, but also he needed a place where he could create some programs himself. But personally he had nothing against the Dean. In fact, the Dean did everything possible to facilitate *P*'s attending international conferences and other activities. Basically, the university did not permit its faculty members to go to foreign countries during the duration of their courses. But the Dean did whatever he could under the circumstances. *H* and *P* remained in excellent friendship until and even after *P*'s moving to the other university.

Before summarizing the non-additive as well as additive effects of various stresses which led to *P*'s decision, we must consider also the degree of the stresses as indicated by delayed physiological symptoms which did not surface before the time of the decision.

3.2.8 Delayed effects of the stresses

Sometimes the effects of stresses appear long after the stresses are removed, especially the effects on the body. *P* had some delayed physiological or somatic effects which appeared after he had made his decision to move. They are reported here to indicate the severity of the stresses, even though they were latent and were not visible or tangible at the time of the decision making.

Toward the end of February 1996, *P* noticed a darkening in his left eye visual field which he kept ignoring mainly because this symptom resembled a problem he had in 1974 which had no serious consequences. But it was a dangerous problem, very different from the one he had in 1974. Let

us first see the 1974 problem, then return to the 1996 problem

In the spring of 1974, *P* was under a severe psychological pressure which will be explained later. His eye problem occurred during that time. While driving at night, the lower quarter of his right eye visual field suddenly went blind. He immediately stopped his car. Realizing that his left eye was normal, he drove home. By the time he reached his home, the right eye almost recovered: the blind quadrant was by then a dark quadrant, in which he could see things tinted purple. Next day he went to an ophthalmologist. The physician could find nothing wrong, and concluded that it was probably due to a temporary blood occlusion which might or might not have damaged the retinal nerves, and therefore the darkening might or might not disappear. In any case the blood circulation was normal when the physician examined *P*'s right eye.

It turned out that it took the symptom three or four months to disappear completely. The darkening was most noticeable when *P* blinked his right eye. Immediately after the opening of the eye, the "shadow" appeared but gradually disappeared. The shadow was always there, but the brain adjusted for the darkness. As days passed, the dark quadrant receded to a smaller and smaller shape, becoming like a slanted hammer. By mid summer, even the hammer disappeared.

In 1996, recalling that in 1974 he was given no therapy nor medication, but the eye healed itself, *P* thought that it was unnecessary to go to an ophthalmologist immediately when a new problem began toward the end of February 1996.

An exciting and pleasurable experience had begun in October 1995 which alleviated *P*'s psychological pain. A graduate student of his from Canada was making a progress in her very interesting research project. She devised and sent out several hundred copies of a questionnaire, one third of which was printed in English, another third in French, and the rest in German, to European and North American managers working in Tokyo. Her questionnaire was a composite of several psychological tests, ranging from Adorno-Sanford F-scale and Rokeach scale of open-mindedness to more recent tests. She had several objectives: (1) to check the valid-

ity of these tests when used with subjects from many cultures; (2) to find correlations among various tests; (3) to see whether these European and North American managers working in Tokyo would score differently from those staying in their home countries; (4) whether the differences was the "cause" or the "effect" of coming to Tokyo; (5) whether the individual epistemological types would be found across national or cultural boundaries, i.e. any type found in one national or cultural group could be found also in other national or cultural groups. She was getting a very high percentage of questionnaire return. The data would be sufficient to meet all five of her objectives, and could generate more than one Ph.D. dissertations even though she was going to use them for her master's degree. By mid-January 1996 the data were beginning to pour in, and she was making surprising discoveries which excited him. He was in touch with her every day and night over the telephone and by fax. Her oral defense of the thesis would take place on March 6. He decided not to go to an ophthalmologist until after March 6.

The darkening in his left eye visual field in 1996 somewhat stayed steady in the lower left portion, similar to the one in 1974. But around March 6 it began to spread toward the center, gradually spreading beyond the center. On March 7 he began to look for an ophthalmologist. In the morning of March 9 he went to an ophthalmologist near his university, and she advised him to see a specialist for this type of problems. He went to the special clinic, and it immediately alerted him of the seriousness of the problem: bleeding in the retina, very different from the problem he had in 1974. He was given a medication. The bleeding stopped spreading, but did not recede. In March his left eye was quite dysfunctional, especially for reading and writing.

On June 6 another eye problem struck. He was taking a short walk before breakfast as a daily routine. Suddenly his right eye went almost blind. Dark purple color covered his visual field and the trees and the sky disappeared. It was as if ink was spilled over a somewhat bumpy table, leaving islands which gradually submerged. He stood stupefied. Fortunately the "ink" began to subside after about three minutes and disappeared gradually. It was almost similar to what happened in

1974. Therefore he did not panic, even though he thought he should see the ophthalmologist (hereafter called *W*) as soon as possible. *P* had an urgent work in that morning, and decided to postpone his visit to *W* until after the work. When *W* looked into *P*'s right eye, *W* almost panicked: the main vein in the retina was almost completely clogged and *P* could go blind at any moment. *W* immediately put *P* on an intravenous drip treatment. *W* said *P* should continue the treatment for one hour every day for a week, and that *W* would come to the clinic on the weekend just for *P*. *W* was tired and exhausted, but did come on the Saturday and Sunday. *W* saved *P*'s right eye.

In 1996 his two eyes had opposite problems: in the left eye he had bleeding, and in the right eye he had an occlusion. The treatment for occlusion may increase bleeding. But curiously his left eye began to get better during the intravenous drip treatment. Therefore some invisible occlusion in a small vein in the left eye retina may have been the cause of the bleeding because a blockage at one point may increase the blood pressure at a point upstream. Consequently the ophthalmologist added an anti-clogging medication to stop *P*'s left-eye bleeding even though it seemed contradictory.

Judging from the symptom of the "ink", what happened to *P* in 1974 must have been also a blood occlusion, much lighter in degree and without heavy clogging. The spring of 1974 was a time of great psychological stress for *P*: he was undergoing the process of divorce with his first wife. The lawyers of both sides prohibited them to communicate directly, and this created tremendous anxiety and worries in *P*'s mind, even though after the divorce *P* and his first wife remained as good friends. Considering this, in the spring of 1996 *P* must have been under a stress worse than in 1974.

If standardized tests had been used to measure *P*'s level of satisfaction with separate aspects of the university in Tokyo, and if the scores had been computed additively with weight coefficients, then the satisfaction level would have come out quite positive, and no real understanding could be forthcoming. On the other hand if we use situation-contextual and time-contextual non-additive combination analysis, the diagram in Fig. 1 emerges.

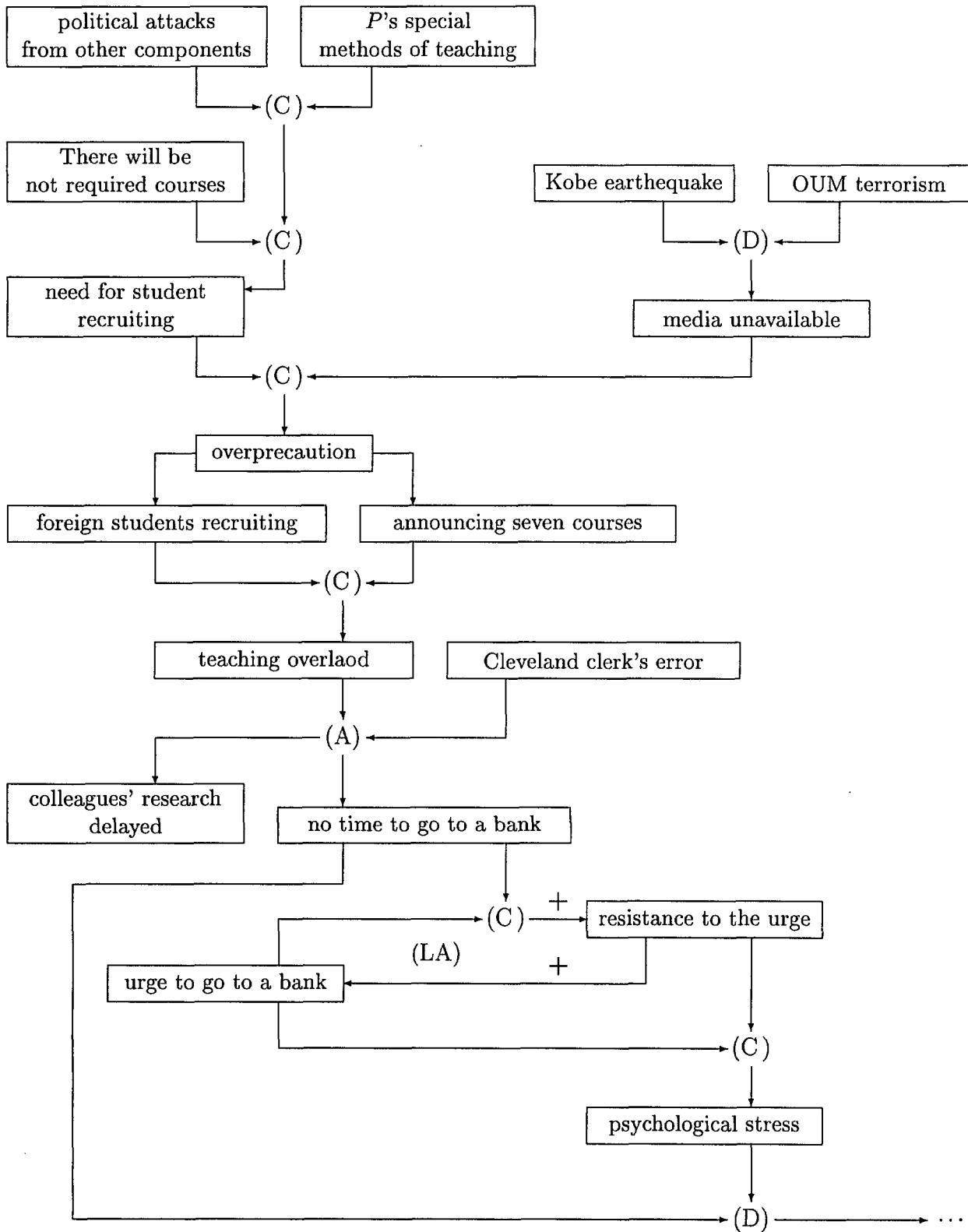


Figure 1: Keys: (A) additive; (C) conjunctive; (D) disjunctive; (LA) change-amplifying causal loop; (LC) change-counteracting causal loop; (P) parajunctive; and (I) interlocking.

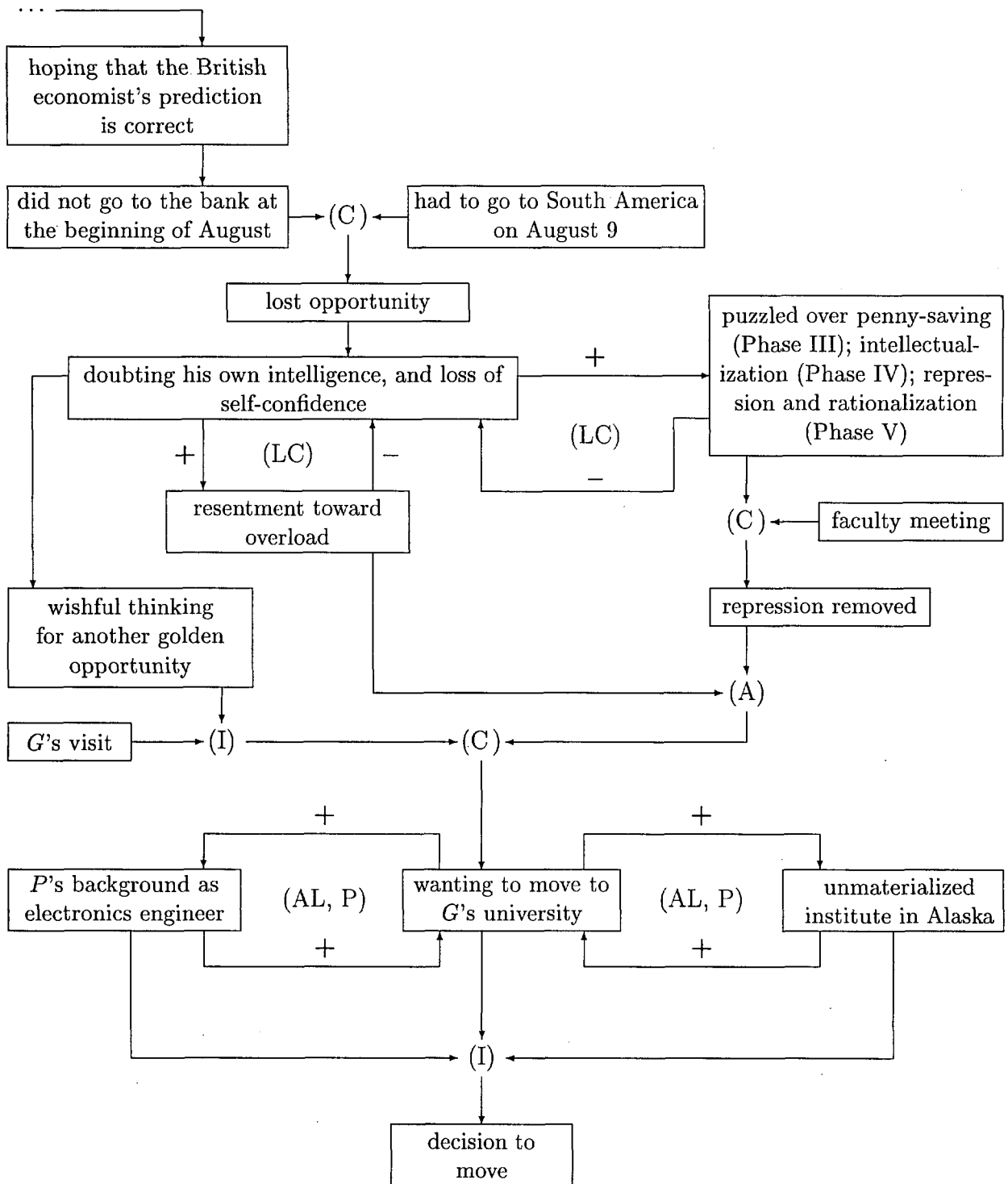


Figure 1: Continuation of the diagram (from the low-positioned '...' on the left side to the high-positioned '...' on this side).

References

- [1] HIDAKA, K. 1995. Professor who sends out the business sleuth. *Asahi Evening News* April 24, 1995.
- [2] MARUYAMA, M. 1989. Morphogenetic teaching in international business. *Journal of Teaching in International Business* 1: 77-93.
- [3] MARUYAMA, M. 1994. Successive modifications of study guidelines for student projects with business firms. *Cybernetica* 37: 59-72.
- [4] MARUYAMA, M. 1990. International proactive marketing. *Marketing Research* (June 1990) 36-48.
- [5] MARUYAMA, M. 1992. Changing dimensions in international business. *Academy of Management Executive* 6: 88-96.
- [6] MARUYAMA, M. 1987. Japan's agricultural policy failure. *Food Policy* (May 1987) 123-126.
- [7] TOYO KEIZAI SHIMPO SHA. 1994. *Nippon no daigaku 1995*. Toyo Keizai Shimpo Sha. Tokyo.

The Definition of an Integrated Software Life-Cycle Tool

Rick Leonard
 PEI Electronics
 110 Wynn Drive, Huntsville, Alabama 35812
 USA

Keywords: computer aided software engineering, convergent metrics, life-cycle models, software engineering, software metrics

Edited by: Xindong Wu

Received: July 12, 1996

Revised: January 17, 1997

Accepted: January 22, 1997

For Computer Aided Software Engineering to fulfill its promise, every aspect of the software product should be accomplished within an integrated software engineering environment. Although integrated software engineering environments have been attempted in the past, several areas of concern still exist. Too often, the life-cycle models and their resulting products are incongruous between all phases of the software life-cycle. With this incongruity comes a loss of previously captured information which, in turn, results in a larger, overall work effort to complete the software product. Incongruity between life-cycle phases also causes inconsistent evaluation metrics and, as a result, improper life-cycle management. In addition, techniques for reconciliation of errors with previous life-cycle products are practically non-existent. This paper attempts to overcome the aforementioned problems by providing the background, methodology, and technical analysis to demonstrate an interactive software engineering environment capable of addressing the entire software life-cycle in an integrated, automated fashion. In addition, a related metric set is identified that converges with actual results as the life-cycle progress from one phase to the next.

1 Introduction

1.1 Statement of the Problem

As software professionals, we tend to decompose the problems associated with software development into a finite set, rather than address them as an integrated whole. Consequently, the problems associated with requirements definition, design, implementation, and test are usually addressed with entirely different tool sets. Since the tool sets are unrelated among life-cycle phases, an inconsistency among life-cycle products results. The inconsistency among life-cycle products gives rise to a wide assortment of problems including, the duplication of work, unrelated metrics, difficult to harvest metrics, and the inability to efficiently reconcile errors between life-cycle phases in an efficient manner. As a result, a life-cycle process that generates homogeneous products in an automated fashion remains under utilized by the soft-

ware profession.

To overcome these problems, we provide the background, methodology, and technical analysis to define an integrated software life-cycle environment, that uses structured analysis and design as the keystone. As an expert system, this tool provides the software project with the following features:

- A set of related metrics that will objectively:
 - Estimate the software development, maintenance, and test efforts during the earliest phases of the life-cycle,
 - Converge with actual results as the life-cycle progresses from one phase to the next.
- A software engineering tool set that will automatically:

- Integrate each life-cycle phase with products from previous phases,
 - Integrate formal documentation with each life-cycle phase,
 - Generate variable specifications, input/output range tests, and function calls during implementation,
 - Generate boundary-level test code for each module during formal and informal CSU, CSC, and CSCI testing.
- Reverse engineer the system to provide:
- Errors of estimate between life-cycle phases and allow these estimates to be used as metrics to determine subsequent programming efforts,
 - Automatic reconciliation of product errors between phases.

1.2 Organization of the Paper

This paper defines the components of an automated software engineering environment that integrates software life-cycle methodology and metrics with an interactive computer program, hereinafter called, the software engineering tool. The software engineering tool requires the definition of a formal software life-cycle model plus the identification of any software development techniques that could enhance the definition and/or refinement of a program size metric. Each of these components are further analyzed in subsequent sections of this document.

Section 2 provides an analysis of currently available software engineering tools and their deficiencies. The analysis covers pertinent metrics, a common life-cycle approach, and a limited review of relevant structured analysis and design techniques. Since the structured analysis and design review is constrained, a complete examination of the subject matter may be found within the literature identified by the bibliography.

The third section provides solutions to the problems identified in Section 2 and reconciles the objectives set forth in Section 1. As such, a set of convergent program size metrics are identified, methods of integrating life-cycle products into later phases are presented, while code generation, test, and documentation are automated to the greatest extent practical.

2 Analysis of Available Tools

2.1 Introduction

A survey was conducted to distinguish research related to the identification of metrics used for measuring software programming effort. As a result, program size, in terms of the number of lines of code (LOC), was identified as one of the better metrics for measuring programming effort. Subsequently, a mature life-cycle approach to the software development process was identified. This led to further research involving the availability of tools designed to attack the various problems inherent to the software life-cycle. In addition, this survey identified top-down software development techniques that enhance software productivity and integrate well with the life-cycle model and software engineering tools chosen. Each of these topics are further addressed in subsequent paragraphs.

2.2 Pertinent Metrics

Techniques for objectively estimating the size of a software product during the early stages of the software development effort are currently unreliable. Generally, this task has been consigned to the experienced software professional whose collective estimation methods are limited to his or her own personal judgments or intuition. Hopefully, these estimates converged with actual results as the software development process advanced from one phase to the next.

As a part of the software evaluation process, metrics are used as a quantitative measurement of the effort and subsequent cost of the problem solution. Once the appropriate metrics are established within the software life-cycle, they may be used during the earlier project phases for predicting effort and cost. Current studies indicate that program size is very closely associated with the amount of effort required to accomplish the software task and, subsequently, the overall cost of the software product.

Consider the following equation as the basis for determining the level of programming effort required to develop a software product:

$$E = S/P \quad (1)$$

where E = level of effort, S = size of job, and P = productivity of workers.

Assume that the level of effort may be expressed in terms of person-months. In addition, assume that the size of the job may be expressed in terms of the number of lines of code (LOC) in the finished product. Also, assume that the productivity of workers may be expressed in terms of number of lines code (LOC) accomplished per person-month. By substitution, Equation (1) translates into the following:

$$E = (LOC)/(LOC/Person - months) \quad (2)$$

Most research efforts have centered on identifying factors that affect the productivity of workers (the P designator of Equation (1)). Factors affecting the size of a project (the S designator of Equation (1)) have only recently been addressed. Moreover, most research assumes that the size of the job, in terms of the number of lines of code (LOC), remains constant, or at least uniform, throughout the life-cycle of the project. In fact, when various models of productivity (such as Boehm's COCOMO model) are used to determine the level of effort (E), job size (S), in terms of LOC, generally remains static throughout the development life-cycle. As a result, a wide disparity exists between the LOC estimated at the beginning of a project and the LOC delivered in a finished software product. Techniques for reconciling this lack of convergence are fully developed in Section 3.

2.3 Life-Cycle Models

A software life-cycle model partitions the processes required for the development of a software product. In addition, it provides the engineer with a framework that is used for evaluating the impact of alternative solutions to the specific problems encountered as the product is being developed. The building of this framework should take the developer from the conception of the project through its phase out. If this task is approached using a life-cycle model, then the first step is to separate the software development process into sequential phases. The sequential phases include requirements analysis, preliminary design,

detailed design, implementation, test, integration, and delivery. This life-cycle approach was chosen because it reflects current military and industrial standards. Beginning with requirements analysis, the following paragraphs identify techniques designed to overcome the problems inherent to each phase and their deficiencies.

2.4 The Tools of Structured Analysis

Proper requirements specification is critical to the success of any project. The most difficult aspect of properly specifying a software system is the translation of the system concepts into a tangible form. Accomplishing this requires two steps [6]. The first step is the top-down decomposition of the software system into smaller, more manageable processes. The second step is the bottom-up synthesis of the smaller, more manageable processes into a workable system. For this paper, data flow diagrams (DFD's) are used as the principal tool in converting the concept of a system into a tangible form. The specification tools include data flow diagrams (DFD's), a method of accessing the DFD, a data dictionary, and a method of describing the specification.

Data flow diagrams specify systems by portraying the processes and data elements comprising a software system as a network. The entire DFD network can be expressed using the data flow, represented as a line; the process, depicted as a bubble; the data store, drawn as a pair of straight lines; and the source/sink, symbolized by a box (see Figure 1).

The data dictionary produced during requirements analysis gives rigor to data flow diagrams by providing the analyst with a central repository of data definitions. This includes definitions of all data stores, unique data flows, and functional primitives. A data flow that can be decomposed farther is defined in terms of its components. Components consist of other data flows and data elements. A data element is a data flow that cannot be decomposed any further (in Section 3 these are referred to as data element (DE) variables). It is defined in terms of the meaning of each of the values that it can assume [6]. For example:

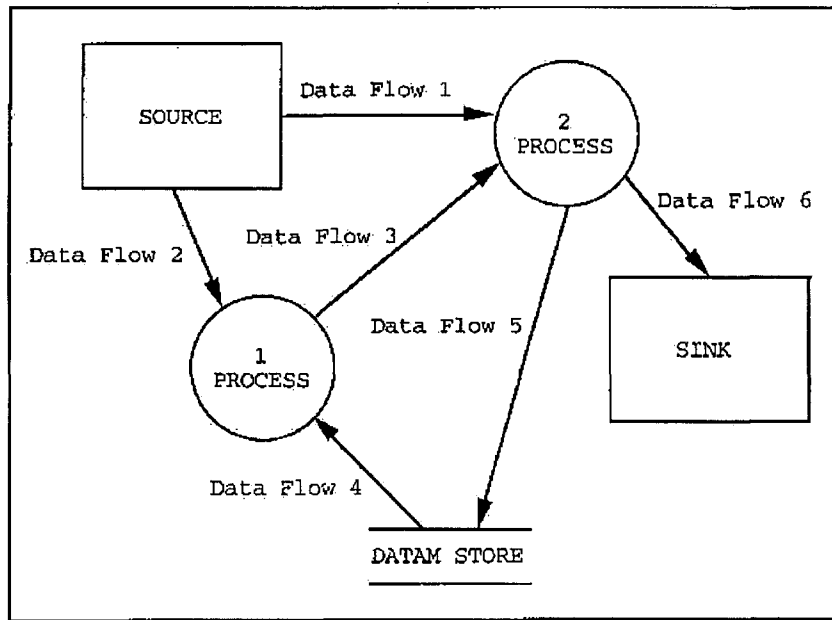


Figure 1. DFD Components [6]

DATA FLOW	DATA ELEMENTS
telephone number	area code local number extension

Data flow diagrams alone do not specify a software system, but they do provide a computerized tool for decomposing the system into successively smaller processes. The processes that cannot be decomposed any farther are called functional primitives. It is the functional primitives of the system that are formally specified. If each functional primitive is specified in a concise and independent manner, then processes at higher levels do not need to be specified since they are nothing more than a collection of lower-level processes [6]. The simple mini-spec in Figure 2 demonstrates the ability of structured English to express a functional primitive in a concise manner.

2.5 The Tools of Structured Design

The design phase of the software life-cycle may be considered the bridge between the specification phase and the implementation phase of the software project. Its primary goal is to translate a specification document into a rigorous design document that can be used by a software engineer as

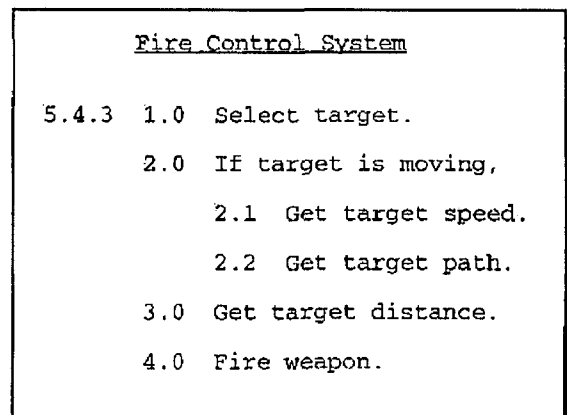


Figure 2. Mini-Spec Example [6]

a guide for implementing code. In this regard, a structured design document may be considered as the blueprint for programming a software system [6]. Structured design provides the system designer with a disciplined approach for transforming the products of structured analysis into the products of structured design. In particular, the computerized data flow diagrams of structured analysis are refined into a more detailed graphical product called structure charts, while the computerized mini-specs that describe the processes within the data flow diagrams are refined into a more detailed description of the functions parti-

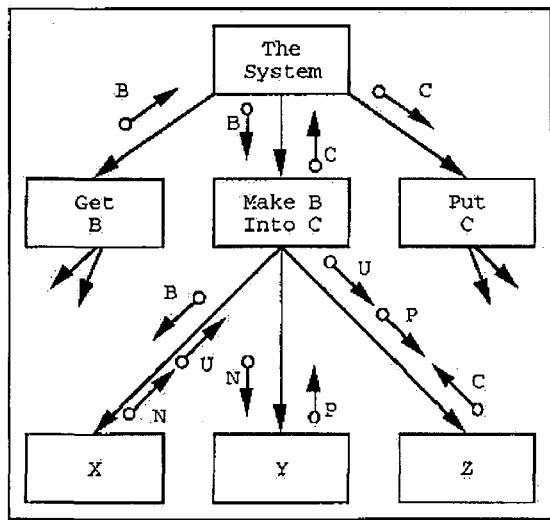


Figure 3. The Structure Chart [6]

tioned by the structure charts.

A structure chart is a graphical tool that represents the manner in which a system will be implemented. It is derived from the data flow diagrams that were developed during structured analysis. Structure charts are used to partition large or complex systems into smaller more manageable modules for coding (see Figure 3).

As the primary tool in preliminary design, structure charts are used to graphically depict the partitioning of a process into smaller modules, the top-down organization of the modules, the data flowing into and out of a module (in Section 3 these are referred to as a logic and control (LC) variables), and a functional description of the modules [6].

From Figure 3, a structure chart is composed of the following three symbols, which are used to illustrate the entire software system:

1. modules - depicted as a rectangular box. A module represents a single subroutine within the system.
2. connections - drawn as a line linking two modules. It is used to symbolize a call from one module to another.
3. couples - indicated by a short arrow with a circular tail. A couple represents a data flow from one module to another.

The transformation of data flow diagrams into

structure charts is accomplished by transform analysis, which enables the software engineering to translate the network of processes that comprise a data flow diagram into the hierarchy of modules that form a structure chart. By deriving the structured design from a structured specification, continuity is provided between the specification and design phases of the software life-cycle.

At this time, the only task left within the design phase is the translation of the mini-specs into a detailed description of the processes partitioned by the structure charts. Most texts recommend pseudo code in the form of a program design language or program statement language to accomplish this task.

2.6 Conclusions

Data flow diagrams and structure charts provide excellent methods of quantifying requirements then translating them into an acceptable design. However, most products do not translate into the implementation phase. In particular, a number of phase-dependent products are lost. These products include previously defined documentation, data dictionary declarations, and the opportunity to generate any associated code. In addition, none of the structured analysis and design products translate over to the test phase. Consequently, independent IV&V tasks are extremely costly. Finally, the number of lines of code (LOC) metric is not fully realized until the implementation phase has been completed. As a result, empirical estimations of LOC can not be achieved during the earlier phases of the life-cycle. Solutions to each of these problems will be addressed in the following section.

3 An Integrated Software Engineering Tool

3.1 Introduction

Consider the structured analysis and design techniques identified previously. The products produced during requirements analysis flow nicely into the design phase, however, few of the resulting products provide benefit to the implementation or test phases of the life-cycle. To overcome this problem, the following paragraphs identify

structured analysis and design process enhancements that maximize product flow throughout the life-cycle, minimize duplication of effort, and automate the life-cycle wherever possible. In addition, a metric set is presented that converges on the LOC as the life-cycle progresses from one phase to the next. It is to this metric set that we now focus our attention.

3.2 Program Size Metrics

Although research by Wang [2] indicates that the number of unique variables (VARS) within a program can be used to predict LOC as a simple linear regression, this metric isn't available until very late in the software development life-cycle. Further research [5] indicates that subsets of VARS exist such that the number of data element variables (DE), the number of logic and control variables (LC), and the number of static variables (SV) are equal to VARS. In addition, data flows (DF) link the DFD's and identify the interaction of the individual tasks with other tasks. The logic for developing these equations follows.

For data structure metrics concentrating on the data elements flowing between software modules, consider the following equation,

$$VARS = DFE + SV \quad (3)$$

For Equation (3), the data flow elements (DFE) are equal to the number of unique DE and LC variables flowing between modules, while SV is equal to the number of unique variables that exist only during the execution of the module. For DFE, Equation (4) follows:

$$DFE = DE + LC \quad (4)$$

Substitution into Equation (3), provides the following relationship,

$$VARS = DE + LC + SV \quad (5)$$

Considering that DF, DE, LC, and SV are subsets of VARS, and that these subsets are identified at progressive points in the life-cycle by the continuous refinement of the software module's data flows (as described in the previous section), then, the assumption can be made that these subsets can be used to predict LOC during the early stages of the life-cycle. The credibility of this assumption is validated by regression analysis [5],

the results of which may be found in Table 1 and Table 2.

3.3 Requirements Analysis

If the structured analysis methodology described in Section 2 is automated as an interactive program, additional integrated software tools are required. In particular, an editor should be provided that addresses both formal documentation demands (as presented to the customer) and the requirements specification products (as provided by the analyst). To further automate the documentation process, the editor should generate the appropriate formal documentation templates including appropriate paging, section numbers, paragraph numbers, and headers. The analyst then updates the templates up to, but not including, the system requirements paragraphs. Additionally, system requirements paragraphs are linked to a data flow diagram drawing package. The DFD drawing package allows on-screen process illustrations as depicted in Figure 1. Once illustrated, the processes, data flows, data elements, mini-specs, etc. may be further described within the data dictionary utilizing the same editor. As an added feature, the data dictionary requires both a valid and invalid range specification for each data element. Reasons for this requirement will become clear during the implementation and test phases.

As a result of the formal documentation template, its linkage into the systems requirements paragraphs, and the special editor, process documentation may be generated automatically after the system is defined. Specifically, process and sub-process definitions, data flow definitions, and mini-specs collected within the data dictionary are then written to the formal documentation in a prescribed format consisting of paragraph numbers, requirements description paragraphs, input variable definitions, mini-specs, and output variable definitions. Consequently, the analyst is not required to define and analyze the system with one tool, then transfer the system information to a formal documentation standard using another tool. This minimizes duplication of effort, both in this life-cycle phase and in subsequent phases.

COCOMO provides estimates of schedules, person-power, cost, etc. based on program size, program structure, and certain productivity met-

Life-Cycle Phase	Metric	Correlation	T Significance	F Significance
Early Spec.	DF	0.694	P0.1	P0.1
Late Spec.	DF	0.941	P0.02	P0.01
	DE		P0.01	P0.01
Early Design	DF	0.984	P0.02	P0.0001
	LC		P0.001	P0.0001
Late Design	DF	0.987	P0.2	*
	DE		P0.5	*
	LC		P0.3	*
	SV		P0.7	*
	VARS		0.905	P0.01

* Not Computed

Table 1: A VARS Components Comparison

Life-Cycle Phase	Metric	Equation
Req. Defin.	DF	$LOC \approx 29.027 + 3.892 * DF + E$
Req. Analy.	DF	$LOC \approx -11.434 + 3.072 * DF + 5.843 * DE + E$
	DE	
Prel. Design	DF	$LOC \approx -5.729 + 1.942 * DF + 12.654 * LC + E$
	LC	
Detail Design	VARS	$LOC \approx 1.583 + 5.429 * VARS + E$

Table 2: Equations for Predicting LOC

rics. Although the productivity metrics generally remain static during requirements analysis, program size and structure may vary. As a result, the system may be analyzed against time, budget, and person-power constraints by automatically supplying LOC estimates to COCOMO from the DF and DE metrics identified previously. In addition, program structure metrics may be determined from the data flow diagramming network itself, then supplied to COCOMO automatically. Consequently, certain managerial aspects of the life-cycle become automated without any additional efforts.

3.4 Automated Design

Although data flow diagramming techniques provide a well defined set of products flowing from the requirements specification phase into the design phase, some process adjustments must be made to accomplish the goals set forth herein. As with the requirements analysis phase, consider a

formal design documentation template, its linkage into the design package, an editor capable of addressing each of the design products, and an expanded data dictionary as the principle enhancements to the design phase tool set.

Enhancements to the formal documentation process allow the appropriate requirements documentation to flow directly into the applicable design documentation. Subsequently, undefined paragraphs are identified to the user. As with requirements analysis, design is enhanced by the linkage between the documentation and the design processes. If additional requirements are identified during design, the relevant requirements products are immediately updated via backlink. The information then flows into the appropriate design products.

To generate the formal design document, process definitions, data definitions including logic and control variables, and mini-specs are collected from within the data dictionary, then written into the formal documentation package in a

prescribed format consisting of section numbers, section headers, paragraph numbers, paragraph headers, description paragraphs, input variable definitions, mini-specs, and output variable definitions. As during requirements analysis the software engineer is not required to design and analyze the system with one tool, then transfer the information to a different tool set to complete formal documentation or continue additional analysis. This reduces duplication of effort, both in this life-cycle phase and in subsequent life-cycle phases.

At this point, a computer aided transform analysis occurs, the system is decomposed into functional primitives, and the logic and control variables are added to the data dictionary. Subsequently, schedule, budget, and cost constraints may be addressed by automatically updating CO-COMO's program size estimates using the DF and LC metrics identified previously. As a further contribution to this analysis, the program structure metrics may be updated based on the design network.

3.5 Automated Implementation

Based on previously captured information, most of the documentation required during the implementation phase may be automated. Consider that information regarding the author, requirements paragraph number, requirements description, design paragraph number, design paragraph description, input variables, input variable descriptions, process description (mini-specs), output variables, and output variable descriptions could be placed into the code module as comments based on the information captured during the requirements and design phase. Additional comments based on the data flow diagramming network could include calling program references (which are very difficult for a programmer to maintain) and called program references. Furthermore, all of this information could be included in the module before any coding begins.

As the software product enters the maintenance phase of the life-cycle, comments regarding change history could be tracked and referenced in this manner also. In addition, a programmers updates to the previously described information could be reversed engineered into the formal documentation from the code module using the editor

described previously. As a result, consistent, errorless, up-to-date requirements and design documentation could be provided to the customer almost immediately. While this may cause some anxiety to the software seller, a software buyer, could monitor product milestones, changes, etc. in a very efficient manner.

Although comments embedded within the code describing specific processes would still be provided by the programmer, obviously, the integrated approach reduces the programmers overall commenting effort, reduces discrepancies between comments in the formal documentation and the code module, provides uniform documentation standards for the programmers (studies indicate that commenting styles vary widely), and aids in the overall programming effort by clarifying the purpose and intent of the code module to the programmer in an efficient manner.

Using the requirements analysis and design methods described previously, a certain amount of code could be automatically generated for the programmer, completely independent of language. Consider both the data dictionary and the logic and control variables identified previously. If variable type is included in the definition, then the variable declarations could be automatically generated for the programmer. As a further enhancement, these variables could be locked by the software engineering tool to provide consistency throughout the system, or, unlocked to provide quick updates to all modules. The only variable declarations left to the programmer are the static variables.

If the invalid variable range is included in the definition, input and output checks could be automatically generated by the software engineering tool also, although specific instructions regarding exceptions would be left to the programmer. In addition, code regarding calls to other modules could be automatically generated within the module based on the data flow diagramming network. As with the exception code, the process code generated between calls would be the responsibility of the programmer.

A quick visual inspection of FORTRAN modules developed under structured analysis and design techniques indicates approximately 25% to 35% of a small to medium sized software modules code could be generated in this manner; a larger

percentage for smaller modules and a smaller percentage for larger modules. The choice of programming language probably plays a key role in this estimate. Assuming that the code generated in this manner is virtually errorless, the amount of implementation, test, and integration time required to deliver the module should be greatly reduced. In addition, some coding conformity among software modules could be enforced by the software engineering tool.

3.6 Automated Test

The cost of completely testing a software product would be enormous. Consequently, methods of selecting effective test sets and methods of automating software testing are highly desirable. Consider a given specification S and a program P . As the primary goal of testing, we want to determine if P is correct with respect to S . Let f_s and f_p denote functions representing S and realized by P , respectively over domain D . For statistical testing, we want to select a test set T of test inputs over D such that T has the following property P1:

P1: If for all $t \in T$, $f_p(t) = f_s(t)$, then f_s and f_p are probably equivalent.

A test data selection method is said to be statistically effective if there is a mechanical procedure to generate test set T with the property P1.

Consider boundary level testing, where software modules are tested by manipulating the input and output variables above a valid boundary, on the boundary, and below the boundary in order to reduce test cases. If the variable specifications defined in the previous steps include valid and invalid ranges, then the software engineering tool described herein could automatically (mechanically) generate all the test cases for a given software module, all the software needed to drive each case, compare the case against known results, and mark the case as valid or invalid based on those results. Consequently, the programmer is allowed to conclude a well defined cycle of design, code, and test for each module, in a complete and thorough manner.

Formal test documentation can be generated in a similar manner. Consider that the previously captured information is linked to a test document template. The appropriate test documentation

information could flow into the applicable paragraphs. Design process definitions, data definitions, and mini-specs collected from within the data dictionary are then written into the formal documentation package in a prescribed format consisting of section numbers, section headers, paragraph numbers, paragraph headers, description paragraphs, input definitions, mini-specs, and output definitions. In addition, the software engineering tool would generate all boundary level test cases, then incorporate them into the documentation as well. As during the previous phases, the software engineer is not required to test the system with one tool, then transfer the information to a different tool to complete formal documentation or continue additional analysis. Again, duplication of effort, both in this life-cycle phase and in subsequent life-cycle phases is reduced.

While the benefits to the three life-cycle phases mentioned above are obvious, the requirements analysis and integration phases should benefit, to varying degrees, from this methodology also. While the requirements test documentation and test cases could be generated as before, the requirements for some systems (real-time, simulation, etc.) could not be completely tested using this philosophy. Nevertheless, the independent test cycle could be reduced to validating system requirements, auditing software module, component, and system test compliance, then generating the appropriate documentation under the methods described. Obviously this would result in a substantial overall savings compared to current methods.

4 Conclusions

4.1 Introduction

At this point all of the components of the integrated software engineering tool have been identified and developed. While some components may not be unique, we have yet to embrace a single software engineering tool that comprises all of them. By allowing a software engineer to implement each component through an interactive, integrated software engineering tool, many aspects of the software life-cycle could be addressed in an automated manner. In addition, the impact of changing any one particular factor could be analyzed and the best alternatives could be chosen

without a detailed knowledge of the tools components. Consequently, duplication of efforts, lack of automation, inability to identify problem areas early, and a larger, overall work effort could be minimized.

4.2 Contributions

Consider the software processes (data flows, data diagrams, etc.) proposed. The development process converges on a finished product as the software life-cycle progresses from one phase to the next. At each phase, metrics (DF, DE, LC, SV, VARS) become available that allow the size of the finished program (S) to be predicted more accurately. When used in conjunction with the various models of COCOMO, effort (E) estimates may be determined more accurately as the life-cycle converges on a finished product. This allows problems with under budgeting and restrictive due dates to be addressed as early in the software life-cycle as possible.

Enhancements to the data flow diagramming process allow integration of phase dependent software products throughout the life-cycle. These products are also linked to the formal documentation effort. As a result, data is captured once, then made available throughout the life-cycle as necessary. During implementation, documentation is virtually automated with the resulting comments uniformly provided while coding is automated to the greatest extent practical. In addition, testing is virtually automated from the programmer's standpoint. The benefits from the capability of reverse engineering legacy systems into this product are intuitively obvious.

Using these techniques, the software engineer is provided with a well defined set of products, including automatic reconciliation of product errors, as the life-cycle progresses from one phase to the next. As a result, the software life-cycle process is moving from a macro analysis perspective to a micro analysis perspective.

4.3 Further Research

Although extensive research has been conducted on the credibility of the convergent metrics, it has centered on the completed software product. Consider that most software products are incompletely specified and/or mispecified during the re-

quirements analysis phase. This results in additional work to complete the final product. While some of these errors may be corrected during the design phase, additional errors may arise resulting in the software product being incompletely designed or misdesigned with respect to the final product.

By taking this error refinement to its logical conclusion, errors of estimate between all life-cycle phases could be resolved with errors of estimate from subsequent phases. Consequently, more accurate equations, providing better estimates of LOC, probably exist. This scenario lends itself well to analysis with Markov chains.

If the effort and time required to complete each module can be estimated, then, intuitively, the DFD's could be translated into a Critical Path Method (CPM) or, possibly, a Program Evaluation and Review Techniques (PERT) network. Either method could provide a more detailed analysis of the system. These methodologies are based in operations research and employ various cost and scheduling analysis tools. Further analysis should determine a well defined methodology for implementing the translation.

A complexity metric could be determined from the DFD network based on a combination of the number of data elements and logic and control variables flowing into and out of the program versus the LOC. This metric influences the amount of effort required to complete the software product. Regression analysis could shed more light on the viability of this metric as compared to other complexity metrics.

The metrics and processes previously defined should be integrated into object-oriented methodologies wherever possible. While object-oriented methodologies provide certain distinct advantages over functions, many of the metrics are still immature. Further analysis and metric maturity are required before this process can take place efficiently.

Finally, the software engineering tool defined herein could be enhanced as an aid in any of the previously identified research. As a result, metrics could be harvested, and translations into other operations analysis tools could occur, automatically.

References

- [1] Boehm, B. *Software Engineering Economics*. Englewood Cliffs, New Jersey: Prentice-Hall, 1981.
- [2] Conte, S.; Dunsmore, H.; and Shen, V. *Software Engineering Metrics and Models*. Menlo Park, California: Benjamin/Cummings, 1986.
- [3] DeMarco, T. *Structured Analysis and System Specification*. New York: Yourdon Press, 1978.
- [4] Jensen, R. and Tonies, C. *Software Engineering*. Englewood Cliffs, New Jersey: Prentice-Hall, 1979.
- [5] Leonard, Ricky Jack. A Convergent Model for Predicting Software Programming Effort. *MSOR Thesis*, Department of Systems Engineering, University of Alabama in Huntsville, June 1991.
- [6] Page-Jones, Meilir. *The Practical Guide to Structured Systems Design*. New York: Yourdon Press, 1980.
- [7] Yourdon, E. *Managing the System Life Cycle: A Software Development Methodology Overview*. New York: Yourdon Press, 1982.

Qualitative Reasoning and a Circular Information Processing Algebra

Franjo Jović

Institute of Computer Engineering & Faculty of Electrical Engineering
University of J. J. Strossmayer in Osijek
HR-31000 Osijek, Istarska 3, Croatia

Keywords: qualitative modeling, circular algebra, quantitative/qualitative transformation, information, qualitative noise

Edited by: Anton P. Železnikar

Received: September 4, 1996

Revised: January 9, 1997

Accepted: February 6, 1997

Quantitative versus qualitative system modeling. Qualitative algebra. Events spaces and measure of their complexity. Ordinality and cardinality of models. Continuous Gödel numbering. Qualitative correlation algebra. Model sensitivity amplification by a circular quantitative/qualitative transformation. Qualitative noise.

1 Introduction

Any physical system can be described and later modeled based on both quantitative and qualitative aspect of its information. Thus measuring air temperature one can obtain only a partial indication of either a pleasant or harsh weather. Quantitative data on temperature, pressure, strain etc. describe a system as well as its comfort, efficiency or beauty. Qualitative description of a system or its end effect possesses necessarily an individual note and both the simplicity of its impact to the user. This has led to the canonization of using qualitative aspect of system behavior to the level of qualitative reasoning about systems as a very useful tool in system modeling.

The core issues of qualitative reasoning include qualitative and causal modeling of the systems, automated modeling and qualitative simulation ([1]). There has been a strong tendency toward a qualitative approach to solving engineering problems in the last twenty years ([2], [3], [4], [5], [6], [7]). Several different algebraic theories and innovative proposals of qualitative and both qualitative and quantitative calculations have been proposed by many authors ([4]).

The mainstream of development of the field as attributed to Benjamin Kuipers ([6]) is as follows:
a) There is always a time independent structure of a mechanism, or a plant, and a time dependent

behavior of the structure when involved with a process. The distinction between these two terms cannot be mapped clearly into a causal network but fits better with Johan de Kleers work on qualitative envisionment ([8]).

b) Qualitative structure and behavior can be most clearly understood and analyzed as abstractions of the ordinary differential equations and their solutions ([9]). It results in legitimate term qualitative differential equation or QDE where the variables in the equation are pure qualitative values given in the continuous manner enabling thus their differentiation.

c) Several steps in modeling a physical system and its behavior were enabled by the usage of models in the following order:

- physical scenario
- abstract elements of the scenario
- qualitative differential equation of abstract elements behavior
- quantitative behavior.

d) The sensory data from the plant and process were taken into account especially by Dennis DeCoste ([2]) where the problem of interpreting observations of a system over time is fundamental to intelligent reasoning about the physical world. The interpretation was viewed as the task of determining which possible behavior, predicted by

the current model, are consistent with the sensory data including which are the most plausible. There were many reports and papers on interpretations of relatively simplified systems based on sensory data and on its failures ([4], [7], [10], [11]).

e) The qualitative interpretation of dynamic across-time measurement has been made both with a bottom-up diagnostic concerns such as MIMIC model ([12]) and top-down such as DATMI approach ([2]). Here the integration of qualitative into quantitative observational data was viewed as clearly desirable for constraining the interpretation space.

f) A special algebra of combining qualitative interval data was developed ([3]) where temporal knowledge may take the form of collections of qualitative relations between intervals. Temporal reasoning tasks include determining a consistent scenario and deducing new relations from those that are known. Because of the exponential character of the problem solution, the so called Point Algebra has been introduced ([3], [13]).

Independent of the research mainstream the two classical approaches to the reasoning process were taken in the industrial application:

- reasoning process in the measurement control and supervisory systems and equipment ([14]) and
- reasoning process in the operators backed-up by the training and artificial intelligence equipment ([15]).

As observed from the historical point of view, the reasoning process in machines is influenced by the following mechanisms:

1. Machine structure follow-up of the process desired behavior and changing its behavior at every exception of the state as for example in PID mechanisms - so called state observers;
2. Machine learning according to some rules that are defined as mathematical model such as the system feedforward case;
3. Stochastic approximation of the desired system behavior such as a missing connection among measurable and unmeasurable variables;
4. Fuzzy set control in rule controlled processes ([16]);

5. Neural network used for pattern recognition and learning;

6. Genetic algorithm ([17]).

These simple reasoning processes are in the domain of machines for many production processes.

When viewed from the outside, the process and its structure (plant) can be observed as a composed dynamic interaction of qualitative and quantitative part of its variables. The behavior model is essential for system semantic. This exemplifies the so called inherent process semantic ([18]) with its four semantic parameters (actuality, reachability, relevance and importance) out of which the most valuable and unique is the so called importance parameter ([19]).

Permanent interaction of qualitative and quantitative aspect of process data, although unnoticed by user, technical and research staff, takes place in an effective way in many practical means and tools of the system equipment such as in ([20], [21]):

1. Neural network, by transformation of input quantitative data at each neuron of each network layer into qualitative reaction type at its output (firing of the neuron based on the exceeding threshold level after its summation point);
2. Fuzzy set logic, by transforming input quantitative data set into qualitative descriptive data on process behavior and later by transforming qualitative data set into the process specific quantitative reaction;
3. Genetic algorithms, where the transformation goes from qualitative (behavior of the observed set) into quantitative (codes of the artificial genes) and back to the qualitative (the behavior of the selected and mutated set).

The interaction of qualitative and quantitative part of the process information does not mean unifying or melting. It is the mutual action of qualitative and quantitative aspect of the information, a dialectical unity and by no means the algebraic one ([22]).

Thus the approach of unifying qualitative and quantitative aspect of the process information as presented in the research mainstream has shown

some defects. As the illustration of these defects let us point to the few lacks of the approach given in ([4]):

1. Model based reasoning is founded on the supposition of the known model which is a tautological approach because by the definition qualitative reasoning is a technique enabling qualitative modeling;
2. The SR1 qualitative/quantitative algebra's are defined on the same set of elements, with the qualitative domain being a subset of the quantitative domain, which is a mixture of different set domains;
3. Qualitative expressions are related by set equivalence, subset and nonempty intersection which is a sharp restriction of the possible relations among expressions;
4. Operations on multiplicands as allowed in SR1 are usually not allowable in "sound" qualitative operations.

The approach to a circular transformation of the qualitative/quantitative "field" data was originated by the problem of modeling elderly people injuring in road traffic accidents ([23]). The solution resulted in a qualitative stochastic difference equation, a model that enabled the prediction of the injuries for subsequent year with the ± 3000 PPM precision. The model precision enabled the discovery of some misreported data on the age of injured people in a few previous cases. Following to this was the model of the steam production unit of the thermal power plant ([24]) that resulted in a qualitative algebraic relation of seven out of 31 process variables. The critical part of the model was the indication of the model sensitivity to the burner subpressure being a small amount in the denominator of the basically qualitative model. The resulting annual repair showed that the misbehavior was properly indicated.

Next was a corn seed dryer modeled with a very small number of variables ([25]). The problem to the applied method might have been the small and slow changes of the variables because the temperatures of the dryer shouldn't increase above 42 degree centigrade constantly for about 100 hours of the drying procedure. During that period the drying abilities of the corn are going through three different types of behavior ([26]).

Constant or slow changing variables are very sensitive to any qualitative treatment. Nevertheless, the obtained model of input/output energy ratio in the starting and in the stationary part of the process has shown a very good level of determinism ([21], [25]).

Next application has been done for modeling a restricted food market in a so called tautological model – or a type of Pareto analysis ([27], [28]). The value of the market has been predicted with the precision of 0.93% of the realized sale taking into account that an idealized prediction of critical variables has been made for the defined time interval.

Some basic research supporting the obtained results has been reported as the proposed language of the procedure named Quacol (*qualitative correlation language*) ([29]).

The aim of the paper is to explain the Quacol algebra which is at the base of the qualitative/quantitative circular transformation method, and it will follow the steps:

- definition of the intended model space and its measure of communication
- continuous Gödel numbering in the ordinal model space
- presentation of the Quacol algebra
- modeling steps with the proposed algebra
- results and discussion.

2 Model events, spaces and measure of communication

Essential for this part is the communication aspect of the qualitative content of the information on process state and change of state.

A reasoning system comprises an on-line engine where the connection to the process and plant is realized, software for support of the reasoning process and a connection to the system evaluation operator. The usual form of such a system is given in *Fig. 1.* as an expert system environment.

Events that occur in the process or plant are fed to the system memory. They are later used as a source of producing a system model in the reasoning process. Events are simple when only one variable with one measurable parameter is

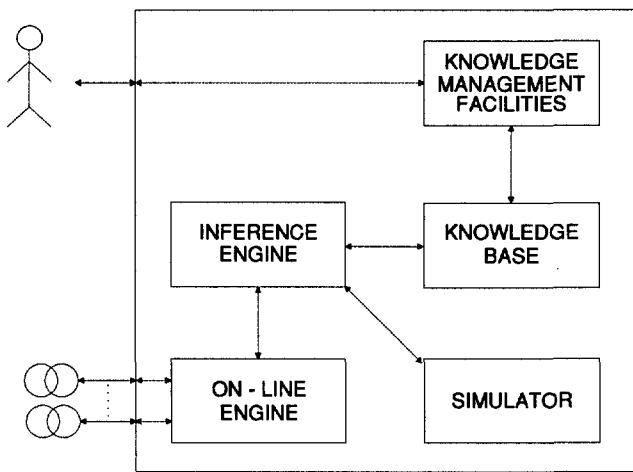


Figure 1: A complex reasoning system environment in real-time process control

fed to the system memory, or complex when a large set of variable measurements are fed at an instance. A series of complex event instances produces a space in the system memory. Events in the memory have to be somehow coded in order to correspond to actual and future system states if the reasoning system is to be used for prediction. A coding system will be proposed that reflects the process state in a given structure in the most efficient way. Essential to the coding system is the state of plant and type of process reaction involved.

Definition 2.1 A structure (or plant) state is ergodic if the change of plant states goes through well defined procedures from one to another structure state without being stuck and unable to recover.

Definition 2.2 A process reaction is dull if all process variables are measurable, known and predictable.

A dull process in an ergodic structure possesses the feature of monotonicity, i.e., the increase or decrease of any of its variable at any place changes in some way at other places other variables with the predictable increase, decrease or constancy depending on actual process reaction type and structure involved.

Definition 2.3 The information on process behavior and plant structure is nested in the surrounding if this surrounding enables the expression of its content ([22]).

Usually the nesting occurs in computer memory

of the reasoning system making it a nested system.

Definition 2.4 A nested system creates its position in the surrounding by action.

The surrounding of the process and plant is the reasoning system memory and the surrounding of the nested (reasoning) system are the process, plant and reasoning system itself. Thus the action of the reasoning system results either in the reinforcement value from the process + plant, indicating either the favorable position of the current state of the environment, or gaining the information on the current state of the reasoning system.

Definition 2.5 The origin of information can be used in the so called external information nesting. Process and plant are external origin of information for the reasoning system.

Definition 2.6 Using its own or reference information in the system is called internal information nesting.

When the external information source is used and modified and based on it a model made, then the transformation occurs from external into internal information nesting. However criteria which source to use for what purpose is on the side of system designer. Ultimately, the decision of external nesting (operator) have to be obeyed.

Definition 2.7 The extent of the information content consists of the squeezing of the set of possible information message set.

Theorem 2.1 ([30]) The extent of information nesting is measurable with the entropy decrease in the system when modeling a dull process and ergodic plant.

Proof of the Theorem 2.1: For a dull process and ergodic plant there is practically no information change obtained from the process events or variables in subsequent events. The plant behavior is completely described with the system model. Also, when the reasoning system is properly functioning there are no additional data on its behavior change.

Collecting external data from the process results, for a reasoning system in the process model, with the information parameters close to the expected model values. Any content of a process variable event $C(e_i)$ can be measured with its entropy being equal

$$C(e_i) = p(e_i) \log p(e_i), \quad (1)$$

where e_i is the i -th event and $p(e_i)$ is its probabil-

ity of its occurrence. When there is certainty in process and plant behavior then all event variable probabilities tend to have the value 1 and total entropy

$$C_{tot} < \sum C(e_i) \quad (2)$$

tends to zero; \sum stands for sum over all events. A properly functioning model shows a negligible small difference to the actual events behavior enabling thus its expression in entropy units (bits). Thus the extent of the information nesting or the value of the model can be measured in the entropy decrease in the reasoning system. \square

Theorem 2.2 *The amount of information nesting is inversely proportional to the entropy content regarded from the center of information nesting.*

Proof of the Theorem 2.2:

The content of process information needs to be changed during modeling procedure in order to be appropriate for the user and reasoning process as well. As each process or process part has its working point, however changeable during time course, the organization of process data needs to be done according to this working point(s). Thus the process information nesting can be organized around the working point(s). The behavior of the process model in subsequent event points can be regarded through its variables and their analytical relations such as given in the abstract form

$$\begin{aligned} \text{Input } (R_1, \dots, R_j) &\rightarrow \text{ corresponds} \\ &\rightarrow \text{ Output } (R_k, \dots, R_n), \end{aligned} \quad (3)$$

where R_i denotes analytical relation of either input or output variables. The amount of correspondence depends on the model quality. With the process in its working point and with the known analytical relation of the model as given in (3) any given input variable values will correspond to given output variable values. The importance parameter ([18]) of the semantic value of the model given in (3) will be equal to zero, meaning that in the working point there is no need to make any action because the process behavior is completely in accordance with the expected behavior. Thus the importance function can be traced for any process controlled by a reasoning system as a single valued function as depicted in *Figure 2*. The excursions of model variables and their effects given in relations R_1 to R_n are then summarized

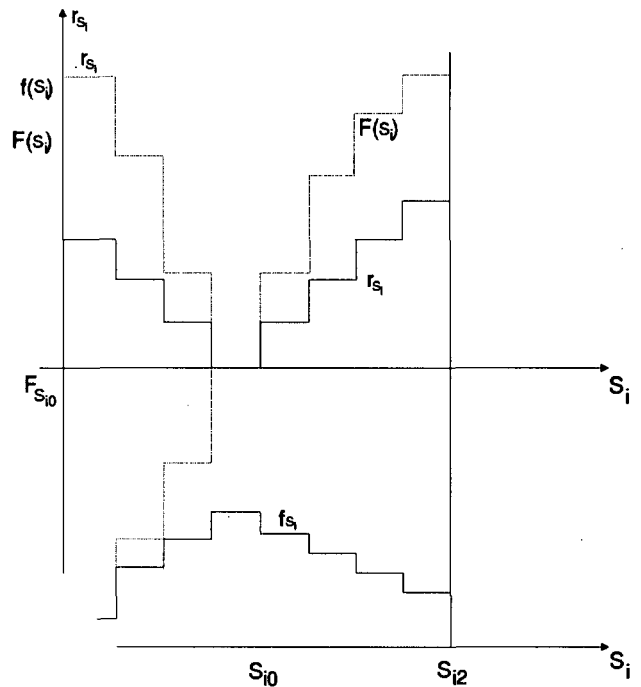


Figure 2: The importance parameter function in a supervisory control system

as variable space S_i with corresponding probability of excursion $F(S_i)$. The importance parameter r_{s_i} is then calculable from the probability density function $f(S_i)$ starting from the process working point.

Any change from the working point introduces entropy. As the information content of the importance function is gradually increased from the center of nesting so its nesting amount is decreasing as an inverse function of the importance function. \square

Theorem 2.3 *The communication of information content in case of information nesting change is more effective by means of a qualitative (pictographic) language.*

Proof of the Theorem 2.3:

The efficiency of any message can be measured by its coding length.

The change of nesting due to process changes results from the discrepancy between observed and modeled and results in the unexpected change of the working point. The change can be communicated in the quantitative manner by sending at least a percentage and direction of change. When using qualitative manner of communication then

only by sending the data on ranks of two consecutive importance parameter will suffice. Thus coding with qualitative data is more effective. \square

Any qualitative communication is a type of a pictographic language such as more/less, up/down, first/next, good/evil pairs of symbols.

3 Continuous Gödel numbering

Evaluation marks that are given to students are a simple example of qualitative or ordering variables. Many technical terms are expressed and used in qualitative form without even a notice of the user. Thus for example the notion of hardness in metallurgy is practically applied using the numbers of relative material hardness. The usage refers to the specific discrete continuous numbering system of graduate hardness of different materials which is only qualitative in its nature.

Thus qualitative approach to technical and scientific issues is sometimes subsumed without any “critical” revision or rigor. The basis for such state in case of hardness numbers is the natural behavior of materials that harder material cuts the softer one and naturally imposes the continuous discrete ordering in the system.

The aim of this part is to introduce a general procedure of continuous (discrete) numbering. Kurt Gödel introduced discrete set numbering in 1905 and it was later used extensively in computer science ([31]). By introducing a continuous (discrete) numbering the reduction of problem dimensionality can be obtained. On the other hand, such a method introduces naturally more nonlinearity and needs to be analyzed.

3.1 Ordinality and cardinality of models

As pointed earlier qualitative reasoning forms a substantial part of our everyday experience. There are two central properties of qualitative reasoning ([32]):

- (1) The variables under consideration are, at best, loosely specified in terms of inequality relations or ranges of values;
- (2) The system properties that interest us are not sensitive to changes in either the form of functional relations or the specific geometry's.

Thus in a group of potentially equal products one can find the most favorable without consideration of bigger geometric tolerances compared to the next favorable one.

Qualitative reasoning treats variables that are scaled ordinarily, or put into order, rank, hence they are defined only up to an arbitrary strict monotonic transformation. As a net result any ordinaly treated system observed and modeled in n events can take at maximum n different ranks of its qualitative behavior.

Cardinal or numeric values cannot be attributed to ordinal variables, but ordinal properties can be always attributed to cardinal variables. This points to the primality of ordinal values, nevertheless.

Ordinal variables are invariant under arbitrary positive monotonic transformations of the measurement scale as will be presented in relation (2) while cardinal relations are invariant only under affine transformations ([32]). Process models are usually describable at least by nonlinear differential equations. An example of such a model is the case of a three variable nonlinear system

$$\frac{dx}{dt} = f(x, y) \quad \frac{dy}{dt} = g(x, y). \quad (1)$$

The ratio of these two equations gives

$$\frac{dy}{dx} = \frac{g(x, y)}{f(x, y)}. \quad (2)$$

The equation (2) gives the system's path from any initial conditions. At any point in such a path equation (2) gives the path's slope. To study the qualitative properties of the phase portrait in detail one needs to define what is meant by qualitative properties in dynamics. Following the work by Andronov et al. ([33]) qualitative properties are those properties of paths, sets of paths and phase portraits that are topological invariants.

Topological invariant properties are properties preserved under arbitrary topological mappings of the region or set. A topological mapping is one-to-one or bicontinuous. Each point M maps to exactly one point M' of the same plane or set. Distinct points M_1 and M_2 map to distinct points M'_1 and M'_2 and any two arbitrary close points M_1 and M_2 map to arbitrary close points M''_1 and M''_2 . The inverse of topological mapping is also topological. A topological transformation can drastically change the shape of curves and regions but

a closed curve will remain closed and a straight line will generally become an arc that does not intersect itself.

The fundamental problem of the qualitative properties is the topological structure of the phase portrait. Singular paths, closed paths and separatrices are crucial in the determination of system behavior. Thus the indication of information on system paths is needed to ensure the qualitative investigation as complete as possible. These are:

- (1) The number and type of equilibrium states — this is often a nontrivial task ([33]);
- (2) The existence of closed paths, such as isolated limit cycles, or whole regions as in conservative systems. For limit cycles one needs to know the number, their relative position and their stability;
- (3) The behavior of the separatrices, especially when x and y are approaching infinity.

These properties are invariant under topological transformations of x and y . Reasoning about qualitative properties of global nonlinear differential equations have not yet been developed except for the case of stability invariance ([32]). Our approach is to invert the approach by constructing algebraic and difference equations based on qualitative behavior of process variables ([21], [23], [25], [34]).

3.2 Cardinality of a complex reasoning system

Although the ordinal number of a complex reasoning system can take only n or less qualitative ranks or values in n observing events, the cardinality of such a system is principally determined with the number of independent variables in the system number of events taken, and resolution of the measurement variables. The cardinal number of any system is thus a set of numbers attached to all its measured variables at any process event. The space of such a set is big and yet not satisfactory because lacking of system behavior estimates. Such estimates, presented in *Figure 1*, as an expert knowledge, require its inclusion in the reasoning system and increase its cardinality.

The inclusion of expert knowledge is done by means of a questionnaire to be answered by plant

and process experts and put into any knowledge representation scheme such as tables, databases or rules. Such expert data form a multivariable space of answers on plant and process estimated states.

When, for instance, ten groups of six questions with binary answers are elaborated, then an additional space of 2^{60} is to be added to the system cardinal number.

3.3 Continuous Gödel numbering

The notion of Gödel numbering is a rather well known method in algorithm theory ([31]). Having any program or data line written in a given language or a data code, an equivalent of ASCII binary code can be stated as well as its equivalent in base ten notation, or in any other base notation. Thus any program activity or a pertinent data set can be recognized in any program step as a numeric equivalent with different numeric outcome corresponding to some command code etc.

Gödel numbering is thus a discrete numbering system of computer commands and its outcomes.

When a similar numbering system is introduced that enables continuous discrete numbering of complex process and plant states then the possibility arises for an automatic ordering of otherwise discontinuous discrete space of process and plant states.

Let us suppose that the whole space of system states is defined with the reasoning system questionnaire, integrating the space of variables, events and resolutions of measurement into the questionnaire. With the number of plant and process state parts m each with k questions and each question having j different answers there is a space with q different answers

$$q = \prod_{l=1}^m l \prod_{i=1}^k j_i, \quad (3)$$

where \prod represent the product of m respectively k members, each member having k respectively j_i values.

In order to obtain the required continuous numbering, the answers have to be grouped into relevance categories. The answers to the questionnaire grouped into the same relevance category can be evaluated by process and plant experts into

broader and coarser groups having the following features of corresponding marks:

- (1) evaluation marks of process/plant states can be given in numerical form;
- (2) numerical classification of marks can be put into a continuous system preserving its original ordinality;
- (3) continuous evaluation although deliberately nonlinear is uniquely connected to the ordinality of the respective variable.

The only condition for the fulfillment of the points (2) and (3) is the change of the numbering system for each feature group according to the numbering base of the particular mark. Thus having two students, with all their marks equal except marks for two different subjects, can be put into a continuous Gödel numbering system by unequal number base of these two subjects.

Basic relation for such continuous transformation takes the form of

$$z = C(x_i^{a_i}), \quad i = 1, 2, 3, \dots, v, \quad (4)$$

where z is the total continuous Gödel number of particular evaluation questionnaire, a_i are numbers exceeding each feature group evaluation by one, x_i are marks given to each feature group, C is designation of positional connection of specific mark and v is the total number of feature groups.

Example: For the municipal thermal power plant the following daily report based on hourly data has been issued for process operators and plant technical people regarding the overall state of the plant, *Table 1* ([35]).

The plant response was coded using a continuous Gödel numbering from equation (4). Thus a series of data was obtained as given in *Table 2*. The coding was based on plant total efficiency and economy of fuel expressed in Gödel numbers, while the total Gödel number z was expressed as a transformation

$$z = (x \text{ on base } c)C(y \text{ on base } d), \quad (5)$$

where $c = 27$, $d = 4$ and $x = \text{efficiency}$, $y = \text{economy of fuel}$, *Table 2*. Total number of continuous states equals to decimal 108.

Theorem 3.1 *Any dull process is continuously Gödel enumerable.*

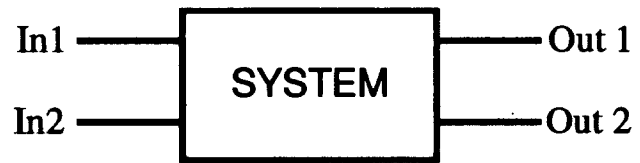


Figure 3: A hypothetical simple system with two inputs and two outputs

Proof of the Theorem 3.1: Any dull process does not produce any unmeasurable variable and any measurable variable can be put into a continuous Gödel numbering system by a simple ranking procedure. \square

4 The Quacol (qualitative correlation) algebra

Whether in the qualitative form or given in the continuous Gödel presentation any process variable taken in n events forms a single valued discrete function with maximum n distinct values. Such discrete single valued function will be named an n -point graph. The subject of this part is to present the usual way of using such graphs in reasoning systems i.e. for modeling purposes where a specific algebra arises connected with multiple transformation of such graphs from quantitative to qualitative content during modeling procedure.

4.1 Basic modeling procedure

Any system can be described in terms of its input and output variables. A simple system with two input and two output variables is presented in *Figure 3*.

Let us suppose that both input and output variables are measurable in all events. Thus four n -point graphs can be obtained when these measured values are taken and converted into qualitative form or put into a ranking procedure of any kind.

If interested into the system behavior, one would like to make any type of system model. The simplest way to start with would be to correlate qualitatively any combination of simple input and output n -point graphs. In our case there are four combinations of correlations: In 1 corr Out 1, In 1 corr Out 2, In 2 corr Out 1, and In 2 corr Out 2.

hour	1	2	3	...	22	23	24
Ambient temperature (°C)	3.03	2.62	2.27		8.56	8.12	8.7
Hot water load (MWh)	8.71	9.00	22.31		8.52	8.98	9.18
Vapor rate at plant level (ton/h)	23.73	22.63	23.05		26.63	27.22	27.58
Masoot rate (ton/h)	8.18	8.02	9.57		8.31	8.40	8.04
Gas rate (in 1000 normal m ³ /h)	0.15	0.15	0.15		0.14	0.14	0.14
Vapor rate at turbine input (ton/h)	51.58	51.08	74.00		49.56	48.61	49.94
Vapor rate at condenser input (ton/h)	50.39	49.32	43.29		49.27	47.45	44.44
Electrical energy at generator (MWh)	14.63	14.36	17.68		13.82	13.49	12.63
Total fuel energy (GJ/h)	297.7	291.8	347.4		302.1	305.3	292.5
Fuel energy used for electrical energy (GJ/h)	172.9	169.9	168.8		167.3	166.4	151.4

Table 1: Overall variables of the power plant taken from 31 plant variables (only the first and last three hours of the daily report data are given)

hour	1	2	3	4	5	23	24
Plant efficiency	0	0	8	25	26	1	3
Economy of fuel	0	0	1	3	3	0	0
Gödel number*	0A	0A	8B	25D	26D	1A	3A

Table 2: Coding of the plant global behavior in Gödel numbers; * the set of variable *y* is designated as A, B, C, and D

We can make an assumption that the second combination gives the highest correlation coefficient. Here stops the usual investigation of such a type. But when we analyze reasons why the second combination lacks in better correlation we may find that the In 1 *quantitative data added* to the In 2 *quantitative data* and converted to its new *n-point graph* exhibit a better *qualitative correlation* with Out 2 data.

Thus turning again to the quantitative aspect of process information and back to the qualitative evaluation of the behavior of such a treatment represents a circular way of system modeling by using such transformations.

Example:

A hypothetical case is presented with data in *Table 3-5*, for a system from *Figure 3*, and for the case of a very short three-point *n-graph*. Presented in *Table 3*, are four measured values from three events as well as corresponding ranked values of these measured values. Squared differences of all corresponding ranks and their sums are given in *Table 4*. The case of summing two input variables and comparing the sum with the output variable Out 2 is given in *Table 5*. As visible from *Table 3c*, there is a complete qualitative

event	measured values			ranked values		
	1	2	3	1	2	3
In 1	20	10	5	1	2	3
In 2	5	10	15	3	2	1
Out 1	10	10	10	2	2	2
Out 2	4	2	2	1	2.5	2.5

Table 3: Measured and ranked variables from three events of the system from Fig. 3

correlation between the added combination of inputs and the output variable Out 2.

4.2 More complex algebraic operations and their consequences on rank graphs

The subject of this part is to introduce some common features of algebraic operations on quantitative aspects of *n-graphs* and results of these operations. Such *n-graphs* are strictly defined on dynamic across-the-time series of measurable processes although a lot of freedom in relative proportions of time measure is acceptable. The time intervals between steps depend on the nature of

event	rank differences squared			sums of rank differences squared
	1	2	3	-
In 1/Out 1	1	0	1	2
In 1/Out 2	0	0.25	0.25	0.5
In 2/Out 1	1	0	1	2
In 2/Out 2	4	0.25	0.25	4.5

Table 4: Qualitative evaluation of different input/output variable combinations

event	sum of measured values			ranked values		
	1	2	3	1	2	3
(In 1 + In 2)	25	20	20	1	2.5	2.5
(In 1 + In 2) / Out 2	rank differences squared			sum of rank differences squared		
	0	0	0	0		

Table 5: The case of complete qualitative correlation of the sum of input variables *In 1* and *In 2* and the output variable *Out 2*. **Table 3-5:** A simple hypothetical case of complete qualitative correlation after an algebraic operation on input variables of the system given in Fig. 3.

the process and are outside the scope of the paper. The ranking of time independent data series is considered elsewhere ([36]).

Definition 4.1 An *n*-point graph is a single valued discrete function defined in *n* points and obtained from the ranking procedure of a process measured values or from ranking procedure of a Gödel numeration of more complex system behavior, where *n* > 1.

However processed the sum of all *n*-graph values is equal to

$$S = 1 + 2 + \dots + n = \frac{n(n+1)}{2} \quad (1)$$

and its mean value is $\frac{n+1}{2}$.

The *n*-graphs *i*, *j*, and *k* will be designated as *g_{in}*, *g_{jn}* and *g_{kn}*.

Theorem 4.1 The *n*-point graph *g_{in}* is transparent to the scaling procedure, i.e. any linear operation on its quantitative part with the constants *a* and *b*, does not change the graph or $a + bg_{in} = g_{in}$ for $a > 0, a < 0, b > 0,$

$$a = const 1, \quad b = const 2. \quad (2)$$

The proof is trivial since multiplying all values with a constant does not change the order of ranked values, as well as adding a positive or negative constant. The meaning of relation (2) is the invariance of Quacol algebra under positive metric transformation.

For $b < 0$ the *n*-point graph is mirrored around its mean value i.e.

$$a + bg_{in} = \frac{1}{g_{in}} \quad \text{for } a > 0, a < 0, b < 0,$$

$$a = const 1, \quad b = const 2. \quad (3)$$

Definition 4.2 An *n*-point graph *g_{in}* is strictly increasing (or decreasing) when the consecutive graph points are in the decreasing (or increasing) order.

Definition 4.3 An *n*-point graph *g_{in}* is a monotonically increasing or decreasing when some or all of its consecutive graph points are of the same rank in the increasing or decreasing order.

Graphs can be partially strict or monotonic when some of their parts exhibit the features given in Definitions 4.2 and 4.3.

Theorem 4.2 The addition operation on two (or more) *n*-point graphs yields a new *n*-point graph i.e.,

$$g_{in} + g_{jn} = g_{kn}. \quad (4)$$

When two monotonically increasing or decreasing graphs are added, a less monotonous graph is obtained because of independence of its monotonicity type.

Theorem 4.3 The addition operation on two strictly increasing *n*-point graphs *g_{in}* and *g_{jn}* gives only the same strictly increasing *n*-point graph, or

$$g_{in} + g_{jn} = g_{in} = g_{jn}. \quad (5)$$

Theorem 4.4 *The multiplication operation on two n-point graphs is commutative, i.e.,*

$$g_{in}g_{jn} = g_{jn}g_{in} \tag{6}$$

Because the multiplication takes part in their corresponding quantitative value pairs the commutativity property is unquestionable.

Theorem 4.5 *The inversion operation on the n-point graph g_{in} changes the graph into its mirror n-point graph g_{jn} symmetrical to g_{in} around the mean value. Particularly the monotonically increasing n-point graph g_{in} is converted into its monotonically decreasing counterpart g_{jn} , i.e.,*

$$g_{in} = \frac{1}{g_{jn}} \tag{7}$$

Proof: when any two quantitative values of the n-point graph are inverted then their ranking is inverted as well, maxima are converted to minima e.t.c. □

Theorem 4.6 *The division operation of any two n-point graphs g_{in} and g_{jn} is always possible even when some values of the quantitative part of the divisor graph g_{jn} possess zero values, i.e.,*

$$\frac{g_{in}}{g_{jn}} = g_{kn} \tag{8}$$

Proof: Due to *Theorem 4.1* the quantitative values of the n-point graph g_{jn} can be all set to nonzero values and then the division operation can be performed. Nevertheless the division operation exhibits highly nonlinear results in system modeling. □

Definition 4.4 *The similarity of two n-point graphs g_{in} and g_{jn} can be measured. The measure of similarity SM can be expressed as a function of the sum of rank differences squared among corresponding graphs, as given in Table 3. In this sense the similarity between two n-point graphs is proportional to the inverse value of the sum of rank differences squared, i.e.,*

$$SM = \frac{1}{\sum D_i^2} \tag{9}$$

where D_i is the difference between two ranks at point i of the n-point graph and \sum designates the sum above all n points. Usually the sum of rank differences squared or $(\sum D_i^2)$ can be used as the similarity measure.

Two n-point graphs can be evaluated for their similarity by using the *qualitative correlation coefficient* r ([37]) given with the equation (for all different ranks)

$$r = 1 - \frac{6 \sum D_i^2}{n(n^2 - 1)} \tag{10}$$

where n is the length of the n-point graph.

Definition 4.5 *The goal n-point graph is any n-point graph that is to be mimicked during the modeling procedure by a set of other n-point graphs using different algebraic and other mathematical relations among them in order to obtain as similar result as possible.*

Definition 4.6 *Any n-point graph possesses a certain amount of systematic pictographic content when its shape resembles to some simple symmetric or unsymmetric, odd or even line function such as constant, monotonically increasing or decreasing function, U or inverse U shape, J or inverse J, N or inverse N, M or inverse M (W) etc.*

Definition 4.7 *Two n-point graphs can possess nonsystematic pictographic content when by means of any type of algebraic operations on their quantitative contents they can be made more similar to the required goal n-point graph without using any n-point graph with a systematic pictographic content.*

The measure of their similarity can be expressed by (9) or (10).

Apart from or sometimes combined with algebraic operations other types of operations can be used such as differentiation, point shift, inversion etc.

Theorem 4.7 *Principal reason for the decrease of similarity of a given n-point graph to a given goal function lies in dispersion of otherwise expected monotonous ranks to other parts of the n-point modeling function by means of any algebraic operation.*

Proof: any two rank positions can differ in a very small amount of corresponding quantitative contents. Thus any algebraic operation can remove any rank to any point of the n-point modeling function. Thus caution should be put whenever the n-graphs are treated possessing small amounts of the corresponding quantitative content. □

Definition 4.8 *Qualitative noise is any unpredictable change of rank positions due to infinitesimally small changes of the corresponding quantitative variables.*

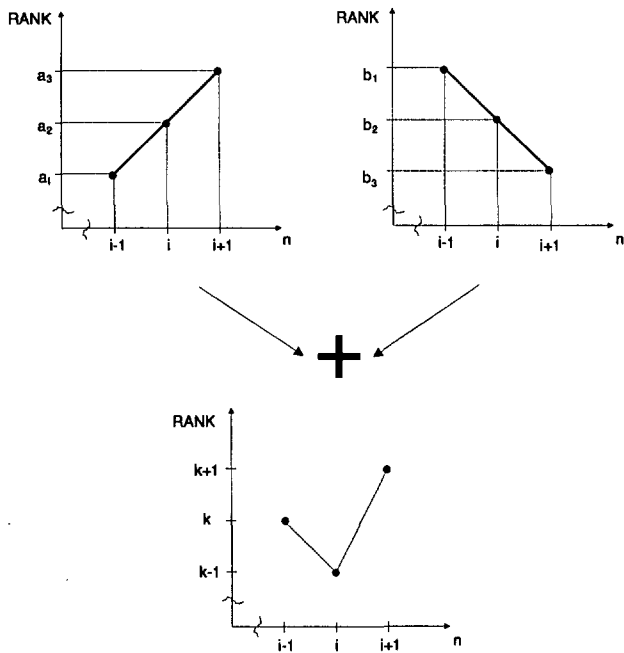


Figure 4: The example of qualitative noise calculation

Example:

Let us consider the example in *Figure 4*, where two graphs each with three points is presented. Let us suppose that the goal of addition of these two graphs is to obtain the (goal) n -point graph as given in *Figure 4c*.

The case can be described with the following set of consistent inequalities

$$\begin{aligned}
 aa_1 + bb_1 < aa_2 + bb_2 & \text{ or } a(a_1 - a_2) < b(b_2 - b_1) \\
 aa_3 + bb_3 < aa_2 + bb_2 & \text{ or } a(a_3 - a_2) < b(b_2 - b_3) \\
 aa_3 + bb_3 < aa_1 + bb_1 & \text{ or } a(a_3 - a_1) < b(b_1 - b_3)
 \end{aligned}$$

that can be reduced under less general but still rather strict conditions to

$$x - 1 < c, \quad 1 - y > c, \quad x - y > 2c, \quad (12)$$

where $x = a_1$, $1 = a_2$, $y = a_3$ are points on graph A from *Fig. 4a*, and $b_1 = 1$, $b_2 = 2$, $b_3 = 3$ are points on graph B from *Fig. 4b*, and $c = \frac{a}{b}$.

In the case of taking $c = \frac{1}{2}$ the limit of the solution is obtained meaning that even the infinitesimally small change of c above $\frac{1}{2}$ will induce the jump in the shape of the graph in *Fig. 4c*.

Definition 4.9 *The sensitivity of an n -point graph is equal to the amount of decimal numbers of the corresponding quantitative data of the graph.*

Theorem 4.8 *Any algebraic operation on an n -point graph increases the sensitivity of the obtained result.*

Proof:

The initial sensitivity of any n -point graph equals to the measurement precision of process variables. When two n -graphs of the same sensitivity are added then the result is the average increase of the sensitivity by a half decade. The multiplication operation increases the sensitivity by factor two or for measured variables by two to three decades etc. □

Theorem 4.9 *The limits of model building are determined with the qualitative noise.*

Proof:

The goal of model building can be simply stated as a series of comparisons of results of algebraic operations on input and output variable relations such as given in the general form in expression (3). When more and more complex relations are used for model building then the sensitivity of the obtained n -point graphs increases in such a way that the resultant quantitative data are so big that they finally induce the effect of qualitative noise due to stochasticity of used algebraic transformations ([38]). □

5 Modeling procedure with the Quacol algebra

The circular modeling procedure with the presented Quacol algebra is briefly given in *Figure 5*. It consists of the following parts:

- a) preparatory part, where process variables are selected, event samples taken and the conversion into n -point ranked graph is done (modules 1, 2, 3 in *Fig. 5*); the result of this modeling phase are relevant model variables put into the form of n -point graphs
- b) model selection part (module 4), where model goal function is defined either as a simple output n -point graph or similar input graph or as a complex algebraic relation of such input and/or output graphs
- c) determination of input/output model difference (module 5) with the choice of difference behavior in a systematic manner, as given in

Definition 4.6 (module 18) or in a nonsystematic manner, as given in *Definition 4.7* (module 12)

- d) the difference in nonsystematic manner leads to model compensation by selection and inclusion of a new n -point graph (module 6), and the difference in systematic manner leads to the selection of a new model analytic form (module 7) and both lead to the model scaling procedure in quantitative form, as given in expression (2) (module 8)
- e) model conversion into quantitative form and its testing for adequacy by using expression (9) after converting into qualitative form is performed in modules 9 and 16; the result of model adequacy leads to its conversion into quantitative form (module 10) and its application in the process automation (module 11); the resulting inadequacy leads to the procedure of trial to make a better system model
- f) the part of making a better system model can be concluded either by changing the scale of the model (module 13) or by dropping out the included new variable or a new analytic expression (modules 19 and 17) or by arranging a complete new way of input/output goal functions (module 14)
- g) the case of process behavior change with time is arranged in the modules 20 and 21 where after a definite time interval a completely new procedure of modeling takes place.

The application of this procedure was done for the case of the thermoelectric power plant where the modeling of input/output relation led to the final model as given in ([19])

$$(a_1v_1 + \frac{a_2v_2}{a_3 + a_4v_3}) - \text{corr} - (v_4 + \frac{a_5v_5v_6}{v_7}), \quad (1)$$

where the model variables are as follows:

- v_1 – feedwater flow
 - v_2 – masoot flow
 - v_3 – masoot temperature
 - v_4 – steam flow
 - v_5 – temperature of exhaust gases
 - v_6 – burner air flow
 - v_7 – burner pressure
- and a_1, \dots, a_5 are model constants.

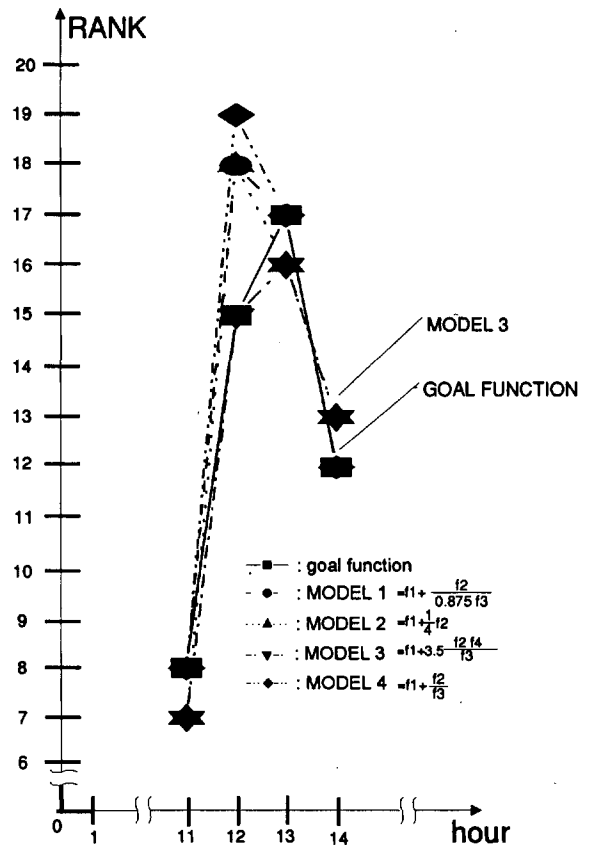


Figure 6: Modeling input/output relations in a thermoelectric power plant ([19])

The numeric value of the obtained model relations was 11 decades and the sum of ranks squared was 46. The length of the model was $n = 24$. The most critical part of the modeling procedure was model sensitivity to the slightest change of the analytical form that resulted in the drastically change of the corresponding ranks. The most critical were the ranks between the 11th and 14th point of the n -point model graph, as depicted in *Figure 6*.

6 Discussion

The circular nature of system modeling is described in points e), f) and g) of *part 5*. In its nature it is a repetitive procedure that after testing of model inadequacy leads to the change of its form in either inclusion of new variables or new analytic forms but always based on testing numerical differences of the qualitative n -point graphs.

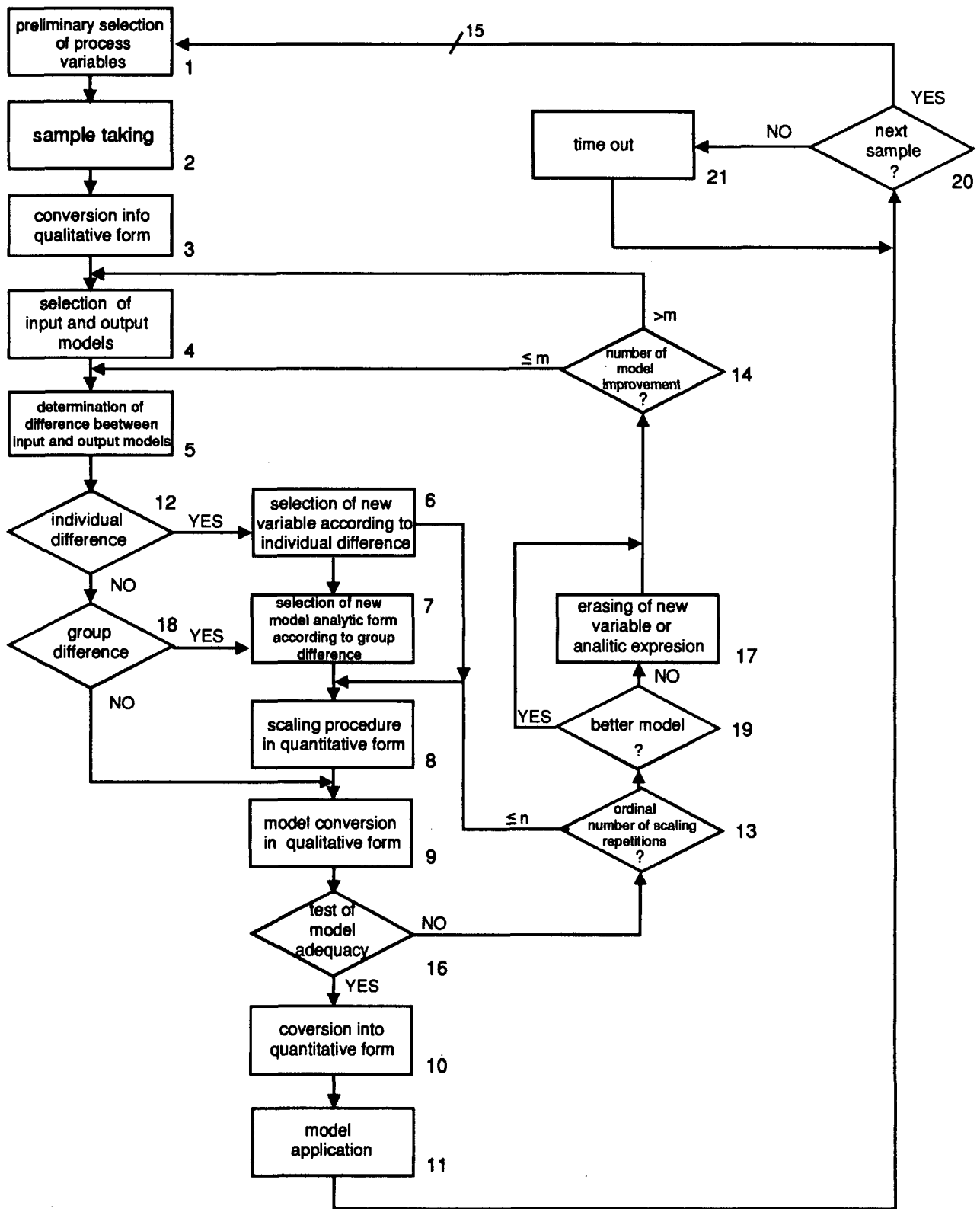


Figure 5: Circular information processing in nonlinear process modeling

Thus in its base it is a qualitative procedure, although it can include both qualitative and quantitative process description that after its conversion into qualitative rank data can be evenly treated. This makes the algebra feasible for a very wide application area especially in quality control operations and complex human related modeling. The engineering of such procedures is ensured by introduction of the continuous Gödel enumeration.

Such a qualitative procedure leads to models that are weaker in its determinacy degree as compared to the same obtained quantitative correspondent models by using quantitative correlation. However a slight improvement can be done toward a better determinacy of the model by using scaling feature of the method because of its invariance to the scaling procedure.

Essential for the procedure are two parallel processes that are exhibited through such modeling: a strong and measurable increase of model sensitivity and intrinsic existence of so called qualitative noise.

Nevertheless, all algebraic operations in Quacol algebra are not of the same strength. Thus addition increases very slightly the modeling sensitivity compared to the multiplication procedure. Both subtraction and division increase drastically the content of noise and thus limit the number of modeling steps. On the other side not all operations in Quacol algebra are preserving systematic features of the n -point graphs in the same way.

A special problem might present the length of the n -point graph. While for short and very short graphs, meaning graphs with less than 10 points, the exact correlation of the input/output model is obtainable, longer graphs are not prone to this feature, although a very high precision of models can be obtained.

A special field of features are hidden in operations of the Quacol algebra that are not exactly algebraic one such as inverting, differentiating and delaying operations. They can be also included into the algebra since the behavior of usual algebraic operation are extended in Quacol algebra. They extend the type of modeling possibilities toward nonlinear stochastic difference equations ([23]).

The division operation in Quacol algebra is completely permissible which puts this algebra into a very favorable position for model building.

The only almost completely unknown region of the investigation is the feasibility of mixing various quantitative and qualitative variables in obviously a very free manner. The response to this problem will be treated later although it has been already observed that processes tend to exhibit the feature of "gestalts" ([39]), i.e., their features are not expressible in a unique additive way.

Acknowledgments. The author is thankful to Prof. Lee J. White CWRU for suggestions and comments.

References

- [1] Nishida T., Tomiyama T. and Kiriyama T. : Eighth International Workshop on Qualitative Reasoning about Physical Systems, AI Magazine, Summer 1995,7-8.
- [2] DeCoste D.: Dynamic across-time measurement interpretation, *Artificial Intelligence* 51 (1991) 273-374.
- [3] Gerevini A. and Schubert L.: Efficient algorithms for qualitative reasoning about time, *Artificial Intelligence* 74 (1995) 207-248.
- [4] Williams B.C.: A theory of interactions: unifying qualitative and quantitative algebraic reasoning, *Artificial Intelligence* 51 (1991) 39-94.
- [5] van Beek P.: Reasoning about qualitative temporal information, *Artificial Intelligence* 58 (1992) 297-326.
- [6] Kuipers B.J.: Reasoning with qualitative models, *Artificial Intelligence* 59 (1993) 125-132.
- [7] Raiman O.: Order of magnitude reasoning, *Artificial Intelligence* 51 (1991) 11-38.
- [8] de Kleer J. and Brown J.S.: A qualitative physics based on confluences, *Artificial Intelligence* 24 (1984) 7-83.
- [9] Forbus K.D.: Qualitative process theory, *Artificial Intelligence* 24 (1984) 85-168.
- [10] Falkenhainer B. and Forbus K.D.: Compositional modeling: finding the right model for the job, *Artificial Intelligence* 51 (1991) 95-143. - 20 -

- [11] Skeirik R.D.: Modular neural network process control system with natural language configuration, International Patent Classification G06F 15/80. Int. Publ.No.: WO 92/02896. 1992.
- [12] Dvorak D. and Kuipers B.J.: Process monitoring and diagnosis: a model based approach, IEEE Expert 6 (3) (1991) 67-74.
- [13] Vilain M., Kauty H. and van Beek P.: Constraint propagation algorithms for temporal reasoning: a revised report in Reading in Qualitative Reasoning about Physical Systems (Morgan Kaufmann, San Mateo, CA, 1990) 373-381.
- [14] Shin K.G. and Cui X.: Design of a Knowledge Based Controller for Intelligent Control Systems, Proc of the 1990 American Control Conference, San Diego CA, 1990, 1461-1466.
- [15] Jović F.: Expert Systems in Process Control, Chapman and Hall, London 1992.
- [16] Isaka S. and Sebald A.V.: An Optimization Approach for Fuzzy Controller Design, Proc. of the 1990 American Control Conference, San Diego CA, 1990, 1485- 1491.
- [17] Goldberg D.E.: Genetic Algorithms in Search, Optimization and Machine Learning, Reading MA : Addison Wesley, 1988.
- [18] Jović F. and Zupanec R.: Entropy and semantics of process model variables (in Croatian), VI Symposium on Automatic Control Systems, Proc of the 35th Jurema, Zagreb 1990, 121-125.
- [19] Jović F.: Possibilities of entropy and semantic application in the estimation of process model variables, Elektrotehnika 35, No. 12, 1992, 45-54.
- [20] Jović F.: Causality function in expert production systems, 6th International DAAAM Symposium, Krakow, October 1995.
- [21] Jović F.: Process Modeling by Multiple Conversion of Nonlinear Analytic Expressions of Process Variables into Qualitative Form, Patent Claim HP950207A, Croatian Patent Office, Zagreb 1995.
- [22] Železnikar A.P.: On the way to information, Ljubljana 1990, Authors publication, ISBN 869012510.
- [23] Vorko-Jović A. and Jović F.: Macro Model Prediction of Elderly Peoples Injury and Death in Road Traffic Accidents in Croatia, Accid. Anal. and Prev. Vol.24 No.6, 1992, 667-672.
- [24] Jović F.: Information in Nested Expert Systems, Proc of the Croatian Systems Society, Artificial Intelligence in Measurement and Control, Zagreb 1992, 113-118, Editors: N. Bogunović and F. Jović.
- [25] Jović F.: Modeling Energy Ratios in Corn Seed Drier, XI Int. Conf. of Technologists for Drying and Storing, Stubičke toplice 1995. 191 - 196.
- [26] Petric P.: Technological Aspects of the Drying Process of Grainy Agricultural Products (in Croatian), Končar Stručne Informacije 1, 1989, 21-23.
- [27] Mc Graw Hill Dictionary of Modern Economics, 3rd edition, Mc Graw Hill Corp. New York 1983. p 337.
- [28] Jović F.: A Food Market Estimation from Characteristic Purchase Variables, to be published.
- [29] Jović F.: On Grammar of Quacol - Language for Process Modeling, Eurosime Congress '95, TU Vienna 1995, Poster Book, Poster No.49.
- [30] Jović F.: On Continuous Goedel Numbering In Expert Systems, IX Int. Wissenschaftliches Kolloquium Hochschule Bremen - Universitaet Osijek, Bremen 1993, 41-46.
- [31] Brookshear J.G.: Computer Science, The Benjamin/Cummings Publishing Company, Inc., Redwood City, CA 1991.
- [32] Kalagnanam J. et al.: The mathematical bases for qualitative reasoning, IEEE Expert, April 1991, 11-19.
- [33] Andronov A.A. et al.: Qualitative Theory of the Second Order Dynamic Systems, John Wiley and Sons, New York, 1973.

- [34] Jović F.: Drying Process as a Nonlinear Dynamic System, IX Int. Conf. of Technologists for Drying and Storing, Stubičke toplice, 1993, 123 - 129.
- [35] Anić-Ivičić S. et all.: A process control and supervisory system, Proc. of the MIPRO Conf. PU 51-55, Opatija-Rijeka 1988.
- [36] Jović F., Piližota V. and Ugarčić-Hardi Ž.: CIM in Food Industry - a Missing Link in Process Prediction, Int. Conf. on Computer Integrated Manufacturing, Zakopane, May 1996, 131-137.
- [37] Petz B.: Basic Statistical Methods (in Croatian) SNL Zagreb, 1985, p 192.
- [38] Kohonen T.: Content-Addressable Memories, Springer Verlag, Berlin, Heidelberg and New York, 1980.
- [39] Jović F.: Uber eine Gestalttheorie tautologischer Prozessenmodellierung, Presentation at the 25th Anniversary of the Polytechnic Albstadt - Sigmaringen, Albstadt, June 1996

Hierarchical Classification as an Aid to Browsing

J. Royce Rose and Caroline M. Eastman
 Department of Computer Science
 The University of South Carolina
 Columbia, South Carolina 29208
 USA

Keywords: unsupervised machine learning, hierarchical classification, browsing

Edited by: Rudi Murn

Received: July 12, 1996

Revised: January 8, 1997

Accepted: January 16, 1997

An approach to browsing large chemical reaction databases is presented. The method that is described builds on earlier work in which unsupervised hierarchical classification was used to extract generalizations of reaction classes from reaction databases for use in reaction knowledge bases. The method described in this paper involves classification based on both semantic and topological features. It supports the creation of deep hierarchies in which succeeding levels represent increasing degrees of abstraction. The creation of a hierarchy allows the user to quickly locate interesting items or classes of items by performing a tree traversal as opposed to sequentially scanning a hit list. In addition, the depth of the resulting hierarchy is determined interactively by the user.

1 Introduction

Browsing is a common information seeking activity and has been extensively studied [2]. Although browsing is not well defined, a variety of definitions have been proposed. What they all have in common is information seeking behavior that involves scanning a (possibly large) number of items looking for something of interest. The items are not restricted in nature; they may be books, grocery items, TV shows, or database records. Browsing is appropriate for searches involving some uncertainty about the goal of the search or about the way to achieve the goal (or both).

Several broad classes of database browsing requests can be identified:

1. Items related to X. X is a known or hypothetical item. If X is a known item, it might or might not be in the database. Items might be related to X because they are similar to X or for some other reason. In a chemical reaction database, a user might request reactions similar to a known reaction; similarity might be determined on the basis of the end product or the reaction conditions.
2. Items characterized by P. P is a set of properties. In a chemical reaction database, possible classes of properties include reaction conditions, i.e., temperature, solvent, catalyst, and pressure, topological changes such as ring closure/opening, and general mechanism such as the base catalyzed nucleophilic mechanism.
3. Items of interest. This is a vague and ill-specified request. However, people sometimes browse with exactly this kind of vague goal in mind. Such searches might be facilitated by knowledge discovery systems. Such a system might be used in a chemical reaction database to look for interesting and previously unidentified groups of reactions.
4. Kinds of items in the database. A user might be interested in finding out what kinds of information are in the database. This form of exploration is facilitated by a classification of the contents of the database. Although this could be done manually, an automatic classification is both more convenient and potentially more flexible. It makes it practical to create several classifications based on differ-

ent dimensions.

Imposing a hierarchical classification, in which succeeding levels represent increasing degrees of abstraction, on either the entire database or the results of a user query can be used to support these four broad classes of browsing requests. The creation of a hierarchical classification on a hit list allows the user to examine the hit list by performing a tree-traversal. This makes it possible to rapidly evaluate the contents of the hit list and quickly locate those items or sets of items the user is searching for or to determine that they are not present. Browsing by traversing such a hierarchy is equivalent to being able to query by similarity. We have chosen to evaluate this approach to browsing in large chemical reaction databases.

2 The Domain

Chemistry is the science that among other things deals with the transformations that substances undergo. Two key problems in this field are reaction prediction and synthesis design. Reaction prediction addresses the question of what chemical reaction or reactions will take place with a given starting material under particular conditions. In the case of synthesis design, the chemist has a target compound in mind. The question here is what should be used as starting material and what reaction or series of reactions should be used in order to transform the starting material into the desired target compound.

Reaction prediction and synthesis design require the chemist to have a very good understanding of the types of reactions that may possibly occur with a given set of materials and the influence that reaction conditions have. Where does the chemist get this information? Historically, chemists have learned about chemistry by reasoning from individual examples and by inducing generalizations from sets of related reactions. The chemist may be able to accurately predict the resulting transformation on a set of starting materials if these materials and the reaction conditions are similar to a known reaction. On the other hand, this prediction may also be made possible by an understanding of the underlying chemical processes. This deep understanding can be derived by generalizing from a set of related reactions.

Both inductive generalization and reasoning from individual examples are predicated on the chemist having access to an appropriate collection of reactions. For this reason, chemistry has always been a field in which databases have been compiled. Thus chemistry databases have existed long before the advent of the modern digital computer. In earlier times, these databases took the form of multi-volume compilations much like very large cookbooks. Today the field of chemistry is well supported by computerized databases [7,22]. These databases provide access to information about the scientific literature, chemistry hand books, patent information, business and industry data, chemical substance information, and reaction information. Textual, structural, and factual information is supported. In recent times, databases with more than one million reactions have been compiled [1]. Other reaction databases are growing by as much as 60,000 reactions per year [17].

3 The Problem

Chemistry is a field in which the amount of information available has consistently exceeded the capability of database technology. The explosive growth of reaction databases brings its own set of problems. One of the most pressing problems is not how the data is stored but how the user navigates through such a vast amount of information. This is usually not a problem if the database happens to contain the particular piece of information that the chemist is searching for. However, if this information is not contained in the database and the user must search for similar or related data, then current technology does not provide an adequate solution. Query methods that were adequate for reaction databases comprising tens of thousands of reactions are woefully inadequate when the database grows by one or two orders of magnitude.

An important aspect of the problem that users have with such databases relates to finding a good match between the generality/specificity of their queries and the contents of the database. An optimal match results in a hit list containing only that portion of the database the user is actually interested in. Even in very large reaction databases it may be the case that very little of the chem-

istry that the user is interested in is contained by the database. In this case, the user may have to start with a very general query in order to select the examples representing that chemistry. On the other end of the spectrum, the database may contain a rich complement of reactions, perhaps even the actual example the user is interested in. Here, the user will want to restrict the query to focus on the most relevant reaction or set of reactions.

Typically, the user scans the resulting hit list and then modifies the query in order to better target the relevant portion of the database. This may involve submitting a modified query to the entire database or just to the portion contained in the hit list. This type of query modification is both time consuming and wasteful of resources. One of the more tedious aspects occurs when the user must try to extract a summary of the hit list in order to decide how to modify the query. Quite often this is done by glancing at the first few entries and then modifying the query to exclude the kinds of entries in the hit list that the user does not find relevant.

This process of iterative query modification and hit list summarization results in an incomplete *ad hoc* hierarchical classification. Recognition of this fact leads us to propose hierarchical classification based on unsupervised learning as an efficient method for hit list processing in databases of organic reactions. However, since this problem is very general, we expect that many of the lessons learned will be applicable to other domains in which very large databases of complex objects are used.

4 Classification Methodology

The approach to hierarchical classification that we have taken is based on both semantic and topological features. It builds on the previous work of Rose and Gasteiger [19,20] which in turn was based on an earlier scheme that primarily considered topological features [14]. It supports the creation of deep hierarchies in which succeeding levels represent increasing degrees of abstraction. Our initial efforts have focussed on classifying the retrieved set (hit list) and not the entire database. We believe that providing the hit list with a hierarchical structure is more related to the needs of the user than would be reorganizing the entire

database. Another reason for not restructuring an entire database at the very beginning is that such an approach would demand extremely close cooperation with a database provider. However, we expect that the experience we gain from structuring hit lists will be valuable for later work involving entire databases.

4.1 The HORACE Algorithm

The HORACE hierarchical classification algorithm was developed for classifying and generalizing sets of chemical reactions. The primary motivation for this earlier work was the extraction of generalized reaction descriptions for use in chemical knowledge bases to support synthesis design and reaction prediction systems. Consequently, the hierarchies that are produced are created with the specific goal of producing reaction class descriptions with the degree of abstraction appropriate for a synthesis design or reaction prediction knowledge base. The resulting hierarchy is simply a means to an end. This algorithm for which a detailed description has already been published[19] is shown schematically in Figure 1.

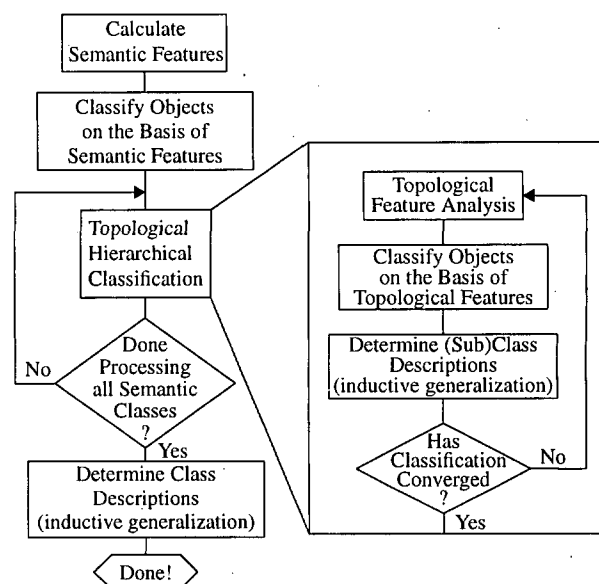


Figure 1: Hierarchical classification algorithm combining semantic and topological metrics.

The algorithm starts by calculating the semantics of the objects being classified. The methods for performing this characterization build on empirical methods developed by the EROS group

during the last 15 years [8, 9, 10, 11, 12, 15]. These are used to characterize the electronic and energy effects operative at the atoms and bonds of the reaction center. Classification at this level then is based on the comparison of corresponding atoms and bonds of the reaction centers of the reactions with respect to the dimensions defined by these parameters.

During the topological phase of hierarchical classification, the reactions are analyzed for topological features to support classification. HORACE uses a list of 114 features which are essentially chemical subgraphs recognized by chemists as functional groups. This set of 114 target features is stored in an external file which can easily be modified by adding or removing features. At this level, the classification of reactions involves the comparison of their complements of topological features. The precise details of HORACE's semantic and topological classification can be found in Rose and Gasteiger[19].

A hallmark of this approach to classification is the alternation between phases of classification and generalization and the way in which semantic and topological classification is combined. A key feature of this algorithm is the manner in which it combines structural and semantic classification approaches. It does not simply compose the two classification methods. Rather, it propagates constraints from the semantic phase of classification into the topological phase. This is done by first computing the semantic classification and then creating a topologically-based hierarchy on each of the resulting clusters (Figure 2). Since the topological algorithm is processing only reactions from one semantic cluster at a time, it cannot mistakenly combine reactions from separate semantic clusters that might appear to be topologically similar. The semantic features in the case of chemical reactions consist of descriptions of chemical structure in terms of electronic and energy parameters. These describe the meaning of the structure and make it possible to create chemically valid equivalence classes of reactions. The semantic classification is extended by alternating phases of topological classification and generalization of both semantic and topological descriptions. After the topological classification stabilizes, a final generalization based on the initial semantic classification is performed.

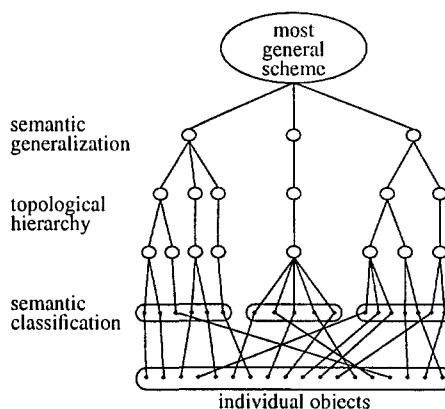


Figure 2: Stylized classification tree.

As can be seen in Figure 2, the topological hierarchical classification actually expands the classification tree in between the semantic classification and the final semantic generalization level. In a given hierarchy, each level represents a different degree of abstraction. The original objects being classified are at the lowest level. These items are then classified on the basis of similarity. The next layer consists of generalizations of the classes formed by classification of these items. Each level in the hierarchy is an abstraction of the level below it. The goal is to provide class summaries which are stored at the next highest level of abstraction in the hierarchy. The topmost item in the hierarchy summarizes all of the objects in the tree and is therefore the most general description.

4.2 The Modified HORACE Algorithm

In order to derive substantial benefit from giving hierarchical order to data, the resulting classification trees should strike a balance between depth and breadth. For this reason, one important goal in the design of the classification algorithm was to produce classification hierarchies expressing a large range of abstraction. This requirement motivated the design of a classification algorithm combining both phases of semantic and topological classification. A classification based on semantic features makes it possible to recognize similarity between objects that may be topologically dissimilar. On the other end of the spectrum, consideration of topological features makes it possible to refine a classification by extending it

in the direction of greater specificity in a manner that is intuitive to the chemist.

Notice that the hierarchy shown in Figure 2 is not particularly deep. Typically, HORACE hierarchies have the number of levels shown here. Occasionally, however, hierarchies that are shallower or deeper by one level are produced. This is a result of the data driven nature of the algorithm. If the reactions in a given semantic cluster are either topologically very similar or very dissimilar then only a single level of topological classification will be produced [20]. Clearly, such shallow hierarchies are inadequate for supporting the browsing of large numbers of reactions. Consider the case where the hit list contains several hundred reactions. A hierarchy of only a four or five levels lacks balance between breadth and depth. The resulting hierarchy would look more like a fat bush than a tree and would do little to reduce the information overload placed on the user.

The relative shallowness of the hierarchies produced by HORACE has been overcome by modifying the algorithm to increase the number of levels produced on the basis of semantic classification. This is done by varying the distance threshold which is used to determine cluster membership. The user supplies a starting threshold value and all intervening threshold values interactively so that a well-proportioned hierarchical classification tree, from the perspective of the user, results. Although the computed distances between reactions are normalized by the number of atoms and bonds in the reaction centers, selecting an appropriate threshold will depend on the nature of the reactions under consideration. If the reactions are quite similar, then a very low distance threshold will be required to split the clusters of one level into significantly smaller clusters in a deeper level. The threshold defines the upper distance limit allowable for a reaction to still be considered as matching the elements of a cluster. Lowering the threshold corresponds to requiring a closer degree of similarity. Consequently, the depth of the hierarchy is determined by the user interactively.

Once the size of a semantically based cluster drops below a user-specified size, it is no longer considered for further semantic classification. It is then automatically extended by consideration of topological features using the topological portion of the HORACE algorithm. Recall that each

internal node of the hierarchy contains a description which summarizes the subhierarchy which extends underneath it. Such summaries are particularly helpful in the case of topologically based clusters since the resulting descriptions highlight the structural similarity among the items comprising the subhierarchy.

5 A Browsing Example

Evaluation of the modified HORACE algorithm for supporting browsing is being carried out on a subset of the ChemInform-RX reaction database[17]. This set, containing approximately 115,000 reactions, corresponds to the reactions compiled in the database during 1991 and 1992. This data set is being accessed directly without going through a database system.

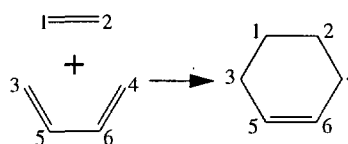


Figure 3: Diels-Alder reaction center.

The transformation shown in Figure 3, which chemists will recognize as the reaction center of the Diels-Alder reaction, was used as a query. The data set was then searched directly using a program written locally. This generated a list of 343 reactions to be treated as a hit list in a simulated reaction database query.

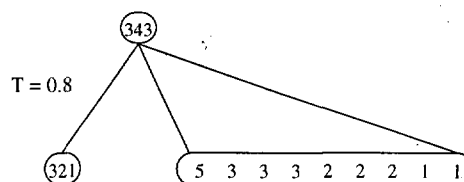


Figure 4: Level 1 of the hierarchical classification.

In Figure 4 we see the first browsing step with the creation of the first and highest level of the hierarchical classification. The user has selected a distance threshold of 0.8 which has partitioned the original 343 reactions into 10 clusters, of which the largest contains 321 reactions. The clusters at each level contain reactions that are mutually dissimilar to those of other clusters at

the same level with respect to the user specified distance threshold. Thus, the twenty-two reactions that are contained in the nine smaller clusters can be interpreted as those reactions most unlike the remaining 321 reactions in the single large cluster. Although a single oval-shaped node has been used to depict the nine smaller clusters in Figure 4 primarily in order to reduce clutter and to make the figure more readable, this depiction also conveys their dissimilarity from the large cluster of 321 reactions. The user that is interested in outliers need only examine these reactions without ever having to scan through the vast bulk that resulted from the simulated query.

As mentioned earlier, once the size of a cluster falls below a user-specified size, it is automatically processed by the topological portion of the HORACE algorithm. The other side of the coin is that these smaller clusters will no longer take part in the refinement of the hierarchy that is based on user selected thresholds. In this particular case, the nine smaller clusters are either so small or similar within a cluster that no further subhierarchy is created on the basis of topological features. However, each cluster is generalized to produce a description which summarizes the cluster content. Thus, the user may choose to look at the generalizations of such small clusters before deciding whether or not to look at the individual reactions. The cluster description shown in Figure 5 is for the cluster containing five reactions from Figure 4. In this figure, the label **R1** denotes the generalization of hydrogen and C_{sp^3} atoms.

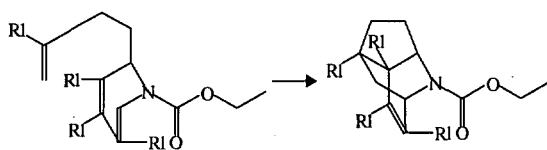


Figure 5: Generalization of the cluster containing five reactions in level 1.

The extension of the classification hierarchy that results from the user having selected a distance threshold of 0.55 followed by a threshold of 0.5 is shown in Figure 6. In the bottom-most level, two large clusters have been produced in addition to 16 smaller clusters. The 16 smaller clusters comprise only 44 of the 305 reactions on this level and in the case of clusters which are not sin-

gletons, the user may initially examine the generalized cluster description before deciding whether or not to look at the individual reactions.

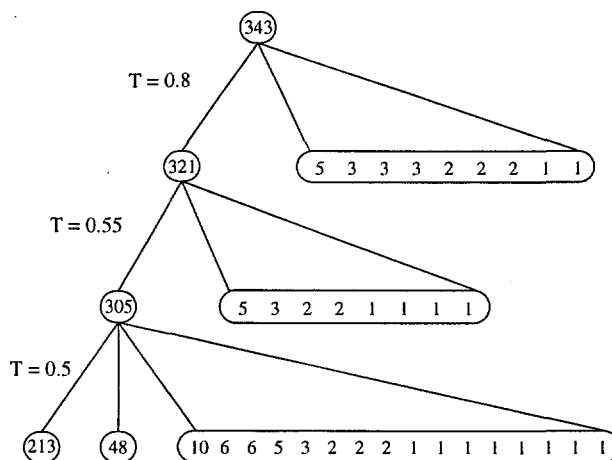


Figure 6: Levels 1-3 of the hierarchical classification.

Figure 7 shows the last level of semantically motivated hierarchical classification construction requested by the user. A threshold of 0.46 was specified by the user to create this level. In noting that the preceding level was created with a threshold of 0.47, we perceive that a critical boundary has been crossed that has resulted in the fragmentation of the cluster containing 202 reactions into 27 clusters, all of which are considerably smaller. It may be reasonable at this juncture for the user to re-specify the cluster size threshold that the system uses to determine when to automatically extend hierarchies with the creation of topologically motivated levels in order to further process the larger remaining clusters. Doing so would, for example, extend the hierarchy rooted at the larger cluster of 59 reactions by the topological-based hierarchy shown in Figure 8. We see that within the span of five user-selected levels the initial monolithic hit list has been systematically reduced to clusters that a user would find much more manageable than the imposing initial set of 343 reactions.

6 Related Work

Clustering has been extensively studied across a wide variety of disciplines, and a large number of clustering algorithms have been developed. Many

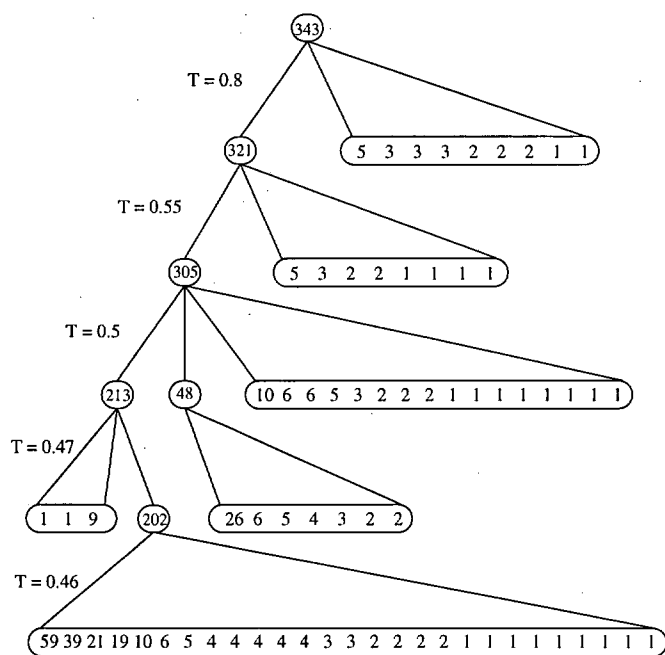


Figure 7: Levels 1-5 of the hierarchical classification.

of these algorithms create hierarchies of clusters, but this is almost always done by splitting or joining existing clusters by varying a cutoff parameter. The algorithm used here is distinctive both in its use of semantic and syntactic information during different phases of the creation of the hierarchy and its use of different, i.e., successively more abstract, information in the creation of each major level in the topological phase. Cutting, Karger, and Pedersen [4] describe a hierarchical approach used in document retrieval applications. However, their clustering algorithm is statistical rather than semantic; it is based on the computation of keyword vector similarity.

Clustering is only one of the techniques that has been used in analyzing and managing chemical information. A complete survey of all of the different approaches proposed or implemented is not feasible here. An overview of storage and processing of chemical structure information is provided by Lipscomb, Lynch, and Willet [16]. They address problems in representation, indexing, and searching in both structure and reaction databases, including similarity based matching and clustering.

7 Future Work

The browsing system described in this paper bases its classification purely on topological and physicochemical attributes. The set of 114 topological features used for classification was derived from a collection of functional group structures used by the SYNCHEM synthesis design system [13]. The structures in this subset have not been rigorously evaluated for their appropriateness as classification features. It is expected that some of them could be discarded without negatively affecting classification accuracy. At present, only the physicochemical features sigma and pi electronegativity along with resonance stabilization parameters are used. Additional physicochemical attributes must be evaluated for their classification utility. One area in which the current system is completely lacking is in the use of reaction conditions as classification criteria. Although reaction conditions by themselves can not support the fine degree of classification possible with topological and physicochemical attributes, they are important and must be taken into con-

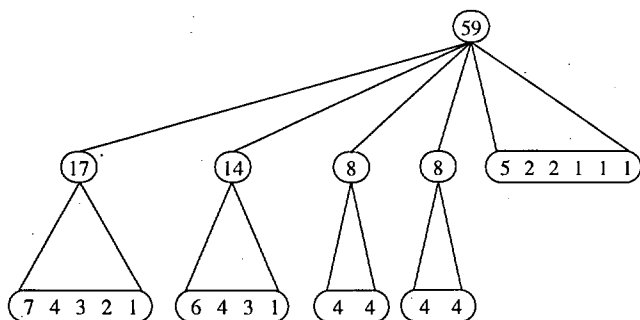


Figure 8: Topological-based hierarchical classification of the level 5 cluster of 59 reactions.

sideration. Additionally, stereo-chemistry is not presently taken into account.

8 Conclusion

Hierarchical restructuring allows the user to quickly evaluate the results of a query and to locate interesting items and classes of items. This is accomplished by performing a tree traversal rather than a sequential perusal of a hit list or a series of *ad hoc* query refinements that is normally required for nonhierarchical approaches. More general classes may be examined by moving up the hierarchy. Conversely, more specific classes may be examined by moving down the hierarchy. In contrast, sibling nodes in the hierarchy represent related classes of approximately the same degree of abstraction. In very large databases where classical querying methods are increasingly inadequate such as chemical reaction databases, such a browsing method is required in order to manage the flood of information with which the user is confronted.

There is a long history of interest in intelligent systems to facilitate chemical information processing beginning in the late 1960's. Much of this work has focussed on the development of knowledge-based systems for reaction prediction and synthesis design [3, 5, 6, 13, 18, 21]. The problems of synthesis design and reaction prediction are much more difficult than was thought when research in this field began. Consequently, intelligent systems developed to address these problems have met with limited success. This has been due in large part to the difficulty experienced in compiling adequate knowledge bases. The research that we propose could be adapted to assist in the compilation of chemical reaction knowledge bases since it is essentially a data-mining tool.

Acknowledgments

The authors would like to thank Prof. Johann Gasteiger for making the empirical methods developed by the EROS group for calculating physicochemical parameters available to us. We would also like to thank Dr. René DePlanque of Fachinformationszentrum Chemie, Berlin, for providing us with the reaction data set used in evaluating the hierarchical classification algorithm.

References

- [1] J. E. Blake and R. C. Dana, *CASREACT: More than a Million Reactions*. ACS Jour. Chemical Information and Computer Science 1990, 30, 394-399.
- [2] S. Chang and D. E. Rice, *Browsing: a multidimensional framework*. Annual Review of Information Science and Technology 1993, 28, 231-276.
- [3] E. J. Corey and W. T. Wipke, *Computer-assisted Design of Complex Organic Synthesis*. Science 1969, 166, 178-192.
- [4] D. R. Cutting, D. R. Karger, and J. O. Pedersen, *Constant interaction-time scatter/browsing of very large document collections*. In proceedings of the Sixteenth Annual International ACM SIG.R Conference on Research and Development in Information Retrieval. June 27-July 1, 1993, Pittsburgh, PA. pp126-134.
- [5] P. A. D. De Maine and M. M. De Maine, *Computer aids for chemists*. Analytica Chimica Acta 1990, 235, 7-26.
- [6] J. Dugundji and I. Ugi, *An algebraic model of constitutional chemistry as a basis for chemical computer programs*. Top. Curr. Chem. 1973, 39, 19-64.
- [7] J. Garnett and V. Calderhead, *Chemistry. In manual of Online Search Strategies, 2nd Edition*. C. J. Armstrong and J. A. Large, eds. G. K. Hall & Co. New York 1992 pp128-157.
- [8] J. Gasteiger and M. G. Hutchings, *Quantification of Effective Polarisability. Applications to Studies of X-Ray Photoelectron Spectroscopy and Alkylamine Protonation*. J. Chem. Soc. Perkin, 1984, 2, 559-564.
- [9] J. Gasteiger, M. G. Hutchings, B. Christoph, L. Gann, C. Hiller, P. Löw, M. Marsili, M. Saller and K. Yuki, *A New Treatment of Chemical Reactivity: Development of EROS, an Expert System for Reaction Prediction and Synthesis Design*. Topics Cur. Chem. 1987, 137, 19-73.

- [10] J. Gasteiger and M. Marsili, *Iterative Partial Equalization of Orbital Electronegativity - A Rapid Access to Atomic Charges*. Tetrahedron, 1980, 36, 3219-3228.
- [11] J. Gasteiger, M. Marsili, M. G. Hutchings, H. Saller, P. Löw, P. Röse and K. Rafeiner, *Models for the Representation of Knowledge about Chemical Reactions*. ACS Jour. Chemical Information and Computer Science 1990, 30, 467-476.
- [12] J. Gasteiger and H. Saller, *Calculation of the Charge Distribution in Conjugated Systems by a Quantification of the Resonance Concept*. Angew. Chem. 1985, 97, 699-701. Angew. Chem. Ed. Engl. 1985, 24, 687-689
- [13] H. Gelernter, G. A. Miller, D. L. Larsen and D. J. Berndt, *Realization of a large expert problem-solving system: SYNCHEM2, a case study*. IEEE 1984 Proceedings of the First Conference on Artificial Intelligence Applications. IEEE Computer Society Press: Silver Spring, MD, 1984.
- [14] H. Gelernter, J. R. Rose and C. Chen, *Building and Refining a Knowledge Base for Synthetic Organic Chemistry via the Methodology of Inductive and Deductive Machine Learning*. ACS Jour. Chemical Information and Computer Science, (30):492-504, 1990.
- [15] M. G. Hutchings and J. Gasteiger, *Residual Electronegativity - An Empirical Quantification of Polar Influences and its Application to the Proton Affinity of Amines*. Tetrahedron Lett. 1983, 24, 2541-2544.
- [16] K. J. Lipscomb, M. Lynch and P. Willet, *Chemical structure processing*. Annual Review of Information Science and Technology 1989, 24, 189-238.
- [17] A. Parlow, C. Weiske and J. Gasteiger, *ChemInform - An Integrated Information System on Chemical Reactions*. ACS Jour. Chemical Information and Computer Science 1990, 30, 400-402.
- [18] P. Röse and J. Gasteiger, *EROS 6.0, a Knowledge Based System for Reaction Prediction - Application to the Regioselectivity of the Diels-Alder Reaction*. In Software-Development in Chemistry. Editor: J. Gasteiger. Springer-Verlag, Heidelberg, 1990.
- [19] J. R. Rose and J. Gasteiger, *HORACE: An Automatic System for the Hierarchical Classification of Chemical Reactions*. ACS Jour. Chemical Information and Computer Science, 1994, 34, 74-90.
- [20] J. R. Rose and J. Gasteiger, *Hierarchical Classification as an Aid to Database and Hit-List Browsing*. Third International Conference on Information and Knowledge Management Conference Proceedings, Gaithersburg, Maryland, pp408-414, 1994.
- [21] T. D. Salatin and W. L. Jorgensen, *Computer-assisted mechanistic evaluation of organic reactions*. Journal of Organic Chemistry, 45 (11), 1980.
- [22] H. Schulz and U. Georgy, *From CA to CAS online*. In Databases in Chemistry, 2nd Edition. Springer Verlag, Berlin 1994.

Towards Recursive Models—A Computational Formalism for the Semantics of Temporal Presuppositions and Counterfactuals in Natural Language

Stefano Mizzaro

Department of Mathematics and Computer Science

University of Udine

Via delle Scienze, 206, Loc. Rizzi, 33100 Udine, Italy

Tel: +39 (432) 55.8456, Fax: +39 (432) 55.8499

E-mail: mizzaro@dimi.uniud.it

WWW: <http://dimi.uniud.it/~mizzaro>

Keywords: artificial intelligence, computational linguistics, natural language processing, semantics, pragmatics, temporal presuppositions, counterfactuals, linguistic knowledge, extra-linguistic knowledge, ontology, content, computational models, recursive models, TOBI

Edited by: Vladimir Fomichov

Received: October 10, 1996

Revised: February 24, 1997

Accepted: March 18, 1997

The linguistic phenomena of temporal presuppositions and counterfactuals, situated on the boundary line between semantics and pragmatics, are common to many languages, and the computational treatment of such phenomena is difficult because of their non-monotonic aspect.

These phenomena are presented through a corpus of examples; they are studied emphasizing the various types of knowledge underlying them; and the fragment of language that encloses such phenomena is defined in a way not dependent from a specific language. Then, Recursive Models, a formalism for modeling the semantics of utterances containing temporal presuppositions and counterfactuals, are proposed, described from both functional (by formal specifications) and structural points of view, and compared with related work. Finally, the adequacy of Recursive Models is empirically verified: TOBI (Temporal presuppositions and counterfactuals: an Ontological Based Interpreter), a system that interacts with the user in natural language using the recursive models, is illustrated. TOBI is not based on a deductive system, but uses the more primitive and flexible notion of model-based evaluation; its architecture, flow of control and internal data structures are presented.

1 Introduction

This paper¹ sketches a formalism, named *recursive models*, that can be used for representing, at a semantic-pragmatic level, utterances containing temporal presuppositions and counterfactuals. The power of this formalism is tested by using it in a natural language processing system named TOBI (Temporal presuppositions and counterfactuals: an Ontological Based Interpreter).

The paper is structured in the following way.

In Section 2 the linguistic phenomena of temporal presuppositions and counterfactuals are presented and analyzed, and the fragment of natural language relevant for such phenomena is formally defined. Section 3 presents recursive models, the data structures used for modeling the semantics of natural language utterances; such presentation is given from a functional-formal, a structural, and a behavioral point of view. Furthermore, a survey of related work is proposed. Section 4 describes the TOBI system, illustrating its architecture, flow of control, and internal data structures.

¹This work is a revised and extended version of [33; 34].

Section 5 summarizes the work done so far and proposes some future extensions.

2 The language fragment

The linguistic phenomena considered in this work are situated on the boundary between semantics and pragmatics. In the following three subsections: (i) a corpus of examples that informally describe such phenomena is presented; (ii) the examples are analyzed with respect to a classification of various kinds of knowledge; and (iii) a formal (syntactical) definition of the fragment of the language studied is presented.

2.1 The linguistic phenomena

To completely understand the meaning of each utterance, it is important to analyze its relations with the other utterances in the discourse. Following Gazdar [18], an utterance *implies* another utterance if the latter is a consequence of the former (here I give no formal definition of implication). For example, utterance (1)

“Mary met John before she left” (1)

(utterances are enclosed in double quotes) implies utterances (2) and (3):

“Mary met John” (2)

“Mary left.” (3)

A particular case of implication between utterances is *entailment*: utterance (1) entails (2). However, entailment is not the only type of implication, as utterance (3) proves: the relation between (1) and (3) is not an entailment, as shown by the fact that the utterance

“Mary met John before she left and he persuaded her to stay at home” (4)

is consistent. If we admit that (3) is entailed by (1), then (3) is also entailed by (4). But (4) entails

“Mary did not leave”

which contradicts (3). Utterance (3) is a (*temporal*) *presupposition* of (1) [18, 23, 24, 27, 29, 30]. A presupposition is a form of implication weaker

than entailment: the second part of (4), *asserting* that Mary stayed at home, *cancel*s the presupposition, so we do not have a contradictory utterance.

It is important to remark that although the event ‘Mary left’ did not happen, it is used in (4) to date the event ‘Mary met John’. Moreover, from a logical point of view it seems more correct to say

“Mary met John before she *did not* leave and he persuaded her to stay at home” (5)

instead of (3), but no human would do so. In other words, the problem is in *nonmonotonicity*: utterance (1) implies (3) only *by default* and the second part of (4) deletes the default. Then, entailment can be seen as a *certain* inference, while presupposition as a default (and so *uncertain*) one.² Therefore, a system handling such phenomena must be nonmonotonic. The most widely used formalism for this purpose is represented by nonmonotonic logics [11, 21]; however, the approach followed in this work is different, as will be shown later.

It has to be noted that ‘after’ is not the symmetric counterpart of ‘before’, as shown by the fact that in the utterance

“Mary met John after she left”

the leaving event cannot be deleted, as done in utterance (4): the utterance

“Mary met John after she left. She did not leave”

is clearly inconsistent.

Furthermore, relationships between events are necessary for example to explain the utterance

“Mary left before meeting John”, (6)

² Note that there are two different views of temporal presuppositions (and of presuppositions in general). On the one side (what might be called an *a priori* view) they are necessary for giving a truth value to the whole sentence: the name ‘presupposition’ comes from here. On the other side (*a posteriori* view) they leave a trace as a defeasible inference: as it was pointed out before, temporal presuppositions can be seen as a kind of implication weaker than entailment. Here I am interested in the latter aspect; the former is analyzed in a lot of works [23, 24, 29].

in which the meeting event is presupposed, but it is immediately deleted on the basis of *world knowledge*; so the leaving event prevents the meeting.

Another linguistic phenomenon strictly related with the previous ones is that of (*conditional counterfactuals*). In fact, (4) implies:

“If Mary had not met John, she would have left”, (7)

that is used for referring to an hypothetical course of events, or a *non-real world* (the world in which Mary did not meet John). It is important to observe the (perhaps unexpected) fact that the meaning of an utterance as “ α before β ” is sometimes more similar to an utterance of type “if not α' , β' ” (where α' stays for the subjunctive form of α and β' for the conditional form of β) than to an utterance of kind “ β after α ”.

Two other related linguistic phenomena have been considered. The first one is exemplified by

“The bullet deviated before hitting Mary. Nevertheless it hit her.” (8)

What happens in this utterance can be explained in the following way:

- analogously to utterance (6), it is presupposed that the bullet hit the target, but such presupposition is immediately deleted on the basis of world knowledge: human beings know that if a thing is deviated from its trajectory, usually it does not hit the original target;
- in the second part of the utterance it is asserted that the bullet hit the target anyway. To do this, it is not correct to use the conjunction ‘and’ as done in (4) to cancel a presupposition. A more powerful way, the use of the conjunction ‘nevertheless’, is needed. The reason is that what has to be deleted in this case (the non-occurrence of the non-hitting, derived from world knowledge considerations) is something ‘stronger’ than the temporal presupposition of (4).

The second phenomenon is shown by the following utterance, implied by (8):

“Even if the bullet had not deviated, it would have hit Mary.” (9)

Such ‘even if’ utterances (that I shall call *weak counterfactuals*) play, concerning ‘nevertheless’ utterances (i.e. utterances like (8) above), the same role that usual counterfactuals have in the case of ‘and’ sentences. That is, utterance (9) is for (8) what (7) is for (4).

The standard treatment of temporal presuppositions [18, 23, 24, 27, 29, 30] is not entirely satisfactory: there is no deep explanation of why ‘before’ should introduce a presupposition, while ‘after’ should introduce an entailment. The point is that an *ontology of time* is not taken into account: time is ordered and the future unknown and partially unpredictable, and these facts must be taken into account when dealing with utterances containing ‘before’ and ‘after’. In this way, no *linguistic* explanation of why an event introduced by ‘before’ can be deleted and one introduced by ‘after’ cannot is required. Linguistically, one can—and ought—only say that secondary sentences started by ‘after’ and ‘before’ introduce a presupposition that can be deleted later. The explanation of the asymmetry between ‘before’ and ‘after’ must be found at a deeper level, in the way we, human beings, perceive and treat the time. This point is investigated in the next section.

2.2 Linguistic and extra-linguistic knowledge

It is common usage [29] to divide the knowledge utilized for making inferences about an utterance into two classes: *linguistic knowledge* (LK) and *world knowledge* (WK). In this section I propose a more subtle distinction, that will be useful for both understanding and treating the linguistic phenomena at hand.

First, it is possible to distinguish between LK and *extra-linguistic knowledge* (ELK). LK is used for deriving facts from an utterance through pure linguistic rules.³ For instance: a proper noun

³Let us note that ‘knowledge’ and ‘inference’ can be defined from the standpoint of mathematical logic. A formal calculus [16] is made of axioms (that represent known facts about a domain) and inference rules (that model the inference process). Starting from the axioms, and using inference rules, one can derive (inference) other facts. The axioms may be divided into groups corresponding to different kinds of knowledge involved. In the same way, also the inference rules may be grouped. Inferences and derived facts can be classified according to the kind of the axioms

stands for an individual; if a noun phrase is plural, then it denotes more than one individual; if the tense of a verb is 'simple past' ('future'), then the event that it denotes happened in the past (will happen in the future); if an event is described in the main (secondary) proposition, then it is entailed (presupposed); and so on. ELK inferences instead are not directly derived from the utterance through linguistic considerations, but from other knowledge sources (i.e. from the world as we know it): a human proper noun like 'Mary' usually denotes a female human being; if someone is dead, he cannot do anything; if an event happened in the past, it cannot be modified; if an event is expected to happen in the future, it may or may not happen; and so on. The distinction between LK and ELK is not so clear-cut, being sometimes difficult (or arbitrary) to classify an axiom or an inference. Anyway it is interesting to study how far it is possible to push this dichotomy.

Second, both the LK and ELK inferences and derived facts can be *uncertain* or *certain*. The uncertain LK inferences were called in the previous section 'presuppositions', the certain ones 'entailments'. Another kind of uncertain LK inferences are *implicatures* [30]. ELK inferences are often uncertain (the 'real' world is very difficult to model: the research on WK, or *common sense* [14, 25] is one of the main subfields of artificial intelligence): 'Mary' usually denotes a female human being, but it might denote a hurricane, or a boat, or something else; if a bullet is deviated, usually it does not hit the target, but sometimes this could happen anyway; and so on. But ELK inferences can also be certain: if an event happened in the past it cannot be modified; if an event is said to happen in the future, it might happen or not happen; and so on. In the following I will call *ontology* the certain ELK and *content* the uncertain ELK. Informally speaking, ontology is the component of knowledge that has a general logical status; on the contrary, content is the component of knowledge that is highly situation dependent.

Let us consider a concrete example. In Table 1

LK	Uncertain (presupposition)	An event of 'hitting' (from the before-clause) happened in the past
	Certain (entailment)	'The bullet' and 'Mary' denote individuals An event of 'deviation' happened in the past The deviation-event happened before the hitting-event An event of 'hitting' (from the nevertheless-clause) happened in the past
ELK	Uncertain (content)	The individual denoted by 'Mary' is a female human being The hitting-event, because of the deviation-event, did not happen
	Certain (ontology)	The hitting-event is in the future for what concerns the before-clause, so it is uncertain.

Table 1: Inferences from utterance (8).

some of the facts that can be derived from utterance (8) reported here below are shown and classified along the LK/ELK and uncertain/certain dimensions.⁴

"The bullet deviated before hitting Mary. Nevertheless it hit her." (8)

The phenomenon of temporal presuppositions seems to be an expression of the ontology of time, not of the content of time. The ontology of time is its *ordering* and the fact that while the past is in a sense *closed*, the future is *open*. This leads to certain inferences. On the other side, the *metric* of time is a content characteristic, in that the subjective evaluation of the duration of a time interval may vary depending on the situation, and this usually leads to uncertain inferences. Then, the phenomenon of temporal presuppositions can be explained in the following way: an event in the future cannot be certain, because of the ontology of time (partial unpredictability of the future).⁵

and inference rules used. Therefore, it is possible to speak of axioms (inference rules, inferences, and facts) of LK and ELK type. Examples of distinctions can be, besides the WK/LK in [29], the terminological/assertional [9], or the symbolic/subsymbolic [38] dichotomies.

⁴The case of the certain ELK inference might seem a bit awkward. A more convincing example is the fact that in utterance "Mary met John after she left" the leaving event did certainly happen.

⁵It is important to point out that 'future' refers to the

This is why ‘before’ introduces a temporal presupposition, while ‘after’ does not.

In this work, I am interested in those parts of LK and ELK that are related with temporal presupposition and counterfactuals. The content is not the focus of this research, but it plays a role (indeed a marginal one) into the above described linguistic phenomena. As a matter of fact, content inferences can contradict presupposed and/or entailed events, thus sometimes (but only if relevant and necessary) it will be necessary to take content into account. Note that entailments overcome content inferences (as, for instance, in utterance (8)), and that content inferences overcome presuppositions (as, for instance, in (4), (6) and (8)).

2.3 Abstract syntax

In order to analyze the above introduced phenomena, it is sufficient to work on a restricted language fragment, defined in this section. The usual way to formally define a fragment of the language is to provide a *grammar*. Since the considered phenomena occur in many natural languages (almost every western language has the syntactic constructs necessary for expressing the previous utterances), I prefer here a more abstract description, to some extent independent from the particular language adopted. I shall call such formalism *abstract syntax*.

The first step to define the abstract syntax of the relevant natural language fragment (that will be denoted with L) is to specify a family of *syntactic functions*, functions that syntactically manipulate sentences of the natural language to obtain other sentences. The definition of the abstract syntax of L is then obtained by means of a set hierarchy: starting from a set of simple sentences, other sets containing *complex* and *compound* sentences [42] are obtained as the range of syntactic functions. The union of these sets will be L .

The syntactic functions used to cover all the linguistic phenomena presented in the previous section are the following:

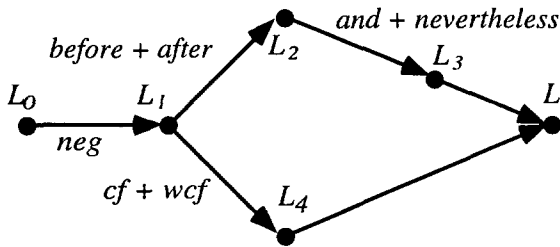
- $neg(s)$: returns the negation of sentence s .

point of reference, not to the point of speech [36]. In utterance (1), both the events happened in the past (‘met’ and ‘left’), but the second is in the future of the point of reference.

For example, if s is ‘Mary left’ (sentences are enclosed in single quotes), $neg(s)$ is ‘Mary did not leave’;

- $before(s_1, s_2)$: returns the complex sentence formed by the main clause s_1 and the temporal subordinate s_2 , introduced by ‘before’. Observe that the syntactic functions do not only concatenate the strings given as arguments, but also (syntactically) manipulate them to obtain the correct result. For example, from ‘Mary met John’ and ‘Mary left’, using the syntactic function $before$, one should obtain ‘Mary met John before she left’ and not ‘Mary met John before Mary left’;
- $after(s_1, s_2)$: returns the complex sentence formed by the main clause s_1 and the temporal subordinate s_2 , introduced by ‘after’;
- $and(s_1, s_2)$: returns the compound sentence constituted by the two sentences s_1 and s_2 joined by the conjunction ‘and’;
- $nevertheless(s_1, s_2)$: returns the compound sentence constituted by the two sentences s_1 and s_2 joined by the conjunction ‘nevertheless’. Usually, $nevertheless(s_1, s_2)$ is a pair of sentences separated by a full stop. Here this detail is not important, in that the two sentences, from a semantic point of view, are co-ordinated;
- $cf(s_1, s_2)$: returns the counterfactual sentence with s_1 as antecedent and s_2 as consequent. For example, if s_1 is ‘Mary met John’ and s_2 is ‘Mary left’, $cf(neg(s_1), s_2)$ is ‘If Mary had not met John, she would have left’;
- $wcf(s_1, s_2)$: returns the weak counterfactual, i.e. a sentence (syntactically) differing from a counterfactual one in that ‘even if’ substitutes ‘if’. For instance, if s_1 and s_2 are the two sentences just met for the cf function, then $wcf(neg(s_1), s_2)$ is ‘Even if Mary had not met John, she would have left’.

Now, the syntactic functions listed above are used to formally define the fragment L : as it was said before, a set hierarchy is built, the last set of the hierarchy being L . The process of L ’s construction is illustrated in Figure 1. Nodes indicate

Figure 1: The construction of L .

the subsets of L , arcs mean set inclusion and arc labels show which syntactic functions are used to obtain the following sets.

The first set of the hierarchy, L_0 , contains simple sentences, like 'Mary met John' or 'Mary left', and so on.

Using the *neg* function, the set L_1 can be defined as⁶

$$L_1 = L_0 \cup \text{neg}(L_0).$$

L_1 contains sentences and their negations, so sentences as 'Mary did not meet John' belong to L_1 .

The next set is defined by the *before* and *after* functions:

$$L_2 = L_1 \cup \text{before}(L_1, L_0) \cup \text{after}(L_1, L_1).$$

Note that the temporal clauses introduced by 'before' are always affirmative, as observed in [24] and as indicated by the utterances and sentences (in particular (5)) presented above.

The following steps are:

$$L_3 = L_2 \cup \text{and}(L_2, L_1) \cup \text{nevertheless}(L_2, L_1),$$

$$L_4 = L_1 \cup \text{cf}(L_1, L_1) \cup \text{wcf}(L_1, L_1).$$

The final set, L , is then obtained as

$$L = L_3 \cup L_4.$$

In this section, only sentences have been dealt with, but the extension to the case of utterances is immediate. In fact, if u is an utterance, then u is a pair (s, c) , where s is the sentence and c the context. Then,

$$\text{neg}(u) = (\text{neg}(s), c)$$

and similarly for the other syntactic functions.

⁶Here and in the following of this section, the standard notation for using sets as functions arguments is used: $\text{neg}(L_0)$ stands for $\{\text{neg}(s) \mid s \in L_0\}$, and similarly for the other syntactic functions, paying attention to their arity.

3 Recursive models

This section presents *recursive models* (RM), a formalism that can be used to represent the meaning of utterances at a semantic/pragmatic level. In Section 3.1 the RMs are defined as an instance of the class of computable models. In Section 3.2 RMs are seen as an abstract data type, whose formal specifications are given. In Section 3.3 the structure of RMs is described. In Section 3.4 the functions that build and use an RM are analyzed and a possible implementation is sketched. In Section 3.5 related work is discussed.

3.1 Computable models

From a computational perspective, two approaches are possible for representing the semantics of a discourse,⁷ and for using such representation in finding implications between the discourse and following utterances. In the first, 'inferential', approach, the discourse is translated into a theory (a set of logical formulas) Γ ; the same happens to a following utterance, obtaining, say, the logical formula ϕ ; then, to discover whether the discourse implies the utterance, an inference procedure \vdash is used for testing whether $\Gamma \vdash \phi$.⁸

In the second, 'model-theoretic', approach, the discourse is used to build a model M , and an evaluation function (usually denoted by \models in mathematical logic) is used in order to test whether $M \models \phi$.

These are obviously two quite different approaches: in the former the central notions are a set of axioms (to which further ones can be added for taking into account new utterances) and a set of inference rules; the latter is based on the two functions that, respectively, *integrate* (*int* in the following) a previous model with the information of a new utterance, and *evaluate* (*eval* in the following) an utterance in a previously built model.

If the representation of the semantics of a discourse has to be used by an algorithm, both these approaches reveal some decidability problems. In the inferential approach, this happens when nei-

⁷A *discourse*, or a *text*, can be defined as a sequence of utterances. The concepts of implication, entailment and presupposition described in Section 2.1 can be extended in a natural way in order to deal with discourse.

⁸For an explanation of the concepts derived from mathematical logic, see for instance [12, 16].

ther the utterance (ϕ) nor its negation ($\neg\phi$) are an entailment of the discourse (Γ), and this is a common situation, in that the logical theory Γ is not necessarily *complete*. The standard solution is to abort the inference process when it is too long, the length of the process being the number of inference steps or the computation time. In the model-theoretic approach, similar decidability problems arise when the evaluation function is not computable. This leads to a constraint on the models: their expressivity has to be sacrificed, for obtaining a computable *eval* function. I shall call the models with such property *computable models*.

In the next subsections I will propose an instance of computable models named *recursive model* (RM) that can be used to represent the semantics of utterances belonging to the language fragment defined above. I will not formally prove the computability of the corresponding *eval* function; instead, the approach is empirically tested by utilizing RMs in a system whose implementation will be described in Section 4.

3.2 Formal specifications of recursive models

This section describes the RMs from a functional point of view, formally specifying their behaviour without referring to their structure. In other words, I propose the *formal specifications* of the Abstract Data Type (ADT) RM. The formal specifications of the ADT RM, being rather complex, only a brief sketch is presented here. I will define (some of) the *sorts*, (some of) the functions that define the ADT RM, together with their *signature*, and (some of) the *axioms* that describe the behaviour of the functions.⁹

The sorts of ADT RM are:¹⁰

- U , the set of all utterances. On this sort,

⁹Note that I said ‘formal’, not ‘algebraic’ specifications: in algebraic specifications [7, 39] the axioms must be equations, in order to have an executable object. Here I am interested only in obtaining a formal definition of the behaviour of the ADT RM, not in the computational aspect (that will be tackled in the following), so I prefer not to have restrictions on the shape of axioms.

¹⁰These are not all the sorts needed to completely specify the ADT RM. Another sort, the set E of all events, on which the functions that describe the causal links between events must be defined, is necessary.

all the syntactic functions presented in Section 2.3 are assumed to be defined;

- M , the set of all RMs;
- B , the set of boolean values ($\{true, false\}$). I assume that the usual logical connectives are defined as functions on this sort;
- Bu , the set obtained adding the undefined value to the set of boolean values ($\{true, false, undef\}$). Also on this sort I assume that some logical operations are pre-defined. There exist various 3-valued logics; among them I need Bochvar’s logic [8, 40], in which the *undef* value is ‘contagious’ (i.e., if *undef* is one of the arguments of a logical operation, the result will be *undef* too).

On such sorts, the following functions are defined (together with the signature of the functions, I also present an informal description of their behaviour):

- *create*: $\rightarrow M$, that returns an empty RM;
- *int*: $U \times M \rightarrow M$, that returns a new RM obtained integrating the information of a new utterance in a previously existing model;
- *eval*: $U \times M \rightarrow Bu$, that evaluates the truth value of an utterance in a model;
- *modify*: $U \times M \rightarrow M$, that, given a counterfactual utterance and an RM as arguments, returns the RM obtained modifying the original RM in such a way that the antecedent of the counterfactual utterance is evaluated *false*. The model obtained is named *counterfactual model*;
- *pref*: $2^M \rightarrow M$, that selects the preferred RM among the set of plausible ones. For example, in the case of utterance (1), *pref* should choose the RM in which Mary left, and not the one in which Mary did not leave. This function, together with the following three, is needed because of the nonmonotonic aspect of the phenomenon of temporal presuppositions;
- *contr*: $U \times M \rightarrow B$, that is *true* iff an utterance, once integrated in an RM, leads to a contradiction. This happens, for example

when integrating the second part of (4) in the RM obtained from (1), where it is not longer true that Mary left;

- *rev*: $U \times M \rightarrow M$, that operates a revision of an RM when it, together with an utterance, leads to a contradiction;
- *intmon*: $U \times M \rightarrow M$, that can integrate utterances that do not present contradiction with the existing RM. Therefore, *intmon* cannot treat nonmonotonicity, but it is the core of *int* function;
- *intset*: $U \times 2^M \rightarrow 2^M$, that from the set of previous plausible RMs and an utterance returns another set of RMs. This function is needed because it is possible to build more than one RM from an utterance, as is shown, for example by utterance (1) and (4);
- *evalset*: $U \times 2^M \rightarrow Bu$, that is *true* iff the utterance given as the first argument is evaluated *true* in all the RMs belonging to the set given as the second argument;
- *entail*: $U^* \times U \rightarrow Bu$, that is *true* iff an utterance is an entailment of a discourse, i.e. a sequence of utterances (with the * operator I indicate the concatenation of utterances);
- *imply*: $U^* \times U \rightarrow Bu$, that is *true* iff an utterance is evaluated *true* in the preferred model of a discourse. These two last functions can be defined in terms of the previous ones, see below.

As it was said, I present here only some examples of the axioms needed for the ADT RM. One of such axioms defines the *int* function using functions *intmon*, *rev* and *contr*:¹¹

$$\begin{aligned} \text{int}(u, m) = & \text{if } \text{contr}(u, m) \\ & \text{then } \text{intmon}(u, \text{rev}(u, m)) \\ & \text{else } \text{intmon}(u, m). \end{aligned}$$

The evaluation of counterfactual and weak counterfactual utterances takes place in a peculiar way. A counterfactual utterance $cf(u_1, u_2)$ is evaluated *true* if and only if its antecedent u_1 and consequent u_2 are evaluated *false* and

the event represented in the consequent should have happened if the event in the antecedent had happened. In other words, the evaluation of $cf(u_1, u_2)$ in an RM m takes place evaluating u_1 and u_2 in m and then evaluating u_2 in a model obtained *modifying* the RM m on the basis of the antecedent u_1 . A weak counterfactual $wcf(u_1, u_2)$ behaves in the same way, with the exception that the consequent u_2 must be evaluated *true*. This is formalized by two axioms:

$$\begin{aligned} \text{eval}(cf(u_1, u_2), m) = & \\ \text{if } \text{eval}(u_1, m) = \text{false} \text{ and} & \\ \text{eval}(u_2, m) = \text{false} & \\ \text{then } \text{eval}(u_2, \text{modify}(u_1, m)) & \\ \text{else } \text{false} & \end{aligned}$$

$$\begin{aligned} \text{eval}(wcf(u_1, u_2), m) = & \\ \text{if } \text{eval}(u_1, m) = \text{false} \text{ and} & \\ \text{eval}(u_2, m) = \text{true} & \\ \text{then } \text{eval}(u_2, \text{modify}(u_1, m)) & \\ \text{else } \text{false} & \end{aligned}$$

(note the use of the syntactic functions *cf* and *wcf*).

Another axiom defines the *imply* function in terms of *eval* and *int*:

$$\text{imply}(\mathbf{u}_1, u_2) = \|\text{eval}(u_2, \text{int}(\mathbf{u}_1, \text{create}()))\|,$$

where the symbol \mathbf{u}_1 denotes a sequence of utterances, i.e. a discourse, and the symbol $\|\cdot\|$ indicates Bochvar's 'assertion operator', that maps the *undef* value in *false* and does not affect the other two logic values.¹²

A similar axiom can be given for the definition of the *entail* function. Here the notion of *set of models* must be used: an utterance is entailed by another utterance only if the former is *true* in all the models of the latter. Such an axiom is:

$$\begin{aligned} \text{entail}(\mathbf{u}_1, u_2) = & \\ \|\text{evalset}(u_2, \text{intset}(\mathbf{u}_1, \{\text{create}()\}))\| & . \end{aligned}$$

¹²Note that the first argument of *int* is a sequence of utterances, while *int* should have as argument a single utterance (*int*: $U \times M \rightarrow M$). But it is easy to define by recursion *int'*: $U^* \times M \rightarrow M$ in the following way:

$$\begin{aligned} \text{int}'([], m) & = m; \\ \text{int}'([u_1|u_2], m) & = \text{int}'(\mathbf{u}_2, \text{int}(u_1, m)) \end{aligned}$$

(where the standard symbology of Prolog lists is used in order to indicate a sequence of utterances) and redefine *int* as *int'*. The same remark has to be made for the *intset* function in the following equation.

¹¹The if-then-else operator used here has to be intended as a declarative one, without any procedural meaning.

As a last example, the following axiom defines the connection among the *int*, *intset* and *pref* functions:

$$\text{int}(u, m) = \text{pref}(\text{intset}(u, \{m\})).$$

The meaning of this axiom should be clear: the RM obtained by *int* is the preferred one in the set of all plausible models, as generated by the *intset* function.

3.3 Structure of recursive models

The previous section has shown how to formally define the properties that RMs must have. Here, the *structure* of the RMs is presented.

Roughly speaking, an RM is constituted by *instances* of classes of an encyclopedia and *relations* among those instances. Therefore, an *encyclopedia* is needed, that is a taxonomy of *categories* and *concepts*. The encyclopedia is a knowledge base, and is needed in order to know that Mary and John are persons, hence living beings, and so on; that the meeting of Mary and John is an event, etc.

Using the operation of *instantiation* it is possible to create a *token* for each individual mentioned in the utterance. Referring to utterance (1), there will be tokens for ‘Mary’, ‘John’ (instances of the class *person*), ‘met’ and ‘left’ (instances of the class *event*). Every token has an associated identifier; I shall use uppercase letters for instances of objects (M for ‘Mary’, J for ‘John’), and lower case letters for events (m for ‘met’, l for ‘left’, etc.). As usual, tokens inherit *slots* from their parent concepts, so M is the value of the slot *agent* of m and J is the value of the slot *theme* of m. Moreover, between tokens m and l there is a temporal *relation* to indicate that the meeting took place before the leaving.

Tokens, slots and relations are not sufficient to obtain a complete RM, since by using only these components, one would obtain the same RM for the utterance

“Mary did not meet John before she left”

and this is clearly a problem. To deal with event occurrence and object existence, other elements are introduced in the RM: *spaces*, *attachments* and *signs*.

A *space* is needed because not only an object exists, or an event takes place; it is more correct to say that an object exists (or an event takes place) *in a world*. Consider utterance (4): Mary did not leave in the *real world*, but it is correct to say that Mary left in the counterfactual world (see utterance (7)) in which she did not meet John. Analogously, it is possible to say that Donald Duck does not exist in the real world, but he exists in Walt Disney’s world.

So, a space is a formal tool for representing alternative worlds. I indicate the real world with \square . It is possible to represent the object existence and the event occurrence *attaching* every token to the right world: the relation between token and world is named *attachment*. Finally, attachments are labelled with a *sign* in order to deal with non-existence and non-occurrence, both of which are represented by a negative sign, whereas a positive sign obviously means existence and occurrence.

As illustrated in Section 2.1, the occurrence of an event may be certain (the meeting of (1)) or uncertain (the leaving of (1)); this can be dealt with using *certain* and *uncertain* signs. In the RM of (1), the signs labelling the attachments of the tokens for ‘met’ and ‘left’ are both positive, but only the first is certain, while the second is uncertain.

The RM obtained for (1) is illustrated in Figure 2. Only the portion of the encyclopedia needed to build the RM of the utterance is represented (in the upper gray area, while in the lower white area, the proper RM is sketched): each rectangle stands for a concept. The relations *is-a* (between two concepts) and *instance-of* (between a concept and a token) are represented by labelled grey arcs, tokens are shown as circled letters, slots are illustrated by means of oriented arcs, relations, as usual in entity-relationship diagrams used in data base theory [13], are represented by arcs labelled with a rhombus (the symbol $<$ stands for ‘precedes temporally’), a dashed arc represents an attachment, a bold sign is certain and a plain text sign is uncertain. For the sake of simplicity, in the graphic representation the names of the slots are not illustrated.

The RM in Figure 2 models the meaning of (1). Nevertheless, there is another element to add for dealing with the *causal links* relating the occurrence (or non-occurrence) of events. Examples

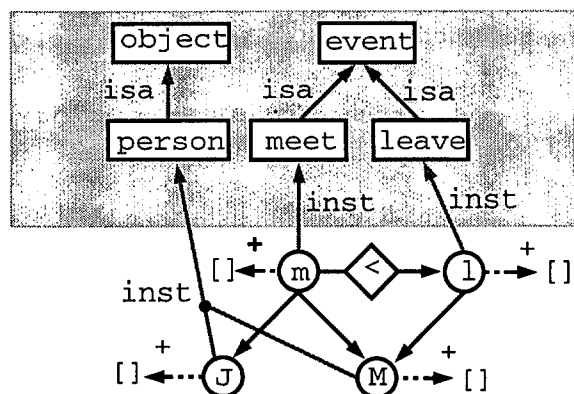


Figure 2: Graphic representation of the RM of the utterance (1).

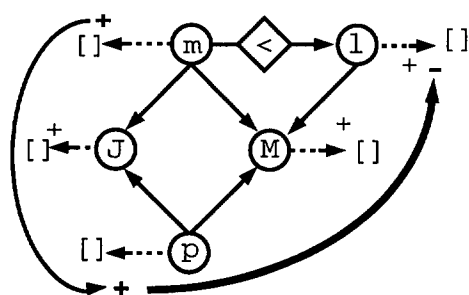


Figure 3: Graphic representation of the RM of (4).

can be found in utterances (4) and (7) (the occurrence of the meeting with John causes the occurrence of the event ‘Mary stayed at home’) and (6) (the occurrence of Mary’s leaving causes the non-occurrence of the meeting with John).

The elements used in RMs to represent such causal relations are named *justifications*, and are represented by curved arcs. As signs, justifications may also be certain or uncertain. In order to understand the role of these new elements, consider Figure 3, in which the RM of (4) is represented. Here and in the following, for the sake of simplicity, I have omitted the representation of the encyclopedia (i.e. the classes and the *isa* and *inst* relations): the letters labelling the tokens should be sufficient for understanding which class each token is an instance of. Furthermore, the token *p* is assumed to be an instance of the ad-hoc class *persuade to stay at home*.

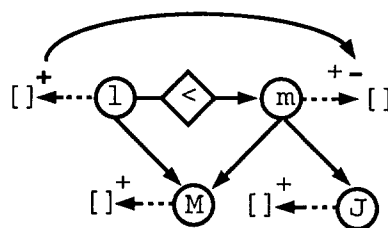


Figure 4: Graphic representation of the RM of (6).

The justification between the signs of tokens *m* and *p* is uncertain (graphically represented by a thin curved line), whereas the one that links the signs of *p* and *l* is certain (thick curved line). The reason for this distinction is that the meeting implies persuading in a very weak sense (it is a precondition), while persuading (to stay at home) entails non-leaving.

Note furthermore that in Figure 3 *l*’s attachment is labelled with two signs: the positive one (uncertain) models the presupposition of the leaving and the negative one (certain) reflects the fact that the leaving actually did not take place. The last sign is the *preferred* sign (and it overrides the uncertain one); graphically, this is represented putting it near the end of the arc.

Justifications are needed not only by abstract completeness considerations, but also to deal with counterfactual utterances, as is explained in the next section.

3.4 The implementation of *eval* and *int* functions

At this point, the structure of the RMs should be clear. Now, I present via a couple of examples the algorithms that implement the *eval* and *int* functions (that build an RM for an utterance and evaluate a question in an RM, respectively). Both algorithms can be defined in the same way (by structural recursion on the *logical form* of an utterance, see below), therefore I describe only the way the model of an utterance is built.

A raw RM is built on the ground of LK and ontology and is then refined using content knowledge. Let us consider for example the RM of utterance (6) represented graphically in Figure 4. The following steps take place during its creation:

- token 1, from 'left', is created and it is attached to the space [] with a positive and certain sign. The sign is certain because of linguistic considerations: 'left' belongs to the main proposition;
- token M is created and it becomes the value of slot **agent** of token 1. Now, the building of the RM of the main proposition is terminated;
- token m, from the event of the secondary proposition, is created and attached to [].¹³ The sign of this attachment is still positive, but uncertain because the event is in a secondary proposition;
- the slot **agent** of token 1 assumes as value the token M, already present in the RM; token J is instead created and it becomes the value of slot **theme** of token m;
- the temporal relation between the tokens 1 and m is created;
- all the above operations take place on the ground of linguistic and ontological considerations. However, to complete the construction of the RM, some content inferences are needed to create a negative certain sign (preferred to the positive uncertain one) on the attachment of m and the corresponding justification.

Thus, the division of linguistic, ontological and content work seems clear. Linguistically and ontologically, tokens are created, slot values are filled, relations explicitly referred in the utterance are produced and attachments are created. On the ground of content considerations, justification arcs, representing the causal relations between events implicit in the utterance, are added, and the same happens for new signs.

However, the separation between LK, ontology and content is not so simple: temporal relations

¹³The attentive reader might note that the processing of the clause containing the presupposition, the secondary one, takes place after the main one's. This is in contrast with the nature of the presuppositions, which should be tackled as first. But, I pointed out in footnote 2 in Section 2.1, here it is the 'a posteriori' aspect of temporal presuppositions (and of the whole sentences encompassing them) that is studied, so this is not a relevant difference.

may be created on the basis of content, and justifications on the basis of LK. This happens, for example, in the creation of the RM of (7), that is similar to the one represented in Figure 4: the only differences are the attachment of m (that is labelled by only one negative certain sign) and the justification (that is certain too). In this case, the temporal relation is created on the basis of the content, in that the fact that the leaving takes place before the meeting is indubitably a content inference. Furthermore, the justification derives from LK considerations, in that it appears explicitly in the word 'if' of the utterance.

As already specified, the discussion above regards exclusively the function *int*. Nevertheless, the algorithm that implements the function *eval* can work in a similar way; instead of creating tokens, it verifies that they already exist in the RM.

The algorithm implementing *eval* must work in a particular way for the evaluation of counterfactual utterances. Such evaluation takes place in three steps: first, the antecedent and the consequent of the counterfactual utterance are evaluated in the current RM; second, the current RM is modified accordingly to what it was said in the antecedent of the counterfactual, obtaining the counterfactual model; third, the consequent of the counterfactual is evaluated in the counterfactual model. Let us consider the evaluation of utterance (7) in the RM for (4) (the RM in Figure 3). The evaluation takes place in the following way:

- the antecedent and the consequent of (7) are evaluated in the RM; both of them are *false* (and they must be *false* in order to evaluate the counterfactual utterance *true*);
- the counterfactual model, that is obtained modifying the original RM in such a way that the antecedent is evaluated *false*, is illustrated in Figure 5. Observe that the token m is attached with a negative sign to [], in that the antecedent must be evaluated *false*. This, by means of the justification between the signs of m and 1 (see the original RM in Figure 3), leads to removing the positive sign on p's attachment and labelling this token with an opposite (negative) one. The same happens with token 1; here the removal of the negative sign brings up the positive sign;
- the consequent of the counterfactual ("Mary

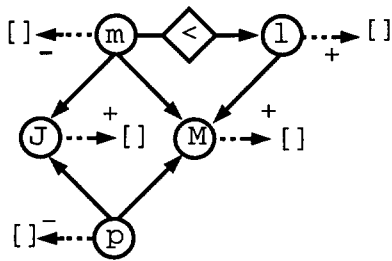


Figure 5: Counterfactual model for the evaluation of (7).

left”) is evaluated in the counterfactual model, obtaining *true* as result. The counterfactual utterance itself is then evaluated *true*.

From the informal description in this section, it should not be difficult to extract the algorithms for *int* and *eval*, implemented in the system described in Section 4.

3.5 Related work

A brief look at related work is mandatory, in order to emphasize the differences between RMs and other proposals. In this section, researches on *discourse models* and *discourse representation theory* (DRT) are briefly compared with RMs, and it is shown how RMs can handle in a simple way the concepts of *belief* and *situation*.

RMs can be seen as models of previous discourse context, into which information from sentences is merged, and against which queries are evaluated. There are a lot of studies on *discourse models* in which it is investigated how the various structures that can be individuated in a discourse ought to be used to understand the meaning of the sentences forming such discourse: see for instance [22, 28, 31, 35, 37, 41]. RMs could be a new instrument for this research, even if it might be more appropriate to say that RMs are a computational tool for modeling the meaning of sentences, and that they do not seem to suffer from any intrinsic limitation for being used at the level of discourse.

RMs are also comparable to DRS (Discourse Representation Structures), the ‘models’ used in DRT [26], but here also there are some differences. First of all, Kamp and Reyle themselves say in

their book on DRT [26, page 627] that they don’t tackle the problems I have analyzed here:

There exists the possibility of using before-phrases in a kind of “virtual” sense which is not possible for prepositional phrase with after. In a case where the sentence “George died before the completion of his novel” is true, the completion of the novel presumably never took place. [...] This use of before has given semanticists a good deal of trouble. [...] It is an issue which we will not pursue here.

Notwithstanding that, one might try to treat temporal presuppositions in DRT—and encounter some difficulties. Consider for instance the standard DRS of utterance (6) reported here

“Mary left before meeting John”, (6)

namely the DRS of Table 2. The DRS is divided in 3 groups, separated by empty lines: the first one models the main clause, the second one the word ‘before’, and the third one the subordinate clause. In such DRS there is nothing representing the facts that the event e_2 (the meeting one) is only presupposed (and then uncertain), that it has not happened, that there is a causal link between the occurrence of the two events and there is no ‘first-order’ object representing the occurrence of the events.

t_1	n	e_1	x	t_2	e_2	y
$t_1 < n$						
$e_1 \subseteq t_1$						
$mary(x)$						
$e_1 : leave(x)$						
$t_1 < t_2$						
$t_2 < n$						
$e_2 \subseteq t_2$						
$john(y)$						
$e_2 : meet(x, y)$						

Table 2: The DRS of (6).

Obviously, DRS could be extended in the direction indicated by RMs, but this is not so simple,

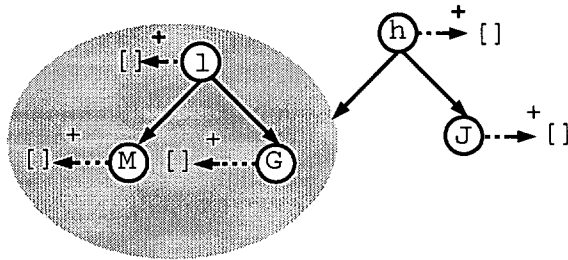


Figure 6: Situations in RMs.

in that in DRS there is nothing like RMs' spaces, attachments, signs and justifications, which are central concepts in RMs. So, DRSs might be situated at a semantic level, while RMs work on the semantic-pragmatic boundary: a DRS is more similar to a logical form [2] than to an RM.

RMs can be extended in a natural way for taking into account the concepts of *beliefs* and *propositional attitudes* [5, 15]. For instance, spaces allow to easily represent Mary's *intention* to leave in utterances (1) and (4): it is sufficient to attach the token 1 in Figures 2 and 3 to a space, $[int(M)]$, representing the world of the events that should have happened if everything had gone as presupposed. In this way, one can create a family of operators on worlds ($int(X)$ for intentions, $bel(X)$ for beliefs, and so on), indexed on the tokens of the RM. These operators can transform one world (for instance $[\]$) in other ones ($[int(M)]$, $[bel(M)]$, etc.)

Finally, RMs might easily be improved for handling utterances like

“Mary left with George. This hurt John”

in which it is not the event per se that ‘hurt John’, but the whole context. In order to treat this kind of utterances, it will be necessary to introduce the concept of *situation* [5, 15] in RMs: the RM of this utterance could look like the one in Figure 6, where the grey circle is the graphic representation of ‘what hurt John’.

4 The TOBI system

This section presents TOBI, a system that communicates with the user in natural language (En-

```
User> Mary met John before she left.
User> Did Mary leave?
CS> Yes.
User> Did Mary kiss John?
CS> I don't know.
User> Ann met George before she left and
      he persuaded her to stay at home.
User> Did Ann leave?
CS> No.
```

Figure 7: An example of interaction CS - user.

glish) and uses the RMs illustrated in the previous sections as internal representations of utterances. TOBI is implemented in LPA Prolog on a Macintosh, and it can handle all the examples presented in Section 2.1 (and similar ones). The following subsections illustrate: the class of natural language processing systems to which TOBI belongs, the architecture of the system, its data flow, and its internal data structures.

4.1 Comprehension systems

TOBI is a natural language processing system. It is indeed a particular case of such systems, a *comprehension system* (CS): it has the *unique* aim of interacting with the user in natural language. This section describes a CS using the concepts presented in Section 3.1 and, on the basis of this description, some design choices made in TOBI are motivated.

A CS simulates the typical human activities of comprehension and production of natural language utterances: it can understand a discourse (sequence of utterances) and provide correct answers to questions regarding the discourse. For the sake of simplicity, only *polar* questions are considered, i.e. questions admitting as answers only ‘yes’ (*true*), ‘no’ (*false*) or ‘I don't know’ (*unknown*). In Figure 7 an example of dialogue between a hypothetical CS and a user is shown.

CSs work by building some internal representation of a discourse, and using such representation to answer successive questions. On the basis of what it was presented in Section 3.1, the implementation of a CS can be accomplished in two ways. The first (and traditional, see [20]) one is to build a nonmonotonic inferential system, that uses an inference procedure \vdash (and usually a TMS, Truth Maintenance System). This kind

of CS will be named *CS Formulae & Inference* (F&I). The second way of realizing a CS is to implement a system that builds a (computable) model of the discourse and evaluates the question in that model in order to obtain the right answer. Systems of this kind are named *CS Models & Evaluation* (M&E), and (with respect to CS F&I) work at the more primitive and flexible level of models and model-based evaluation.

TOBI is a CS M&E that uses the above described RMs to model the meaning of utterances. Since CSs F&I may rely on well known basis, developed in mathematical logic, the attempt to follow the new way of CSs M&E must be justified. The most persuasive critique of CSs F&I concerns the way they have to abort the process of inference if they obtain no answer. This is an unnatural way of working, and it has no cognitive plausibility. On the other hand, CSs M&E present many interesting features: they seem to have more cognitive plausibility (it is widely recognized that human beings build a model of the utterance they hear, and that they don't use an inferential mechanism to answer questions), they might deal with the problem of termination in a better way than CSs F&I do, and they show a natural treatment of implications weaker than entailment (like the presuppositions met in the examples in Section 2.1).

These observations motivate the attempt to follow the approach of CSs M&E. However, it must be said that CSs F&I are preferable in handling entailments and incomplete knowledge, fields in which the inferential approach demonstrates all its power.

Summarizing, TOBI is a CS M&E, and not a F&I one for the following reasons:

- the kind of phenomena it has to deal with: mainly presuppositions, not entailments;
- the greater cognitive plausibility;
- the supposed better control of the weakening of the system's inferential capacities;
- the examination of what can be done using models and evaluation in place of classical and well known logical calculi.

4.2 TOBI's architecture

Figure 8 presents the architecture of TOBI. Here is a list of TOBI's modules with a short descrip-

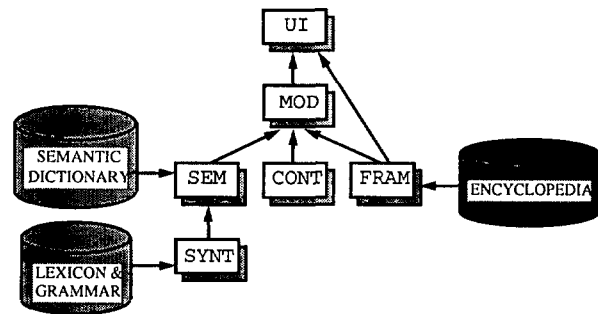


Figure 8: TOBI's architecture

tion of their tasks:

- SYNT: morphoSYNTactic analyzer that parses the input utterance, producing its *syntactic structure*. SYNT uses a *lexicon* and a *DCG grammar* [19] as knowledge bases;
- SEM: SEMantic analyzer; it takes the syntactic structure produced by SYNT and produces as output the *logical form*, that is a representation of the utterance in a slot-filler notation, in which events and semantic roles are singled out. This module uses a *semantic dictionary* associating syntactic terms with the corresponding concepts;
- FRAM: FRAMe Manager; manager of the *encyclopedia* (a taxonomy of categories and concepts) and models. It implements the procedures needed to work on classes (the encyclopedia) and instances (the models);
- CONT: the module devoted to handling CON-Tent knowledge;
- MOD: MODel builder; module that implements the functions *int* and *eval* using procedures from SEM, FRAM and CONT;
- UI: User Interface; it accepts utterances from the user (via keyboard) and answers his (her) questions. This interface is developed using the features of LPA Prolog for windows and menus management.

4.3 TOBI's data flow

In Figure 9 the data flow of TOBI is presented, in order to illustrate the process that takes place

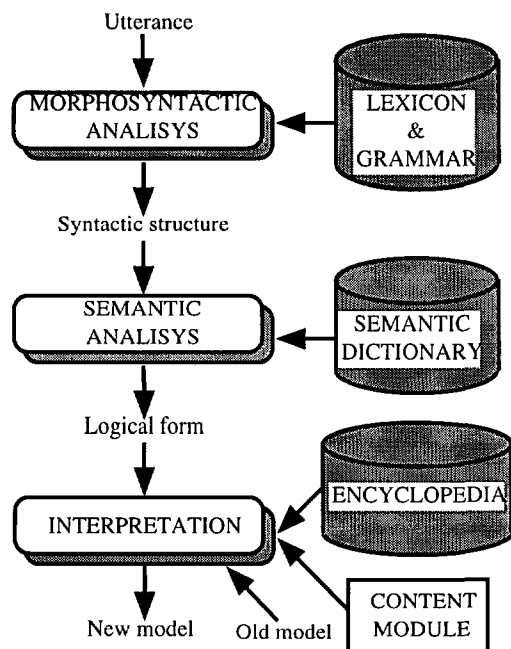


Figure 9: TOBI's data flow for the interpretation of an utterance.

when the system builds an RM from an utterance. TOBI processes the utterance in three steps. The first step is the *morphosyntactic analysis*: the input utterance is parsed into its syntactic structure.

The syntactic structure is input to the *semantic analysis*, that produces another representation of the initial utterance, namely its logical form.

The last step is the *interpretation*: here the logical form is used to build the RM of the utterance (or, more generally, to *integrate* the old RM with the new information in the utterance). It is in this phase that TOBI's peculiarity comes in evidence. In most natural language systems, content knowledge is encapsulated in the encyclopedia, together with ontological knowledge. In TOBI the two kinds of knowledge are separated; the encyclopedia contains only ontological knowledge, that can easily be dealt with in symbolic terms; the content part is handled by another module.

As it was said in Section 2.2, the phenomenon of temporal presuppositions is based on the ontology of time, not on its content. But a system that works only at an ontological level could do very little. For example, to understand utterance (6) content considerations are necessary for keep-

```

1 s(asser,
2   vg(sing,meet,trans,ind,past,aff),
3   subj(np(sing,f,det,[],
4         pNoun(person(mary)),[])),
5   obj1(np(sing,m,det,[],
6         pNoun(person(john)),[])),
7   obj2(nil),
8   [es(
9     prep(before),
10    s(asser,
11      vg(sing,leave,intr,ind,past,aff),
12      subj(np(sing,f,det,[],pronoun,[])),
13      obj1(nil),obj2(nil),[]))])

```

Figure 10: The syntactic structure that TOBI generates when interpreting (1).

ing into account the relation between the leaving and meeting events. Then TOBI has to deal with content inferences too. I have assumed that ontology can be handled using classical symbolic methods; there are reasons, however, to believe that this might not be true for content (see for instance [1]). Furthermore, the linguistic phenomena studied rely on the ontology, not on the content. Therefore, in the present version of TOBI, content inferences are replaced by an interface to an external user, activated upon request of a master module, which fully implements ontological inferences. The clear division between ontology and content gives a conceptually clean system, and the implementation of a 'real' content module can be tackled in an independent way.

4.4 TOBI's data structures

In this section I go deeply into the internal details of TOBI's work, illustrating in a concrete example the utterance analysis process. The data structures passed across the three steps described in Section 4.3, namely the utterance, the syntactic structure, the logical form and the RM, are explicitly shown. Let us consider the interpretation of utterance (1)

"Mary met John before she left" (1)

The syntactic structure, that the SYNT module builds starting from the utterance (1), is the Prolog term showed in Figure 10, where (see [42] or [2] for a description of the terminology used here):

- **s** stands for ‘sentence’ and **asser** means that the sentence is assertive;
 - line 2 represents the verb group (**vg**) that is singular, has head ‘meet’, is transitive, is in the indicative form, in the past tense and affirmative;
 - line 3 models the subject of the main clause; it is a noun phrase (**np**), singular, female, definite, without modifiers (**[]**), with head the proper noun ‘Mary’ and without qualifiers (**[]**);
 - lines 4 and 5 represent the direct (‘John’) and indirect object (not present here) of the main clause, respectively;
 - in lines 6 to 10, the embedded sentence (**es**) introduced by the temporal presupposition ‘before’ is represented, in a recursive manner. The symbols have the same meaning as in the main clause.
- the distinction between linguistic and extra-linguistic knowledge, and the role played by different kinds of knowledge and inferences (entailments, presuppositions, ontology and content) in the linguistic phenomena studied;
 - the abstract syntax of the fragment of language related to temporal presuppositions and counterfactuals;
 - the recursive models, an instance of computational models for naturally dealing with temporal presuppositions and counterfactuals. I have sketched the formal specifications of recursive models, described their structure and compared them with related proposals;
 - the consideration that the nonmonotonic linguistic phenomena of temporal presuppositions and counterfactuals are more naturally handled by comprehension systems Models & Evaluation than Formulae & Inference;
 - the implementation, based on the RMs, of the TOBI system, a comprehension systems Models & Evaluation indicating that RMs are an effective tool for treating temporal presuppositions and counterfactuals and that the dichotomy ontology-content seems reasonable.

The syntactic structure is then input to the SEM module that (recursively) transforms it in the logical form of Figure 11. Here the events (meet and leave) and the semantic roles (agent and theme) are singled out and the anaphoric references are made explicit.¹⁴ The notation should be clear, after noting that the logical form is expressed in a slot-filler notation, that **sLf** and **npLf** stand for ‘sentence logical form’ and ‘noun phrase logical form’ respectively and that **<VAR>** stands for an unspecified value.

The last data structure is the recursive model. The RM of (1) was illustrated in Figure 2. In TOBI, it is represented as the set of Prolog facts of Figure 12, where, again, the meaning should be clearly understandable, when compared with the graphic representation of Figure 2.

5 Conclusions and future work

The main points discussed in this paper are:

- the linguistic phenomena of temporal presuppositions and counterfactuals;

¹⁴I know that this is not an easy problem, but here I am not interested in it. In TOBI, anaphoric references are handled via a simple *history list* mechanism (see [2]).

From an epistemological point of view, RMs make explicit some considerations about the use of *negation* by human (or more generally living) beings (see [6, 10]). In fact, the first way that one can imagine for representing the non-existence of an object (or the non-occurrence of an event) is probably the use of a slot ‘existence’ (‘occurrence’), with the opportune value for each token. In RMs, the more general mechanism of spaces, attachments and signs allows not only to deal with existence and occurrence, but also to explicitly represent the fact that the causal relations hold between occurrences (or non-occurrence) of events, and not merely between events.

In the near future, TOBI will probably be enhanced in various ways. To extend the set of cases it can deal with, an extension of the vocabulary is needed. This, in conjunction with an improvement of the grammar, will allow for the treatment of utterances syntactically different from the ones considered in this work, but

```

sLf(meetConc(aff),asser,past,
  [slot agent:
    npLf(person,sing,f,<VAR>,slot name:mary,slot sex:f),
  slot theme:
    npLf(person,sing,m,<VAR>,slot name:john,slot sex:m),
  slot atTime(before):
    sLf(leaveConc(aff),asser,past,
      slot agent:
        npLf(person,sing,f,det,slot name:mary,slot sex:f)]]

```

Figure 11: The logical form of (1).

```

model(m1, inst(leaveConc1, leaveConc)).
model(m1, inst(person2, person)).
model(m1, inst(person1, person)).
model(m1, inst(meetConc1, meetConc)).
model(m1, instanceSlot(leaveConc1, agent, person1)).
model(m1, instanceSlot(meetConc1, theme, person2)).
model(m1, instanceSlot(person2, sex, m)).
model(m1, instanceSlot(person2, name, john)).
model(m1, instanceSlot(meetConc1, agent, person1)).
model(m1, instanceSlot(person1, sex, f)).
model(m1, instanceSlot(person1, name, mary)).
model(m1, relation(beforeTime, meetConc1, leaveConc1)).
model(m1, attach(a4, meetConc1, [])).
model(m1, attach(a3, leaveConc1, [])).
model(m1, attach(a2, person2, [])).
model(m1, attach(a1, person1, [])).
model(m1, attachSign(a4, (s4,plus,cert))).
model(m1, attachSign(a3, (s3,plus,uncert))).
model(m1, attachSign(a2, (s2,plus,uncert))).
model(m1, attachSign(a1, (s1,plus,uncert))).

```

Figure 12: TOBI's internal representation of the RM of Figure 2.

with some common semantic-pragmatic characteristics. For example, counterfactual phenomena are very common in language, and do not need a specific syntactic construction: another common case is for instance the use of the verb 'to wish', as in "Mary really wishes she had left". Also, the extensions regarding beliefs and situations illustrated in Section 3.5 will surely be considered. Finally, it is also planned to formalize the theory that underlies the RMs, on the basis of Allen's theory of action and time [3, 4], of McDermott's temporal logic [32], and of Fomichov's theory of K-calculuses and K-languages [17] using the formal specifications presented in Section 3.2.

Acknowledgements

I am indebted to Marco Colombetti for many valuable suggestions, and for long and stimulating discussions; also to Paolo Giangrandi, Elena Not, Fabio Rinaldi, and two anonymous referees, for many useful comments on an earlier draft of this paper.

References

- [1] G. Airenti and M. Colombetti. Ontology of mind, subjective ontology, and the example of temporal presuppositions. Technical Report 92-018, Dipartimento di Elettronica e Informazione, Politecnico di Milano, Milano, Italy, March 1992.

- [2] J. Allen. *Natural Language Understanding*. The Benjamin/Cummings Publishing Company, Inc., 2nd edition, 1994.
- [3] J. F. Allen. Maintaining knowledge about temporal intervals. *Communications of the ACM*, 26(11):832-843, November 1983.
- [4] J. F. Allen. Towards a general theory of action and time. *Artificial Intelligence*, 23(2):123-154, 1984.
- [5] J. Barwise and J. Perry. *Situations and attitudes*. MIT Press, Cambridge, MA, 1983.
- [6] G. Bateson. *Steps to an Ecology of Mind*. Chandler Publishing Company, 1972.
- [7] H. K. Berg, W. E. Boebert, W. R. Franta, and T. G. Moher. *Formal methods of program verification and specification*. Prentice Hall, 1982.
- [8] D. Bochvar. On three-valued logical calculus and its application to the analysis of contradictions. *Matematicheskij Sbornik*, 4:353-369, 1939.
- [9] R. J. Brachman and J. G. Schmolze. An overview of the KL-ONE knowledge representation system. *Cognitive Science*, 9(2):171-216, 1985.
- [10] M. D. S. Braine. On the relation between the natural logic of reasoning and standard logic. *Psychological Review*, 85(1):1-21, 1978.
- [11] G. Brewka. *Nonmonotonic Reasoning: Logical Foundations of Commonsense*. Cambridge University Press, Cambridge, 1991.
- [12] C. C. Chang and H. J. Keisler. *Model Theory*. Studies in Logic. North Holland, Amsterdam, 1973.
- [13] C. J. Date. *An Introduction to Database Systems*, volume 1. Addison-Wesley, Reading, MA, 5th edition, 1989.
- [14] E. Davis. *Representations of Commonsense Knowledge*. Morgan Kaufmann, San Mateo, CA, 1990.
- [15] K. Devlin. *Logic and information*. Cambridge University Press, 1991.
- [16] H. B. Enderton. *A Mathematical Introduction to Logic*. Academic Press, New York, 1972.
- [17] V. A. Fomichov. A mathematical model for describing structured items of conceptual level. *Informatica*, 20(1):5-32, 1996.
- [18] G. Gazdar. *Pragmatics: Implicature, presupposition and logical form*. Academic Press, 1979.
- [19] G. Gazdar and C. S. Mellish. *Natural Language Processing in Prolog*. Addison Wesley, Wokingham, England, 1989.
- [20] M. R. Genesereth and N. J. Nilsson. *Logical Foundations of Artificial Intelligence*. Morgan Kaufmann Publishers, Los Altos, California, 1987.
- [21] M.L. Ginsberg, editor. *Readings in non-monotonic reasoning*, Los Altos (California), 1987. Morgan Kaufmann.
- [22] B. Grosz and C. L. Sidner. Attention, intentions, and the structure of discourse. *Computational Linguistics*, 12(3):175-204, July-September 1986.
- [23] O. Heinämäki. Before. In J.N. Peranteau, P.M. and Levi and Phares G.C., editors, *Papers from the Eighth Regional Meeting of the Chicago Linguistic Society*, Chicago, Illinois, 1972. University of Chicago.
- [24] O. Heinämäki. *Semantics of English Temporal Connectives*. PhD thesis, Department of Linguistics, University of Texas at Austin, Austin, Texas, 1974. Reproduced by Indiana University, Linguistic Club, November 1978.
- [25] J. R. Hobbs and R. C. Moore, editors. *Formal Theories of the Commonsense World*. Ablex, Norwood, NJ, 1985.
- [26] H. Kamp and U. Reyle. *From Discourse to Logic*. Kluwer Academic Publisher, Dordrecht, The Netherlands, 1993.
- [27] L. Karttunen. Presuppositions of compound sentences. *Linguistic Inquiry*, 4(2):169-193, 1973.

- [28] A. Lascarides and N. Asher. Discourse relations and defeasible knowledge. In *Proceedings of 29th Annual Meeting of the Association for Computational Linguistics*, pages 55-62. University of California at Berkeley, June 1991.
- [29] A. Lascarides and J. Oberlander. Temporal connectives in a discourse context. In *Proceedings of the European ACL*, pages 260-268, 1993.
- [30] S. C. Levinson. *Pragmatics*. Cambridge University Press, Cambridge, 1983.
- [31] W. C. Mann and S. A. Thompson. Rhetorical structure theory: A theory of text organization. Reprint ISI/RS-87-190, Information Science Institutes, University of Southern California, 4676 Admiralty Way/Marina del Rey/California, June 1987.
- [32] D. McDermott. A temporal logic for reasoning about processes and plans. *Cognitive Science*, 6(2):101-155, 1982.
- [33] S. Mizzaro. TOBI - An Ontological Based Interpreter for Temporal Presuppositions and Counterfactuals. In R. Trappl, editor, *Cybernetics and Systems '94*, volume 2, pages 1879-1886, Singapore, 1994. World Scientific. Proceedings of the Twelfth European Meeting on Cybernetics and System Research, organized by the Austrian Society for Cybernetics Studies, held at the University of Vienna, Austria, 5-8 April 1994.
- [34] S. Mizzaro. Recursive models: A computational tool for the semantics of temporal presuppositions in natural language. In *Proceedings of the fifth meeting of the AI*IA (Italian Association for Artificial Intelligence), Workshop 'Natural Language Processing'*, pages 43-46, Naples, Italy, 26-28 September 1996.
- [35] L. Polanyi. A formal model of the structure of discourse. *Journal of Pragmatics*, 12:601-638, 1988.
- [36] H. Reichenbach. *Elements of Symbolic Logic*. Macmillan, London, 1947.
- [37] R. Scha, B. Bruce, and L. Polanyi. Discourse understanding. Technical Report 391, Bolt Beranek and Newman Inc., 10 Moulton Street, Cambridge, MA, October 1986.
- [38] P. Smolensky. On the proper treatment of connectionism. *Behavioral and Brain Science*, 11(1):1-74, 1988.
- [39] I. Sommerville. *Software Engineering*. Addison-Wesley, Reading, MA, 3rd edition, 1989.
- [40] R. Turner. *Logics for artificial intelligence*. Ellis Horwood Limited, Chichester, England, 1984.
- [41] B. L. Webber. Discourse deixis and discourse processing. Technical Report MS-CIS-88-75, Department of Computer and Information Science, University of Pennsylvania, Philadelphia, PA 19014, September 1988.
- [42] T. Winograd. *Language as a cognitive process*, volume 1: Syntax. Addison-Wesley, Reading, MA, 1983.

Informational Graphs

Anton P. Železnikar
 An Active Member of the New York Academy of Sciences
 Volaričeva ulica 8
 SI-1111 Ljubljana, Slovenia
 Email: anton.p.zeleznikar@ijs.si

Keywords: graph definition; informational gestalt, graph, operand, operator, star gestalt, transition; parenthesis pairs; primitive parallel formula system; serial-circular formula system

Edited by: Jiří Šlechta

Received: August 6, 1996

Revised: December 3, 1996

Accepted: December 17, 1996

Informational graph seems to be one of the most basic underlying structures (circuits [1, 4], frontal lobes functions [10], schemata [15], impressions, etc.) for the concept and possibilities of informing and its understanding. The graph behaves as a regular parallel informational system of formulas (entities) with its own possibilities of informational spontaneity and circularity. By means of informational graph, it is possible to explain the origin of the so-called informational gestalt and, besides, the arising of informational formulas especially concerning the so-called causality in regard to the position of the formula parenthesis pairs. Another view of the graph lies in the moving along the arrows in the graph, that is, a formula construction, when choosing a path and setting parenthesis pairs in the emerging well-formed formula, in a spontaneous and circular way. This approach, together with the arising of the graph itself, can represent one of the keystones of the informational arising of formulas, the vanishing of their parts, and the changing of the structure during the informational moving through the graph. The paper shows how the informational graph can be understood by the phenomenalism of informational gestalts exerting the causal possibilities of formulas with the same length but differently displaced parenthesis pairs. Several examples are formalized.

1 Introduction

Informational graph¹ is a graphical imitation (informational presentation, and circuits [4], frontal lobes functions [10], schemata [15] in the neurological sense) of a serial informational formula system (a system of parallel serial formulas) or also a parallel system of basic (atomic) informational transitions (without any parentheses pairs). Informational graph performs like an imprint along which different formula interpretations are possible. In this sense, an informational graph preserves the sequence (direction) of the occurring informational operands and operators, but does not consider (that is, ignores) the parenthesis pairs of

original (initial) formulas. As such, it appears as a schematic pattern of operands and operators, in which the user can set parenthesis pairs spontaneously, getting an arbitrary causal dependence of operands (informational entities) within the graph's pattern².

Informational graphs can be comprehended as generalizations of informational formulas (as parallel systems of certain serial formulas) from which various formula reconstructions are possible, the number of which depends on the involved informational operators, that is, on the formula length. The number of possible graph interpre-

¹This paper is a private author's work and no part of it may be used, reproduced or translated in any manner whatsoever without written permission except in the case of brief quotations embodied in critical articles.

²Informational phenomenalism joins the terms representing phenomenology, ontology and causation in regard to an informational entity. An informational operator \models exerts an existential (Being-like) as well as causal property of the operand(s), to which the operator belongs (connects them).

tations by formulas grows rapidly by the number of the occurring binary operators in the formula as we shall show in the study which follows. We shall learn also in which way the most rational (unique) formal description of an informational graph is possible and how interpretations of the circular graphs by circular formulas can carry a substantial degree of expressional redundancy.

On the other side we have to determine a set of new concepts concerning graphs and their interpretations especially by *informational gestalts*. It has to be answered rigorously what does an informational graph represent and how can it be used for the generation (emerging) of different formula interpretations. For the sake of the understanding clarity we can introduce special primitive graphical symbols by which graphs of any complexity can be presented in the form of graphical sketches (schemes, circuits). For example, complex circular informational graphs can be studied from the different points of view. On one side, such a graph appears as a relatively clear picture to the user; on the second side it can be described formally in the most rational form by a parallel system of the primitive informational transitions; on the third side, it can be expressed by a parallel system of arbitrary serial and circularly serial formulas, considering only those of them which in a given case represent the reality (rationality) of the problem.

An informational graph represents all possible interpretations which number depends solely on the involved binary operators. Additionally, as interpretations of the graphs, the so-called star gestalts of graphs can be introduced which illustrate the moving through the graph from an initial operand, constructing serial and circularly serial formulas of different lengths, for example from the length $\ell = 1$ (basic or atomic transition) on to an arbitrary length $\ell = n$. Thus, systems of formulas belonging to a circular star gestalt from a circular graph can be constructed (by a parenthesizing) in an arbitrarily lasting way.

Informational graphs are visual (texts, images), acoustic (voices, music, noise), tactile (Braille script), taste (food evaluation), smelling (perfume competition), etc. They are static and dynamic. Some practical examples of static informational graphs are literature texts, artistic pictures, drawings, music notes, etc. Examples of dynamic informational graphs are theatre performance, TV

and radio transmissions (mixed visual and acoustic), everyday happenings in characteristic situations (common patterns of behavior and uncommon reactions), etc. For all these graphs it is typical that they are not ‘parenthesized’. Understanding of the mentioned phenomena (together with Parenthesizing, causation, interpretation) is left to the observer. It means that the causal structuring of a graph belongs to the domain of the observer and that different observers can causally-differently structure one and the same static or dynamic informational graph.

2 Elements of Informational Graphs

Elements of informational graphs are circles (or ovals, if their informational markers are longer or complex) and arrows (vectors). Parallel and alternative input and output buses contribute only to the compactness of graphs. Circles (operand atoms) are marked by operands which represent informational entities in informational formulas. Arrows (operator atoms) are marked and unmarked. Unmarked arrows represent the most general operator \models and are used also in situations where all of the unmarked arrows belong to one and the same operator, e.g., to the implication operator \implies in the global implication axioms of Hilbert [8]. The symbolic atoms of informational graphs are presented in Fig. 1.

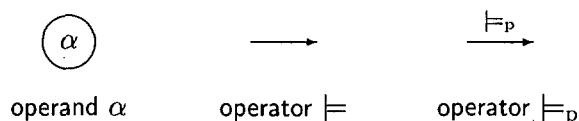


Figure 1: *Graphical symbols for informational operands and operators (atomic elements).*

The symbols in Fig. 1 can be connected in the form of a graph representing a formula or formula system, but without parenthesis pairs. In this way the order of operands and operators remains preserved (exactly in cases of a single formula, and to some extent in cases of formula systems).

In Fig. 2 some elementary graphs for the most basic phenomenal occurrences of an informational entity α are presented. These occurrences are

(1) operand (entity) α as such,

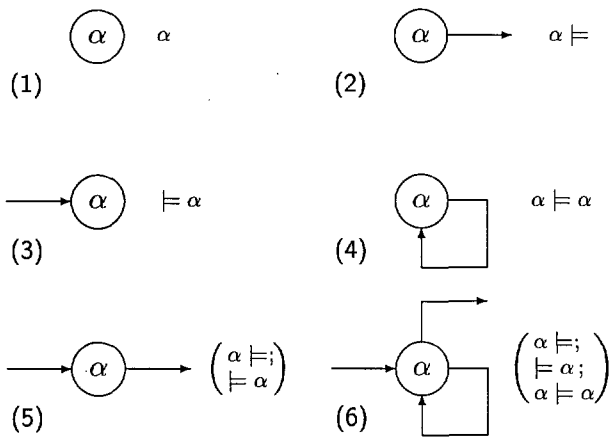


Figure 2: Informational graphs for the basic (phenomenalistically structured) informational formulas.

- (2) α 's externalism $\alpha \models$ (informing for others),
- (3) α 's internalism $\models \alpha$ (informing for itself),
- (4) α 's metaphysicalism $\alpha \models \alpha$ (informing in or within itself),
- (5) α 's phenomenalism $(\alpha \models; \models \alpha)$ (informing as such), and
- (6) for the sake of clarity, the entire *modus agendi*³ of an informing entity α , that is, its phenomenalistic and simultaneously metaphysicalistic informing as a system of $(\alpha \models; \models \alpha; \alpha \models \alpha)$.

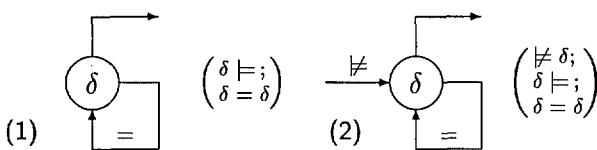


Figure 3: Informational graphs representing (phenomenalistically) the structure of data δ .

Data, marked by δ , is a characteristic informational entity which can be graphically presented by a particularized graph being essentially different from the *modus agendi* belonging to a general informational entity, marked by α . The data

³Modus agendi of an informational entity, exerting an entity characteristic internalism, externalism, and metaphysicalism, has the potentiality to be compared with the Self behaving self-sensitive and self-active in regard to its environment (exterior) and interior (the subject regarded as the object of its own activity).

graph is sketched in Fig. 3, where the feedback loop is marked by the equality operator '=', realizing the metaphysical situation $\delta = \delta$, belonging to data. In case (2) one can introduce also the explicit operator of non-informedness of data δ , that is $\not\models \delta$. In this case, the non-informedness of δ means the absence or impossibility of any exterior or interior impact on data δ .

In the case of input and/or output parallelism concerning an informational entity α , we introduce, for the sake of graphical transparency, a kind of input and output lines (parallel buses), respectively. Some specific cases are presented in Fig. 4.

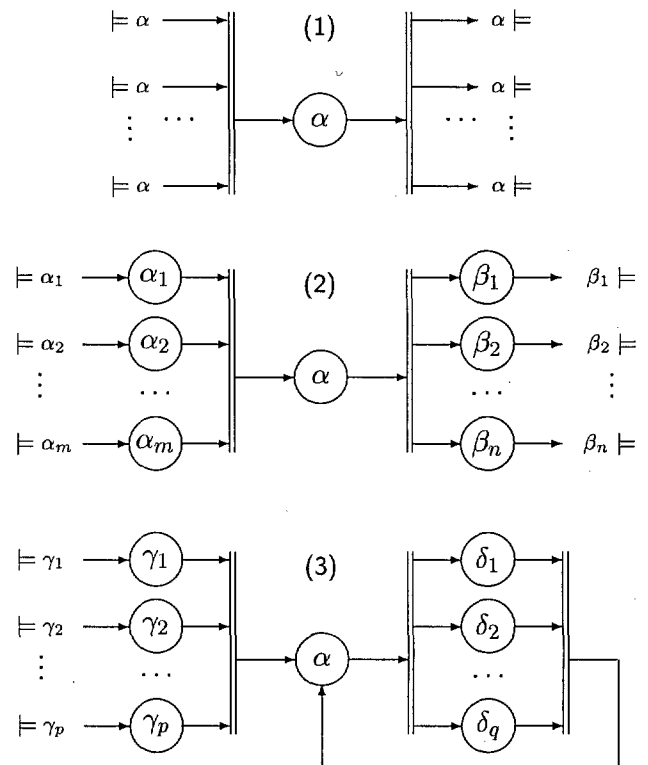


Figure 4: Informational graphs representing characteristic cases of informational parallelism, where vertical double-bars (||) are parallel input and output collection lines of operand (entity) α .

The graph in Fig. 4, case (1), shows how the parallel decomposition of operand α might be sensible, that is, in the form of implication

$$\alpha \Rightarrow \left(\left(\begin{array}{c} \models \alpha; \\ \models \alpha; \\ \vdots \\ \models \alpha \end{array} \right); \left(\begin{array}{c} \alpha \models; \\ \alpha \models; \\ \vdots \\ \alpha \models \end{array} \right) \right)$$

which expresses the α 's unlimited input (internalistic) and output (externalistic) potentiality. The unique description of the graph is presented by a system of the incomplete basic transitions, that is, $\alpha \rightleftharpoons (\models \alpha; \models \alpha; \dots; \models \alpha; \alpha \models; \alpha \models; \dots; \alpha \models)$.

Case (2) in Fig. 4 can be described formally as

$$\left(\left(\begin{array}{c} \models \alpha_1; \\ \models \alpha_2; \\ \vdots \\ \models \alpha_m \end{array} \right) \models \alpha \right) \models \left(\begin{array}{c} \beta_1 \models; \\ \beta_2 \models; \\ \vdots \\ \beta_n \models \end{array} \right)$$

But, as we see looking at the graph, there exists still another formula interpretation of the graph, in the form

$$\left(\begin{array}{c} \models \alpha_1; \\ \models \alpha_2; \\ \vdots \\ \models \alpha_m \end{array} \right) \models \left(\alpha \models \left(\begin{array}{c} \beta_1 \models; \\ \beta_2 \models; \\ \vdots \\ \beta_n \models \end{array} \right) \right)$$

The unique description of the graph by elementary transitions is in the form of the parallel formula system, which is

$$\left(\begin{array}{cccc} \models \alpha_1; & \models \alpha_2; & \dots; & \models \alpha_m; \\ \alpha_1 \models \alpha; & \alpha_2 \models \alpha; & \dots; & \alpha_m \models \alpha; \\ \alpha \models \beta_1; & \alpha \models \beta_2; & \dots; & \alpha \models \beta_n; \\ \beta_1 \models; & \beta_2 \models; & \dots; & \beta_n \models \end{array} \right)$$

Case (3) in Fig. 4 includes a loop of parallel operands and a possible formal description of this graph is

$$\left(\begin{array}{c} \models \gamma_1; \\ \models \gamma_2; \\ \vdots \\ \models \gamma_p \end{array} \right) \models \left(\left(\alpha \models \left(\begin{array}{c} \delta_1 \models; \\ \delta_2 \models; \\ \vdots \\ \delta_q \models \end{array} \right) \right) \right) \models \alpha$$

The reader can guess in which way still four other interpretations of the graph (3) in Fig. 4 are possible. On the other side, the very unique graph description, including all possible graph interpretations, is the parallel formula system

$$\left(\begin{array}{cccc} \models \gamma_1; & \models \gamma_2; & \dots; & \models \gamma_p; \\ \gamma_1 \models \alpha; & \gamma_2 \models \alpha; & \dots; & \gamma_p \models \alpha; \\ \alpha \models \delta_1; & \alpha \models \delta_2; & \dots; & \alpha \models \delta_q; \\ \delta_1 \models \alpha; & \delta_2 \models \alpha; & \dots; & \delta_q \models \alpha \end{array} \right)$$

Let us show two examples which explain the graphical presentation of parallelism.

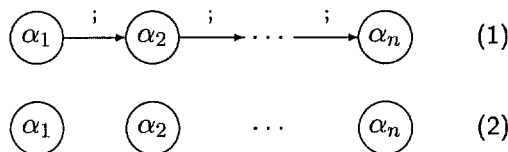


Figure 5: Graphically, the parallel connection of operands $\alpha_1, \alpha_2, \dots, \alpha_n$ (1) by semicolons (operators of informational parallelism, \models) is replaced by isolated parallel operands (2).

Case (1) in Fig. 5 shows the graphical presentation of a strictly, that is operationally, parallel structured formula

$$(\dots (\alpha_1 \models \alpha_2) \models \dots \alpha_{n-1}) \models \alpha_n$$

(also, with any other distribution of parenthesis pairs) which is replaced with the common semicolon (symbol of parallelism) formula

$$\alpha_1; \alpha_2; \dots; \alpha_n$$

Case (2) in Fig. 5 shows this semicolon-simplified formula example graphically.

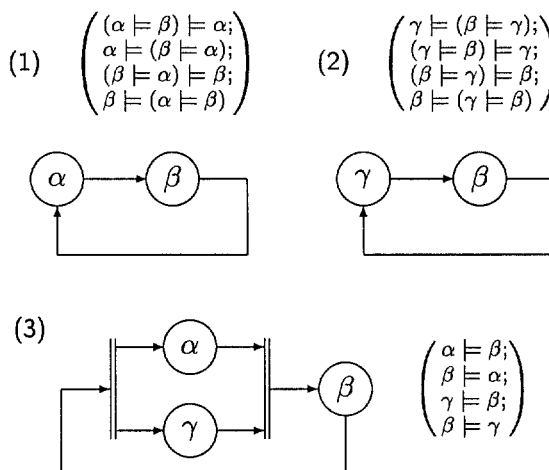


Figure 6: The joining of two serial graphs given by formula systems (1) and (2) into a unique serial-parallel graph determined by the adequate formula system (3).

Fig. 6 shows in which way two circular serial graphs (1) and (2) can be joined into the circular serial-parallel graph (3). In this way the graphical transparency of the resulting formula system is essentially improved. The serially circular formula system, being parallelized by joining graphs (1) and (2), means a (graphically) equivalent form

$$\left(\begin{array}{l} \alpha \models \beta; \\ \beta \models \alpha; \\ \gamma \models \beta; \\ \beta \models \gamma \end{array} \right) \equiv \left(\begin{array}{l} \left(\left(\begin{array}{l} \alpha; \\ \gamma \end{array} \right) \models \beta \right) \models \left(\begin{array}{l} \alpha; \\ \gamma \end{array} \right); \\ \left(\begin{array}{l} \alpha; \\ \gamma \end{array} \right) \models \left(\beta \models \left(\begin{array}{l} \alpha; \\ \gamma \end{array} \right) \right); \\ \left(\beta \models \left(\begin{array}{l} \alpha; \\ \gamma \end{array} \right) \right) \models \beta; \\ \beta \models \left(\left(\begin{array}{l} \alpha; \\ \gamma \end{array} \right) \models \beta \right) \end{array} \right)$$

This formula describes the joined systems (1) and (2) in Fig. 6 (compare to the eight formulas). There exists a sort of meaning equivalence (operator \equiv) between the transition formula system describing to the graph (3) in Fig. 6 and the serial-parallel formula system on the right side of operator \equiv .

Similar role as the parallel informational operator \models , that is semicolon ‘;’, has the alternative informational operator $\models_{\text{alternatively_to}}$, that is comma ‘,’. In an informational graph we use a specially marked alternative bus (instead of parallel bus) to distinguish the particular case of alternativeness. For instance, if in cases of Fig. 6, all semicolons are replaced by commas, the alternative situation is presented in Fig. 7.

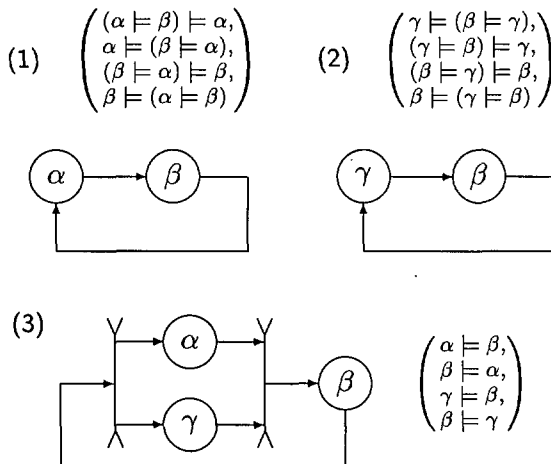


Figure 7: The joining of two serial graphs given by formula systems (1) and (2) into a unique serial-alternative graph determined by the adequate formula system (3).

After the discussion by examples we have to answer the question, what does an informational

graph mean at all, what is its informational representing potentiality?

3 Informational Graph as an Informational Entity

After the introductory interpretation concerning the informational graph on one side and the informational formula (also formula system) phenomenalism on the other side we have to construct a precise definition of the informational graph and its meaning, and deduce some consequences of this definition. Further, how can an informational graph arise informationally? To answer this question we must determine which kind of informational entity or even entities (operands, formulas) does the informational graph represent.

DEFINITION 1 (A General Graph Determination)
An informational graph is a connection of informational operands (entities) and informational operators (entities' informational properties) not containing the parenthesis pairs of a formula. *Operands* are marked circles or ovals representing arbitrary informational entities. *Operators* are unmarked or marked arrows (vectors) representing adequate informational properties of entities which they connect. An *informational graph* as a structure of connected and unconnected (isolated) graphical elements can represent any possible well-formed informational formulas—serial, parallel, and circular—which can be constructed by a consequent use of the parentheses pairs. \square

DEFINITION 2 (A General Primitive Parallel Formula System) A general primitive parallel formula system (GPPFS, for short), marked by φ'_{\parallel} , is a parallel sequence (system) of the most simple internalistic, externalistic, transitional (serial, serially circular) formulas of length $\ell = 1$ (having a single operator) and unconnected (marker) formulas of length $\ell = 0$, concerning the informationally involved operands. Such primitive formulas are, for example,

- $\models \iota$; [internalistic (input) formula]
- $o \models$; [externalistic (output) formula]
- $\tau_1 \models \tau_2$; [transition formula as a serial or serially circular formula]
- v ; [unconnected marker formula]

$$\left. \begin{array}{l} \iota, o, \\ \tau_1, \tau_2, \\ v \end{array} \right\} \in \mathcal{P} \quad [\text{primitive operands domain}]$$

where set \mathcal{P} represents all possible primitive (the most elementary) operands or operand markers. Thus, for example,

$$\varphi'_{\parallel}(\iota, o, \tau_1, \tau_2, v) \Rightarrow \left(\begin{array}{l} \models \iota; o \models; \\ \iota \models \tau_1; \tau_1 \models \tau_2; \\ \tau_2 \models o; v \end{array} \right)$$

marks a general form of GPPFS. \square

According to the last definition we can represent an informational graph graphically, as already done in the previous section in Fig. 2, or also symbolically, that is by formulas or formula systems, using an appropriate and unique (well-formed) informational symbolism. However, there exists only one form for unique graph representation by informational formulas without parenthesis pairs.

THEOREM 1 (A Graph Interpretation by the General Primitive Parallel Formula System) *An arbitrary informational graph (drawing), being formally marked by*

$$\mathfrak{G}(\varphi'_{\parallel}(\gamma_1, \gamma_2, \dots, \gamma_{n_\gamma}; v_1, v_2, \dots, v_{n_v}))$$

concerning the connected operands $\gamma_1, \gamma_2, \dots, \gamma_{n_\gamma}$ and the unconnected (informationally isolated) operands v_1, v_2, \dots, v_{n_v} , can uniquely be described by the primitive parallel formula system φ'_{\parallel} of the form

$$\varphi'_{\parallel}(\gamma_1, \gamma_2, \dots, \gamma_{n_\gamma}, v_1, v_2, \dots, v_{n_v}) \Rightarrow \left(\begin{array}{l} \models \gamma_i; \gamma_j \models; \gamma_p \models \gamma_q; \\ \gamma_i, \gamma_j, \gamma_p, \gamma_q \in \{\gamma_1, \gamma_2, \dots, \gamma_{n_\gamma}\}; \\ v_1; v_2; \dots; v_{n_v} \end{array} \right)$$

Preliminarily unconnected operands $v_1; v_2; \dots; v_{n_v}$ function as entities which could become connected in the course of further decomposition of the system φ'_{\parallel} . \square

Proof 1 (Graph \mathfrak{G} and Formula System φ'_{\parallel}) Each graph \mathfrak{G} can completely be described by the adequate formula system φ'_{\parallel} . The proof of the theorem is the following. For any two by the arrow

connected marked circles (or ovals) in the direction from γ_p to γ_q , there is, evidently, $\gamma_p \models \gamma_q$, where the arrow represents the operator \models , being marked or unmarked. For each arrow in the graph, there is exactly one basic transitional formula. In this way, all operator connections of the graph can be formalized. In principle, a graph can include also unconnected (isolated) operands which are not connected elsewhere. The input arrows leading to some operand markers are formalized in the form $\models \gamma_i$, while output arrows leading from some operands have the form $\gamma_j \models$. This completes the proof of the theorem. \square

The next question we have to solve is what an informational entity, that is, formula or formula system, does an informational graph, formally determined in Theorem 1, imply. We have to prove the informational transformation possibilities of the parallel system in the previous theorem. Evidently, a parallel informational system hides the power of another parallelism of serial and circular structure which has a *causal* origin.

PRESUMPTION 1 (Interpretation Possibilities of the Graph) An informational graph represents a parallel system of serial and serial-circular formulas of different lengths in which operands can be variously interwoven and the parenthesis pairs in the occurring formulas can be arbitrarily displaced. This means that the parallel system of the most basic transition formulas, representing the graph (as in Theorem 1), can be informationally transformed in a causally more transparent and for the human observer more understandable parallel system of serial and/or circular-serial formulas of different lengths. In this way transformed formula system brings to the surface the explicit possibilities of a causal structure of the occurring formulas which, then, can be chosen as the best fitting ones for the description of an informational situation and attitude. \square

Serial and circularly serial formulas will be superscribed by the formula length n and $n + 1$, respectively, and subscripted by the formula index i in an interval $1, 2, \dots, N_{\downarrow}$ and $1, 2, \dots, N_{\downarrow}^{\circ}$, respectively, where N_{\downarrow} and N_{\downarrow}° , respectively, the so-called *numeri causae* (causal numbers), will depend on the length of a serial and circularly serial formula. Usually, subscript 1 will be given to a serial formula

$${}^n_1\varphi_{\rightarrow}(\sigma, \sigma_1, \sigma_2, \dots, \sigma_{n-1}, \sigma_n) \equiv ((\dots((\sigma \models \sigma_1) \models \sigma_2) \models \dots \sigma_{n-1}) \models \sigma_n)$$

where $\sigma, \sigma_1, \sigma_2, \dots, \sigma_{n-1}, \sigma_n$ are serial operands, and to a circular formula

$${}^{n+1}_1\varphi_{\rightarrow}^{\circ}(\omega, \omega_1, \omega_2, \dots, \omega_{n-1}, \omega_n) \equiv (((\dots((\omega \models \omega_1) \models \omega_2) \models \dots \omega_{n-1}) \models \omega_n) \models \omega)$$

where $(\omega, \omega_1, \omega_2, \dots, \omega_{n-1}, \omega_n)$ are circular serial operands. The subscripting proceeds consequently (systematically) from 1 to $N_{\rightarrow}^{\rightarrow}$ and 1 to N_{\rightarrow}° , respectively.

DEFINITION 3 (A Serial Formula and Its Gestalt) By a *serial* formula ${}^n_i\varphi_{\rightarrow}$ let us mark any well-formed arrangement of the operands, operators and parenthesis pairs, for which

$${}^n_i\varphi_{\rightarrow}(\sigma, \sigma_1, \dots, \sigma_{n-1}, \sigma_n) \in \left\{ \begin{array}{l} {}^n_1\varphi_{\rightarrow}(\sigma, \sigma_1, \dots, \sigma_{n-1}, \sigma_n), \\ {}^n_2\varphi_{\rightarrow}(\sigma, \sigma_1, \dots, \sigma_{n-1}, \sigma_n), \\ \dots, \\ {}^n_{N_{\rightarrow}}\varphi_{\rightarrow}(\sigma, \sigma_1, \dots, \sigma_{n-1}, \sigma_n) \end{array} \right\};$$

$$i \in \{1, 2, \dots, N_{\rightarrow}\}; N_{\rightarrow} = \frac{1}{n+1} \binom{2n}{n};$$

$$\Gamma({}^n_i\varphi_{\rightarrow}(\sigma, \sigma_1, \dots, \sigma_{n-1}, \sigma_n)) \equiv \left(\begin{array}{l} {}^n_1\varphi_{\rightarrow}(\sigma, \sigma_1, \dots, \sigma_{n-1}, \sigma_n); \\ {}^n_2\varphi_{\rightarrow}(\sigma, \sigma_1, \dots, \sigma_{n-1}, \sigma_n); \\ \dots \\ {}^n_{N_{\rightarrow}}\varphi_{\rightarrow}(\sigma, \sigma_1, \dots, \sigma_{n-1}, \sigma_n) \end{array} \right)$$

where n is the length of serial formula being equal to the number of binary operators \models in the formula, i is a systematic (causal) numbering of serial formulas of length n , N_{\rightarrow} is the number of all possible serial formulas obtained from a serial formula by the displacements of the parenthesis pairs, and $\Gamma({}^n_i\varphi_{\rightarrow}(\sigma, \sigma_1, \dots, \sigma_{n-1}, \sigma_n))$ is the so-called gestalt of the serial formula representing the parallel system of all formulas of length n obtained from the original formula by all possible displacements of the parenthesis pairs in the original formula. As evident, operands and operators remain on the initially fixed positions, according to the original (initial) formula ${}^n_i\varphi_{\rightarrow}(\sigma, \sigma_1, \dots, \sigma_{n-1}, \sigma_n)$. \square

DEFINITION 4 (A Circularly Serial Formula and Its Gestalt) By a *circularly serial* informational formula ${}^{n+1}_j\varphi_{\rightarrow}^{\circ}$ let us mark any serially and circularly well-formed arrangement of the operands, operators and parenthesis pairs, for which

$${}^{n+1}_j\varphi_{\rightarrow}^{\circ}(\omega, \omega_1, \dots, \omega_{n-1}, \omega_n) \in \left\{ \begin{array}{l} {}^{n+1}_1\varphi_{\rightarrow}^{\circ}(\omega, \omega_1, \dots, \omega_{n-1}, \omega_n), \\ {}^{n+1}_2\varphi_{\rightarrow}^{\circ}(\omega, \omega_1, \dots, \omega_{n-1}, \omega_n), \\ \dots, \\ {}^{n+1}_{N_{\rightarrow}^{\circ}}\varphi_{\rightarrow}^{\circ}(\omega, \omega_1, \dots, \omega_{n-1}, \omega_n) \end{array} \right\};$$

$$j \in \{1, 2, \dots, N_{\rightarrow}^{\circ}\}; N_{\rightarrow}^{\circ} = \frac{1}{n+2} \binom{2n+2}{n+1};$$

$$\Gamma({}^{n+1}_j\varphi_{\rightarrow}^{\circ}(\omega, \omega_1, \dots, \omega_{n-1}, \omega_n)) \equiv \left(\begin{array}{l} {}^{n+1}_1\varphi_{\rightarrow}^{\circ}(\omega, \omega_1, \dots, \omega_{n-1}, \omega_n); \\ {}^{n+1}_2\varphi_{\rightarrow}^{\circ}(\omega, \omega_1, \dots, \omega_{n-1}, \omega_n); \\ \dots \\ {}^{n+1}_{N_{\rightarrow}^{\circ}}\varphi_{\rightarrow}^{\circ}(\omega, \omega_1, \dots, \omega_{n-1}, \omega_n) \end{array} \right)$$

where circularity concerns the first (main) operand, that is, ω and the parenthesis pairs can be arbitrarily displaced in another way, according to the gestalt of the formula.

The difference between formulas ${}^n_i\varphi_{\rightarrow}(\sigma, \sigma_1, \sigma_2, \dots, \sigma_{n-1}, \sigma_n)$ and ${}^{n+1}_j\varphi_{\rightarrow}^{\circ}(\omega, \omega_1, \omega_2, \dots, \omega_{n-1}, \omega_n)$ is in their lengths ℓ_{\rightarrow} and $\ell_{\rightarrow}^{\circ}$, that is, $\ell_{\rightarrow}^{\circ} = \ell_{\rightarrow} + 1$. Thus,

– gestalt $\Gamma({}^n_i\varphi_{\rightarrow}(\sigma, \sigma_1, \sigma_2, \dots, \sigma_{n-1}, \sigma_n))$ includes

$$\frac{1}{\ell_{\rightarrow} + 1} \binom{2\ell_{\rightarrow}}{\ell_{\rightarrow}}$$

formulas of length ℓ_{\rightarrow} ;

– gestalt $\Gamma({}^{n+1}_j\varphi_{\rightarrow}^{\circ}(\omega, \omega_1, \omega_2, \dots, \omega_{n-1}, \omega_n))$ of a circular formula includes, analogously to the serial structure of its circularity,

$$\frac{1}{\ell_{\rightarrow}^{\circ} + 1} \binom{2\ell_{\rightarrow}^{\circ}}{\ell_{\rightarrow}^{\circ}}$$

formulas of length $\ell_{\rightarrow}^{\circ}$ formulas of length $\ell_{\rightarrow}^{\circ} = \ell_{\rightarrow} + 1$; and

– the so-called circular gestalt of the form $\Gamma^\circ \left({}^{n+1}_j \varphi_{\rightarrow}^\circ(\omega, \omega_1, \omega_2, \dots, \omega_{n-1}, \omega_n) \right)$ includes, considering all operands in the circular structure (loop, cycle),

$$\frac{\ell_{\rightarrow}^\circ}{\ell_{\rightarrow}^\circ + 1} \binom{2\ell_{\rightarrow}^\circ}{\ell_{\rightarrow}^\circ}$$

formulas of length $\ell_{\rightarrow}^\circ = \ell_{\rightarrow} + 1$.

In this respect, for the gestalts of a serial and circularly serial formula, and for the circular gestalt of a circularly serial formula, the corresponding numeri causae are the following:

$$N_{\rightarrow} = \frac{1}{\ell_{\rightarrow} + 1} \binom{2\ell_{\rightarrow}}{\ell_{\rightarrow}} = \frac{1}{n+1} \binom{2n}{n};$$

$$N_{\rightarrow}^\circ = \frac{1}{\ell_{\rightarrow}^\circ + 1} \binom{2\ell_{\rightarrow}^\circ}{\ell_{\rightarrow}^\circ} = \frac{1}{n+2} \binom{2n+2}{n+1};$$

$${}^\circ N_{\rightarrow}^\circ = \frac{\ell_{\rightarrow}^\circ}{\ell_{\rightarrow}^\circ + 1} \binom{2\ell_{\rightarrow}^\circ}{\ell_{\rightarrow}^\circ} = \frac{n+1}{n+2} \binom{2n+2}{n+1}$$

In this way, the so-called *causal difference* between the complex circular and the pure serial case is, expressed by the running subscript n (being equal in all three cases),

$$\frac{n+1}{n+2} \binom{2n+2}{n+1} - \frac{1}{n} \binom{2n}{n}$$

Causal possibilities are essentially different in a pure serial and in a complex circular case. \square

THEOREM 2 (Transparency of the Operand Circularity in a Circular Formula) *The circularity of operand ω in a serially circular formula ${}^{n+1}_j \varphi_{\rightarrow}^\circ(\omega, \omega_1, \omega_2, \dots, \omega_{n-1}, \omega_n)$ implies the circularity of the remaining operands $\omega_1, \omega_2, \dots, \omega_n$, that is,*

$${}^{n+1}_j \varphi_{\rightarrow}^\circ(\omega, \omega_1, \omega_2, \dots, \omega_{n-1}, \omega_n) \implies \left(\begin{array}{l} {}^{n+1}_{j_1} \varphi_{\rightarrow}^\circ(\omega_1, \omega_2, \dots, \omega_n, \omega); \\ {}^{n+1}_{j_2} \varphi_{\rightarrow}^\circ(\omega_2, \omega_3, \dots, \omega_n, \omega, \omega_1); \\ \vdots \\ {}^{n+1}_{j_n} \varphi_{\rightarrow}^\circ(\omega_n, \omega, \omega_1, \dots, \omega_{n-2}, \omega_{n-1}) \end{array} \right)$$

In a serially circular formula, the circularity concerns all operands, that is, $\omega, \omega_1, \omega_2, \dots, \omega_{n-1}, \omega_n$. \square

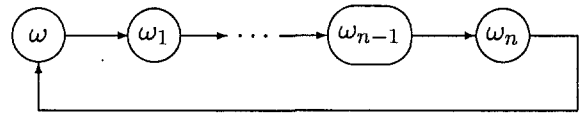


Figure 8: *The serially circular graph representing $\frac{n+1}{n+2} \binom{2n+2}{n+1}$ serially circular formulas of length $\ell = n + 1$.*

Proof 2 (Circular Informing of Operands in a Circular Formula) For a serially circular formula

$${}^{n+1}_1 \varphi_{\rightarrow}^\circ(\omega, \omega_1, \dots, \omega_{n-1}, \omega_n) \implies ((\dots(\omega \models \omega_1) \models \dots \omega_{n-1}) \models \omega_n)$$

we can construct the circular graph in Fig. 8. In the loop of the graph, for the remaining operands $\omega_1, \dots, \omega_{n-1}, \omega_n$, the existence of the serially circular formulas (viewed as by the parenthesis-pairs-displaced specific functions of $n + 1$ operands)

$$\begin{array}{l} {}^{n+1}_{j_1} \varphi_{\rightarrow}^\circ(\omega_1, \omega_2, \dots, \omega_n, \omega); \\ {}^{n+1}_{j_2} \varphi_{\rightarrow}^\circ(\omega_2, \omega_3, \dots, \omega_n, \omega, \omega_1); \\ \vdots \\ {}^{n+1}_{j_n} \varphi_{\rightarrow}^\circ(\omega_n, \omega, \omega_1, \dots, \omega_{n-2}, \omega_{n-1}) \end{array}$$

is evident. This proves the validity of the theorem. \square

THEOREM 3 (A Graph Interpretation by the Circularly Serial Formula System) *An informational graph (drawing in Fig. 9), being formally marked by*

$$\mathfrak{G} \left\| \Phi_{\rightarrow}^\circ \left(\begin{array}{l} \iota_0, \iota_1, \iota_2, \dots, \iota_{n_\iota}; \\ o_0, o_1, o_2, \dots, o_{n_o}; \\ \sigma_{p0}, \sigma_{p1}, \sigma_{p2}, \dots, \sigma_{pm_p}; \\ p = 0, 1, \dots, A; \\ \omega_{q0}, \omega_{q1}, \omega_{q2}, \dots, \omega_{qn_q}; \\ q = 0, 1, \dots, B; \\ v_0, v_1, v_2, \dots, v_{n_v} \end{array} \right) \right.$$

concerns the systematically grouped and also overlapping

- input (*internalistic*) operands, marked by $\mathbb{I} = \{\iota_0, \iota_1, \iota_2, \dots, \iota_{n_\iota}\}$,
- the output (*externalistic*) operands, symbolized by $\mathbb{O} = \{o_0, o_1, o_2, \dots, o_{n_o}\}$,

- the A groups of serial (transitional) formula operands, denoted by $\mathbb{S} = \{\sigma_{p0}, \sigma_{p1}, \sigma_{p2}, \dots, \sigma_{pm_p} \mid p = 0, 1, \dots, A\}$,
- the B groups of circularly serial formula operands, represented by $\Omega = \{\omega_{q0}, \omega_{q1}, \dots, \omega_{qn_q} \mid q = 0, 1, \dots, B\}$, and
- the unconnected (informationally isolated) operands, $\mathbb{U} = \{v_0, v_1, v_2, \dots, v_{n_v}\}$.

The unconnected operands do not overlap with other operands and are foreseen as those which could become informationally involved. The integral gestalt, being composed of the respective partial gestalts, is a formula

$$\Gamma \left(\underset{\parallel \Phi \rightarrow}{\left(\begin{array}{l} \iota_0, \iota_1, \iota_2, \dots, \iota_{n_\iota}; \\ o_0, o_1, o_2, \dots, o_{n_o}; \\ \sigma_{p0}, \sigma_{p1}, \sigma_{p2}, \dots, \sigma_{pm_p}; \\ p = 0, 1, \dots, A; \\ \omega_{q0}, \omega_{q1}, \omega_{q2}, \dots, \omega_{qn_q}; \\ q = 0, 1, \dots, B; \\ v_0, v_1, v_2, \dots, v_{n_v} \end{array} \right)} \right) \equiv$$

$$\left(\begin{array}{l} \Gamma(\models \iota_0; \models \iota_1; \models \iota_2; \dots; \models \iota_{n_\iota}); \\ \Gamma(o_0 \models; o_1 \models; o_2 \models; \dots; o_{n_o} \models); \\ \Gamma \left(\underset{i_p \rightarrow}{\overset{m_p}{\varphi}} (\sigma_{p0}, \sigma_{p1}, \dots, \sigma_{pm_p}) \right); \\ p = 0, 1, \dots, A; \\ \Gamma \left(\underset{j_{q0} \rightarrow}{\overset{n_q+1}{\varphi}} (\omega_{q0}, \omega_{q1}, \omega_{q2}, \dots, \omega_{qn_q}) \right); \\ \Gamma \left(\underset{j_{q1} \rightarrow}{\overset{n_q+1}{\varphi}} (\omega_{q1}, \omega_{q2}, \dots, \omega_{qn_q}, \omega_{q0}) \right); \\ \dots; \\ \Gamma \left(\underset{j_{qn_q} \rightarrow}{\overset{n_q+1}{\varphi}} (\omega_{qn_q}, \omega_{q0}, \omega_{q1}, \dots, \omega_{qn_{v-1}}) \right); \\ q = 0, 1, \dots, B; \\ \Gamma(v_0; v_1; v_2; \dots; v_{n_v}) \end{array} \right)$$

where

$$1 \leq i_p \leq \frac{1}{m_p+1} \binom{2m_p}{m_p};$$

$$1 \leq j_{qk} \leq \frac{1}{n_q+2} \binom{2n_q+2}{n_q+1};$$

$$k = 0, 1, \dots, n_q$$

If the elements of the connected operands are represented by the union set $\mathbb{C} = \mathbb{I} \cup \mathbb{O} \cup \mathbb{S} \cup \Omega$, then $\mathbb{C} \cap \mathbb{U} = \emptyset$. Evidently, the different sorts of graphs corresponding to the input, output, serial, and serially circular gestalts can mutually (spontaneously, arbitrarily) overlap. \square

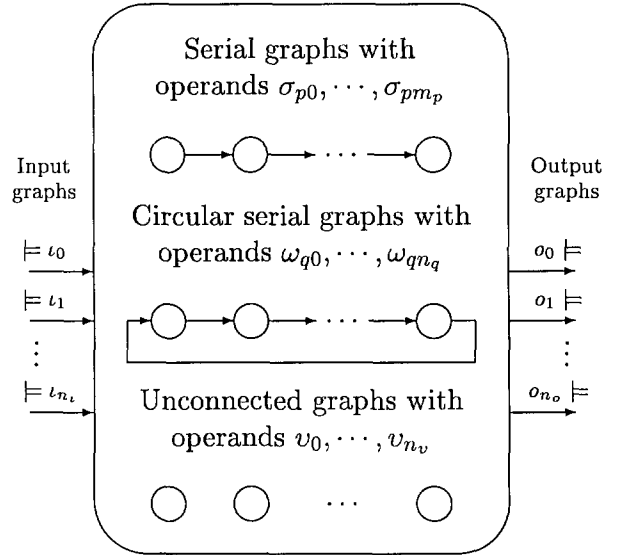


Figure 9: A complex (circularly parallel-serially structured) graph concerning input (internalistic operands), output (externalistic operands), serial (transitional), serially circular, and isolated operands, corresponding to the gestalt in Theorem 3.

Proof 3 (Covering the Graph by Serial and Circular Formulas) It is to stress that any operand of the system formula $\varphi_{\parallel, \rightarrow}^{\circ}$ appears in the graph one time only. This situation leads to the overlapping of input, output, serial, and serially circular graphs. Overlapping means the occurrence of one and the same operand in several formulas concerning the input, output, serial, and serially circular arrangement of operands and operators.

The proof of the theorem can be realized by the inspection of the graph. In this situation, serial and circular formulas can be constructed, concerning different paths and loops, respectively, with the aim, to cover the entire graph by this formula description technique. In this process, any form of the formula, irrespective of the setting of the parenthesis pairs within it, can be chosen. Certainly, for each chosen path or loop of the graph, a formula and to it corresponding gestalt can be determined. Evidently, by such a technique, the graph with all its circles, ovals, and arrows can be covered, that is described by formulas in whole. It is evident that for, the isolated operands and for internalistic and externalistic formulas, there is

$$(v_1; v_2; \dots; v_{n_v}) \equiv \Gamma(v_1; v_2; \dots; v_{n_v}) \text{ and}$$

$$(\models \iota_i) \equiv \Gamma(\models \iota_i); (o_j \models) \equiv \Gamma(o_j \models)$$

respectively. Thus, the number of causal variations of system $\parallel\Phi_{\rightarrow}^{\circ}$ is

$$\sum_{p=0}^A \frac{1}{m_p + 1} \binom{2m_p}{m_p} + \sum_{q=0}^B \frac{1}{n_q + 2} \binom{2n_q + 2}{n_q + 1}$$

This completes the proof of the theorem. \square

DEFINITION 5 (Graphical Equivalence of Formulas and Formula Systems) Two formula systems (or formulas), marked by Φ_1 and Φ_2 , are said to be graphically equivalent, if they describe (cover, have, possess) exactly (completely) one and the same informational graph \mathfrak{G} . In this case, we introduce

$$\Phi_1 \equiv_{\mathfrak{G}} \Phi_2$$

where $\equiv_{\mathfrak{G}}$ marks the operator of the informational graphical equivalence. The meaning of this graphical equivalence can be expressed by the formula

$$(\Phi_1 \equiv_{\mathfrak{G}} \Phi_2) \Leftrightarrow (\mathfrak{G}(\Phi_1) \equiv \mathfrak{G}(\Phi_2))$$

in which the general informational equivalence (operator \equiv) can be used. For a formula system (or formula) Φ_1 and its graph, and a formula system (or formula) Φ_2 and its graph, the following is alternatively (the comma) informationally implied:

$$(\mathfrak{G}(\Phi_1) \equiv \mathfrak{G}(\Phi_2)) \Rightarrow \left(\begin{array}{l} \Phi_1 \equiv_{\mathfrak{G}} \mathfrak{G}(\Phi_2), \\ \mathfrak{G}(\Phi_1) \equiv_{\mathfrak{G}} \Phi_2, \\ \mathfrak{G}(\Phi_1) \equiv_{\mathfrak{G}} \mathfrak{G}(\Phi_2) \end{array} \right)$$

Thus the use of the general equivalence operator \equiv and the graphical equivalence operator $\equiv_{\mathfrak{G}}$ is uniquely determined. \square

THEOREM 4 (An Equivalence of the Possible Graph Descriptions by Different Formula Systems) *There exists always the one and only one graph \mathfrak{G} being described simultaneously by the circular primitive parallel formula system*

$$\psi_{\parallel}^{\circ'} \left(\begin{array}{l} \iota_0, \iota_1, \iota_2, \dots, \iota_{n_i}; \\ o_0, o_1, o_2, \dots, o_{n_o}; \\ \sigma_{p0}, \sigma_{p1}, \sigma_{p2}, \dots, \sigma_{pm_p}; \\ p = 0, 1, \dots, A; \\ \omega_{q0}, \omega_{q1}, \omega_{q2}, \dots, \omega_{qn_q}; \\ q = 0, 1, \dots, B; \\ v_0, v_1, v_2, \dots, v_{n_v} \end{array} \right)$$

(Theorem 1), consisting of input operands ι_i , output operands o_i , serial (elementary transitional) operands $\sigma_{i\sigma}$, circular (elementary transitional) operands $\omega_{i\omega}$ and isolated operands v_{i_v} , which can (with exception of the isolated operands) arbitrarily overlap one another, on the one side, and the graphically equivalent parallel circular serial formula system $\parallel\Phi_{\rightarrow}^{\circ}$, on the other side. In this system, there exist circular serial formulas (of length 2 and greater), which can be derived from the graph. There is

$$\parallel\Phi_{\rightarrow}^{\circ} \equiv_{\mathfrak{G}} \psi_{\parallel}^{\circ'}$$

Moreover, $\psi_{\parallel}^{\circ'}$ means nothing else than the primitive parallelization of $\parallel\Phi_{\rightarrow}^{\circ}$, by which serial and circular serial formulas of $\parallel\Phi_{\rightarrow}^{\circ}$ are decomposed into primitive (basic) transitions. In this way,

$$\Pi'(\parallel\Phi_{\rightarrow}^{\circ}) \equiv \psi_{\parallel}^{\circ'}$$

where

$$\psi_{\parallel}^{\circ'} \left(\begin{array}{l} \iota_0, \iota_1, \iota_2, \dots, \iota_{n_i}; \\ o_0, o_1, o_2, \dots, o_{n_o}; \\ \sigma_{p0}, \sigma_{p1}, \sigma_{p2}, \dots, \sigma_{pm_p}; \\ p = 0, 1, \dots, A; \\ \omega_{q0}, \omega_{q1}, \omega_{q2}, \dots, \omega_{qn_q}; \\ q = 0, 1, \dots, B; \\ v_0, v_1, v_2, \dots, v_{n_v} \end{array} \right) \equiv \left(\begin{array}{l} \models \iota_r; r = 0, 1, \dots, n_i; \\ o_s \models; s = 0, 1, \dots, n_o; \\ \sigma_{pi} \models \sigma_{p,i+1}; \\ p = 0, 1, \dots, A; \\ i = 0, 1, \dots, m_p - 1; \\ \omega_{qj} \models \omega_{q,j+1}; \omega_{qn_q} \models \omega_{q0}; \\ q = 0, 1, \dots, B; \\ j = 0, 1, \dots, n_q - 1; \\ v_u; u = 1, 2, \dots, n_v \end{array} \right)$$

The obtained serial and circular serial basic transitions are seen in the lower formula array. \square

Proof 4 The proof of the last theorem is evident and proceeds from the previous definitions and theorems. For a graph \mathfrak{G} , there is evidently,

$$\mathfrak{G} \left(\begin{array}{c} \psi_{\parallel}^{\circ'} \\ \left(\begin{array}{l} \iota_0, \iota_1, \iota_2, \dots, \iota_{n_\iota}; \\ o_0, o_1, o_2, \dots, o_{n_o}; \\ \sigma_{p0}, \sigma_{p1}, \sigma_{p2}, \dots, \sigma_{pm_p}; \\ p = 0, 1, \dots, A; \\ \omega_{q0}, \omega_{q1}, \omega_{q2}, \dots, \omega_{qn_q}; \\ q = 0, 1, \dots, B; \\ v_0, v_1, v_2, \dots, v_{n_v} \end{array} \right) \end{array} \right) \equiv \mathfrak{G} \left(\begin{array}{c} \Phi_{\parallel}^{\circ} \\ \left(\begin{array}{l} \iota_0, \iota_1, \iota_2, \dots, \iota_{n_\iota}; \\ o_0, o_1, o_2, \dots, o_{n_o}; \\ \sigma_{p0}, \sigma_{p1}, \sigma_{p2}, \dots, \sigma_{pm_p}; \\ p = 0, 1, \dots, A; \\ \omega_{q0}, \omega_{q1}, \omega_{q2}, \dots, \omega_{qn_q}; \\ q = 0, 1, \dots, B; \\ v_0, v_1, v_2, \dots, v_{n_v} \end{array} \right) \end{array} \right)$$

Both parallel formula systems, $\psi_{\parallel}^{\circ'}$ and Φ_{\parallel}° describe exactly one and the same graph \mathfrak{G} . This proves the theorem. \square

Theorem 4 says that there exist at least two formal and to the graph completely corresponding interpretations of an informational graph: that in the form of the primitive parallel formula system (PPFS) and the other in the form of the parallel circular serial formula system (PCSFS). Other forms can certainly be circularly mixed parallel-serial systems, including explicit expressions of various main operands of complexly mashed informational systems.

4 Serial and Circular Informational Graphs Hiding a Parallel Informational Structure

4.1 Introduction

In general, an informational graph represents a complex (gestalt-like) informational structure, which under certain circumstances, can simultaneously represent parallel, serial and circular informing of the involved (mutually informing) operands (entities). An informational graph, as an optical or formalistic scheme, does not say anything of that what concretely, in a true situation, comes actually in the foreground. It can be only a simple, very particular serial or circularly serial scheme of informing. On the other

side, such a graph, obtained from the pure serial or circularly serial situation, comes to the foreground simultaneously as a pure, that is, extremely parallel informingly structured organization, and through this view, covers not only one particular case but all possible other particular cases of particular causalisms, as a consequence of the parenthesis pairs displacements in particular formulas in respect to the original (initial) particular formula(s).

On the other hand, we have learned so far, how informational parallelism can cause the phenomenon of informational serialism and informational circular serialism (as an implicit property of elementary informational transitions). Within informational serialism of any kind, the so-called causalism can come into existence. The causalism is nothing other than a different serial interpretation of an initial formula, in which parenthesis pairs are not once for all determined or are simply left out. This happens with sentences of a natural language where a formal syntax analysis cannot be performed unambiguously. Also in a speech act, the speaker or hearer cannot follow an abstract and depth-structured syntax analysis (synthesis), but more or less a spontaneous linguistic informing. Also, a living linguistic performer, by his/her consciousness, cannot inform in the discussed primitive parallel way, when long and circularly structured sentences (with hundreds or thousands causal possibilities) appear in the discourse. A consequent primitive informing would be possible solely by an informational machine⁴ [22].

Causalism appears as nothing other than a specific informational organization (structure) of informational serialism. There does not exist a causalism of this sort in a pure parallel (simultaneous) case. It seems that causalism is “timely” (serially) conditioned, where serially could mean time-consequently.

4.2 A Serial Formula and Its Graph

The pure serial informational graphs (without informational circles) correspond to the pure serial informational formulas in which there are only bi-

⁴An informational machine is a multiprocessing (multi-processor) device possessing the property of an informational entity (spontaneity, circularity, parallelism, serial and circular causality).

nary informational operators between operand entities (parenthesized or single) in a formula. To this pure serial structure one can (must) add possibilities of the input and output informing of the serial structure. The input informing is internalistic when a serial operand component σ is being openly informed from the environment by $\models \sigma_i$. The output informing is externalistic when an operand of the serial structure informs openly to the environment by $\sigma_i \models$. Such a structure is realistic from the viewpoint that entities (operands) of a serial structure (formula) can be informed from exterior operands and can inform to exterior operands.

DEFINITION 6 (A Pure Serial Formula) A pure serial formula, ${}^n_i\varphi_{\rightarrow}$, is determined in the following manner:

$${}^n_i\varphi_{\rightarrow}(\sigma_0, \sigma_1, \dots, \sigma_n) \equiv \left(\begin{array}{l} \models \sigma_i; \sigma_o \models; \\ (\sigma_i, \sigma_o \in \{\sigma_0, \sigma_1, \dots, \sigma_n\}); \\ (\dots(\sigma_0 \models \sigma_1) \models \dots \sigma_{n-1}) \models \sigma_n \end{array} \right)$$

where the serial formula of the form $(\dots(\sigma_0 \models \sigma_1) \models \dots \sigma_{n-1}) \models \sigma_n$ can be substituted by any other serial formula with displaced parenthesis pairs, that is by a formula which is graphically equivalent to the original formula (Definition 5). The formula subscript i systematically varies in the interval $1 \leq i \leq \frac{1}{n+1} \binom{2n}{n}$. \square

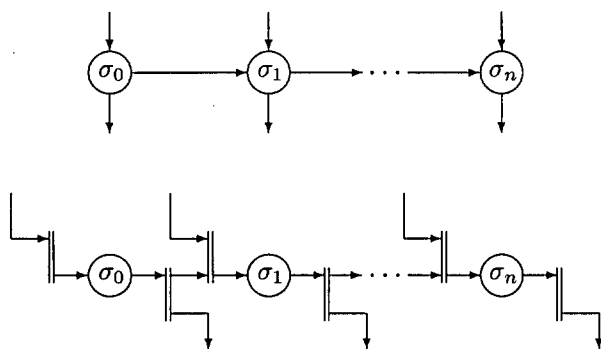


Figure 10: Two graphical interpretations corresponding to the pure serial formula system in Definition 6. The equivalent bottom graph explicates the parallel character of the serial formula.

4.3 Primitive Serial Parallelism and Serial Formula Gestalt

A primitive serial parallelism denotes the existence of the primitive transitions which are serially connected, e.g., in the form of a small paragon system $\alpha \models \beta; \beta \models \gamma; \gamma \models \delta$. A serial formula gestalt is the system of all the possible serial formulas which parallelization delivers the primitive parallel system. For the paragon system, these gestalt formulas have the form $((\alpha \models \beta) \models \gamma) \models \delta$, with all possible replacement of the parenthesis pairs.

DEFINITION 7 (Primitive Parallelism of a Pure Serial Formula) The graph corresponding to Definition 6 is presented in Fig. 10. The parallel collection lines \parallel in the bottom graph of the figure show the potentiality of further parallel inputs and outputs of the serially coupled entities. According to Theorem 4, the graph in Fig. 10 can be equivalently described by the primitive parallel formula system

$${}^n_{\parallel}\varphi'(\sigma_0, \sigma_1, \dots, \sigma_n) \equiv \left(\begin{array}{l} \models \sigma_i; \sigma_o \models; (\sigma_i, \sigma_o \in \{\sigma_0, \sigma_1, \dots, \sigma_n\}); \\ \sigma_0 \models \sigma_1; \sigma_1 \models \sigma_2; \dots; \sigma_{n-1} \models \sigma_n \end{array} \right)$$

For this formula it is characteristic that it determines the gestalt Γ of formula ${}^n_i\varphi_{\rightarrow}(\sigma_1, \sigma_2, \dots, \sigma_n)$ in Definition 6. \square

THEOREM 5 (Graphical Equivalence of a Serial Formula Gestalt and a Pure Serial, Primitive Parallel Formula System) The graphical equivalence of the form

$$\Gamma({}^n_i\varphi_{\rightarrow}(\sigma_0, \sigma_1, \dots, \sigma_n)) \equiv_{\mathfrak{G}} {}^n_{\parallel}\varphi'(\sigma_0, \sigma_1, \dots, \sigma_n)$$

determines one and the same graph for the gestalt of a serial formula of length n and the corresponding primitive parallel formula. This equivalence shows the power of the basic transition formulas (of the length 1, and informing in parallel) compared to the long serial formulas of length n , and belonging to the gestalt $\Gamma({}^n_i\varphi_{\rightarrow}(\sigma_0, \sigma_1, \dots, \sigma_n))$. \square

Proof 5 We have to prove the graphical equivalence of the two parallel formula systems in the theorem. Gestalt $\Gamma({}^n_i\varphi_{\rightarrow}(\sigma_0, \sigma_1, \dots, \sigma_n))$ is a parallel system of serial formulas of the form ${}^n_i\varphi_{\rightarrow}(\sigma_0, \sigma_1, \dots, \sigma_n)$ which length is n . In a pure

serial formula of length n there are exactly n binary operators and $n + 1$ serial operands. As described, parenthesis pairs can be displaced in a spontaneous manner, giving to the formulas different causal meanings (the corresponding causal subscript i of formula ${}^n_i\varphi_{\rightarrow}$). Thus, the formulas can be systematically numbered (subscripted), where, for example,

$$\begin{aligned} {}^n_1\varphi_{\rightarrow}(\sigma_0, \sigma_1, \dots, \sigma_n) &\equiv ((\dots(\sigma_0 \models \sigma_1) \models \dots \sigma_{n-1}) \models \sigma_n); \\ {}^n_2\varphi_{\rightarrow}(\sigma_0, \sigma_1, \dots, \sigma_n) &\equiv ((\dots(\sigma_0 \models \sigma_1) \models \dots \sigma_{n-1} \models \sigma_n)); \\ &\vdots \\ {}^n_{\frac{n+1}{2}}\varphi_{\rightarrow}(\sigma_0, \sigma_2, \dots, \sigma_n) &\equiv (\sigma_0 \models (\sigma_1 \models \dots (\sigma_{n-1} \models \sigma_n) \dots)) \end{aligned}$$

Thus, the gestalt of any of these formulas is a parallel system of $N_{\rightarrow} = \frac{1}{n+1} \binom{2n}{n}$ serial formulas, that is, in a shortened symbolic form,

$$\Gamma({}^n_i\varphi_{\rightarrow}) \equiv \left({}^n_1\varphi_{\rightarrow}; {}^n_2\varphi_{\rightarrow}; \dots; {}^n_{\frac{n+1}{2}}\varphi_{\rightarrow} \right)$$

for $1 \leq i \leq N_{\rightarrow}$. Each of these formulas determines one and the same graph \mathfrak{G} .

On the other hand, a graph \mathfrak{G} , corresponding to a formula ${}^n_i\varphi_{\rightarrow}$ can be described, according to Theorem 1, by formula ${}^n\varphi'_{\parallel}$. This proves the theorem. \square

4.4 A Circular Serial Formula and Its Graph

DEFINITION 8 (A Pure Serially Circular Formula) A pure serially circular formula, ${}^{n+1}_{j_1}\varphi^{\circ}_{\rightarrow}$, is determined in the following manner:

$$\begin{aligned} {}^{n+1}_{j_1}\varphi^{\circ}_{\rightarrow}(\omega_0, \omega_1, \dots, \omega_n) &\equiv \\ &(\models \omega_i; \omega_o \models; (\omega_i, \omega_o \in \{\omega_0, \omega_1, \dots, \omega_n\});) \\ &(((\dots(\omega_0 \models \omega_1) \models \dots \omega_{n-1}) \models \omega_n) \models \omega_0) \end{aligned}$$

where the serially circular formula of the form $((\dots(\omega_0 \models \omega_1) \models \dots \omega_{n-1}) \models \omega_n) \models \omega_0$ can be substituted by any other serial formula with displaced parenthesis pairs, that is by a formula which is graphically equivalent to the original formula (Definition 5). Formula subscript j systematically varies in the interval $1 \leq j_1 \leq \frac{1}{n+2} \binom{2n+2}{n+1}$. \square

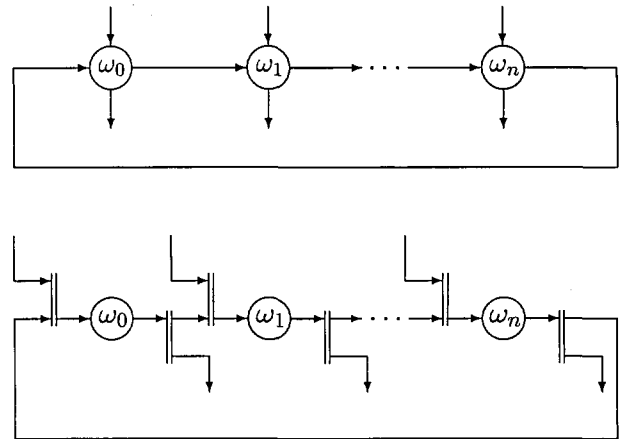


Figure 11: Two graphical interpretations corresponding to formula system in Definition 8. The equivalent bottom graph explicates the parallel character of the serially circular formula.

4.5 Primitive Circular Serial Parallelism, and Gestalt of Circular Serial Formula

For the circular serial parallelism, one of the basic transition formulas must informationally connect the ‘last’ operand with the ‘first’ one, that is, $\alpha_{\text{last}} \models \alpha_{\text{first}}$.

THEOREM 6 (Graphical Equivalence of a Circular Serial Formula Gestalt and a Pure Circular Serial, Primitive Parallel Formula System) The graphical equivalence of the form

$$\Gamma\left({}^{n+1}_j\varphi^{\circ}_{\rightarrow}(\omega_0, \omega_1, \dots, \omega_n)\right) \equiv_{\mathfrak{G}} {}^{n+1}\varphi^{\circ'}_{\parallel}(\omega_0, \omega_1, \dots, \omega_n)$$

determines one and the same graph for the gestalt of a circular serial formula of length $n + 1$ and the corresponding primitive circular parallel formula system. This equivalence shows the power of the basic transition formulas (of the length 1, and informing in parallel) compared to long circular serial formulas of the length $n + 1$, belonging to the gestalt $\Gamma\left({}^{n+1}_j\varphi^{\circ}_{\rightarrow}(\omega_0, \omega_1, \dots, \omega_n)\right)$. \square

Proof 6 We have to prove the graphical equivalence of the both circularly structured parallel formula systems in the theorem. Gestalt $\Gamma\left({}^{n+1}_j\varphi^{\circ}_{\rightarrow}(\omega_0, \omega_1, \dots, \omega_n)\right)$ is a parallel system of the serial formulas of the form

${}^{n+1}_j \varphi_{\rightarrow}^{\circ}(\omega_0, \omega_1, \dots, \omega_n)$ which length is $n + 1$. In a pure circular serial formula of the length $n + 1$ there are exactly $n + 1$ binary operators and the $n + 1$ serial operands (one of them appears twice, that is, the title operand of the circular formula, at the beginning and at the end of the formula. As described, parenthesis pairs can be displaced in a spontaneous manner, giving to the formulas different causal meanings (the corresponding causal subscript i of formula ${}^{n+1}_j \varphi_{\rightarrow}^{\circ}(\omega_0, \omega_1, \dots, \omega_n)$). Thus, the formulas can be systematically numbered (subscripted), where, for example,

$$\begin{aligned} & {}^{n+1}_1 \varphi_{\rightarrow}^{\circ}(\omega_0, \omega_1, \dots, \omega_n) \rightleftharpoons \\ & \quad (((\dots(\omega_0 \models \omega_1) \models \dots \omega_{n-1}) \models \omega_n) \models \omega_0); \\ & {}^{n+1}_2 \varphi_{\rightarrow}^{\circ}(\omega_0, \omega_1, \dots, \omega_n) \rightleftharpoons \\ & \quad ((\dots(\omega_0 \models \omega_1) \models \dots \omega_{n-1}) \models (\omega_n \models \omega_0)); \\ & \vdots \\ & {}^{n+1}_{\frac{n+2}{2}} \varphi_{\rightarrow}^{\circ}(\omega_0, \omega_1, \dots, \omega_n) \rightleftharpoons \\ & \quad (\omega_0 \models (\omega_1 \models \dots (\omega_{n-1} \models (\omega_n \models \omega_0)) \dots)) \end{aligned}$$

Thus, the gestalt of any of these formulas is a parallel system of $N_{\rightarrow}^{\circ} = \frac{1}{n+2} \binom{2n+2}{n+1}$ circular serial formulas, that is, in a shortened notation,

$$\Gamma \left({}^{n+1}_j \varphi_{\rightarrow}^{\circ} \right) \rightleftharpoons \left({}^{n+1}_1 \varphi_{\rightarrow}^{\circ}; {}^{n+1}_2 \varphi_{\rightarrow}^{\circ}; \dots; {}^{n+1}_{\frac{n+2}{2}} \varphi_{\rightarrow}^{\circ} \right)$$

for $1 \leq j \leq N_{\rightarrow}^{\circ}$. Each of these circular formulas determines one and the same graph \mathfrak{G} .

On the other hand, graph \mathfrak{G} , corresponding to a formula ${}^{n+1}_j \varphi_{\rightarrow}^{\circ}$ can be described, according to Theorem 1, by formula ${}^{n+1}_{\parallel} \varphi_{\rightarrow}^{\circ'}$. This proves the theorem. \square

4.6 Circular Gestalt

The circular gestalt of a circular formula concerns the parity of the formula operands. Within the cycle of the formula graph, each operand can be rotated to the title (initial, leftmost) position, and in this situation, it can function also as the leftmost and simultaneously as the rightmost operand in the circular formula.

THEOREM 7 (Graphical Equivalence of Primitive Circular Parallel Formula Systems) *If the expression ${}^{n+1}_{\parallel} \varphi_{\rightarrow}^{\circ'}$ represents a primitive circular parallel formula system (of basic $n + 1$ transitions), then*

$$\begin{aligned} & {}^{n+1}_{\parallel} \varphi_{\rightarrow}^{\circ'}(\omega_0, \omega_1, \dots, \omega_n) \equiv_{\mathfrak{G}} \\ & \quad {}^{n+1}_{\parallel} \varphi_{\rightarrow}^{\circ'}(\omega_i, \omega_{i+1}, \dots, \omega_n, \omega_0, \omega_1, \dots, \omega_{i-1}) \end{aligned}$$

for $i = 1, 2, \dots, n$. Each primitive circular formula system

$${}^{n+1}_{\parallel} \varphi_{\rightarrow}^{\circ'}(\omega_i, \omega_{i+1}, \dots, \omega_n, \omega_0, \omega_1, \dots, \omega_{i-1})$$

corresponds to a gestalt Γ of the circular serial formula

$${}^{n+1}_{j_i} \varphi_{\rightarrow}^{\circ}(\omega_i, \omega_{i+1}, \dots, \omega_n, \omega_0, \omega_1, \dots, \omega_{i-1})$$

The parallel system of such gestalts is called the the circular gestalt Γ° of a circular serial function ${}^{n+1}_{j_0} \varphi_{\rightarrow}^{\circ}(\omega_0, \omega_1, \dots, \omega_n)$. There is

$$\Gamma^{\circ} \left({}^{n+1}_{j_i} \varphi_{\rightarrow}^{\circ}(\omega_i, \omega_{i+1}, \dots, \omega_n, \omega_0, \omega_1, \dots, \omega_{i-1}) \right) \rightleftharpoons \left(\begin{array}{l} \Gamma \left({}^{n+1}_{j_0} \varphi_{\rightarrow}^{\circ}(\omega_0, \omega_1, \dots, \omega_n) \right); \\ \Gamma \left({}^{n+1}_{j_1} \varphi_{\rightarrow}^{\circ}(\omega_1, \omega_2, \dots, \omega_n, \omega_0) \right); \\ \vdots \\ \Gamma \left({}^{n+1}_{j_i} \varphi_{\rightarrow}^{\circ}(\omega_i, \omega_{i+1}, \dots, \omega_n, \omega_0, \omega_1, \dots, \omega_{i-1}) \right); \\ \vdots \\ \Gamma \left({}^{n+1}_{j_n} \varphi_{\rightarrow}^{\circ}(\omega_n, \omega_0, \omega_1, \dots, \omega_{n-1}) \right) \end{array} \right)$$

\square

Proof 7 In a parallel formula system, formulas can occur in an arbitrary sequence. The formula ordering does not influence the informing of the system. Therefore, instead of $(\alpha \models \beta) \models (\gamma \models \delta)$, simply the semicolon system $(\alpha \models \beta; \gamma \models \delta)$ can be used (taking a semicolon instead of \models), where semicolon performs as an associative operator of parallel occurrences of formulas in an informational system.

The proof of the first part of the theorem consists of the graphical equivalence of informational systems of the form

$$\left(\begin{array}{l} \omega_0 \models \omega_1; \\ \omega_1 \models \omega_2; \\ \vdots \\ \omega_{i-1} \models \omega_i; \\ \omega_i \models \omega_{i+1}; \\ \omega_{i+1} \models \omega_{i+2}; \\ \vdots \\ \omega_{n-1} \models \omega_n; \\ \omega_n \models \omega_0 \end{array} \right) \equiv_{\mathfrak{G}} \left(\begin{array}{l} \omega_i \models \omega_{i+1}; \\ \omega_{i+1} \models \omega_{i+2}; \\ \vdots \\ \omega_{n-1} \models \omega_n; \\ \omega_n \models \omega_0; \\ \omega_0 \models \omega_1; \\ \vdots \\ \omega_{i-2} \models \omega_{i-1}; \\ \omega_{i-1} \models \omega_i \end{array} \right)$$

for $i = 0, 1, 2, \dots, n$. The right formula system is nothing else than the parallel reordered left formula system. So,

$$\begin{aligned} \varphi_{\parallel}^{\circ'}(\omega_0, \omega_1, \dots, \omega_n) &\equiv_{\mathfrak{G}} \\ \varphi_{\parallel}^{\circ'}(\omega_i, \omega_{i+1}, \dots, \omega_n, \omega_0, \omega_1, \dots, \omega_{i-1}) \end{aligned}$$

is proved. But, evidently, in this particular case, instead of the operator of the graphical equivalence $\equiv_{\mathfrak{G}}$, the operator of the general informational equivalence \equiv could be used.

In the second part of the theorem we must prove

$$\begin{aligned} \Gamma^{\circ} \left(\varphi_{j_i \rightarrow}^{n+1}(\omega_i, \omega_{i+1}, \dots, \omega_n, \omega_0, \omega_1, \dots, \omega_{i-1}) \right) \\ \equiv_{\mathfrak{G}} \varphi_{\parallel}^{\circ'}(\omega_0, \omega_1, \dots, \omega_n) \end{aligned}$$

There also is, evidently,

$$\begin{aligned} \Gamma \left(\varphi_{j_i \rightarrow}^{n+1}(\omega_i, \omega_{i+1}, \dots, \omega_n, \omega_0, \omega_1, \dots, \omega_{i-1}) \right) \\ \equiv_{\mathfrak{G}} \varphi_{\parallel}^{\circ'}(\omega_0, \omega_1, \dots, \omega_n) \end{aligned}$$

However, there are $n + 1$ different gestalts $\Gamma \left(\varphi_{j_i \rightarrow}^{n+1}(\omega_i, \omega_{i+1}, \dots, \omega_n, \omega_0, \omega_1, \dots, \omega_{i-1}) \right)$ for a circular formula, and so, $\overset{\circ}{N}_{\rightarrow} = \frac{n+1}{n+2} \binom{2n+2}{n+1}$ possibilities of the circular formula interpretation through all circular operands, where each of the operands can occupy the title position in the corresponding circular formula. This proves the second part of the theorem. \square

To remind, graphical equivalence means one and the same informational graph for two differently structured informational formulas or formula systems. As we have seen, the following graphical equivalences hold:

$$\begin{aligned} \varphi_{\parallel}^{\circ'} &\equiv_{\mathfrak{G}} \Gamma \left(\varphi_{j_i \rightarrow}^{n+1} \right) \equiv_{\mathfrak{G}} \Gamma^{\circ} \left(\varphi_{j_i \rightarrow}^{n+1} \right) \equiv_{\mathfrak{G}} \\ &\mathfrak{G} \left(\varphi_{j_i \rightarrow}^{n+1} \right) \equiv_{\mathfrak{G}} \mathfrak{G} \left(\varphi_{j_k \rightarrow}^{n+1} \right) \end{aligned}$$

where $i, k = 0, 1, 2, \dots, n$. We see how operator $\equiv_{\mathfrak{G}}$, in fact, has the property of the graphical projection of an informational formula onto its informational graph.

4.7 Star Gestalt Γ^*

The star gestalt of a formula or a (complex) formula system is obtained by moving through the graph from an arbitrary operand by an arbitrary chosen path (in the direction of operator arrows). In this way, two formulas, obtained by such a moving, can have one and the same graph. The moving can run along an infinite path when formula or formula system is circularly structured. In this sense, a star gestalt concerning a circular system, can consist of an unbounded number of formulas (the infiniteness of recursiveness).

The reader can imagine how formulas of a star gestalt can be arbitrarily not only cycled, but also re-cycled, that is, certain parts of the paths can be passed again and again. Such a circulating does not cause an additional part of the graph, but it can have complex causal, especially, self-referencing and self-constructing consequences, reaching to arbitrary causal (self-informing, hermeneutic, historical, memorizing, consciousness) depths. The phenomena using the attribute of the self can have a circular organization of an arbitrary depth. For instance, phenomena of self-observation, self-reference, self-replication, self-consciousness, etc. can be conceptually and notionally captured (grasped, understood) by sufficiently complex circular informational formalism, possessing means of spontaneous informational arising (emerging, coming into existence, to the surface, etc.). The reader can perform such experiments by means of presented circular informational graphs in this paper.

For a simple (pure) circular formula of the form $\varphi_{j_i \rightarrow}^{n+1}(\omega_i, \omega_{i+1}, \dots, \omega_n, \omega_0, \omega_1, \dots, \omega_{i-1})$, the cycle can be repeated several times, and can begin at an arbitrary operand and end by an another (arbitrary) operand. Such a case is not particularly interesting in the sense of a star

\subset	entworfen	(ist) entworfen in [(is) projected in]
\Vdash	erschließen	erschließen [disclose]
\Vdash	für	ist für (für) [is for (for)]
\Vdash		sein (bin, bist, ist, sind) [to be (am, are, is)]
\nVdash		nicht sein (ist nicht, wird nicht) [not to be (is not, does not become)]
\implies		impliziert [implies]
\Vdash	sondern	sondern (ist) [but (is)]
\Vdash	verstehen	verstehen [understand(s)]
\Vdash	zu	ist zu (zu, gegen) [is towards (towards)]
\Vdash_{Ψ}		ist eine Funktion von [is a function of]
\subset		hat, einschließt, besitzt [has, include(s), possess(es)]
\circ		und (Operatorkomposition) [and (operator composition)]
$;$		Formelparallelismus [parallelism (of formulas)]
$‘, ’$		Formelalternativität [alternativeness (of formulas)]
\approx		bedeutet, bedeuten [mean(s)]

$$\parallel V_{\rightarrow}^{\circ'} = \left(\begin{array}{ll} a_{\text{andere}} \subset A; & \left(\begin{array}{l} A_{\text{Ausarbeitung}} \\ \Vdash_{\Psi} M \end{array} \right); \\ A_{\text{Ausbildung}} \Vdash_{\Psi} V; & A \Vdash \text{sondern } V; \\ A \Vdash \text{verstehen } V; & A \nVdash K; \\ D \Vdash \text{als } V; & D \Vdash \text{auf } M; \\ D \Vdash \text{zu } M; & D \Vdash \text{erschließen } R; \\ E \Vdash_{\Psi} V; & e_{\text{etwas}} \Vdash_{\Psi} a_{\text{andere}}; \\ K \Vdash_{\Psi} V_{\text{Verstandene}}; & M = M; \\ M \Vdash_{\Psi} S; & M \Vdash \text{verstehen } S; \\ M \Vdash S_{\text{Seinkönnen}}; & M \Vdash V; \\ M \Vdash_{\Psi} E; & M \subset \text{entworfen } V; \\ R \Vdash_{\Psi} M; & S \Vdash_{\Psi} D; \\ S_{\text{Seinkönnen}} \Vdash \text{für } S; & S_{\text{Seinkönnen}} \subset D; \\ \left(\begin{array}{l} V_{\text{Verstandene}} \\ \Vdash \text{verstehen } V \end{array} \right); & \left(\begin{array}{l} V_{\text{Verstandene}} \\ \Vdash \text{sondern} \\ A_{\text{Ausarbeitung}} \end{array} \right); \\ V \Vdash \text{entwerfen } S; & V \Vdash \text{als } S_{\text{Seinkönnen}}; \\ V \subset V; & V \subset E; \\ V \Vdash \text{ausbilden } M; & V \Vdash \text{ausbilden } E; \\ V \Vdash \text{ausbilden } V; & V = A; \\ V = V; & V \Vdash A_{\text{Ausbildung}}; \\ V \Vdash \text{eignen } V_{\text{Verstandene}}; & V \subset A; \\ V \nVdash e_{\text{etwas}}; & V \Vdash \text{verstehen } A; \\ V \implies V; & V \Vdash \text{entwerfen } M \end{array} \right)$$

Evidently, according to Fig. 12, each operand in in some way circularly connected with other operands occurring in the graph. The graph is a mesh of feedback loops for each occurring operand, a kind of creative loops⁵ interacting in the informational mesh. It means also, that Heidegger was probably well aware that a proper philosophical system for understanding and interpretation has to be structured circularly in the sense of interconnected loops (informational circles), if possible, also in the presented initial philosophical text detail.

This complete circular system can be formally described by the primitive circular parallel understanding system $\parallel V_{\rightarrow}^{\circ'}$, consisting of elementary transitions, that is (alphabetically ordered by the left operands of elementary transitions),

Understanding $\parallel V_{\rightarrow}^{\circ'}$ of something α requires the additional transition

$$\alpha \Vdash V$$

to the system $\parallel V_{\rightarrow}^{\circ'}$, causing $\parallel V_{\rightarrow}^{\circ'}(\alpha)$, and thus all operands of system $\parallel V_{\rightarrow}^{\circ'}$ become functions of α , that is, take the general functional form $\varphi(\alpha)$ [20].

5 Complexly Interweaved Circular Informational Graphs

Real informational systems are complexly circularly interweaved. This is a condition sine qua non, for only circular systems have the potentiality of emerging from that what already is, to that what unforeseeably could arise. In the most normal situation, each occurring operand (representing an informational entity) is circularly connected in one or more loops (circular graphs), and

⁵See, for instance, E. Harth: *The Creative Loop, How the Brain Makes a Mind*. Penguin, Harmondsworth (1996); or the book review in *Journal of Consciousness Studies* 3 (1996) No. 2, pp. 186–187, by G. Sommerhoff.

each loop is in some way connected to all other loops in the system. For such a system we say that it is *completely circularly connected* or informationally closed. In some way, each operand of such a system informs (indirectly via other operands) all other operands and vice versa: each operand is indirectly (or, in particular cases, directly) informed by all other operands. This means that a singular operand impacts the system entirely and is entirely impacted by the system in which it occurs.

This type of informational closeness does not mean that system operands cannot be impacted from the exterior and that they cannot impact the exterior operands (entities). According to informational axioms [21], this property of informational openness or independence of operands is given, in general, to any informational entity. Only in clearly explicated cases, it can be determined in which case an entity does not inform another entity.

Besides, there can exist operands which do not appear in a loop, for instance, merely in linearly serially structured formulas, where the last (end) operands of such formulas function like final destinations, informing for the sake of its own purpose, as a kind of final receptors. In some cases, such final informational destinations can be foreseen for a later looping into the system, when their informing will begin to impact the other entities of the system.

In the described complexly interweaved circular informational system, presented by the corresponding graph, the only senseful and significant function of each operand is to be circularly connected to the system, that is, to play a developmental role of the system by its arising and diminishing⁶. Otherwise, the existence of an unconnected or partly connected informational item is not within an informationally senseful and significant function of system development, emergence, and function.

In the same sense, parallel informational graphs can exist, but, in a senseful situation, they must be in some way connected. Isolated graphs are presentations of possible informational informing and as such, that is mutually isolated, they perform as a kind of informational reductionism.

⁶For instance, in the sense of an estimate of the precision or certainty or definiteness according to [11].

This especially holds in the cases when informing of system entities is studied, grasped, and finally presented under essentially limited circumstances. Sooner or later, the need for informational complexity in the form of a serial, parallel, and circular phenomenalism comes to consciousness. Both in the living and artificial systems, as well, the facts of this informational complexity can be considered.

6 A Classification of Informational Graphs

6.1 Isolated Graphs

Isolated graphs perform as informationally isolated entities. It simply means that they are not connected to and from the other graphs or entities. They usually emerge as a consequence of the so-called reductionistic reasoning, where each graph, as such, represents a reductionistic situation of a particularly isolated view or interest.

Isolated graphs as informational entities are in no way senseless since they can become, through the emergence of conceptual and informing systems, parts of systems, also with essential informational modification, further decomposition, connectivity, and the like. As such, they can become suitable for the so-called bottom-up composition. In this sense, isolated informational graphs hide the potentiality for their future involvement into a linearly (serially), parallel (simultaneously), and circularly (cyclically, with regard to a loop structure and organization) connected informational realm.

By definition, isolated informational graphs do not lose the axiomatically given property of the input (internalistic) and output (externalistic) possibility of operands, that is, their connection to and from the potential environment. Considering this situation for isolated graphs, the following definition becomes meaningful.

DEFINITION 9 (Isolated Graph) An *isolated informational graph* is an arbitrary organized graph of operand circles and/or ovals and them connecting operator arrows in a serial (linear), parallel, and circular way, but *not* connected to or from other

informational graphs. Operands of the graph retain their potentiality for undetermined internalistic and externalistic informing, from and to the virtual graph environment. Isolated graphs and operands are graphically presented as circles and ovals without arrows. □

As a consequence, each finite informational graph is isolated, that is obtained on the basis of a reductionistic approach. However, it hides the possibility for its further decomposition of operands and operators [26]. Graphs can contain isolated operands and isolated subgraphs.

6.2 Primitive Transition Graph

A primitive transition of the form $\alpha \models \beta$, and its decomposition, [26] is the key form of any informational system, particularly of the primitive serial and serially circular parallel systems of basic transitions (${}^n\varphi'_{\parallel}$ and ${}^{n+1}\varphi^{\circ}'_{\parallel}$). Its operand and operator decomposition possibilities were exhaustively treated and discussed in [26].

DEFINITION 10 (Transition Graph) A *transition informational graph* consists of two operand circles or ovals, connected by a single operand arrow. This arrangement of both operands and an arrow is treated as an informational unity, that is, as a transition from the first operand to the second one. □

6.3 Pure Serial Graph

Pure serial graph is a graphical presentation of the formula ${}^n\varphi_{\rightarrow}$ with $n + 1$ operands and n binary operators. Pure serial graph represents a system of different formulas ${}^n\varphi_{\rightarrow}$ of length n , where $1 \leq i \leq \frac{1}{n+1} \binom{2n}{n}$.

6.4 Pure Circular Serial Graph

A pure circular serial graph is a graphical presentation of formula ${}^{n+1}\varphi^{\circ}_{\rightarrow}$ with $n + 1$ operands and $n + 1$ binary operators. A pure circular serial graph represents a system of different formulas ${}^{n+1}\varphi^{\circ}_{\rightarrow}$ of length $n + 1$, where $1 \leq j \leq \frac{1}{n+2} \binom{2n+2}{n+1}$.

6.5 Parallelism of Graphs

The reader can see that informational graphs are by arrows connected circles and/or ovals. Both

circles/ovals and arrows are marked: circles/ovals by operands and arrows by operators. An unmarked arrow represents the operator \models (an operational joker). At the first glance, such connected circles/ovals in the graph give the impression that the system of operands is informing in a serial (also circularly serial) manner. To exceed this surface impression, the reader should stay aware that any basic informational transition, $\alpha \models \beta$, with its left part α and its right part β , simultaneously means the detachment [23], in the form

$$\frac{\alpha \models \beta}{\alpha; \beta}$$

This detachment says, that α and β inform independently and *in parallel* to the transition $\alpha \models \beta$. This detachment must be understood recursively, irrespective of the complexity of both α and β , which can represent arbitrary formulas or systems of formulas.

Each graph represents parallel informing of all operands and all possible subformulas and formulas which proceed from the graph in the sense of their informational well-formedness, that is as serial formulas ${}^n\varphi_{\rightarrow}$ and ${}^{n+1}\varphi^{\circ}_{\rightarrow}$, primitive formula systems ${}^n\varphi'_{\parallel}$ and ${}^{n+1}\varphi^{\circ}'_{\parallel}$ and all the possible formulas and formula systems of these formulas and formula systems.

Using the last rule of the parallel detachment of subformulas recursively, one can determine how many parallel processors are needed for a simulation of formulas ${}^n\varphi_{\rightarrow}$ and ${}^{n+1}\varphi^{\circ}_{\rightarrow}$ in an informational machine [22]. For serial formula ${}^n\varphi_{\rightarrow}$, the detachment

$$\frac{(\dots((\alpha_0 \models \alpha_1) \models \alpha_2) \dots \models \alpha_{n-1}) \models \alpha_n}{\alpha_0; \alpha_1; \alpha_2; \dots; \alpha_{n-1}; \alpha_n; \alpha_0 \models \alpha_1; (\alpha_0 \models \alpha_1) \models \alpha_2; \dots ((\alpha_0 \models \alpha_1) \models \alpha_2) \dots \models \alpha_{n-1}; {}^n\varphi_{\rightarrow}}$$

delivers $2n + 1$ separate parallel operands (entities), that is, $2n + 1$ parallel informing processors of the machine.

For the circular serial formula ${}^{n+1}\varphi^{\circ}_{\rightarrow}$ the number of processors increases to $2n + 2$.

Through this discussion, the reader can (should) become aware, that a complex graph represents much more of parallelism as it may be

needed in a concrete situation, because it hides also the various possibilities of circularism, by which a graph can be covered, e.g., by the parallelization of a concrete system of formulas.

6.6 Incompletely Structured Graphs

A definition concerning incompletely structured graphs can be useful.

DEFINITION 11 (Incompletely Structured Graph) An *incompletely structured graph* is characterized by several properties when α , β , and γ are operands of the graph:

1. it includes at least one isolated operand α for which neither $\alpha \models \beta$ nor $\gamma \models \alpha$ holds;
2. it includes at least one operand α for which $\alpha \models \beta$ holds (externalistic connectivity) but $\gamma \models \alpha$ does not hold; or
3. it includes at least one operand α for which $\beta \models \alpha$ holds (internalistic connectivity) but $\alpha \models \gamma$ does not hold.

□

6.7 Completely (Circularly) Structured Graphs

Within an informational system, it is sensible to insist, that all graph operands are externalistically as well as and internalistically (mutually) connected.

DEFINITION 12 (Completely Structured Graph) A *completely structured graph* does not include an operand which is externalistically and/or internalistically isolated and there does not exist a loop isolated from other loops of the graph. □

THEOREM 8 (Completely Circularly Structured Graphs) In a *completely structured graph*, each graph operand is circularly connected, and each operand is transitionally (through informational operators and other operands) connected with all the other (remaining) operands of the graph. □

Proof 8 (Circularity of Operands in a Completely Structured Graph) In a completely structured graph all the operands are externalistically and internalistically connected to at least one other

operand of the graph. Let the graph operands be marked by $\alpha_0, \alpha_1, \dots, \alpha_n$. The worst case condition is, for example, that α_i is connected to α_{i+1} , this one to α_{i+2} , and so on, to α_n . Now, let α_n be connected to α_1 , this one to α_2 , and so on, to α_{i-1} . In this way we have exhausted all the available operands of the graph, however, α_{i-1} does not have the connection externalistic to another operand yet. Let it be connected to α_i . In this case, all the operands of the graph are in the longest possible loop of the graph.

Besides the longest loop also any shorter loop is possible and some loops can cover (include) common operands. The common loop operands mean the informational connection of involved loops. The interloop connection must be realized in such a manner that every operand of the graph is indirectly (informationally) connected with the all remaining operands of the graph.

Let graph \mathfrak{G} possess k distinguishable loops λ_ℓ of the set $\Lambda = \{\lambda_1, \lambda_2, \dots, \lambda_k\}$, in which $n + 1$ operands (all operands appearing in graph \mathfrak{G}) of the set $A = \{\alpha_0, \alpha_1, \dots, \alpha_n\}$ occur. Let any two loops, for example $\lambda_{\ell_1}, \lambda_{\ell_2} \in \Lambda$, include some common operands $\alpha_{i_1}, \alpha_{i_2}, \dots, \alpha_{i_q} \in A$, where $q \geq 1$. Let this hold for any two loops and let all the occurring loops cover all the operands of graph \mathfrak{G} . Evidently, under these circumstances, each operand is transitionally connected with the remaining operands of the graph. This means not only that an operand circularly transitionally informs the remaining operands, but also that it is circularly transitionally informed by the remaining operands. □

Evidently, according to the proof, a loop can be arbitrarily structured, from the shortest one, consisting of two operands to the longer ones.

The next request is, that all loops of the graph must be mutually connected, and no loop group must be isolated. For loops of the graph the same holds as for the graph operands. This leads to the conclusion that every operand is in some way circularly connected within the graph.

A completely (circularly) structured graph guarantees an equal informational treatment of any occurring operand and its possibility to become, in some particular context, the title operand of the system, when it observes and is observed as a significant entity, magnifying its phenomenalism within the system presented by the

graph.

7 What Would Mean to Understand an Informational Graph and to Program Its Possibilities?

An informational graph is the most powerful presentation of an instantaneous informational system since it contains all the possibilities of the instantaneous informational system in respect to its (integral) gestalt (gestalts of the possible loops with already occurring graph operands and operators).

To program all such possibilities of an instantaneous graph means simply to process in parallel not only all the operands in parallel, but also to have separate processors for any subformula of the system and, at last, for the system as an entirety. We have seen how a primitive parallel (transition) formula system does not represent (e.g., simulate) only the concrete system of formulas, but also all the other possibilities which the formula system with its operands and operators could represent. This situation is not always an adequate one, although it simulates the all possible situations of the instantaneous system.

The next problem of an informational graph, representing a system, is its emerging, being conditioned by the informational arising of the system it represents. Informational arising is a consequence of different decompositional processes concerning operands (informational entities) together with their operands. It means that a graph changes its graphical structure during the informing of the system. An informational graph is nothing other than a particular (representational) informational entity, which underlies the principles of informational externalism, internalism, and metaphysicalism.

One can imagine a decomposition of a graph in a similar manner as the decomposition of an operand or operator [24, 26]. Between two operands of the graph, connected by an operator arrow, the decomposition means, that instead of a single arrow, a new subgraph arises, being adequately inserted into the graph.

Let $\alpha \models \beta$ be a transition and $\Delta(\alpha \models \beta)$ its decomposition in the form

$$((\dots((\alpha \models \gamma_1) \models \gamma_2) \models \dots \gamma_{n-1}) \models \gamma_n) \models \beta$$

This decomposition of $\alpha \models \beta$ can be grasped in the following way:

1. operand α -decomposition is framed as

$$\boxed{((\dots((\alpha \models \gamma_1) \models \gamma_2) \models \dots \gamma_{n-1}) \models \gamma_n)} \models \beta$$

2. operand β -decomposition is framed as

$$\boxed{((\dots((\alpha \models \gamma_1) \models \gamma_2) \models \dots \gamma_{n-1}) \models \gamma_n) \models \beta}$$

and, finally,

3. operands α - β -decomposition or the original operator \models -decomposition is framed as

$$\boxed{((\dots((\alpha \models \gamma_1) \models \gamma_2) \models \dots \gamma_{n-1}) \models \gamma_n) \models \beta}$$

8 Informational Graphs for Informational Being-in and Informational Being-of

What are informational graphs for informational Being-in and informational Being-of and how could they be reasonably interpreted? In concern to these relatively simple informational cases we can discuss the meaning of the occurring gestalts, circular gestalts, and star gestalts.

8.1 Informational Being-in

In [19], the informational Being-in or informational inclusion (operator \subset) was defined in the following way.

DEFINITION 13 (Informational Includedness) Let the entity α inform within the entity β , that is, $\alpha \subset \beta$. This expression reads: α informs within (is an informational component or constituent of) β . Let the following parallel system of includedness (Being-in) be defined recursively:

$$(\alpha \subset \beta) \Rightarrow_{\text{Def}} \begin{pmatrix} \beta \models \alpha; \\ \alpha \models \beta; \\ \Xi(\alpha \subset \beta) \end{pmatrix}$$

where for the extensional part $\Xi(\alpha \subset \beta)$ of the includedness $\alpha \subset \beta$, there is,

$$\Xi(\alpha \subset \beta) \in \mathcal{P} \left(\left\{ \begin{array}{l} (\beta \models \alpha) \subset \beta, \\ (\alpha \models \beta) \subset \beta, \\ (\beta \models \alpha) \subset \alpha, \\ (\alpha \models \beta) \subset \alpha \end{array} \right\} \right)$$

The most complex element of this power set is denoted by

$$\Xi_{\beta,\alpha}^{\beta,\alpha}(\alpha \subset \beta) \equiv \left((\beta \models \alpha) \subset \beta, \alpha; (\alpha \models \beta) \subset \beta, \alpha \right)$$

Cases, where $\Xi(\alpha \subset \beta) \equiv \emptyset$ and \emptyset denotes an empty entity (informational nothing), are exceptional (reductionistic). \square

If one looks into this definition, irrespective of the complexity and recursiveness of the definition, (s)he can observe the informational interplay solely between two entities, that is, α and β . Informational includedness means that both α and β are under mutual informing and observing. Within this interplay two informational operators appear: \models and \subset .

Let us perform the primitive parallelization Π' of the definition, that is,

$$\Pi' \left(\begin{array}{l} \beta \models \alpha; \\ \alpha \models \beta; \\ \Xi_{\beta,\alpha}^{\beta,\alpha}(\alpha \subset \beta) \end{array} \right) \equiv \left(\begin{array}{l} \beta \models \alpha; \quad \alpha \models \beta; \\ \Xi_{\beta,\alpha}^{\beta,\alpha} \models_{\Psi} \alpha; \quad \alpha \subset \beta \end{array} \right)$$

and

$$\Pi' \left(\Xi_{\beta,\alpha}^{\beta,\alpha}(\alpha \subset \beta) \right) \equiv \left(\begin{array}{l} \beta \models \alpha; \quad \alpha \subset \beta \\ \alpha \subset \alpha; \quad \alpha \models \beta; \\ \beta \subset \beta; \quad \beta \subset \alpha \end{array} \right)$$

This parallelization delivers according to the def-

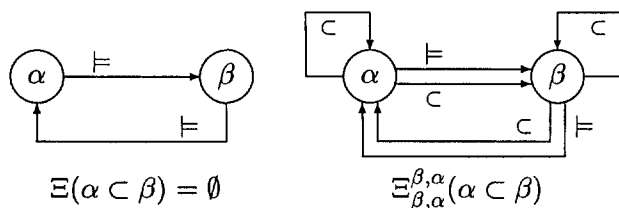


Figure 13: Informational graphs for the simplest and the most complex case of informational includedness.

inition of the informational includedness 16 different graphs, where the minimal and the maximal configuration is presented in Fig. 13.

For the so-called zero or minimal option ($\Xi(\alpha \subset \beta) = \emptyset$), there is,

$$(\alpha \subset \beta) \Rightarrow (\beta \models \alpha; \alpha \models \beta)$$

The remaining Ξ -parts are:

$$\begin{array}{l} \Xi_{\alpha}, \Xi_{\beta}, \Xi_{\beta,\alpha}, \Xi^{\alpha}, \Xi_{\alpha}^{\alpha}, \Xi_{\beta}^{\alpha}, \Xi_{\beta,\alpha}^{\alpha}, \\ \Xi^{\beta}, \Xi_{\alpha}^{\beta}, \Xi_{\beta}^{\beta}, \Xi_{\beta,\alpha}^{\beta}, \Xi^{\beta,\alpha}, \Xi_{\alpha}^{\beta,\alpha}, \Xi_{\beta}^{\beta,\alpha} \end{array}$$

The minimal graph configuration must be included in all the Ξ -parts.

8.2 Informational Being-of

In [20], the informational Being-of or informational function (expressed as $\varphi(\alpha)$) was defined in the following way.

DEFINITION 14 [Informational Function] Let entity φ be an informational function of the entity α , that is, $\varphi(\alpha)$. This expression reads: φ is a function of α . Let the following parallel system of the informational function (Being-of) be defined recursively:

$$\varphi(\alpha) \equiv \left(\begin{array}{l} \varphi \models_{\text{of}} \alpha; \\ \alpha \models \varphi; \\ (\varphi \models_{\text{of}} \alpha) \subset \varphi; \\ (\alpha \models \varphi) \subset_{\text{of}} \varphi \end{array} \right)$$

where, for the first informational includedness of the formula, according to [19], there is

$$((\varphi \models_{\text{of}} \alpha) \subset \varphi) \equiv \left(\begin{array}{l} \varphi \models (\varphi \models_{\text{of}} \alpha); \\ (\varphi \models_{\text{of}} \alpha) \models \varphi; \\ (\varphi \models (\varphi \models_{\text{of}} \alpha)) \subset \varphi; \\ ((\varphi \models_{\text{of}} \alpha) \models \varphi) \subset \varphi \end{array} \right)$$

and, for the second informational includedness, according to [19],

$$((\alpha \models \varphi) \subset_{\text{of}} \varphi) \equiv \left(\begin{array}{l} \varphi \models_{\text{of}} (\alpha \models \varphi); \\ (\alpha \models \varphi) \models_{\text{of}} \varphi; \\ (\varphi \models_{\text{of}} (\alpha \models \varphi)) \subset_{\text{of}} \varphi; \\ ((\alpha \models \varphi) \models_{\text{of}} \varphi) \subset_{\text{of}} \varphi \end{array} \right)$$

\square

This definition recursively determines the parallel informational mechanisms of the informational Being-of, irrespective of the functional-nesting depth. For the complex (recursive) functional def-

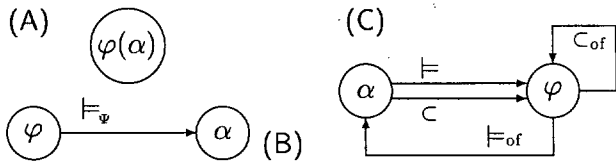


Figure 14: Informational graph for the case of informational function: (A) for $\varphi(\alpha)$, (B) for the functional transition $\varphi \models_{\psi} \alpha$, and (C) for the definition of informational function $\varphi(\alpha)$.

initiation $\varphi(\alpha)$, the standardized informational transition of the form $\varphi \models_{\psi} \alpha$ is introduced, with the meaning

$$\varphi(\alpha) \equiv (\varphi \models_{\psi} \alpha)$$

The definition of $\varphi(\alpha)$ is graphically presented in Fig. 14. Questions which follow are the following: How can α and φ be expressed explicitly in a parallel circular form? Which is the primitive parallel system for the informational function? How complex is the circular gestalt and what is the star gestalt for $\varphi(\alpha)$?

From the graph in Fig. 14, the informational system concerning α is, evidently,

$$\alpha \equiv \left(\begin{array}{ll} (\alpha \models \varphi) \models_{\text{of}} \alpha; & (\alpha \models \varphi) \text{C}_{\text{of}} \varphi; \\ (\alpha \subset \varphi) \models_{\text{of}} \alpha; & (\alpha \subset \varphi) \text{C}_{\text{of}} \varphi \end{array} \right)$$

Similarly, from the system, represented by the graph, functional component φ can be expressed. There is

$$\varphi \equiv \left(\begin{array}{ll} (\varphi \models_{\text{of}} \alpha) \models \varphi; & (\varphi \text{C}_{\text{of}} \varphi) \models_{\text{of}} \alpha; \\ (\varphi \models_{\text{of}} \alpha) \models_{\text{of}} \varphi & \end{array} \right)$$

Now, one can see, how $\varphi(\alpha)$ can be expressed by substitution of φ and α , where the complex operand α (a parallel system of cyclically serially structured formulas) becomes the argument of the similarly complex operand φ (as a parallel system of circularly serially structured formulas). Such a substitution could be quite sensible in case of a particular causal situation.

According to the definition of $\varphi(\alpha)$, there exists a unique parallel primitive system (of basic transitions), marked consequently by $(\varphi(\alpha))'_{\parallel}$, that is,

$$(\varphi(\alpha))'_{\parallel} \equiv \left(\begin{array}{ll} \alpha \models \varphi; & \alpha \subset \varphi; \\ \varphi \models_{\text{of}} \alpha; & \varphi \text{C}_{\text{of}} \varphi \end{array} \right)$$

The circular gestalt must cover the entire graph in Fig. 14 by circularly serial formulas. Thus,

$$\Gamma^{\circ}(\varphi(\alpha)) \equiv \left(\begin{array}{ll} (\alpha \models \varphi) \models_{\text{of}} \alpha; & (\varphi \models_{\text{of}} \alpha) \models \varphi; \\ \alpha \models (\varphi \models_{\text{of}} \alpha); & \varphi \models_{\text{of}} (\alpha \models \varphi); \\ (\alpha \subset \varphi) \models_{\text{of}} \alpha; & (\varphi \models_{\text{of}} \alpha) \subset \varphi; \\ \alpha \subset (\varphi \models_{\text{of}} \alpha); & \varphi \models_{\text{of}} (\alpha \subset \varphi); \\ \varphi \text{C}_{\text{of}} \varphi & \end{array} \right)$$

9 Graphs for Informational Concluding

9.1 Introduction

Informational concluding (modi informationis) was classified, critically discussed, and informationally formalized for the first time in [18]. In this framework, the following modi can be presented systematically and in the form of informational graphs, gestalts, and circular particularities:

1. informational modus agendi (an entity's externalism, internalism, and metaphysicalism);
2. informational implication as a complex structure;
3. informational modus ponens (linear concluding);
4. informational modus tollens (circular concluding);
5. informational modus rectus (detachment of an entity's intention);
6. informational modus obliquus;
7. informational modus procedendi;
8. informational modus operandi (detachment of an entity's informing);
9. informational modus vivendi;
10. informational modus possibilitatis; and
11. informational modus necessitatis.

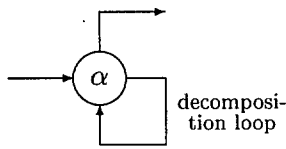


Figure 15: Informational graphs for informational modus agendi with input and output path, and the loop for inner decomposition.

9.2 Graph Investigation for Informational Modus Agendi

The graph for informational modus agendi in Fig. 15 demonstrates the most general phenomenon of informing of entity α . This entity is open to its exterior and interior and it can be innerly decomposed by the informing of its components. There always exists an α -decomposition $\Delta(\alpha)$, which, in general, is a system of formulas (or a simple circular serial formula) in which inner components appears together with α .

Modus agendi of an operand is its possibility to be informed from its exterior by other entities, to inform the exterior components, and to be circularly organized (e.g., metaphysically or otherwise). In an informational graph, the circle or oval represents, in general (in principle) an informational operand (entity) with the presented three categories of arrows (informational operators). Implicitly, the rule of modus agendi, ρ_{MA} , can be formalized as

$$\rho_{MA}(\alpha) \equiv \left(\begin{array}{l} \models \alpha; \\ \alpha \models; \\ \Delta(\alpha \models \alpha) \end{array} \right)$$

This scheme of the operand α presentation is essential for the understanding of the informational operand and informational formulas or formula systems, which are nothing other than forms of operands (informing entities). The parallel formula system ($\models \alpha; \alpha \models; \Delta(\alpha \models \alpha)$) is a form of primitive parallelization of α , that is, $\Pi'(\alpha)$, being equivalent to the graph in Fig. 15.

9.3 Graph Investigation for Informational Modus Ponens

The modus ponens is the most obvious and fully legal rule of inference in mathematics. Its structure realizes logically a conjunction (operator of

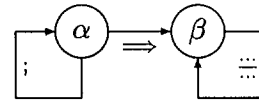


Figure 16: Informational graph for informational modus ponens.

informational parallelism ‘;’), an implication (operator \implies), and the main implication, called the informational detachment (operator in the form of a fraction line, denoted by $\frac{\dots}{\dots}$ in an informational graph). Formally, the rule of informational modus ponens, ρ_{MP} , is a formula of the form

$$\rho_{MP}(\alpha, \beta) \equiv \frac{\alpha; \alpha \implies \beta}{\beta}$$

with the graph in Fig. 16, corresponding to the primitive rule parallelization, that is,

$$\Pi' \left(\frac{\alpha; \alpha \implies \beta}{\beta} \right) \equiv \left(\begin{array}{l} \alpha \models \alpha; \\ \alpha \implies \beta; \\ \frac{\beta}{\beta} \end{array} \right)$$

In this primitive parallelization scheme, the semicolon in the premise was replaced by operator of parallelism \models , to make the primitive formula system more transparent (semicolons are used as separators between elementary transition formulas). By the rule of modus ponens, operand β is detached from the rule premise, that is, follows as a conclusion.

What is characteristic for this rule of inference is its linear serial structure in concern to α , but circular structure in concern to β , being evident from the graph in Fig. 16. It will be presented how other rules of inference are much more informationally circularly structured and, that informational circularity pervades the entire living and artificial (also mathematical) informational realm. Thus, modus ponens, as one of the firmest inference rules in mathematics, is pseudolinear (linear consequential) and, in fact, violates itself (in a hidden form) the mathematical principle of straightforwardness. Namely, the detached operand β is, according to the modus agendi of an informational entity, circularly decomposed in the most consequent form $\frac{\beta}{\beta}$ (or $\beta \implies \beta$).

What could bring a real surprise into the philosophy of modus ponens is a mathematically and informationally legal rearrangement of the rule

premise. The semicolon represents a conjunctive connective in mathematics and a parallel connective in informational theory: both underlie the so-called associative law. This means, that operands on the left and the right of semicolon can be exchanged. This sort of legal conclusion causes a rule of modus ponens in the form

$$\rho_{MP}^1(\alpha, \beta) \Rightarrow \frac{\alpha \Rightarrow \beta; \alpha}{\beta}$$

If this rule with exchanged premise parts is mathematically unacceptable, it just means that behind the philosophy of modus ponens something remains unexplained (hidden, unrevealed), and that the semicolon, also in mathematics, occupies an loosely determined logical connective (separator), being formally excluded from the valid (transparent) field of legal intelligibility. It should mean that parallelism does not enter into the regular mathematical discourse, although it functions as the most normal phenomenon in logical concluding.

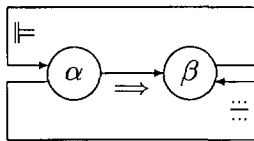


Figure 17: Informational graph for another formal interpretation of informational modus ponens.

This disputable rule of modus ponens delivers the graph in Fig. 17 or the primitive parallelization scheme

$$\Pi' \left(\frac{\alpha \Rightarrow \beta; \alpha}{\beta} \right) \Rightarrow \left(\begin{array}{l} \alpha \Rightarrow \beta; \\ \beta \models \alpha; \\ \frac{\alpha}{\beta} \end{array} \right)$$

As the reader can see, the local (inner) loops for operands α and β disappeared, and instead of them two other loops including both operands α and β appear. Precisely the graphs for the first and the second formal presentation of modus ponens explicate the essential difference of them and bring up the dispute of the ultimate and everlasting validity of this type of concluding.

A radically different concept of modus ponens concerns the decomposition of both operands α

and β . As well α as β hide a decompositional nature Δ_α and Δ_β , which decide about the necessary informational (modus ponens) adequateness between α and β . Thus, the new inference rule $\rho_{MP}^2(\alpha, \beta, \Delta_\alpha, \Delta_\beta)$ must be understood as

$$\rho_{MP}^2(\alpha, \beta, \Delta_\alpha, \Delta_\beta) \Rightarrow \frac{\alpha; \Delta_\alpha(\alpha) \Rightarrow \Delta_\beta(\beta)}{\beta}$$

The graph for this rule of modus ponens is presented in Fig. 18.

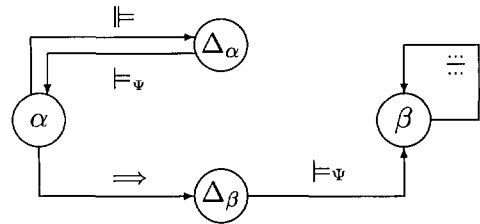


Figure 18: Informational graph for decompositional formal interpretation of informational modus ponens.

9.4 Graph Investigation for Informational Modus Tollens

The informational modus tollens already arises from a more informationally slippery ground, with interweaved informational loops which sometimes might perform explicitly tautologically. Two possible graphs for modus tollens are presented in Fig. 19

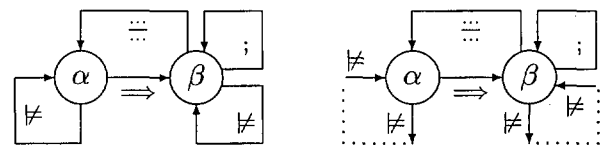


Figure 19: Informational graphs for informational modus tollens.

9.5 Graph for Informational Modus Rectus

The aim of informational modus rectus ρ_{MR} is detaching intention $\iota_{intention}$ of an informational entity α within a transitional intentional informing of α to an entity β . E.g., an exterior observer, say β , of α who observes the α 's intentional informing of β , comes to the conclusion that there exists an α 's intention in the informing

by which it informationally impacts β . The result of this intentional informing is observed (seeable, comprehensible, intelligible) in β as an informational Being-in [19]. Intention of entity α appears as a general circular decomposition, e.g. as $\Delta_{\rightarrow}^{\circ}(l_{\text{intention}}(\alpha))$ or as a metaphysicalistic decomposition $M_{\rightarrow}^{\circ}(l_{\text{intention}}(\alpha))$.

The rule of informational modus rectus has the form

$$\rho_{\text{MR}}(\alpha, \beta, l_{\text{intention}}, \Delta_{\rightarrow}^{\circ}) = \frac{\alpha; ((\alpha \models_{\text{intend}} \beta) \Rightarrow l_{\text{intention}}(\alpha))}{\Delta_{\rightarrow}^{\circ}(l_{\text{intention}}(\alpha)) \subset \beta}$$

The graph for this rule is presented in Fig. 20. Primitively parallelization

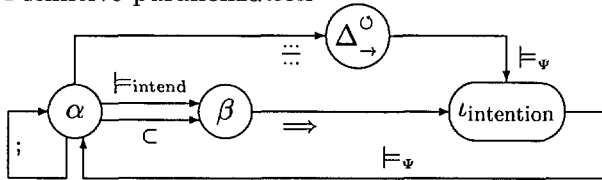


Figure 20: Informational graph for informational modus rectus.

$$\Pi'(\rho_{\text{MR}}(\alpha, \beta, l_{\text{intention}}, \Delta_{\rightarrow}^{\circ})) = \left(\begin{array}{ll} \alpha \models \alpha; & \alpha \models_{\text{intend}} \beta; \\ \beta \Rightarrow l_{\text{intention}}; & l_{\text{intention}} \models_{\psi} \alpha; \\ \frac{\alpha}{\Delta_{\rightarrow}^{\circ}}; & \Delta_{\rightarrow}^{\circ} \models_{\psi} l_{\text{intention}}; \\ l_{\text{intention}} \models_{\psi} \alpha; & \alpha \subset \beta \end{array} \right)$$

is an exact description of the graph in Fig. 20. One can see how modus rectus emerges through a particular moving along the graph paths (arrows). This formula can be understood as the one of the possibilities of belonging to the infinite star gestalt formula system, which arises through all the possible moving along the graph paths.

Many other rules for detaching (extracting) the intentional informing of an entity could be constructed considering various views and beliefs of intentionality as an informational phenomenalism.

9.6 Graph for Informational Modus Obliquus

An *informational modus obliquus* [18, 23] is a broad informational concept for concluding and

inference which has its roots in the Latin conversation practice (e.g., slanting, sideways, oblique, *indirect*, covert, envious discourse). The modus obliquus (MO) concerns indirect adjustment of an absurdly experienced, individually motivated, felt, emotional, interested, etc. consciousness informing. MO as informational detachment of an informational item out of an absurd, disapproved, distrust informational situation uses the inference and concluding forms and processes, even in its conscious realm of informing, in the realm of unawareness, illiteracy, doubt, and falsity. This is the absurd attitude of the MO itself with the aim to surpass the absurdness of a complex informational situation.

The reader will agree that setting an informational graph for various possibilities of MO would require a separate and exhaustive study. But some simplified concepts of MO can be presented by informational graphs and the corresponding formula systems.

As a form of the rule using indirect informational content and meaning, MO obviously deviates from a direct or intentional line (e.g., the line with modus rectus) of discourse, performing roundabout or not going straight to the point. In this respect it performs within a speech game in which behind-the-scenes intentions, views, and purposes remain hidden and must not be disclosed (e.g., in political, ideological, clan-like, deceptive public discourse). As an indirect form of inference, it involves concluding with “commonly noninforming” (secretly, unconsciously informing) entities. But on the other side, right in this manner, MO can reveal information being not openly shown to the participants of a complex, yet not essentially disclosed discourse.

Let a mark a complex controversial and unexamined informational entity. The controversy of a means that a clear absurd informational item b_{absurd} in a can be identified. In this situation, b_{absurd} has to be informationally decomposed (analyzed, synthesized, interpreted, transformed) in the circular form $\Delta_{\rightarrow}^{\circ}(b_{\text{absurd}}(a))$ with the aim to deliver the conclusion $c_{\text{conclusion}}$ in such a way that absurd is informationally included in the conclusion. In this way, the possible scheme for $c_{\text{conclusion}}$ detachment from the controversial origin entity a is

$$\rho_{MO} \left(a, b_{\text{absurd}}, c_{\text{conclusion}}, \Delta_{\rightarrow}^{\circ}, N_{\rightarrow}^{\circ} \right) \Rightarrow \frac{\begin{array}{l} (a \models_{\text{absurdly}} b_{\text{absurd}}(a)) \subset a; \\ \Delta_{\rightarrow}^{\circ}(b_{\text{absurd}}(a)) \Rightarrow c_{\text{conclusion}} \end{array}}{N_{\rightarrow}^{\circ}(b_{\text{absurd}}(a)) \subset c_{\text{conclusion}}}$$

In this rule, N marks the so-called negated circular serial decomposition of the absurd informational entity $b_{\text{absurd}}(c)$. This circular decomposition is something like

$$(\dots ((b_{\text{absurd}}(a) \not\models \alpha_1(a)) \not\models \alpha_2(a)) \dots \not\models \alpha_n(a)) \not\models b_{\text{absurd}}(a)$$

where some derivatives $\alpha_i(a)$ informationally counterinform (informationally oppose) in respect to $b_{\text{absurd}}(a)$. It means that this decomposition is a circular serial function of the form denoted by

$$\overline{\varphi_{j_0}^{\circ}}(b_{\text{absurd}}(a), \alpha_1(a), \alpha_2(a), \dots, \alpha_n(a))$$

with $1 \leq j_0 \leq \frac{1}{n+2} \binom{2n+2}{n+1}$. Now the reader can grasp the importance of the other such functions resulting as a consequence of an operand rotation to the title place. Some of these operands can represent a direct answer (in fact, counter-answer) in regard to the absurd situation $b_{\text{absurd}}(a)$. Thus,

$$\overline{\varphi_{j_i}^{\circ}}(\alpha_i(a), \alpha_{i+1}(a), \dots, \alpha_n(a), b_{\text{absurd}}(a), \alpha_1(a), \alpha_2(a), \dots, \alpha_{i-1}(a))$$

The primitive circular parallel system

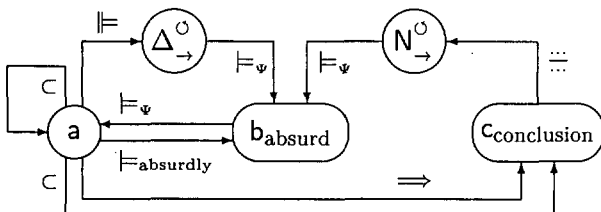


Figure 21: An informational graph for possible informational modus obliquus.

$$\Pi' \left(\rho_{MO} \left(a, b_{\text{absurd}}, c_{\text{conclusion}}, \Delta_{\rightarrow}^{\circ}, N_{\rightarrow}^{\circ} \right) \right) \Rightarrow \left(\begin{array}{l} a \models_{\text{absurdly}} b_{\text{absurd}}; \quad b_{\text{absurd}} \models_{\psi} a; \\ a \subset a; \quad a \models \Delta_{\rightarrow}^{\circ}; \\ \Delta_{\rightarrow}^{\circ} \models_{\psi} b_{\text{absurd}}; \quad b_{\text{absurd}} \models_{\psi} a;^{(*)} \\ a \Rightarrow c_{\text{conclusion}}; \quad \frac{c_{\text{conclusion}}}{N_{\rightarrow}^{\circ}}; \\ N_{\rightarrow}^{\circ} \models_{\psi} b_{\text{absurd}}; \quad b_{\text{absurd}} \models_{\psi} a;^{(**)} \\ a \subset c_{\text{conclusion}} \end{array} \right)$$

is the exact description of the informational graph in Fig. 21. Formulas marked by (*) and (**) in the primitive system are superfluous and are used solely for seeing the circular continuity of the system.

9.7 Graph for Informational Modus Procedendi

A goal or aim of informing of an entity, as something essentially different in regard to informational intention, can become the subject of the detachment, for instance, in a strategic environment. The question is what could a system of goals within a strategy be at all? The Latin *procedo* has the meaning of *to go forth or before, advance, make progress; to continue, remain; and to go on*.

Let an informational strategy entity s include a hidden goal system g_{goal} . Strategy s causes a system of consequences $c(s)$, elsewhere, such that $c(s)$ can be transparently observed. The goal system g_{goal} performs as a cause of $c(s)$, that is, as a system of the form

$$c(s) \subset s; c(s) \models_{\text{cause}} g_{\text{goal}}$$

The \models_{cause} operator has to be particularized to operator $\Rightarrow_{\text{cause}}$ with the meaning *implicatively causes*.

Detachment of g_{goal} must remain within a careful (goal-consequent) decomposition G_{\rightarrow}° of consequences $c, (s,)$, that is, as $G_{\rightarrow}^{\circ}(c(s))$.

Modus procedendi is a rule (ρ_{MPr}) for the detachment of goal-directed organization g_{goal} of the strategy informing entity s , through the observing of $c(s)$. Thus,

$$\rho_{MPr} \left(s, \mathfrak{g}_{goal}, c, G_{\rightarrow}^{\circ} \right) \equiv \frac{\mathfrak{g}_{goal} \subset s; \mathfrak{g}_{goal} \Rightarrow_{cause} G_{\rightarrow}^{\circ}(c(s))}{\mathfrak{g}_{goal}}$$

The graph for this inference rule is presented in Fig. 22. Primitive parallelization of the rule gives

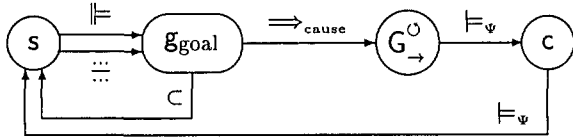


Figure 22: Informational graph for informational modus procedendi.

$$\Pi' \left(\rho_{MPr} \left(s, \mathfrak{g}_{goal}, c, G_{\rightarrow}^{\circ} \right) \right) \equiv \left(\begin{array}{l} \mathfrak{g}_{goal} \subset s; \quad s \models \mathfrak{g}_{goal}; \\ \mathfrak{g}_{goal} \Rightarrow_{cause} G_{\rightarrow}^{\circ}; \quad G_{\rightarrow}^{\circ} \models_{\Psi} c; \\ c \models_{\Psi} s; \quad \frac{s}{\mathfrak{g}_{goal}} \end{array} \right)$$

A more firm and direct inference rule would be that of modus ponens, e.g.,

$$\rho_{MP} (s, \mathfrak{g}_{goal}) \equiv \frac{s; s \Rightarrow \mathfrak{g}_{goal}}{\mathfrak{g}_{goal}}$$

from which c and G_{\rightarrow}° are excluded. But with modus rectus the preceding informational components can be considered in a slightly different way (see Fig. 23, for instance, in the form

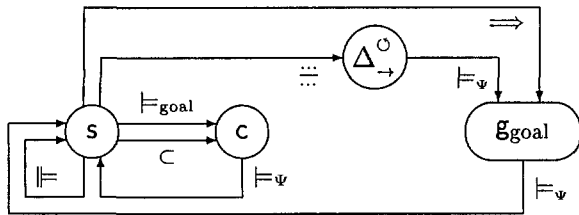


Figure 23: Informational graph for the case when modus procedendi is replaced by slightly modified modus rectus.

$$\rho_{MR} \left(s, c, \mathfrak{g}_{goal}, \Delta_{\rightarrow}^{\circ} \right) \equiv \frac{s; ((s \models_{\mathfrak{g}_{goal}} c(s)) \Rightarrow \mathfrak{g}_{goal}(s))}{\Delta_{\rightarrow}^{\circ}(\mathfrak{g}_{goal}(s)) \subset c(s)}$$

Here, decomposition $\Delta_{\rightarrow}^{\circ}$ is a circular serial decomposition concerning the goal as a function of the strategy, that is, as $\mathfrak{g}_{goal}(s)$. Parallelization of the presented rule for modus rectus is

$$\Pi' \left(\rho_{MR} \left(s, c, \mathfrak{g}_{goal}, \Delta_{\rightarrow}^{\circ} \right) \right) \equiv \left(\begin{array}{l} s \models s; \quad s \models_{\mathfrak{g}_{goal}} c; \\ c \models_{\Psi} s; \quad s \Rightarrow \mathfrak{g}_{goal}; \\ \mathfrak{g}_{goal} \models_{\Psi} s; \quad \frac{s}{\Delta_{\rightarrow}^{\circ}}; \\ \Delta_{\rightarrow}^{\circ} \models_{\Psi} \mathfrak{g}_{goal}; \quad \mathfrak{g}_{goal} \models_{\Psi} s; \\ s \subset c; \quad c \models_{\Psi} s \end{array} \right)$$

9.8 Graph for Informational Modus Operandi

An *informational modus operandi* detaches the (inner, own) operational capabilities of an entity α in the form of an informational function of the entity in the general form $\varphi(\alpha)$ (e.g., informational Being-of in [20]). This function is usually called the entity informing and marked by \mathcal{I}_{α} . But informing \mathcal{I}_{α} , as an operational property of the entity α , performs by itself as an informational entity within α , as an α 's includedness, $\mathcal{I}_{\alpha} \subset \alpha$ (e.g., informational Being-in in [19]).

A modus operandi concerns the decomposition of an entity in respect of its interior informing. Informing \mathcal{I}_{α} is only the initial step in the decomposition process when circular informing of the form $(\alpha \models \mathcal{I}_{\alpha}) \models \alpha$ and/or $\alpha \models (\mathcal{I}_{\alpha} \models \alpha)$ comes to the surface. The deeper operational detachment of further inner components of α can deliver not only the other inner components but also the informational structure (formula) of them. One of the possible forms of the operational detachment concerning an entity is the so-called rule $\rho_{MOp}^{\mu_j}(\alpha, \mathcal{I}_{\alpha}, i_{\alpha}, \mathcal{C}_{\alpha}, c_{\alpha}, \mathcal{E}_{\alpha}, e_{\alpha})$ of metaphysicalistic modus operandi (metaphysicalistic decomposition M_{\rightarrow}° or μ_j -decomposition) in the form

$$\rho_{MOp}^{\mu_{j_1}}(\alpha, \mathcal{I}_{\alpha}, i_{\alpha}, \mathcal{C}_{\alpha}, c_{\alpha}, \mathcal{E}_{\alpha}, e_{\alpha}) \equiv \frac{\alpha; \alpha \Rightarrow \left(\begin{array}{l} ((((((\alpha \models \mathcal{I}_{\alpha}) \models i_{\alpha}) \models \mathcal{C}_{\alpha}) \models \\ c_{\alpha}) \models \mathcal{E}_{\alpha}) \models e_{\alpha}) \models \alpha \end{array} \right)}{\mathcal{I}_{\alpha}, i_{\alpha}, \mathcal{C}_{\alpha}, c_{\alpha}, \mathcal{E}_{\alpha}, e_{\alpha}}$$

where the number j_1 of possible causal interpretations (because of the definite particularization of operators '(', ')', \Rightarrow , and \vdots and the rule firmness)

varies within the interval $1 \leq j_1 \leq 132 (= \frac{1}{7} \binom{12}{6})$. The possible interpretations concern the metaphysicalistic formula ${}^7\mu_{\rightarrow}^{\circ}(\alpha, \mathcal{I}_{\alpha}, i_{\alpha}, \mathcal{C}_{\alpha}, c_{\alpha}, \mathcal{E}_{\alpha}, e_{\alpha})$ (metaphysicalistic symbol μ comes instead of φ for a general symbol⁷).

In fact, the metaphysicalistic components are detached from several loops existing within (in the framework of) the main (one- or two-directional) loop [24], and also from the other parallel loops which particularly (characteristically) concern the interior informing of the entity. In this manner, the detachment of the inner operational components can not only be accomplished but further informationally refined within the process of informational arising, considering the ongoing informational happening, appearing, and phenomenalizing of the entity under the investigation. Thus, according to [25], the improved detachment of an one-directional metaphysicalism (six loops) of the entity delivers

$$\alpha; \alpha \implies \left(\begin{array}{c} ((((((\alpha \models \mathcal{I}_{\alpha}) \models i_{\alpha}) \models \mathcal{C}_{\alpha}) \models c_{\alpha}) \models \mathcal{E}_{\alpha}) \models e_{\alpha}) \models \alpha; \\ (((\mathcal{I}_{\alpha} \models i_{\alpha}) \models \mathcal{C}_{\alpha}) \models c_{\alpha}) \models \mathcal{I}_{\alpha}; \\ (((\mathcal{C}_{\alpha} \models c_{\alpha}) \models \mathcal{E}_{\alpha}) \models e_{\alpha}) \models \mathcal{C}_{\alpha}; \\ (\mathcal{I}_{\alpha} \models i_{\alpha}) \models \mathcal{I}_{\alpha}; \\ (\mathcal{C}_{\alpha} \models c_{\alpha}) \models \mathcal{C}_{\alpha}; \\ (\mathcal{E}_{\alpha} \models e_{\alpha}) \models \mathcal{E}_{\alpha} \end{array} \right) \\ \mathcal{I}_{\alpha}, i_{\alpha}, \mathcal{C}_{\alpha}, c_{\alpha}, \mathcal{E}_{\alpha}, e_{\alpha}$$

This rule of the multiloop detachment could be systematically marked, according to the six loops, as

$$\rho_{\text{MOP}}^{\mu_{j_1}, \mu_{j_{11}}, \mu_{j_{12}}, \mu_{j_{13}}, \mu_{j_{14}}, \mu_{j_{15}}}(\alpha, \mathcal{I}_{\alpha}, i_{\alpha}, \mathcal{C}_{\alpha}, c_{\alpha}, \mathcal{E}_{\alpha}, e_{\alpha})$$

where the second subscript q of μ_{j_pq} corresponds to a subloop in the main loop.

There is to stress that operands below the detachment line are separated by commas. This means that each of operands is detached separately, and that there is not meant the parallel system of the form $\mathcal{I}_{\alpha}; i_{\alpha}; \mathcal{C}_{\alpha}; c_{\alpha}; \mathcal{E}_{\alpha}; e_{\alpha}$. This situation causes a complex and circularly interweaved graph structure (not so easily drawn on a piece of paper). Thus,

⁷Metaphysicalistic formula ${}^7\mu_{\rightarrow}^{\circ}(\alpha, \mathcal{I}_{\alpha}, i_{\alpha}, \mathcal{C}_{\alpha}, c_{\alpha}, \mathcal{E}_{\alpha}, e_{\alpha})$ already considers the semantically determined components (operands) of informing, counterinforming, and embedding.

$$\frac{\alpha}{\beta, \gamma} = \left(\frac{\alpha}{\beta}; \frac{\alpha}{\gamma} \right)$$

where detachments $\frac{\alpha}{\beta}$ and $\frac{\alpha}{\gamma}$ are understood as the basic transitions at the parallelization of the rule. The system of the basic transitions for the discussed metaphysicalistic modus operandi is

$\alpha \models \alpha, \mathcal{I}_{\alpha}, \mathcal{C}_{\alpha}, \mathcal{E}_{\alpha};$	$\alpha \implies \alpha, \mathcal{I}_{\alpha}, \mathcal{C}_{\alpha}, \mathcal{E}_{\alpha};$
$\mathcal{I}_{\alpha} \models \mathcal{C}_{\alpha}, \mathcal{E}_{\alpha}; c_{\alpha} \models \mathcal{E}_{\alpha};$	$\frac{\alpha, \mathcal{I}_{\alpha}, \mathcal{C}_{\alpha}, \mathcal{E}_{\alpha}}{\mathcal{I}_{\alpha}, i_{\alpha}, \mathcal{C}_{\alpha}, c_{\alpha}, \mathcal{E}_{\alpha}, e_{\alpha}};$
$\alpha \models \mathcal{I}_{\alpha}; \mathcal{I}_{\alpha} \models i_{\alpha};$	$i_{\alpha} \models \mathcal{C}_{\alpha}; \mathcal{C}_{\alpha} \models c_{\alpha};$
$c_{\alpha} \models \mathcal{E}_{\alpha}; \mathcal{E}_{\alpha} \models e_{\alpha};$	
$\mathcal{E}_{\alpha} \models \alpha; c_{\alpha} \models \mathcal{I}_{\alpha};$	$e_{\alpha} \models \mathcal{C}_{\alpha}; i_{\alpha} \models \mathcal{I}_{\alpha};$
$c_{\alpha} \models \mathcal{C}_{\alpha}; e_{\alpha} \models \mathcal{E}_{\alpha}$	

The system in the first frame includes 35 basic transitions and represents the part of the rule outside the formula ${}^7\mu_{\rightarrow}^{\circ}(\alpha, \mathcal{I}_{\alpha}, i_{\alpha}, \mathcal{C}_{\alpha}, c_{\alpha}, \mathcal{E}_{\alpha}, e_{\alpha})$, with exception of parallel (semicolon) operators of the formula. In the second frame the “feedback” transitions of the formula ${}^7\mu_{\rightarrow}^{\circ}$ occur. Finally, on the basis of the described parallel system, the graph of the rule can be drawn.

9.9 Graph for Informational Modus Vivendi

An *informational modus vivendi* [18] concerns the information of life (e.g., autopoietic informational entity α) in environmental, individual, technological, and social circumstances. Informational problems of social transition [26] are typical forms of modi vivendi when the governing totalitarian ideological pattern of understanding has to be replaced with a more flexible and life-suited paradigm of social and environmental survival.

The basic living information—conscious as subconscious—existing everywhere the life arises, may be recognized as autopoietic or self-organizing information. It does not organize only the organism for the suited behavior in life circumstances but organizes, through the pressure of the environment, also the information itself for a proper organism behavior, for instance, building up the so-called metaphysicalism μ of autopoietic entity α , marked μ_{α} or together with sensory informational entity σ_{α} .

It is to understand that, in the beginning, α and σ_α impact the arising of μ_α , and then μ_α essentially impacts the emerging of both α and σ_α , thus a basic circular system of the form

$$((\alpha \models \sigma_\alpha) \models \mu_\alpha(\sigma)) \models \alpha$$

is reasonable. Within this cycle, metaphysicalism $\mu_\alpha(\sigma)$ is a constitutive part of α , e.g. $\mu_\alpha(\sigma)$, which has to be extracted from α by a modus vivendi, being essential for survival and adaptation of α in the environment. This rule must belong to α itself as a particular informational entity, being able to identify instantaneous $\mu_\alpha(\sigma)$ during its emerging in life circumstances. This rule is

$$\rho_{MV}(\alpha, \sigma_\alpha, \mu_\alpha(\sigma)) \equiv \frac{(\alpha, \sigma_\alpha); (\alpha, \sigma_\alpha \models \mu_\alpha(\sigma))}{\mu_\alpha(\sigma)}$$

The graph for this rule is presented in Fig. 24. It

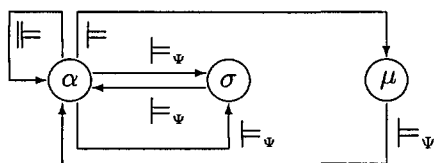


Figure 24: Informational graph for the discussed modus vivendi.

is an initial scheme only which can be decomposed to further details (informationally particularized).

A logically more consistent (rationalistic) form of modus vivendi would, for instance, require a rule of the form

$$\frac{(\alpha, \sigma_\alpha); (\alpha, \sigma_\alpha \implies \mu_\alpha(\sigma))}{\mu_\alpha(\sigma)}$$

representing a kind of modus ponens in the framework of modus vivendi.

9.10 Graph for Informational Modus Necessitatis

By a necessity, the ‘must’ is compelled. In this respect, the meaning of the verb *must* becomes necessity by itself in the realm of the theory of informational. Necessity as informational phenomenon is a pressure of circumstances, which can be grasped also as an essential impossibility of a contrary informational position. It appears as an urgent informational experience, emotion,

memory, need and desire (the consciousness of the must), in such a way, that it as a particular informational entity cannot inform outside itself, that is, in another direction. Within the causalism, necessity can be comprehended as an inevitable informational consequence.

Modus necessitatis is that principle of inference which in several situations can coincide with discussed principles hidden in other modi informatio-nis. Intentionality, which comes to the surface in modus ponens and modus rectus, can be grasped as a particular case of necessity, being consistently bound to the so-called logical mind of human, in the sense of the “best” rationalistic tradition. A more detailed discussion (also formalized) is given in [18]. The attentive reader is already able to proceed into the philosophy of informational, its formalization, and construction of informational graph by himself/herself.

9.11 Graph for Informational Modus Possibilitatis

Possibility is a modal informational phenomenon which opposes the instantaneous reality (essentialness, existentiality) and necessity. A modality (potentiality) by itself is a mood of revealing of experience, emotion, consciousness, and in more general form, of Being, thinking, occurrence; it is a mood of game with conditionalities.

In logic, modality of propositions means the degree of trustability of propositions in regard to possibility, existence, and to necessity. Such a proposition of the possibility is, for instance, $\alpha \models_{\text{can_be}} \beta$, read as α can be β , or informationally consequently as α informs that there could be β . An asserting (existential) proposition is, for example, $\alpha = \beta$ with the meaning α is β or, consequently formally, $\alpha \models \beta$. An apodictic (necessity) proposition is $\alpha \models_{\text{must_be}} \beta$.

Modus possibilitatis is that modus which opens the realm of informational potentiality, including views of exaggeration, inauthenticity, unreliability, controversialism, but also unreasonableness, insolence, and mania. All this concerns the informationally active (intense) and quality creative possible consciousness searching. In this respect, modus possibilitatis can become bound to modus obliquus and modus vivendi, but also to the non-informing (negative) possibilities of modus necessitatis. A basic rule for such a kind of inference

is described in [18] in the form

$$\frac{(\alpha, \beta); (\alpha, \beta \models \alpha)}{\gamma \models_{\pi} \gamma_1, \gamma_2, \dots, \gamma_n}$$

where α is the subject entity, β its exterior impact, γ an entity induced as possibility, components $\gamma_1, \gamma_2, \dots, \gamma_n$ its informational derivatives, while \models_{π} represents a possible informing (possibility operator).

10 An Informational Case for Strategy Decision Making

Maruyama (1993) has invented a practical computer supported scheme for simulation of strategy decision making for business executives and governmental planners. Let us show this concept of a typical computer supported simulation program in the realm of an informational formula system and the corresponding graph, where substantial conceptual changes in understanding the informational nature of the decision making problem take place.

The informing scheme of the problem can be presented in an informationally condensed form in Fig. 25 and parallelized according to the particular entities $e, f, g, h, j, k, m, n, r, v, x, z$ ⁸. According to the graph in Fig. 25, the adequate formal description of the graph \mathcal{G} becomes, considering the gestalt Γ of the formula system,

$$\mathcal{G} \left(\Gamma \left(\begin{array}{l} \models e, f, g, h, j, k, m, n, r, v, x, z; \\ e, f, g, h, j, k, m, n, r, v, x, z \models; \\ \left(\begin{array}{l} (e; \\ f; \\ h; \\ k \end{array} \right) \models j \end{array} \right) \models \left(\begin{array}{l} f; \\ g; \\ h; \\ m; \\ e \end{array} \right); \\ \left(\begin{array}{l} (h; \\ r; \\ x \end{array} \right) \models z \end{array} \right) \models \left(\begin{array}{l} e; \\ m; \\ v \end{array} \right); \\ (v \models n) \models \left(\begin{array}{l} x; \\ g \end{array} \right); \left(\begin{array}{l} g; \\ k \end{array} \right) \models k; \\ (m \models f) \models r \end{array} \right)$$

The scheme in Fig. 25 presents a complex and completely circularly structured informational graph (Subsection 6.7, with additional input and output paths for each involved entity).

11 An Example of Association Graph

Informational connection means entities being operationally linked and joined together through causal or logical relations or sequences. By informational *association* such a connection between informational entities is meant which has a relation in similarity, contrariety, contiguity, causation, etc. For example, a thought (process) is linked in the mind or memory with the other thoughts in the process of thinking (e.g. associative components, an associative system, encoding consciously apprehended information, in [10], pp. 137–138). Further, the sensory and the motor areas of the cortex are supposed to be connected with ideation, etc. For these associative phenomena a general type of the informational circular graph can be constructed in which the so-called parallel association arrays occur as presented in Fig. 26.

Let us describe the graph in Fig. 26 in more detail, concerning a process of thinking association. Let be given a sentence in which operands (language entities like nouns, adjectives, pronouns, etc.)

⁸Maruyama (1993) has given the following meaning to the operand markers (informational entities in our case): e —exchange rate (value of county’s own currency); f —inflation (price); g —government subsidy of inefficient firms; h —import restriction; j —amount of import; k —efficiency of business; m —money supply; n —nationalism; r —interest rate; v —international balance of payment; x —restriction of investment from foreign countries; z —foreigners’ deposits in banks and purchase of government securities, stocks and other investment. As understood, the model was reduced to the corresponding numerical values and by three influence parameters (particularized operators between operands) by which the character of the impact onto the informed operand has been qualified.

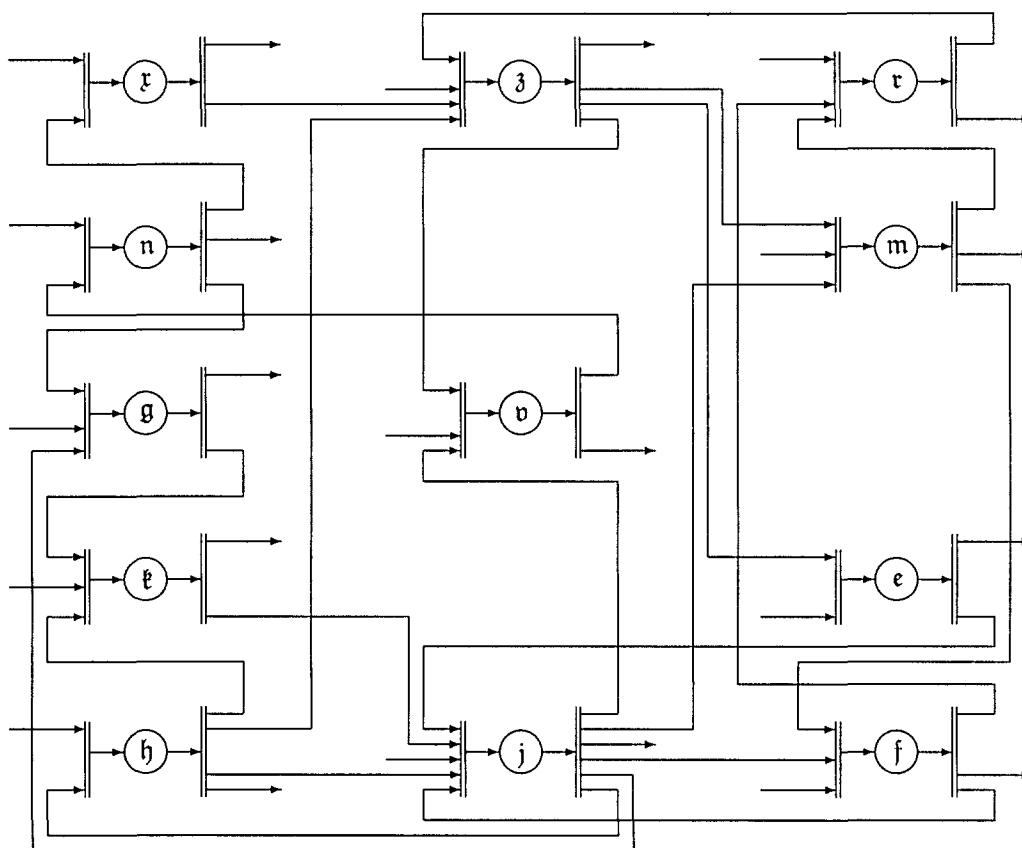


Figure 25: A parallelized scheme of the graph concerning an example of strategic decision making, according to Maruyama (1993), and with explicit input and output informing of the 12 involved informational entities, e, f, g, h, j, k, m, n, t, v, r, z.

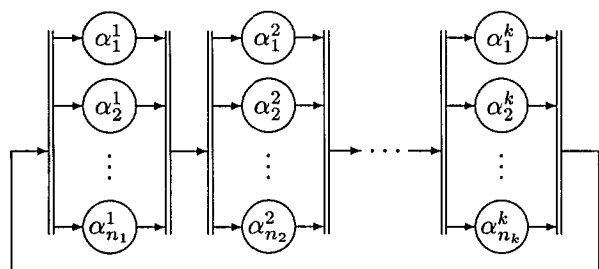


Figure 26: The graph of parallel associative arrays in an associative serial loop.

$$\begin{aligned} &\alpha_1^1, \alpha_2^1, \dots, \alpha_{n_1}^1, \\ &\alpha_1^2, \alpha_2^2, \dots, \alpha_{n_2}^2, \\ &\vdots \\ &\alpha_1^k, \alpha_2^k, \dots, \alpha_{n_k}^k \end{aligned}$$

perform as certain sentence words, but between them the operators (language entities like verbs, adverbs, prepositions, etc.) are set. In the graph,

operators are presented by vectors and the nature (meaning) of them depends on operands which they connect.

A strict causal scheme of an associative mediation of the sentence, presented by the graph in Fig. 26 is

$$\left(\left(\dots \left(\begin{pmatrix} \alpha_1^1; \\ \alpha_2^1; \\ \vdots \\ \alpha_{n_1}^1 \end{pmatrix} \models \begin{pmatrix} \alpha_1^2; \\ \alpha_2^2; \\ \vdots \\ \alpha_{n_2}^2 \end{pmatrix} \right) \models \dots \left(\begin{pmatrix} \alpha_1^{k-1}; \\ \alpha_2^{k-1}; \\ \vdots \\ \alpha_{n_{k-1}}^{k-1} \end{pmatrix} \right) \models \begin{pmatrix} \alpha_1^k; \\ \alpha_2^k; \\ \vdots \\ \alpha_{n_k}^k \end{pmatrix} \right) \models \begin{pmatrix} \alpha_1^1; \\ \alpha_2^1; \\ \vdots \\ \alpha_{n_1}^1 \end{pmatrix} \right)$$

As one can see, while the associationism of the operands α_j^i is explicit, the associationism of operators \models remains implicit and depends on the chosen operands, also on well-formed connections of operands, which the operator connects. It is

certainly possibly to foresee the adequate grouped associative operators. Such an operators associative scheme would take the form

$$\left(\left(\dots \left(\begin{pmatrix} \alpha_1^1; \\ \alpha_2^1; \\ \vdots \\ \alpha_{n_1}^1 \end{pmatrix} \begin{pmatrix} \models_1^1 \\ \models_2^1 \\ \vdots \\ \models_{m_1}^1 \end{pmatrix} \begin{pmatrix} \alpha_1^2; \\ \alpha_2^2; \\ \vdots \\ \alpha_{n_2}^2 \end{pmatrix} \right) \begin{pmatrix} \models_1^2 \\ \models_2^2 \\ \vdots \\ \models_{m_2}^2 \end{pmatrix} \dots \right. \right.$$

$$\left. \begin{pmatrix} \alpha_1^{k-1}; \\ \alpha_2^{k-1}; \\ \vdots \\ \alpha_{n_{k-1}}^{k-1} \end{pmatrix} \begin{pmatrix} \models_1^k \\ \models_2^k \\ \vdots \\ \models_{m_k}^k \end{pmatrix} \begin{pmatrix} \alpha_1^k; \\ \alpha_2^k; \\ \vdots \\ \alpha_{n_k}^k \end{pmatrix} \right) \begin{pmatrix} \models_1^1 \\ \models_2^1 \\ \vdots \\ \models_{m_1}^1 \end{pmatrix} \begin{pmatrix} \alpha_1^1; \\ \alpha_2^1; \\ \vdots \\ \alpha_{n_1}^1 \end{pmatrix}$$

$$\alpha_{j_1}^{i_1} \in \{\alpha_1^1, \alpha_2^1, \dots, \alpha_{n_1}^1\};$$

$$\models_{q_1}^{p_1} \in \{\models_1^1, \models_2^1, \dots, \models_{m_1}^1\};$$

$$\alpha_{j_2}^{i_2} \in \{\alpha_1^2, \alpha_2^2, \dots, \alpha_{n_2}^2\};$$

$$\models_{q_2}^{p_2} \in \{\models_1^2, \models_2^2, \dots, \models_{m_2}^2\};$$

$$\vdots$$

$$\alpha_{j_k}^{i_k} \in \{\alpha_1^k, \alpha_2^k, \dots, \alpha_{n_k}^k\};$$

$$\models_{q_k}^{p_k} \in \{\models_1^k, \models_2^k, \dots, \models_{m_k}^k\}$$

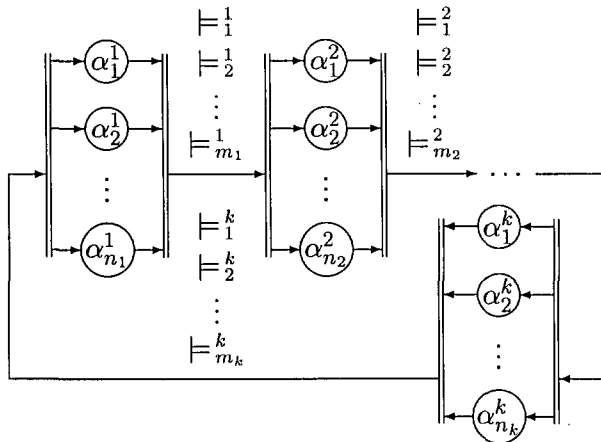


Figure 27: The graph of parallel associative arrays with marked operator arrows in an associative serial loop.

In Fig. 27 the operator paths are marked by the adequate operators which can appear between operands (entities α_j^i). The number of possible operators is given by

$$m_1 \leq n_1 \cdot n_2; m_2 \leq n_2 \cdot n_3; \dots; m_k \leq n_k \cdot n_1$$

In case of the circular association graph in Fig. 27, we have to determine the transition through the graph from an initial operand, for example, $\alpha_{j_1}^{i_1}$. It is possible to take as an example the sentence

$$\left(\left(\dots \left(\alpha_{j_1}^{i_1} \models_{q_1}^{p_1} \alpha_{j_2}^{i_2} \right) \models_{q_2}^{p_2} \dots \alpha_{j_{k-1}}^{i_{k-1}} \right) \models_{q_{k-1}}^{p_{k-1}} \right.$$

$$\left. \alpha_{j_k}^{i_k} \right) \models_{q_k}^{p_k} \alpha_{j_1}^{i_1}$$

where for the associative operands and the corresponding associative operators the choices of the sort

are on disposal. In a natural language, such choices are nothing other than the adequate synonym and antonym word entities, by which the association process in the next associative cycles can come to the surface. It is evident that through such an informational processing the very initial sentence can not only meaningfully change in a substantial way, but can, through the use of antonyms and again synonyms, pass through various mutually oppositional meanings. This process reminds on or approaches to a real associative mediating in the living brain when linguistic thinking is performed on the conscious level (and, for example, by the use of dictionaries).

It is important to stress that a cyclic graph hides the informing of an undetermined length. It only insists to make at least one informational cycle. Afterwards, the informing can still be cyclic, but it can also stop at any operand or operator entity, not closing the ongoing cyclic informing. In this case the part of the last cycle is serial. On the other hand, to some extent, cyclic informing is causal, depending on the concrete form of the cyclic formula, which can not be directly recognized from the graph.

12 Conclusion

Problems of informational graphs reveal the complexity of informational phenomenalism and make the appearance of circularity and possible spontaneity of emerging and arising informational entities more transparent as a pure informational formula and formula system approach could do in such an evident way. On the other hand, graphs as graphical informational entities can have their own informational presentation and can perform as regular informational entities (systems).

The history of the informational theory (since 1987) has gone through substantial principled (axiomatic) and formalistic innovations, so today it can fit the most pretentious requirements for the formalization in the area of consciousness phenomena (e.g., formalistic and graphical treatment of psychological, psychiatric, understanding, economical informational models, presented in [25]). The exposed formalism together with informational graphs (a kind of conscious imprints, expressed as the extremely possible parallel form) seems to be appropriate for defining, handling and observing the problems of consciousness with various consciousness components and systems concerning, for instance, experience, emotion, memory, association, qualia, sensitivity, awareness, attention, intention, significance, meaning, discourse, understanding, self, subconsciousness, unconsciousness, etc., being connected into a complex system of consciousness (conscious thinking). In the discussed sense, the informational graphs could be incorporated into the 'hot' ($T < 0_+$) theory of the brain and society ([12, 13]) in a fuzzy disperse pattern.

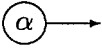
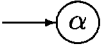
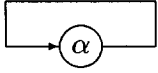
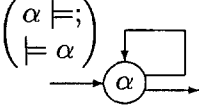
Wheeler [16] has argued persuasively that physics stands to learn a great deal about the world by looking it in terms of information. Information occupies a wonderfully ambiguous place somewhere between the concrete and the subjective [6]. He suggested [17] that information is fundamental to the physics of the universe so that in a double-aspect theory, proposed by Slechta [14] and Chalmers [2, 3], information has both physical and experiential aspects. Hameroff and Penrose [5] stress how experiential phenomena and the physical universe are inseparable (e.g., the duality of energy and information in [14]), and this may imply a necessary non-computability in conscious thought processes; and they argue that this non-computability must also be inherent in the phenomenon of quantum state *self-reduction* — the 'objective reduction'.

Besides others, the theory of the informational fulfills these requirements and the concept of informational graph not only widens the instrumentality of the theory but makes the formalistic approach more evident (technical).

References

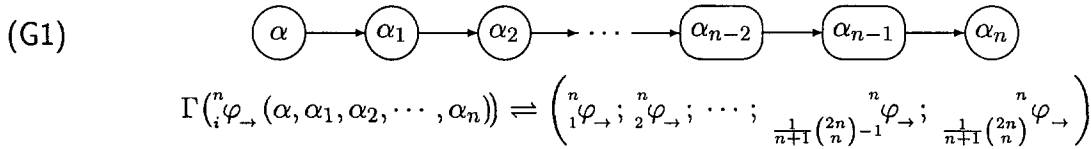
- [1] Structure and Functions of the Human Prefrontal Cortex. 1995. J. Grafman, K.J. Holyoak, & F. Boller, Eds. *Annals of the New York Academy of Sciences* **769**: i–ix + 1–411. The New York Academy of Sciences. New York.
- [2] CHALMERS, D. 1996 Facing up the Problem of Consciousness. *Journal of Consciousness Studies* **2**: 200–219.
- [3] CHALMERS, D. 1996. *The Conscious Mind*. Oxford University Press, New York.
- [4] CUMMINGS, J.L. 1995. Anatomic and Behavioral Aspects of Frontal-subcortical Circuits. *In* [1]: 1–13.
- [5] HAMEROFF, S. & R. PENROSE. 1996. Conscious Events as Orchestrated Space-Time Selections. *Journal of Consciousness Studies* **3**: 36–53.
- [6] HAUSLADEN, P., B. SCHUMACHER, M. WESTMORELAND, & W.K. WOOTERS. 1995. Sending Classical Bits via Quantum Its. *In* *Fundamental Problems in Quantum Theory: A Conference Held in Honor of Professor John A. Wheeler*: 698–705. D.M. Greenberger & A. Zeilinger, Eds. *Annals of the New York Academy of Sciences* **755** i–xiv + 1–908. The New York Academy of Sciences. New York.
- [7] HEIDEGGER, M. 1986. *Sein und Zeit*. Sechzehnte Auflage. Max Niemeyer Verlag. Tübingen.
- [8] HILBERT, D. und P. BERNAYS. 1934. *Grundlagen der Mathematik*. Erster Band. Die Grundlagen der mathematischen Wissenschaften in Einzeldarstellungen. Band XL. Verlag von Julius Springer. Berlin.
- [9] MARUYAMA, M. 1993. A Quickly Understandable Notation System of Causal Loops for Strategic Decision Makers. *Cybernetica* **36**: 37–41.
- [10] MOSCOVITCH, M. & G. WINOCUR. 1995. Frontal Lobes, Memory, and Aging. *In* [1]: 119–150.

Informational Axiomatism

Externalism	Internalism	Metaphysicalism	Phenomenalism	Serial Formula Systems
$\alpha \models$	$\models \alpha$	$\alpha \models \alpha$	$(\alpha \models; \models \alpha)$	${}^n\varphi_{\rightarrow}; {}^{n+1}\varphi_{\rightarrow}^{\circ}; \Gamma({}^n\varphi_{\rightarrow}); \Gamma({}^{n+1}\varphi_{\rightarrow}^{\circ});$
				Parallel Formulas
				${}^n\varphi'_{\parallel}; {}^{n+1}\varphi'_{\parallel}{}^{\circ}; \Gamma^*({}^{n+1}\varphi_{\rightarrow}^{\circ})$

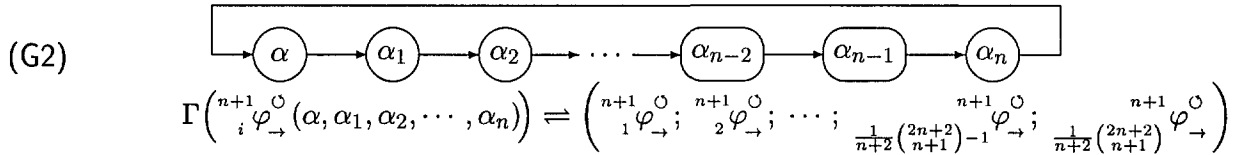
Informational Serialism

${}^n\varphi_{\rightarrow}(\alpha, \alpha_1, \dots, \alpha_n);$
 $1 \leq i \leq N_{\rightarrow}; N_{\rightarrow} = \frac{1}{n+1} \binom{2n}{n}$
 ${}^n_1\varphi_{\rightarrow} \Rightarrow ((\dots((\alpha \models \alpha_1) \models \alpha_2) \models \dots \alpha_{n-1}) \models^* \alpha_n);$
 ${}^n_2\varphi_{\rightarrow} \Rightarrow ((\dots((\alpha \models \alpha_1) \models \alpha_2) \models \dots \alpha_{n-2}) \models^* (\alpha_{n-1} \models \alpha_n));$
 $\dots; {}^n_{N_{\rightarrow}}\varphi_{\rightarrow} \Rightarrow (\alpha \models^* (\alpha_1 \models (\alpha_2 \models \dots (\alpha_{n-1} \models \alpha_n) \dots)))$



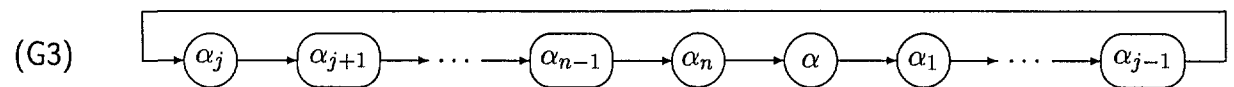
Circular Serialism

${}^{n+1}_i\varphi_{\rightarrow}^{\circ}(\alpha, \alpha_1, \dots, \alpha_n);$
 $1 \leq i \leq N_{\rightarrow}^{\circ}; N_{\rightarrow}^{\circ} = \frac{1}{n+2} \binom{2n+2}{n+1}$
 ${}^{n+1}_1\varphi_{\rightarrow}^{\circ} \Rightarrow (((\dots((\alpha \models \alpha_1) \models \alpha_2) \models \dots \alpha_{n-1}) \models \alpha_n) \models^* \alpha);$
 ${}^{n+1}_2\varphi_{\rightarrow}^{\circ} \Rightarrow (((\dots((\alpha \models \alpha_1) \models \alpha_2) \models \dots \alpha_{n-2}) \models \alpha_{n-1}) \models^* (\alpha_n \models \alpha));$
 $\dots; {}^{n+1}_{N_{\rightarrow}^{\circ}}\varphi_{\rightarrow}^{\circ} \Rightarrow (\alpha \models^* (\alpha_1 \models (\alpha_2 \models \dots (\alpha_{n-1} \models (\alpha_n \models \alpha)) \dots)))$



Circulating the Main Operand α_j ($\alpha_0 = \alpha$)

${}^{n+1}_{ij}\varphi_{\rightarrow}^{\circ}(\alpha_j, \alpha_{j+1}, \dots, \alpha_{n-1}, \alpha_n, \alpha, \alpha_1, \dots, \alpha_{j-1}); 1 \leq ij \leq N_{\rightarrow}^{\circ}; j = 0, 1, \dots, n$



$N_{\rightarrow}^{\circ} = \frac{n+1}{n+2} \binom{2n+2}{n+1}$
 $\Gamma^{\circ}({}^{n+1}_i\varphi_{\rightarrow}^{\circ}(\alpha_j, \alpha_{j+1}, \dots, \alpha_{n-1}, \alpha_n, \alpha, \alpha_1, \dots, \alpha_{j-1})) \Rightarrow$
 Circular Causalism $(\Gamma({}^{n+1}_i\varphi_{\rightarrow}^{\circ}(\alpha_j, \alpha_{j+1}, \dots, \alpha_{n-1}, \alpha_n, \alpha, \alpha_1, \dots, \alpha_{j-1}))); j = 0, 1, 2, \dots, n)$

Primitive Informational Parallelism

$\Pi'({}^n\varphi_{\rightarrow}(\alpha, \alpha_1, \dots, \alpha_{n-1}, \alpha_n)) \Rightarrow (\alpha \models \alpha_1; \alpha_1 \models \alpha_2; \dots; \alpha_{n-1} \models \alpha_n);$ (G1)
 ${}^n\varphi'_{\parallel}(\alpha, \alpha_1, \dots, \alpha_{n-1}, \alpha_n) \Rightarrow \Pi'({}^n\varphi_{\rightarrow}(\alpha, \alpha_1, \dots, \alpha_{n-1}, \alpha_n));$
 $\Gamma({}^n\varphi_{\rightarrow}(\alpha, \alpha_1, \dots, \alpha_{n-1}, \alpha_n)) \subset {}^n\varphi'_{\parallel}(\alpha, \alpha_1, \dots, \alpha_{n-1}, \alpha_n);$

$\Pi'({}^{n+1}_i\varphi_{\rightarrow}^{\circ}(\alpha, \alpha_1, \dots, \alpha_n)) \Rightarrow (\alpha \models \alpha_1; \alpha_1 \models \alpha_2; \alpha_{n-1} \models \alpha_n; \alpha_n \models \alpha);$ (G2, G3)
 ${}^{n+1}\varphi'_{\parallel}{}^{\circ}(\alpha, \alpha_1, \dots, \alpha_{n-1}, \alpha_n) \Rightarrow \Pi'({}^{n+1}_i\varphi_{\rightarrow}^{\circ}(\alpha, \alpha_1, \dots, \alpha_{n-1}, \alpha_n));$
 $\Gamma({}^{n+1}\varphi_{\parallel}{}^{\circ}(\alpha, \alpha_1, \dots, \alpha_{n-1}, \alpha_n)) \subset {}^{n+1}\varphi'_{\parallel}{}^{\circ}(\alpha, \alpha_1, \dots, \alpha_{n-1}, \alpha_n)$

Figure 28: An overview of markers, formulas, formula systems, and graphs concerning axiomatism, serialism, circularism, causalism, parallelism, and gestaltism. By \models^* the main operator \models is marked.

A Study in Generating Readable Modula-2 from Prolog

Dan Resler
Virginia Commonwealth University,
Richmond, VA U.S.A.
Email: dresler@vcu.edu

AND

Danny Crookes
The Queen's University of Belfast,
Belfast, Northern Ireland

Keywords: program translation, program generators, software readability

Edited by: Matjaž Gams

Received: December 23, 1996 **Revised:** February 18, 1997 **Accepted:** March 7, 1997

This article presents an empirical study into generating readable imperative language programs from deterministic Prolog. An overview of a prototype translator is given, followed by a detailed discussion of the required transformations. Two case studies, complete with listings of both the specifications and resultant programs, are presented. An assessment of the generated code and overall project is also given.

1 Introduction

This paper describes an exercise in developing a program generator (called 'Genny') that translates a dialect of Prolog into functionally equivalent, *readable* Modula-2. Generating readable software obviously requires having a clear understanding of what constitutes readable programs. Unfortunately, however, an objective standard for program readability does not yet exist [1, 2]. As a working definition, readable programs are defined as programs that have a similar style to that used by 'good' programmers.

Genny was designed and written with three primary goals in mind:

1. To investigate the development of a good strategy for transforming logic programs into imperative code.
2. To determine the transformations necessary to go from Prolog to Modula-2.
3. To investigate the generation of Modula-2 code that is 'indistinguishable' from hand-written code.

Note that one of the broader goals of the project was to explore the process of *paradigm* transla-

tion. Prolog and Modula-2 were chosen as representative languages for rule-based and procedural paradigms, respectively; other languages could have been selected without changing the main issues that the project addressed.

2 Overview of System

2.1 Specification and target languages

Turbo Prolog [3] was chosen as the basis for the specification language primarily because it requires the explicit defining of domains and predicates. Such definitions greatly simplify the generation of readable type definitions in Modula-2. There is also great benefit in having an *executable* specification language; in the absence of a formal proof of correct transformations from the Prolog specification to the target program, it is possible at least to demonstrate that for given inputs both the specification and resultant programs generate the same outputs.

While it is possible to generate non-deterministic Modula-2 programs [4, 5], the resultant code does not appear to have been written by good imperative language programmers. Therefore Genny was written to translate only *deter-*

ministic Turbo Prolog programs into Modula-2. This restriction is not as severe as it first appears; in *Some Global Optimizations for a Prolog Compiler* [6], Mellish suggests that large portions of existing Prolog programs are already deterministic and directional. Even without backtracking Prolog still remains a powerful programming abstraction; features such as unification, the logic variable, declarative semantics, and the lack of side effects make it an attractive programming tool.

Complete Turbo Prolog programs are not necessary to produce Modula-2 code. Genny will try to do the best it can with what is given; generally all that is essential is that user-defined domains and predicates be specified. Defined predicates with no clauses will produce a subprogram shell (i.e., a procedure heading with an empty BEGIN-END block) while an empty goal section will result in an empty main program.

Genny is a prototype designed to handle only a subset of Turbo Prolog. It does not support global and special ‘database’ declarations, most of the standard predicates, and modular programming—the primary aim was not to generate a ‘complete’ translator but to use this exercise as a means of investigating the problem of constructing a generator.

Genny generates code for the single-pass TopSpeed Modula-2 compiler [7]. Differences between TopSpeed and Wirth’s language definition [8] are slight, with the most noticeable being the different IO routines. Formatting and naming conventions reflect the authors’ personal style and are easily modified.

2.2 Approach

The method used by Genny emulates human translation of code. A significant part of the design of Genny involved hand translating Prolog programs to Modula-2 and noting the process. An attempt was made to go beyond direct line-by-line translation—Genny focuses on translating *paradigms*, from rule-based to procedural (imperative). Therefore whenever problems were encountered or design decisions had to be made, the first question asked was always “How would an imperative language programmer code this, and how can we mimic the process?”

The first consequence of this was the observa-

tion that Genny was not going to be a straightforward ‘single-pass’ translator. Human translation of Prolog involves free movement through the specification to extract the information needed at the moment. Of major importance therefore was an intermediate representation of the specification that allowed for easy extraction of information from virtually any part of the program at any time.

3 Transformations

3.1 Generating types from domains

Genny is able to translate 3 types of Turbo Prolog domain definitions into Modula-2 definitions:

1. *domain_list* = *standard_domain_type*

A *standard_domain_type* can be either an integer, char, real, string, or symbol. A *domain_list* can consist of one or more domains separated by commas. For example,

```
sum, average = real
```

would declare the domains `sum` and `average` to be of type `real`.

2. *domain_list*

```
= compound_object{; compound_object}
```

A *compound_object* consists of a functor and (optionally) the sub-objects belonging to it (e.g., *functor(object1, object2, ..., objectN)*).

A functor without objects is written as *functor()* or just *functor*. Domains can be explicitly extended using the disjunction operator (i.e., the semicolon (;)). For example, if we assume that the sub-objects `title`, `author`, and `name` are defined elsewhere, the domain `articles` could be defined as

```
articles = book(title,author);
           horse(name); boat; ball
```

3. *domain_list* = *domain**

Appending an asterisk (*) to either a standard or user-defined domain declares it to be a list. Therefore the declaration

```
fpList = real*
```

declares a domain for lists of reals (e.g., [2.1,4.2,0.02,93.1]).

Turbo Prolog's rules for defining compound objects allow for some interesting and versatile domain declarations, as illustrated in Figure 1. Domain one consists of a series of functors without sub-objects that functionally is similar to an enumerated type in Modula-2. Functors with and without sub-objects can be mixed freely in declarations, as is shown in the declaration for seven. Note also that in the definition of four, the identifiers *string*, *real*, and *integer* are functors and *not* domains (the same is true for one in the definition of seven). Recursive domain declarations (both indirect and direct) are also allowed in Turbo Prolog, as is demonstrated in the declarations of six and seven.

Simple compound domains in Turbo Prolog can be translated directly to record types in Modula-2. The functor provides a convenient name for the new type; field names, however, are not available, as sub-objects are not named in Turbo Prolog. The convenient but rather unsatisfactory convention of naming fields f_1, f_2, \dots, f_n was adopted in Genny. Since record definitions almost always consist of more than a single field, compound objects with only one sub-object are 'flattened' to simple type definitions, e.g., the domain declaration

```
oneSubObject = f(integer)
```

would be translated to the type definition

```
oneSubObject = INTEGER;
```

Domains that consist of a list of functors without sub-objects are transformed into Modula-2 enumerated types. This method alone will not work, however, if the compound declaration mixes objects with and without sub-objects. In such cases Genny generates a *variant record* type to represent the domain. The functors are used to define an enumerated type whose items become the tags in the variant record. For example, the domain

```
articles = book(title,author);
         horse(name); boat; ball
```

would become

```
TYPE
  articlesTags = (book,horse,boat,ball);
  articles     =
RECORD
  CASE tag:articlesTags OF
```

```
| book:  f1: title;
         f2: author;
| horse: f3: name;
| boat:  f4: articlesTags;
| ball:  f5: articlesTags;
END;
END;
```

Note that the tag names *boat* and *ball* are redundant (Genny at present does not use tags when accessing variant records) and that further analysis would have shown that fields *f4* and *f5* could have been combined (i.e., the tag value would be a sufficient indicator).

In Turbo Prolog, domains can be referenced before they are declared; TopSpeed Modula-2, however, is a single-pass compiler that requires all types to be defined before they are used. Genny therefore must generate declarations in such a way as to avoid referencing undefined types. This requires the use of a dependency graph.

Figure 2 shows the directed dependency graph for the domain declarations of Figure 1. The correct ordering for single-pass compiler type definitions can be derived by topologically sorting the graph nodes; however, topological sorts work only for directed acyclic graphs. The back edges must first be removed before sorting the nodes.

Only in defining pointer types does Modula-2 allow one to reference an unknown type. Genny takes advantage of this by defining all types whose dependency creates a back edge as pointers, thus allowing the removal of back edges from the graph. All edges representing the dependencies of list domains can also be removed as Genny uses generic lists which need only the importation of Genny's 'standard' *Lists* module.

domains

```
one   = aa;bb;cc;dd;ee
two   = ff(real,integer,integer)
three = gg(real,two,seven)
four  = string;real;integer
five  = one*
six   = hh(six,seven)
seven = jj;one;kk(one,two,five,six)
eight = ll(one,two,six,seven)
```

Figure 1: Domain declarations in Turbo Prolog

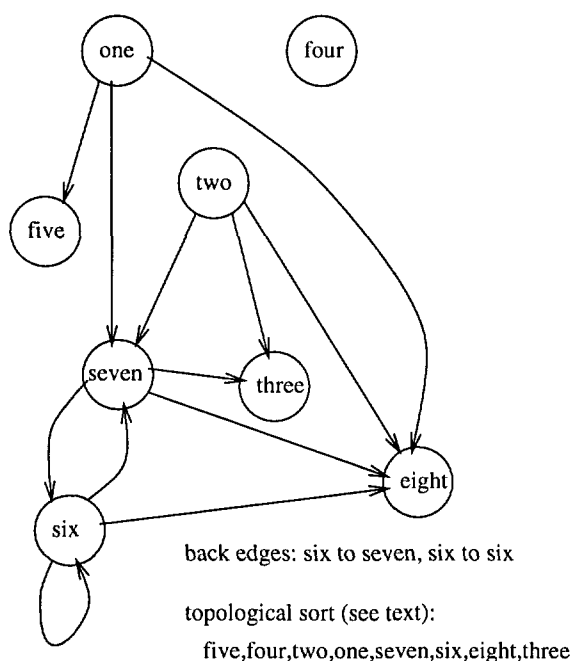


Figure 2: Dependency graph

The resultant type definitions produced by Genny are illustrated in Figure 3. Note that name conflicts are resolved by appending a unique integer to the offending identifier.

While Genny is capable of handling rather convoluted domain declarations, the majority of Turbo Prolog domains used in ‘real’ programs map to simple standard or record type definitions in Modula-2. These definitions, with the exception of record field names, are very close to what a human programmer would produce.

3.2 Generating procedures from clauses

Genny merges all identically named clauses with the same arity and component types into a single Modula-2 procedure. Clauses are considered in the order that they appear and are not at present reordered to possibly generate more natural code. In addition, all procedures are declared as FORWARD to allow the ignoring of the ordering restrictions for single-pass compilers. The re-ordering of procedure declarations to eliminate or minimize forward declarations is a straightforward exercise that has been left for a later time.

Genny maintains a rather limited database of predicates that need not be defined to be referenced. Many of the more commonly used ‘stan-

```

TYPE
STRING      = ARRAY [1..80] OF CHAR;
sixPtr      = POINTER TO six;
five        = Lists.list; (* list type: one *)
four        = (string,real,integer);
two         = RECORD
            f1 : REAL;
            f2 : INTEGER;
            f3 : INTEGER;
            END;
one         = (aa,bb,cc,dd,ee);
sevenTags   = (jj,one_1,kk);
seven       = RECORD
            CASE tag:sevenTags OF
            | jj:   f1 : sevenTags;
            | one_1: f2 : sevenTags;
            | kk:   f3 : one;
                   f4 : two;
                   f5 : five;
                   f6 : sixPtr;
            END;
            END;
six         = RECORD
            f1 : sixPtr;
            f2 : seven;
            END;
eight       = RECORD
            f1 : one;
            f2 : two;
            f3 : sixPtr;
            f4 : seven;
            END;
three       = RECORD
            f1 : REAL;
            f2 : two;
            f3 : seven;
            END;

```

Figure 3: Generated type declarations

ard’ predicates in Turbo Prolog are specified in this database. Warning messages are generated for all predicate calls that are not user-defined or located in the predicate database.

Prolog parameters are mapped directly to Modula-2 parameters (i.e., no attempt is made to create new Modula-2 parameters by combining *in* and *out* Prolog parameters). Genny requires that the parameter type *and* direction be specified in the predicate definition. Since Turbo Prolog does not provide for defining the direction of (non-global) predicates, Genny requires that a parameter’s type be followed by a comment that indicates its flow pattern (or ‘mode’). For example, adding the flow patterns for quicksort would produce

```

predicates
writeList(list/*i*/)
quicksort(list/*i*/,list/*o*/)
split(integer/*i*/,list/*i*/,
       list/*o*/,list/*o*/)

```

```
conc(list/*i*/,list/*i*/,list/*o*/) .
```

Genny looks at the unification conditional at parameter n (moving left to right) of each clause for a given predicate to determine the type of conditional Modula-2 statement to generate. Consider, for example, the clauses

```
editLine(quit, '\27', L, L).
editLine(insert, Ch, line(BC, AC),
         line([Ch|BC], AC)).
editLine(leftArrow, _, line([Ch|BC], AC),
         line(BC, [Ch|AC])).
editLine(leftArrow, _, line([], AC),
         line([], AC)).
editLine(rightArrow, _, line(BC, [Ch|AC]),
         line([Ch|BC], AC)).
editLine(rightArrow, _, line(BC, []),
         line(BC, [])).
```

Genny first considers unifying parameter #1 and determines that only simple comparisons are necessary to select the appropriate clause. Moving left to right, parameter #2 is shown to have only one unification conditional (in the first clause). Finally it would be noted that the last parameter would require a combination of extractions and comparisons for unification to be successful.

Unification conditionals will result in the generation of some combination of IF or CASE statements according to the following criteria:

- Predicates with only one or two conditions for a given parameter will result in an IF statement. (Note that throughout this section 'IF statements' will include, where applicable, all permutations of IF-ELSIF-ELSE statements.)
- Three or more conditions requiring comparisons of ordinal (i.e., enumerated, CARDINAL, INTEGER, CHAR, or BOOLEAN) types will cause Genny to generate a CASE statement.
- More complex comparisons (e.g., comparing more than one list item, floating point or string comparisons, etc.) will always result in an IF statement.
- Situations where both simple and complex comparisons take place will result in the possible nesting of IF and CASE statements according to the rules mentioned above.

For example, the `getCommand` predicate

```
getCommand(Ch, digit) :-
    Ch >= 48, Ch <= 57,
    printstring("New level: "), put(Ch), nl.
getCommand(105, up).
getCommand(60, down).
getCommand(106, left).
getCommand(108, right).
getCommand(32, shoot).
getCommand(113, quit).
getCommand(_, illegalcommand).
```

would be translated into the following code by Genny:

```
PROCEDURE getCommand(    Ch:CHAR;
                      VAR command:SYMBOL);
BEGIN
  IF (Ch >= '0') AND (Ch <= '9') THEN
    IO.WrStr('New level: ');
    IO.WrChar(Ch);
    IO.WrLn;
    command := digit;
  ELSE
    CASE Ch OF
      'i': command := up;
      | ',': command := down;
      | 'j': command := left;
      | 'l': command := right;
      | ' ': command := shoot;
      | 'q': command := quit;
      ELSE command := illegalcommand;
    END; (* CASE *)
  END; (* IF *)
END getCommand;
```

The naming of procedure parameters presents a particular problem as Prolog often does not require it. Where variables are used as parameters in Prolog clauses, Genny tries to use the same name in the generated Modula-2 code. This often works as it's a common practice amongst Prolog programmers to use the same identifier for a given parameter across most or all of the clauses. Parameter names can also be explicitly given in the predicate definitions by appending a '-*idName*' to the flow indicator, e.g.,

```
predicates
  length(chars/*i-str*/,integer/*i*/,
         integer/*o-len*/) .
```

Should Genny not be able to determine the name for a given parameter, a default name of 'pn' will be generated where n is the parameter's position.

The scope of variables in the Prolog specification is maintained by Genny in the Modula-2 code. As mentioned previously, name clashes are resolved by appending a unique integer to one of the identifiers. The global scope is determined by the (optional) goal section (i.e., variables that are local to the goal section are declared globally, and the main Modula-2 program block is generated from this section).

Most of Turbo Prolog's 'standard' predicates were deemed unnecessary for the prototype and therefore were not implemented. In addition, the cut (or '!') is simply ignored as it is irrelevant when translating deterministic Prolog code.

3.3 Generating statements from subgoals

There is almost a one-to-one correspondence of subgoals in the body of a Turbo Prolog predicate to statements in the body of a Modula-2 procedure. Therefore in most instances transforming subgoals into statements is trivial.

There are exceptions, however, with one important example being the Turbo Prolog input and output predicates. The subgoals

```
write("Value ",X," was found in list ",
      List1), nl.
```

would be translated into several TopSpeed Modula-2 statements:

```
IO.WrStr("Value ");
IO.WrInt(X,0);
IO.WrStr(" was found in list ");
WrList(List1);
IO.WrLn;
```

Routine WrList would need to be generated by Genny and would be peculiar to the type of list being written.

Another difficulty arises from Turbo Prolog's substitution of '=' for the 'is' operator; the resulting ambiguity is described in the *Turbo Prolog Owner's Handbook*:

In Turbo Prolog, statements like $N=N1-2$ indicate a relation between the three objects: N , $N1$, and 2 ; or a relation between

two objects: N and the value of $N1-2$. If N is still free, the statement can be satisfied by binding N . This corresponds to what other programming languages call an *assignment statement*; in Turbo Prolog, it is a logical statement. [3, page 66]

As a result testing for equality and assignment have the same syntax in Prolog; the semantics for these operations must therefore be determined by context. In Genny, relational subgoals that occur before any procedural subgoals *and* contain bound variables will result in the generation of conditional expressions that must be satisfied before the rest of the predicate body is executed. Any relational subgoals that either contain unbound variables or occur after a procedural subgoal will result in assignment. For example, the predicate count contains two '=' operators:

```
count([N|T],Key,CIn,COut) :-
  N=Key,
  C=CIn+1,
  count(T,Key,C,COut).
```

In the first instance (i.e., $N=Key$), both variables are bound and the subgoal occurs before a procedural subgoal (i.e., before `count(T,Key,C,COut)`), therefore an IF statement containing the conditional expression $N=Key$ would be generated as part of the unification conditional. And since the second use of '=' contains the unbound variable C , Genny would generate the code $C:=CIn+1$;

3.4 List manipulation

Generic list routines (imported from the 'standard' module Lists) are used for all list manipulations in the programs produced by Genny. Figure 4 gives examples of the common list operations provided by Genny (Note that the ambiguity between the test for list equality and list assignment arises from Turbo Prolog's lack of the 'is' operator. Creating a list constant must be done at run-time as 'standard' Modula-2 does not allow the building of structured constants during compilation). In order to make type list truly general, list item constructors, dereference routines, and comparison functions must be provided for all types. These are included in Lists for the 'standard' types but would have to be generated for all applicable user-defined types.

Operation	Prolog	Using Lists ADT
tests for empty list	[]	empty(L)
test for list equality	L1=L2	areEqual(L1,L2,listCompare)
list assignment	L1=L2	assign(L1,L2);
create an empty list	[]	emptyList()
remove head of list	[_ L]	behead(L);
split integer list into head & tail	[H T]	H:=int(head(L)); T:=tail(L);
add integer N to head of list	[N L]	L:=newList(intItem(N),L);
create list literal	[1,2,3]	newList(intItem(1), newList2(intItem(2), intItem(3)))

Figure 4: Generic list operations in Genny

4 Case Studies

This section will present two case studies of programs transformed by Genny: quicksort and a simple line editor.

4.1 Quicksort

Appendix A.1 lists the Prolog specification for Quicksort; the Modula-2 program generated by Genny from this specification is given in Appendix A.2. The quicksort Prolog specification is of interest because it involves list unification, more than two clauses for a predicate, conditional subgoals, and 'out' parameters.

Genny always postpones clause unification until the clause is selected; note, for example, the Modula-2 code for `writeList`—the list head for the second clause is not extracted until the ELSE branch in the IF statement. Note also that both the second and third clauses of `split` require the extraction of the head of the second parameter (which is unified with 'Y'); Genny pulls this common operation out of the Modula-2 IF statement rather than duplicating the code.

Prolog 'out' parameters present a unique problem as they must be assigned values immediately before exiting a procedure (see the assignments to `Big` and `Small` in the procedure `split`). In general, for each clause Genny executes code for unifying 'in' parameter first, then subgoals, and finally the 'out' parameters are unified.

The subgoal '`X > Y`' in the second clause for

`split` is the condition determining which of the last two clauses will be executed and therefore is the test in the generated IF statement. Genny extracts all subgoal conditionals and includes them in the generated code.

The rather unorthodox building of the list passed to `quicksort` in the main program is due to the lack of a compile time constructor for structured types in Modula-2.

4.2 Line editor

The final example is a one-line screen editor (see Appendix B) that features simple cursor movement and text deletion operations using several of the IBM PC's special keypad keys.

Turbo Prolog has a standard type 'symbol' that maps naturally to enumerated types in Modula-2. Genny scans the code looking for all symbol identifiers that are then used to define the special type `SYMBOL`. Turbo Prolog allows for input and output of data of type symbol; Modula-2, however, does not provide for enumerated type I/O. Consequently Genny will not translate a Turbo Prolog goal of the form '`readln(SymbolVar)`'. A straightforward solution, saved for a later version, would be to have Genny generate special I/O routines for `SYMBOL`.

Unlike the other two examples, `linedit` does not provide adequate information for Genny to name several of the procedure parameters. Many of the actual parameters simply are not unified to *variables*, so either Genny provides default names

or the user supplies connotative names in the predicate section (see Section 3.2). Genny will only use formal parameter identifiers if there are no conflicting parameter names in other clauses for the same predicate. Such a conflict is illustrated in the predicate `editLine`—the second clause (for command ‘insert’) uses variable `C` for parameter #2 whereas clause #12 (command ‘end’) uses `Ch`. Genny is also careful to remove possible conflicts caused by `local` variables. The predicate `getCommand` uses the variable `Ch` differently in 2 different clauses; although in this case it wasn’t necessary, Genny removes all possibilities of conflict by making the local variable unique.

The predicate `editLine` is a good example of how Genny combines many relatively complicated clauses into one Modula-2 procedure. Most of the clauses have two conditional parameters and are paired by the value of the first parameter. This results in Genny generating a large multi-way branching (CASE) statement each with nested IF-ELSE statements.

4.3 Overall Assessment of Genny

Genny is capable of generating reasonable quality, readable Modula-2 code from deterministic Turbo Prolog specifications. Its strengths lie in its generation of readable type definitions, the merging of many scopes (clauses) into a single Modula-2 procedure, and its ability to generate complex selection control structures.

Many of its weaknesses have known solutions [9] that were not implemented because either they did not help meet the original objectives or because of difficulty in implementing them in the time available, e.g.,

- the elimination of the need for (most) FORWARD declarations through the re-ordering of procedure definitions
- automatically determining the ‘mode’ of predicate arguments
- the combining of certain *in* and *out* parameters into a single Modula-2 VAR parameter
- the use of arrays to represent lists in certain cases
- transforming recursion to iterative structures (such as WHILE or FOR loops)

- extending the database of ‘standard’ Prolog predicates that have default transformations
- the automatic generation, when needed, of IO routines for enumerated types
- the automatic generation, when needed, of comparison routines (of type `cproc`) for user-defined types
- the inclusion of specification comments in the generated code

The flexibility of Genny could be improved by allowing the user more control over the form of the generated program. This control could vary from code formatting to choice of data and control structures to specifying the contents of separate modules. Other issues include incorporating a better, more comprehensive method for specifying names (e.g., variable and record field names) and possibly allowing the user to select an interactive mode of operation.

As Genny evolved it became more and more apparent that Modula-2 was perhaps not the best language for *implementing* the system. Repeated manoeuvring through the parse tree to extract information was awkward at best even with special purpose routines. It was often difficult to extract and debug transformations that were hard-coded across many lines of code or several procedures. A rule-based system (using a dialect of Prolog or a transformational language) would have resulted in a more easily modifiable system.

5 Conclusion

Genny generates readable Modula-2 that is very similar to hand-written code (for example, the program presented in Appendix A.2 compares favorably to textbook versions found in [10, 11, 12]. See [9] for a thorough comparison). In addition, the transformations for improving the code for some of the situations where the generated program fails to meet this criterion (e.g., forward procedure declarations and removal of recursion) are known [9] and will be incorporated into future versions.

Genny will also generate ‘better’, more readable code as it becomes more *flexible*. The ability to specify, for example, data structures, naming conventions, type of control structures, etc., will aid

the user to 'tune' the output's readability. Such control is essential until there is a widely recognized standard for code readability.

References

- [1] Paul W. Oman and Curtis R. Cook. A programming style taxonomy. Software Engineering Lab Technical Report #90-05 TR, University of Idaho, College of Engineering, Computer Science Department, Moscow, Idaho, February 1990. Abridged to 'A Taxonomy for Programming Style' in *Proceedings of the 18th Annual Computer Science Conference* (Washington D.C., Feb. 22-23, 1990) ACM, New York, pages 244-250.
- [2] Dan Resler and Danny Crookes. Software readability. Technical report, University of Limerick, Department of Computer Science and Information Systems, Limerick, Ireland, August 1991.
- [3] Borland International, Scotts Valley, CA, USA. *Turbo Prolog Owner's Handbook-version 1.0*, 1986.
- [4] Jørgen Fischer Nilsson. On the compilation of a domain-based Prolog. In R.E.A. Mason, editor, *Information Processing 83*, pages 293-298. IFIP, Elsevier Science Publisher B.V. (North-Holland), 1983.
- [5] J.L. Weiner and S. Ramakrishnan. A piggyback compiler for Prolog. *ACM SIGPLAN Notices*, 23(7):288-296, 1988. In *Proceedings of the SIGPLAN '88 Conference on Programming Language Design and Implementation*, Atlanta, Georgia, June 22-24, 1988.
- [6] C.S. Mellish. Some global optimizations for a Prolog compiler. *Journal of Logic Programming*, 1:43-66, 1985.
- [7] K.N. King. *Topspeed Modula-2 Language Tutorial*. Jensen & Partners International, London, 1990. For Topspeed Modula-2, version 2.0.
- [8] Niklaus Wirth. *Programming in Modula-2*. Springer-Verlag, Berlin, third edition, 1985.
- [9] R. Daniel Resler. *An Investigation into Generating Readable Software from Logic Specifications*. PhD thesis, Queen's University of Belfast, Belfast, N. Ireland, October 1991.
- [10] Richard F. Sinovec and Richard S. Weiner. *Data Structures with Abstract Data Types and Modula-2*. John Wiley, New York, 1986.
- [11] Michael B. Feldman. *Data Structures with Modula-2*. Prentice Hall, Englewood Cliffs, NJ, 1988.
- [12] Daniel F. Stubbs and Neil W. Webre. *Data Structures with Abstract Data Types and Modula-2*. Brooks/Cole, Monterey, CA, 1987.

A Quicksort

A.1 Prolog Specification

```
domains
  list = integer*

predicates
  writeList(list/*i*/)
  quicksort(list/*i*/,list/*o*/)
  split(integer/*i*/,list/*i*/,list/*o*/,list/*o*/)
  conc(list/*i*/,list/*i*/,list/*o*/)

goal
  quicksort([45,1,67,22,897,2,100,4,5,6,2],SortedList),
  write("sorted list:").nl,
  writeList(SortedList).nl

clauses
  writeList([]).
  writeList([H|T]) :- write(H," "), writeList(T).

  quicksort([],[]).
  quicksort([X|Tail],Sorted) :-
    split(X,Tail,Small,Big),
    quicksort(Small,SortedSmall),
    quicksort(Big,SortedBig),
    conc(SortedSmall,[X|SortedBig],Sorted).

  split(_,[],[],[]).
  split(X,[Y|Tail],[Y|Small],Big) :-
    X > Y,
    !,
    split(X,Tail,Small,Big).
  split(X,[Y|Tail],Small,[Y|Big]) :-
    split(X,Tail,Small,Big).

  conc([],L,L).
  conc([X|L1],L2,[X|L3]) :-
    conc(L1,L2,L3).
```

A.2 Generated Modula-2

```
(*
  File: QS.MOD (created 5-20-91 at 10:04)
  Description: created from specification
  file QS.PRO by GENNY v1.0
*)
MODULE QS;

IMPORT Lists,IO;

TYPE
  list = Lists.list; (* list type: INTEGER *)

VAR
  SortedList : list;

PROCEDURE writeList(p1:list);          FORWARD;
PROCEDURE quicksort(p1:list;VAR Sorted:list); FORWARD;
PROCEDURE split(X:INTEGER;p2:list;VAR Small:list;
  VAR Big:list);          FORWARD;
PROCEDURE conc(p1:list;p2:list;VAR L:list); FORWARD;

PROCEDURE writeList(p1:list);

VAR
  H : INTEGER;

BEGIN
  IF Lists.empty(p1) THEN
```

```
    RETURN;
  ELSE
    H := Lists.int(Lists.head(p1)); p1 := Lists.tail(p1);
    IO.WrInt(H,0);
    IO.WrStr(' ');
    writeList(p1);
  END; (* IF *)
END writeList;

PROCEDURE quicksort(p1:list;VAR Sorted:list);

VAR
  X          : INTEGER;
  Small,
  Big,
  SortedSmall,
  SortedBig  : list;

BEGIN
  IF Lists.empty(p1) THEN
    Sorted := Lists.emptyList();
  ELSE
    X := Lists.int(Lists.head(p1)); p1 := Lists.tail(p1);
    split(X,p1,Small,Big);
    quicksort(Small,SortedSmall);
    quicksort(Big,SortedBig);
    conc(SortedSmall,Lists.newList(Lists.intItem(X),
      SortedBig),Sorted);
  END; (* IF *)
END quicksort;

PROCEDURE split(X:INTEGER;p2:list;VAR Small:list;
  VAR Big:list);

VAR
  Y : INTEGER;

BEGIN
  IF Lists.empty(p2) THEN
    Small := Lists.emptyList();
    Big := Lists.emptyList();
  ELSE
    Y := Lists.int(Lists.head(p2)); p2 := Lists.tail(p2);
    IF X > Y THEN
      split(X,p2,Small,Big);
      Small := Lists.newList(Lists.intItem(Y),Small);
    ELSE
      split(X,p2,Small,Big);
      Big := Lists.newList(Lists.intItem(Y),Big);
    END; (* IF *)
  END; (* IF *)
END split;

PROCEDURE conc(p1:list;p2:list;VAR L:list);

VAR
  X : INTEGER;

BEGIN
  IF Lists.empty(p1) THEN
    L := p2;
  ELSE
    X := Lists.int(Lists.head(p1)); p1 := Lists.tail(p1);
    conc(p1,p2,L);
    L := Lists.newList(Lists.intItem(X),L);
  END; (* IF *)
END conc;

BEGIN (* main program *)
  quicksort(Lists.newList(
    Lists.intItem(45),
    Lists.newList(
```

```

Lists.intItem(1),
Lists.newList(
  Lists.intItem(67),
  Lists.newList(
    Lists.intItem(22),
    Lists.newList(
      Lists.intItem(897),
      Lists.newList(
        Lists.intItem(2),
        Lists.newList(
          Lists.intItem(100),
          Lists.newList(
            Lists.intItem(4),
            Lists.newList(
              Lists.intItem(5),
              Lists.newList2(
                Lists.intItem(6),
                Lists.intItem(2)))))))))
SortedList);
IO.WrStr('sorted list:');
IO.WrLn;
writeList(SortedList);
IO.WrLn;
END QS.

```

B Line Editor

B.1 Prolog Specification

```

domains
  chars = char*
  line = line(chars, /* before the cursor */
             chars) /* after the cursor */

predicates
  driver(symbol/*i-command*/,line/*i*/)
  editLine(symbol/*i-command*/,char/*i-currChar*/,
           line/*i-lineIn*/,line/*o-lineOut*/)
  getCommand(char/*i*/,symbol/*o*/)
  specialKey(char/*i-keyStroke*/,symbol/*o-command*/)
  updateScreen(line/*i-newLine*/)
  writestring(chars/*i-str*/)
  length(chars/*i-str*/,integer/*i*/,integer/*o-len*/)
  rev(chars/*i*/,chars/*o*/)
  revzap(chars/*i-l1*/,chars/*i-l2*/,chars/*o-l0ut*/)

goal
  clearwindow,
  driver(nil,line([],[])).

clauses
  /*
   driver: main "loop" of editor
  */
  driver(quit,_) :- !.
  driver(_,LIn) :-
    readchar(Ch),
    getCommand(Ch,Command),
    editLine(Command,Ch,LIn,LOut),
    updateScreen(LOut),
    driver(Command,LOut).

  /*
   editLine: executes an editor command

  Tokens      Commands
  -----
  insert      insert a character at cursor
  leftArrow   move cursor left one character
  rightArrow  move cursor right one character
  backDel     delete character before cursor

```

```

del          delete character at cursor
delLine     delete the entire line
home        move cursor to beginning of line
end         move cursor to end of line
*/

editLine(quit,_,L) :- write("goodbye"),nl,nl.

editLine(insert,Ch,line(BC,AC),line([Ch|BC],AC)).

editLine(leftArrow,_,line([Ch|BC],AC),
          line(BC,[Ch|AC])).
editLine(leftArrow,_,line([],AC),line([],AC)).

editLine(rightArrow,_,line(BC,[Ch|AC]),
          line([Ch|BC],AC)).
editLine(rightArrow,_,line(BC,[],line(BC,[])).

editLine(backDel,_,line([_|BC],AC),line(BC,AC)).
editLine(backDel,_,line([],AC),line([],AC)).

editLine(del,_,line(BC,[_|AC]),line(BC,AC)).
editLine(del,_,line(BC,[],line(BC,[])).

editLine(delLine,_,_,line([],[])).

editLine(home,C,line([Ch|BC],AC),L) :-
  editLine(home,C,line(BC,[Ch|AC]),L).

editLine(home,_,line([],AC),line([],AC)).

editLine(end,C,line(BC,[Ch|AC]),L):-
  editLine(end,C,line([Ch|BC],AC),L).

editLine(end,_,line(BC,[],line(BC,[])).

editLine(illegalCommand,_,L) :- beep.
/*
  getCommand:
  tokenizes input into appropriate command;
  I use the special keypad keys on the IBM PC
  for the corresponding editor commands;
  these keys return special 2 character codes
  which must be interpreted differently */

getCommand(Ch,insert) :- /* printable char */
  Ch >= ' ', Ch <= '~',!.
getCommand('\0',Command) :- /* keypd key */
  readchar(Ch),
  specialKey(Ch,Command),!.
getCommand('\8',backDel) :- !. /* back del */
getCommand('\25',delLine) :- !. /* <ctrl> y */
getCommand(_,illegalCommand).

specialKey('\75',leftArrow) :- !.
specialKey('\77',rightArrow):- !.
specialKey('\83',del) :- !.
specialKey('\71',home) :- !.
specialKey('\79',end) :- !.
specialKey('\45',quit) :- !. /* alt x */

specialKey(_,illegalCommand).

/*
  updateScreen: writes line to screen at row 0
*/
updateScreen(line(BC,AC)) :-
  !,
  clearwindow,
  rev(BC,BC1), /* rev. part before cursor */
  writestring(BC1), writestring(AC),
  length(BC,0,Col),/* compute cursor position */

```

```

        cursor(0,Col). /* move cursor */

writestring([]).
writestring([Ch|S]) :- write(Ch), writestring(S).

length([],N,N).
length([_|L],N,N2) :-
    N1 = N + 1,
    length(L,N1,N2).

rev(L1,L2) :- revzap(L1,[],L2).

revzap([X|L],L2,L3) :- revzap(L,[X|L2],L3).
revzap([],L,L).

```

B.2 Generated Modula-2

```

(*)
File: LINEDIT.MOD (created 5-20-91 at 11:57)
Description: created from specification file
            LINEDIT.PRO by GENNY v1.0
*)

MODULE LINEDIT;

IMPORT Window,Lists,IO;

TYPE
  SYMBOL = (nil,quit,insert,leftArrow,rightArrow,
            backDel,del,delLine,home,end,illegalCommand);
  chars = Lists.list; (* list type: CHAR *)
  line = RECORD
    f1 : chars;
    f2 : chars;
  END;

VAR
  lineBuff : line;

PROCEDURE driver(command:SYMBOL;LIn:line); FORWARD;
PROCEDURE editLine(command:SYMBOL;currChar:CHAR;lineIn:
  line;VAR lineOut:line); FORWARD;
PROCEDURE getCommand(Ch:CHAR;VAR Command:SYMBOL);
  FORWARD;
PROCEDURE specialKey(keyStroke:CHAR;VAR command:SYMBOL);
  FORWARD;
PROCEDURE updateScreen(newLine:line); FORWARD;
PROCEDURE writestring(str:chars); FORWARD;
PROCEDURE length(str:chars;N:INTEGER;VAR len:INTEGER);
  FORWARD;
PROCEDURE rev(L1:chars;VAR L2:chars); FORWARD;
PROCEDURE revzap(l1:chars;l2:chars;VAR lOut:chars);
  FORWARD;

PROCEDURE driver(command:SYMBOL;LIn:line);

VAR
  Ch : CHAR;
  Command : SYMBOL;
  LOut : line;

BEGIN
  IF command = quit THEN
    RETURN;
  ELSE
    Ch := IO.RdKey();
    getCommand(Ch,Command);
    editLine(Command,Ch,LIn,LOut);
    updateScreen(LOut);
    driver(Command,LOut);
  END; (* IF *)
END driver;

PROCEDURE editLine(command:SYMBOL;currChar:CHAR;
  lineIn:line;VAR lineOut:line);

VAR
  Ch : CHAR;

BEGIN
  CASE command OF
    quit : IO.WrStr('goodbye');
          IO.WrLn;
          IO.WrLn;
          lineOut := lineIn;
    | insert : lineOut.f1 :=
              Lists.newList(Lists.charItem(currChar),
                            lineIn.f1);
              lineOut.f2 := lineIn.f2;
    | leftArrow : IF Lists.empty(lineIn.f1) THEN
                  lineOut.f1 := Lists.emptyList();
                  lineOut.f2 := lineIn.f2;
                ELSE
                  Ch := Lists.char(Lists.head(lineIn.f1));
                  lineIn.f1 := Lists.tail(lineIn.f1);
                  lineOut.f1 := lineIn.f1;
                  lineOut.f2 :=
                    Lists.newList(Lists.charItem(Ch),
                                  lineIn.f2);
                END; (* IF *)
    | rightArrow : IF Lists.empty(lineIn.f2) THEN
                  lineOut.f1 := lineIn.f1;
                  lineOut.f2 := Lists.emptyList();
                ELSE
                  Ch := Lists.char(Lists.head(lineIn.f2));
                  lineIn.f2 := Lists.tail(lineIn.f2);
                  lineOut.f1 :=
                    Lists.newList(Lists.charItem(Ch),
                                  lineIn.f1);
                  lineOut.f2 := lineIn.f2;
                END; (* IF *)
    | backDel : IF Lists.empty(lineIn.f1) THEN
                  lineOut.f1 := Lists.emptyList();
                  lineOut.f2 := lineIn.f2;
                ELSE
                  lineIn.f1 := Lists.tail(lineIn.f1);
                  lineOut.f1 := lineIn.f1;
                  lineOut.f2 := lineIn.f2;
                END; (* IF *)
    | del : IF Lists.empty(lineIn.f2) THEN
            lineOut.f1 := lineIn.f1;
            lineOut.f2 := Lists.emptyList();
          ELSE
            lineIn.f2 := Lists.tail(lineIn.f2);
            lineOut.f1 := lineIn.f1;
            lineOut.f2 := lineIn.f2;
          END; (* IF *)
    | delLine : lineOut.f1 := Lists.emptyList();
               lineOut.f2 := Lists.emptyList();
    | home : IF Lists.empty(lineIn.f1) THEN
             lineOut.f1 := Lists.emptyList();
             lineOut.f2 := lineIn.f2;
           ELSE
             Ch := Lists.char(Lists.head(lineIn.f1));
             lineIn.f1 := Lists.tail(lineIn.f1);
             lineIn.f2 :=
               Lists.newList(Lists.charItem(Ch),
                             lineIn.f2);
             editLine(home,currChar,lineIn,lineOut);
           END; (* IF *)
    | end : IF Lists.empty(lineIn.f2) THEN
            lineOut.f1 := lineIn.f1;
            lineOut.f2 := Lists.emptyList();
          ELSE

```

```

        Ch := Lists.char(Lists.head(lineIn.f2));    IO.WrChar(Ch);
        lineIn.f2 := Lists.tail(lineIn.f2);        writestring(str);
        lineIn.f1 :=                               END; (* IF *)
        Lists.newList(Lists.charItem(Ch),         END writestring;
            lineIn.f1);
        editLine(end,currChar,lineIn,lineOut);    PROCEDURE length(str:chars;N:INTEGER;
        END; (* IF *)                               VAR len:INTEGER);
| illegalCommand: IO.WrChar(7C); (* bell *)
        lineOut := lineIn;
END; (* CASE *)
END editLine;

PROCEDURE getCommand(Ch:CHAR;VAR Command:SYMBOL);

VAR
    Ch_1 : CHAR;

BEGIN
    IF (Ch >= ' ') AND (Ch <= '~') THEN
        Command := insert;
    ELSE
        CASE Ch OF
            CHR(0) : Ch_1 := IO.RdKey();
                    specialKey(Ch_1,Command);
            | CHR(8) : Command := backDel;
            | CHR(25): Command := delLine;
            ELSE    Command := illegalCommand;
        END; (* CASE *)
    END; (* IF *)
END getCommand;

PROCEDURE specialKey(keyStroke:CHAR;
                    VAR command:SYMBOL);

BEGIN
    CASE keyStroke OF
        'K': command := leftArrow;
        | 'M': command := rightArrow;
        | 'S': command := del;
        | 'G': command := home;
        | 'D': command := end;
        | '-': command := quit;
        ELSE    command := illegalCommand;
    END; (* CASE *)
END specialKey;

PROCEDURE updateScreen(newLine:line);

VAR
    Col : INTEGER;
    BC1 : chars;

BEGIN
    Window.Clear();
    rev(newLine.f1,BC1);
    writestring(BC1);
    writestring(newLine.f2);
    length(newLine.f1,0,Col);
    Window.GotoXY(Col+1,0);
END updateScreen;

PROCEDURE writestring(str:chars);

VAR
    Ch : CHAR;

BEGIN
    IF Lists.empty(str) THEN
        RETURN;
    ELSE
        Ch := Lists.char(Lists.head(str));
        str := Lists.tail(str);

```

Design of Approximate Identity Neural Networks by Piecewise-Linear Circuits

Jiří Kaderka

Department of Microelectronics, TU of Brno, Udolni 53, 602 00 Brno, Czech Republic

Phone: +42 5 43167 134, Fax: +42 5 43167 298

E-mail: kaderka@umel.fee.vutbr.cz

Keywords: approximate identity neural networks, piecewise-linear circuits

Edited by: Rudi Murn

Received: August 20, 1996

Revised: December 1, 1996

Accepted: December 10, 1996

The design of the approximate identity neural network (AINN) using the piecewise-linear (PWL) approach is presented. AINN realizes linear and non-linear mapping. This net consists of two executive layers. The cells of the first - hidden layer, hidden neurons, are designed according to extended PWL sub-circuits that make the function of the absolute value.

The present theory describes a transfer function and a learning algorithm of AINN. The convergence of the learning algorithm is provided numerically. The electronic circuit of AINN is based on PWL sub-circuits and modeled by Saber.

Results show possibility of AINNs to approximate the sine and squared functions. The theory of PWL AINN can be extended; approximate networks can be defined in more dimensions.

1 Introduction

Approximate identity neural networks belong to the class of neural nets which are able to approximate linear and non-linear functions. These nets have the ability to approximate non-linear mapping to any degree of accuracy. AINN consists of two layers. The degree of accuracy depends on a number of neurons in the hidden layer and a learning algorithm.

There are many possibilities how to create a structure of neural nets. In this paper the design of the analog approximate identity neural networks based on the piecewise-linear circuits is established.

The transfer function of AINN is composed of a sequence of functions $\{y_j(x)\}_{j=1}^{\infty}$. The elementary function $y_j(x)$ has to satisfy the following properties[1]:

- P1) $\int_{-\infty}^{+\infty} y_j(x).dx = C$, for any j
- P2) given ϵ and $\rho > 0$, an index N exists such that for $j \geq N$ it results $\int_{\overline{D}_\rho} |y_j(x)|.dx < \epsilon$,

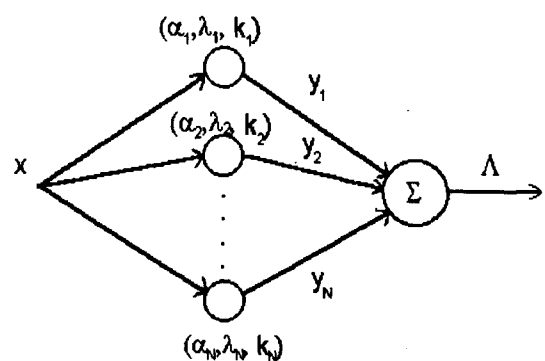


Figure 1: Block scheme of AINN.

where \overline{D}_ρ represents the domain outside the sphere D_ρ or radius ρ and centered at the origin and $\overline{D}_\rho + D_\rho = R^m$. The function $y_j(x)$ describes the elementary analog neurone of the hidden layer.

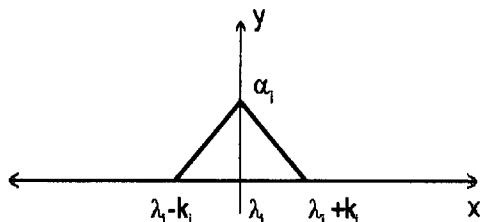


Figure 2: Piecewise-linear transfer function.

2 Definition of the transfer function

The block scheme of AINN is shown in Fig 1. This AINN belongs to the feed-forward neural nets and realizes the one-dimensional mapping. It consists of an input layer, a hidden layer and an output layer. Although the number of layers equals to three in this, in this paper AINN will be denoted as two-layers net as stated in [1]. The AINN is described in Equation (1).

$$\Lambda(x, \alpha, \lambda, k) = \sum_{j=1}^N y_j(x, \alpha_j, \lambda_j, k_j) \quad (1)$$

The proposed AINN is defined in one dimension. It transforms $R \Rightarrow R$. The input layer is composed of one neurone and distributes an input signal into all neurons of the hidden layer. The output layer is also composed of one neurone and sums up output signals of hidden neurons. The neurons of the hidden layer are executive elements. They are described by elementary functions $y_j(x; \alpha, \lambda, k)$.

The transfer function y_j must accomplish notes P1 and P2 which define its properties. As shown, PWL function that has three breakpoints fulfills this definition and can be designed by PWL sub-circuits [2][4].

The function y_j is defined by Equation (2) and its shape is shown in Fig 2.

$$y_j = \frac{1}{2} \alpha_j (|ndist + 1| + |ndist - 1| - 2|ndist|) \quad (2)$$

where $ndist = \frac{1}{k_j}(x - \lambda_j)$

3 Training algorithm of AINN

As was stated in the preceding section the neurons of the AINN are described by 3 sets of parameters - λ, α and k . The parameter α sets a co-ordinate of an elementary function of the hidden neurone on the x -axis and domain of the AINN is the partitioned into $N - 1$ sub-domains. Parameters α and k are adjusted by the learning algorithm so desired outputs have to be obtained. In general the output of AINN will not be the same as the desired value of a primary function $f(x)$; on the other hand parameters α and k [5] can be chosen such that the square of the error

$$E = \frac{1}{2} \sum_{l=1}^p (f(x_l) - \Lambda(x_l))^2 \quad (3)$$

be minimized [1]. A commonly use algorithm is the method of steepest descent in which the incremental changes of parameters $\Delta\alpha$ and Δk are proportional to δE . Equations (4) and (5) make up the core of the learning algorithm.

$$\Delta\alpha_j = \eta_\alpha \frac{\delta E}{\delta\alpha_j} = \quad (4)$$

$$= \eta_\alpha \sum_{l=1}^p ((f(x_l) - \Lambda(x_l)) \cdot$$

$$\cdot \frac{1}{2} (|dist + 1| + |dist - 1| - 2|dist|))$$

where $dist = \frac{1}{k_j}(x_l - \lambda_j)$.

And

$$x_l \Rightarrow (\lambda_j - k_j; \lambda_j + k_j)$$

$$\Delta k_j = \eta_k \sum_{l=1}^p \left((f(x_l) - \Lambda(x_l)) \frac{\alpha_j |x_l - \lambda_j|}{k_j^2} \right) \quad (5)$$

where η_α and η_k are positive valued constants that set the learning rate.

According to Equations (4) and (5) the learning algorithm is implemented in C++ language. The desired primary function $f(x)$, the domain of AINN and initial values of parameters compose the input of the C++ program. Results contain computed parameters of circuits of hidden neurons.

Till these days there is no exact method how to choose initial values of parameters and initial learning condition. Therefore computing of parameters must be repeated to obtain best results.

In addition, constants η_α and η_k control convergence of the learning algorithm. If they are too big convergence will fast point to the minimum but the learning algorithm can exhibit oscillations. If these constants are small the learning algorithm will take a lot of time. In order to solve this problem the constants η_α and η_k are being set to reach the global minimum by the learning algorithm dynamically.¹

4 Implementation using PWL circuits

The particular choice for the transfer functions $y_j(x; \alpha, \lambda, k)$, defined as piecewise-linear in the theory described above, is related to the fact that these terms are suitable for being implemented using piecewise-linear circuits [2].

The structure of the hidden neurone circuit employs elementary circuits of the absolute-value function. This synthesis is very suitable it utilizes the absolute-value function as a basic mathematical and circuit's tool and enables to describe the PWL circuit in explicit close form.

In addition this synthesis makes possible to design multi-dimensional piecewise-linear approximate neural networks. The circuit of a hidden neurone can be easy extended and AINN can approximate more complicated mappings and functions.

On the top of it PWL AINN has simple relation between outputs of the learning algorithm and parameters of PWL sub-circuits. There are many approaches how to design the PWL circuits but in creating of the approximate system it is important that parameters of AINN are easy set in PWL sub-circuits.

As was developed in the present theory and shown in Figure 3 the features of AINN are stated by parameters of summation amplifiers that are easy adapted.

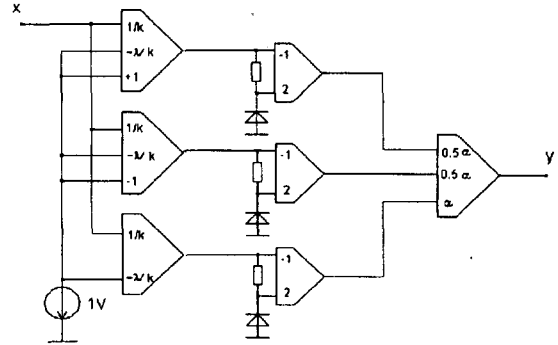


Figure 3: Circuit of analog neurone.

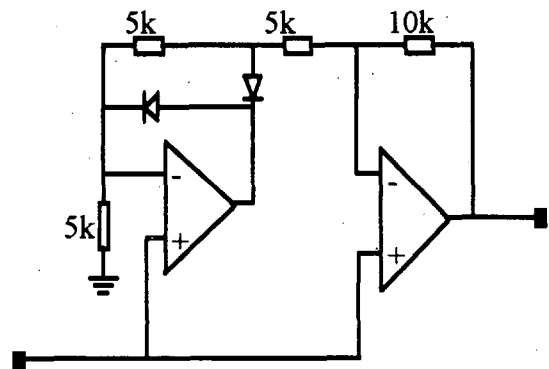


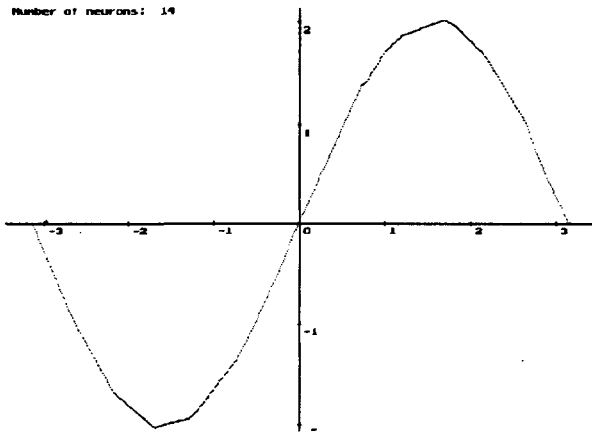
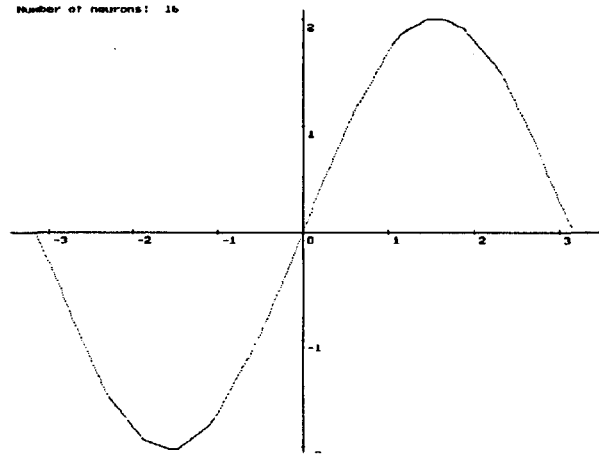
Figure 4: Circuit of the absolute-value function

5 Electronic circuit designing and Saber modeling of PWL AINN

The hidden neurone consists of three main parts of sub-systems. The block scheme of the neurone is shown in Figure 3. **The first part** is composed of three circuits of the absolute-value function; they compose a middle column in block scheme in Figure 3. It is shown the best circuit which implements the absolute-value function is the circuit in Figure 4. This circuit is composed of two operational amplifiers, four resistors and two diodes.

The second part consists of four summation amplifiers. This part is easy created by real op-

¹The source code of the learning algorithm is available on the author's e-mail address.

Figure 5: Function $2 \sin(x)$, 14 hidden neurons.Figure 6: Function $2 \sin(x)$, 16 hidden neurons.

erational amplifiers. The last third part of the hidden neurone is a DC voltage source.

6 Results

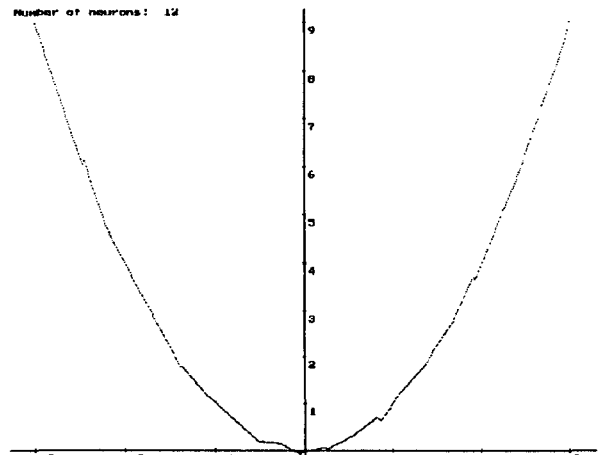
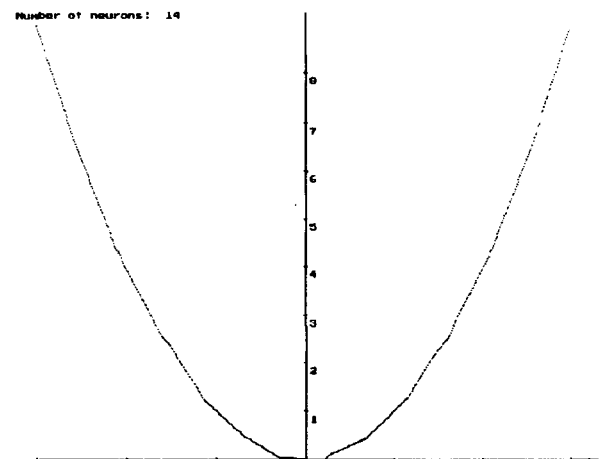
The energy function of PWL AINN has very complicate surface and convergence of the learning algorithm depends on initial values of weights. Changing the initial values the energy function will able to move itself successfully along a learning algorithm's direction toward minimum.

In Figures 5 and 6, application examples with the reference to learning of the function $f(x) = 2 \sin(x)$ are shown. The AINN making a curve in Figure 5 has 14 hidden neurons and this one making a curve in Figure 6 has 16 hidden neurons. Figures 7 and 8 show application examples with reference to learning of the function $f(x) = x^2$. You can note the degree of accuracy depends on a number of hidden neurons.

7 Conclusion

The approximate identity neural networks are universal tools for approximating any mappings and functions [1]. In additions these nets can be designed by piecewise-linear circuits which are easy implemented [5]. The PWL circuits are defined by a new triangular piecewise-linear function.

The numerical paradigm for learning of the

Figure 7: Function x^2 , 12 hidden neurons.Figure 8: Function x^2 , 16 hidden neurons.

PWL approximate identity neural networks is also presented. There are introduced rules for synthesizing of AINN circuits. Experiments show that the approximate identity neural networks could be successfully designed by using of PWL sub-circuits. There are shown examples of approximated sine and squared functions.

References

- [1] Conti M. & Turchetti C. (1994) Approximate Identity Neural Networks for Analog Synthesis of Non-linear Dynamical Systems. *IEEE Transactions on Circuits and Systems: Fundamental Theories and Applications*, vol. 41, p. 841-858.
- [2] Pospisil J., Kolka Z. & Brzobohaty J. (1995) Decomposed Parametric State Model of Piecewise-Linear Systems with the Fully Independent One-Dimensional Mapping. *Proceedings of the SYS 95*, Brno, Czech Republic, p 126-133.
- [3] Pospisil J. & Brzobohaty J. (1996) New generalised approach to the transformation properties of linear and piecewise-linear systems. *Proceedings of IASTED International Conference Modelling, Identification and Control*, Innsbruck, Switzerland.
- [4] Kaderka J. (1995) Universal Approach to Modelling of the Analog Neurone. *Proceedings of the SYS 95*, Brno, Czech Republic, p 34-37.
- [5] Kaderka J. (1996) Using Piecewise-Linear Approach to Synthesis of Approximate Identity Neural Networks. *Proceedings of the ITHURS 96*, Leon, Spain, p. 43-46.

An Integrated Testing Framework for Object-Oriented Programs

Shih-Sung Liao, Kai H. Chang and Chun-Yu Chen

Department of Computer Science and Engineering, Auburn University, Auburn, AL 36849

Keywords: Software testing, Formal methods, Object-oriented programs, Usage profile

Edited by: Rudi Murn

Received: October 4, 1995

Revised: March 15, 1996

Accepted: November 16, 1996

It has been proved that object-oriented technology (OOT) can improve software productivity and quality through object reuse and high level of code modularity. While most OOT research efforts have been devoted to the object-oriented analysis, design, and programming, little attention has been paid to program testing, especially testing beyond class level. This paper presents an integrated object-oriented programs testing framework that distributes its testing effort based on usage and importance. The framework incorporates both formal methods and usage profiles. A usage profile combines the expected data distribution information and the frequencies of anticipated operation sequences of a program. The data distribution information is represented in a grammar form that can be used to generate test data. The frequencies of operation sequences can be derived from state transition diagrams that are enriched by operation probabilities. An object-oriented styled formal specification language, which enables a clearer association between specification and implementation, is used in this framework.

1 Introduction

Object-oriented technology (OOT) is becoming increasingly popular in recent years. Different approaches in testing object-oriented programs have been proposed [5, 10, 14]. Most of these testing approaches have centered around classes or lower levels. This paper presents a new approach that incorporates different techniques into an integrated testing framework for object-oriented programs, with emphasis on system testing and acceptance testing. The main features of this approach include:

1. The use of an object-oriented formal specification in testing — Formal specification has been proved to be a useful means for improving software quality. Using object-oriented styled formal specification not only results in a conceptually clear specification to speed up prototyping and implementation, but also ultimately provides a basis for testing object-oriented programs.

2. The utilization of a usage based testing — Usage based testing tests the system from the perspective of the target user, not the software designer. Testing results can thus reflect the quality of software system with less bias when compared to the conventional coverage testing.

Section 2 reviews the relevant research on object-oriented technology, software testing, formal methods and usage profile. Section 3 gives an overview on the underlying idea of this framework. Section 4 gives an illustrative example, and Section 5 concludes the paper.

2 Background

The importance of quality assurance continues to grow along with the demand for highly complex software systems. The main thrust of software quality assurance has been on the development of formal methods and software testing. Formal methods provides a means to specify and verify the behavior of a system by applying a rigorous

mathematical notation. Software testing represents the ultimate review of software and has been used as a major technique to detect errors. With the increasing application of object-oriented programming, it is also expected that OOT would bring about significant software productivity and quality improvement through its inherent code modularity and code reuse. Object-oriented technology, software testing, formal methods and usage profile all have their impacts on software quality assurance, and are discussed in the following.

2.1 Object-oriented technology and software testing

There is no doubt that object-oriented analysis and design of software are fundamentally different from the conventional structured approach. This also leads to a different approach to testing object-oriented programs. Several properties of object-oriented programming affect its testing, for example, inheritance, overload, and polymorphism. Smith [14] classified testing object-oriented programs into four different levels of abstraction: algorithmic, class, cluster, and system.

For an object-oriented program, it is obvious that the natural units of a program are classes and objects. Class testing is the first level that brings attention to testing object-oriented programs. Smith used a modified data flow testing approach and a class flattening (the inherited parts in a class are flattened out) methodology to address the testing problem at the class level [14]. Also based on inheritance, Harrold and her colleagues [10] presented an incremental approach to test classes by exploiting the hierarchical nature of the inheritance relation and reusing the testing information of a parent class to guide the testing of a subclass. Doong and Frankle [5] built a prototype testing system for the class level, and focused on the issue of whether a sequence of messages puts an object of a class under test into a correct state. In the limited object-oriented program testing research, efforts have been focused on the unit (class) level.

2.2 Formal Methods

The scope of formal methods includes formal specification and verified design [4]. The underlying idea is that the behavior of software is first de-

scribed in a formal specification, then the software is implemented according to its specification, and finally the conformance between specification and implementation is checked. For project managers or high-level designers, a formal specification describes the problem concisely and precisely. For testers or maintainers, the specification provides important information about the product's correct behavior.

With the emergence of formal specification, automated test data generation based on specification has become feasible. Stocks and Carrington presented a specification-based testing framework that is designed for any model-based specification notation [17]. Richardson *et al.* [13] combine real time interval logic and the Z specification language [15] to derive test oracles from specifications and incorporate them in the testing process for reactive systems. Carrington and Stocks also gave a summary of some major efforts that apply formal specification languages (especially Z) to software testing [2]. Formal methods are leading an evolution rather a revolution in the field of software testing. It is noteworthy that none of these approaches to specification-based testing uses an object-oriented specification language.

Another trend in formal methods is the development of object-oriented styled formal specification. Most conventional formal specification languages are not object-oriented. Various object-oriented formal specification methods have been proposed. For example, Fresco [18], an extension to VDM, is intended to facilitate the description of reusable software components and provides an environment for rigorous development of object-oriented software from a specification. A wide-spectrum language, COLD (common object-oriented language for design), along the tradition of VDM and Z, was developed at the Philips Research Laboratories in Eindhoven [8]. COLD can be used as an algebraic specification language, as well as an integrated language unifying algebraic and state-based techniques. More than half a dozen projects describing different approaches for providing Z with object-oriented structuring mechanisms are collected in [16], which includes Mooz, Object-Z, OOZE, Z++ and ZEST+.

2.3 Usage profile

For a moderate to large software product, there are usually an infinite number of possible executions. Since there are too many test cases to choose from, test data selection criteria must be used. Although numerous testing adequacy criteria have been investigated or experienced, none of them can be proven to be better than others for an arbitrary program [9]. When dealing with higher-level testing, such as system testing and acceptance testing, the most critical issue is how the user will perceive the product. Thus, higher level testings, should be conducted from the perspective of the user rather than the designer. Usage profiles provide the fundamental information for test data selection from the user's perspective.

Different user groups may use a software product in different ways, and corresponding profiles can be defined to describe the different usage patterns. Similarly, some infrequently used functions with serious consequences for failure (e.g. emergency handling functions) can also be expressed by special profiles. Five steps to develop an operational profile are described in [11]: (1) Find the customer profile; (2) Establish the user profile; (3) Define the system mode profile; (4) Define the functional profile; and (5) Define the operational profile itself. In these steps, a particular usage is progressively broken down into more and more detailed levels. However, some steps may not be necessary for a particular application. For example, a customer profile is unnecessary if there is only one customer group. According to [12], for an average-sized project, the cost of developing an operational profile at AT&T is about one staff month.

From the perspective of testing, the most-used operations should receive the most testing and should be most reliable. In terms of MTTF, "It is shown based on a study of a number of projects that usage testing improves the perceived reliability during operation 21 times greater than that using coverage testing [3]." Both theory and practical experience indicate that usage profiles can provide a great means of improving software quality from the user's perspective. In fact, software reliability is reflected by the way a user will use the system.

3 An overview

The major difference between testing object-oriented programs and conventional structured programs is the software design architecture. In the conventional structured design, subprograms are basic building blocks. A conceptually clear tree-like structure shows the dependencies among modules. A usage distribution can be easily assigned to this structure. The hierarchy of modules is simple and clearly represented. Unit testing starts with functions and procedures. Integration testing points can be chosen from intermediate nodes based on different approaches, and system testing can be performed incrementally. When applying usage-based testing for a structured design, test cases can be generated to exercise modules according to their probabilities.

To apply usage based testing in an object-oriented environment, an entirely new approach must be developed. For an object-oriented design, the basic building blocks are classes and objects. Data communication between classes is by message passing. The topology of an object oriented design is no longer a tree, and there is no easy way to establish a structural hierarchy. The usage model of an OOP application should be derived from its operation.

The main ideas of this work include (1) the translation of ambiguous requirement descriptions into a rigorous formal specification and (2) the definition of usage profile that define data distribution and operation frequency. It has been reported that formal methods and statistical usage based testing have been successfully combined in the Cleanroom approach [3, 7], which resulted in several ultra-high quality software products. Although the Cleanroom paradigm is not specially designed for object-oriented programs, the ideas of Cleanroom are adopted in our research. Cleanroom uses a box structure formal specification methodology to describe a software system. The specification hierarchy contains three distinct box structure forms: black box, state box, and clear box. The underlying idea of the box structure is abstraction. In Cleanroom, the conventional structural testing is replaced by correctness verification, while the functional testing strategy still remains and is called statistical usage testing. Statistical usage testing requires an investment in front-end analysis and then planning for func-

tional testing based on usage probabilities. The usage probability distribution is expressed in an executable usage grammar called usage distribution language (UDL). By using a translator to convert UDL statements and running a statistical test case generator, any number of test cases can be created according to the distribution [3, 7].

Cleanroom approach works fine for structured program development. However, in order to incorporate the Cleanroom approach into the object-oriented program testing, it is necessary to modify the existing techniques. First, although the box structure gives a clear specification and good representation for verification, extensive human intervention is needed. Also, the box methodology is basically structure-oriented, not object-oriented. An object-oriented formal specification language is needed to mirror the implementation. Second, the usage analysis can be independent of the design methodology, but the mapping between usage and implementation varies drastically for different design methodologies. The mapping between usage and implementation must be studied when deriving the operation profile. Third, data distribution information can be added for statistical test data generation, and will provide a natural data usage simulation.

3.1 The framework

Testing can not be a stand-alone process in the software development life cycle. Instead, test teams must work closely with the analysis and design teams. The framework of our approach is shown in Figure 1. In the analysis and design phases, three aspects of a system are presented: domain knowledge, static class structure, and dynamic interaction of operations. These aspects must be represented precisely, so that the tester can have accurate information regarding the system. An object-oriented formal specification language can serve this purpose. The specification language provides a means for the tester to retrieve the information without going into the implementation details. The formal specification, which provides pre- and post-conditions for each operation, can then be used as a basis for testing and verifying the correctness of execution. The purposes of pre- and post-conditions are to indicate the system states that must hold before exercising an operation and the effects of an operation,

respectively.

Usage profiles include both data usage profiles and operation usage profiles. The data usage profile includes the data dictionary and the data distribution. The operation usage profile describes the operation probability based on the analysis result of projected user applications (or user groups). A state transition diagram is used to describe the operation sequences of the application. A test scenario, which is a sequence of message passing paths along with the expected results, can be derived from the state transition diagram. Any number of test cases can then be generated from a test scenario. If different usage patterns exist among user groups, individual usage profiles can be derived for each group.

Testing is inadequate unless testing results can be verified thoroughly against the specification. Unfortunately, software testing are usually not analyzed thoroughly, either due to the time required, or due to the lack of tools and the absence of specification. Since the formal specification is incorporated into this framework, it becomes possible to generate the desired effects for each cause systematically. During execution, each message passing event will be monitored for its pre- and post-conditions. In practice, no oracle can generate precise expected results for a particular test case, otherwise it could replace the code itself. By improving the testing process in terms of test cases and oracle generation, the quality of software can thus be measured statistically in terms of MTTF. Product released by this testing framework will gain a significant degree of quality assurance.

Our approach is centered around the conformance between implementation and specification, while assuming the specification has met the customer's requirements. Although the underlying ideas can be language independent, the effective use of a particular language is important in object-oriented design. This framework is targeted at programs written in C++ and the formal specification language used in this framework is Object-Z [6]. C++ and Object-Z are chosen for many solid reasons. C++ is a stable and full-scale production language. The Z specification language [15], is a well-known formal method with a very popular user group in both the UK and the USA. Object-Z is an extension language of Z

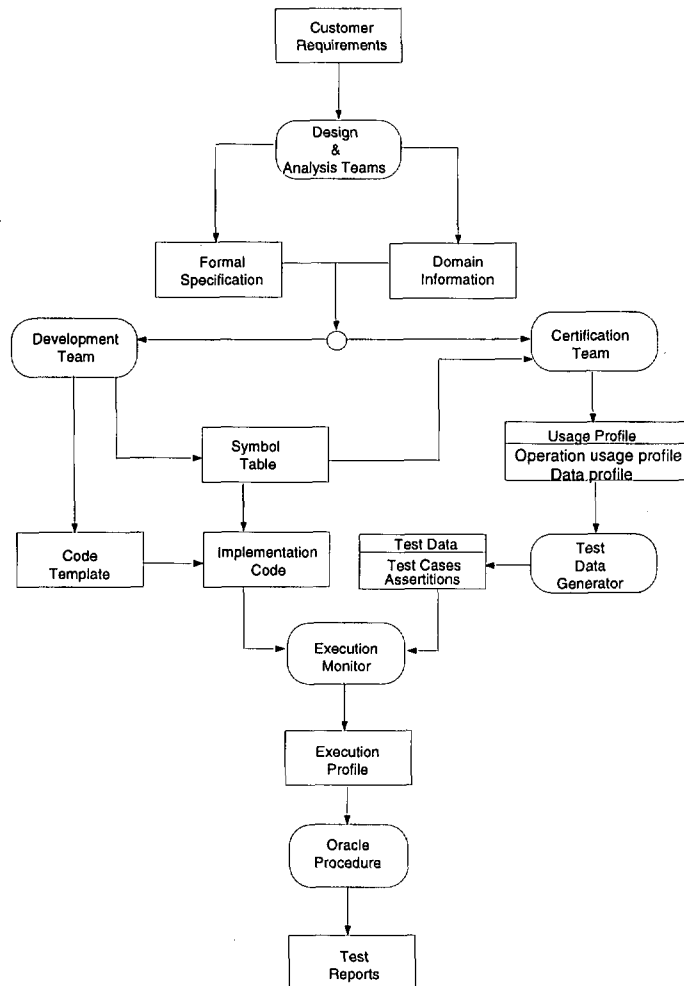


Figure 1: The framework

which embraces the object-oriented style, while still keeps the advantages of Z. In addition, there is also a strong correlation between an Object-Z class and a C++ class, including class constants, inheritance and operations (member functions in C++).

3.2 Specification of the design

A formal specification that embraces an object-oriented style will not only speed up the prototyping and implementation of object-oriented programs, but also direct the selection of test cases and oracles. Object-Z was developed by the Software Verification Research Center at the University of Queensland [6]. Encapsulation, inheritance and polymorphism were key concepts in developing Object-Z. Similar to C++, an object in Object-Z is described by its states and re-

lated operations. A specification box which represents typical Object-Z definition format is given in Figure 2. A class definition in Object-Z may include inherited classes (names of ancestor classes), type and constant definitions (collectively called attributes), at most one state schema (nameless and containing declarations and a predicate), at most one initial schema (initial instances for the class), zero or more operation schemas (enabling communication to the environment), and an optional history invariant. A class consists of a name box in which the features of the class are described and related.

The detailed discussion of Object-Z is beyond the scope of this paper. An introduction to Object-Z can be found in [6]. Figure 3 shows an example of Object-Z specification and its corresponding C++ implementation. The specification clearly specifies the behavior for each schema

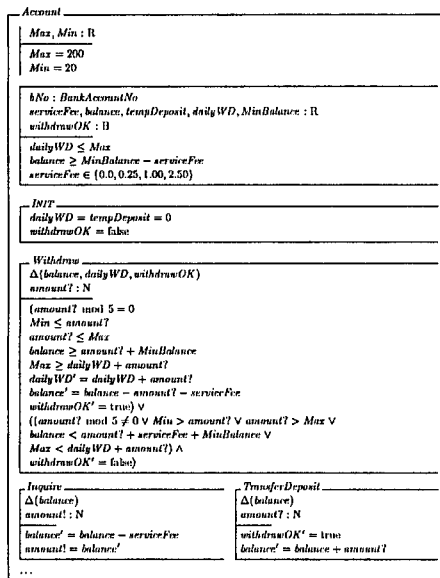


Figure 2: Object-Z class definition.

(operation or method). The specification can be translated into assertions which then can be used to monitor program execution. The mapping between the specification and implementation is very direct. This indicates that formal specification also serves as a very useful means for code verification during development. While verification may not totally replace class level testing, it would definitely improve the correctness of classes. An automatic teller machine (ATM) system is used as one of our target testing systems. Figure 4 shows a portion the Object-Z specification of an account class. A complete specification of the ATM is given in [11].

3.3 Usage of the system

In Booch's definition [1], "A *state transition diagram* is used to show the state space of a given class, the events that cause a transition from one state to another, and the actions that result from a state change." In our approach, a complete system is presented by the state transition diagram. We define an embedded finite-state automaton (EFSA) to describe the state transition diagram. The definition follows:

An Embedded Finite-State Automaton (EFSA) is an nine-tuple

$$(E, X, f, p, R, Op, x_0, S, F)$$

where

E is a *finite event set*

X is a *finite state set*

f is a *state transition function*,

$$f : X \times E \rightarrow X$$

p is a *usage function*, $p : E \rightarrow g, 0 < g \leq 1$

R is a *finite operation information set*

Op is an *operation function*, $Op : E \rightarrow R$

x_0 is an *initial state*, $x_0 \in X$

S is a *set of superstates*, $S \subseteq X$

F is a *set of final states*, $F \subseteq X$

A state that contains nested states is called a *superstate*, and its nested states are called *sub-states*. $\forall s \in S$, s can be defined by another EFSA, i.e., a decomposed state transition diagram. This will provide an abstraction of the state transition diagrams. The state transition diagram is developed incrementally, and normally, has a hierarchical structure. At the highest level, a state transition diagram is constructed according to the major operation steps. At the lowest level, a state transition diagram must be associated with an operation in a class. The multiple levels of presentation will provide the tester a direct mapping between design structure and usage, and allow the user to view the design in an understandable way. After the state transition diagram is developed, different usage profiles representing different user groups or different criteria can be applied to the state transition diagram to develop the usage models. The sequence of operations are then derived from the usage models, which are called *test scenarios*. We first need to prepare testing scenarios for normal usage, i.e., applications without any unusual or exceptional conditions. Different criteria can then be used to define additional test scenarios.

$\forall r \in R$, r includes the information of input, output parameters, a list of called functions, pre- and post-conditions. The names of input and output parameters will be used by the test data generator to generate input data and by the execution monitor to check output results. The pre- and post conditions are obtained from the Object-Z specification. These conditions can be interpreted by human, or by an oracle tool once a reasoning tool of Object-Z is incorporated. A sequence of operations is represented by a sequence

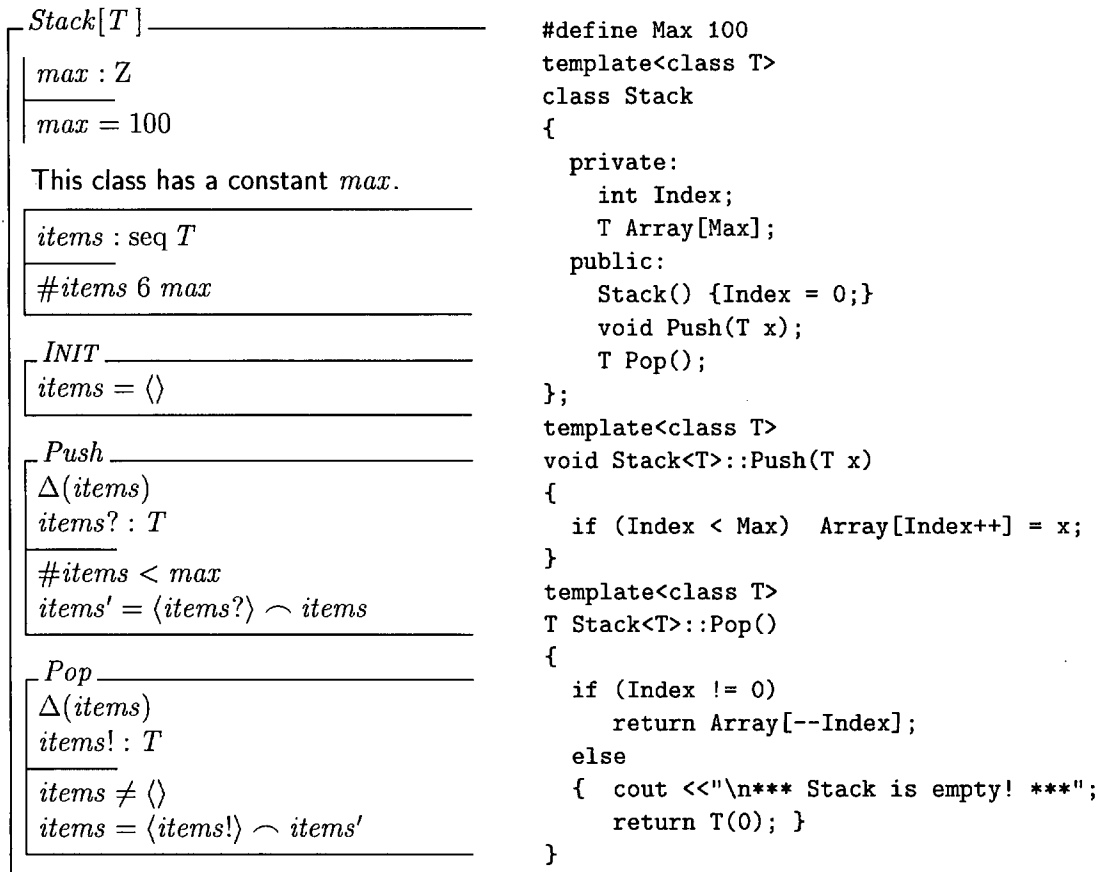


Figure 3: A specification and implementation of a generic stack class.

of events(or states). The probability of a sequence represents the usage probability of this operation, and the probability of a sequence is the product of probability of each event in a sequence. When there is no loop in an EFSA, the set of all possible sequences is finite and can be collected, but this is rarely the case in real applications. A threshold of probability and an upper limit of loop iterations are used to filter out impossible sequences or some sequences with very low probability. In this case, a finite set of sequences will be impossible to cover all sequences. However, it is well known that exhausted white or black box testing is impossible for most applications. The rationale of the testing based on the EFSA is that the testing will be based on the usage of the system, and thus the testing efforts will be distributed according to its importance.

After usage scenarios are constructed, test cases

can be generated according to the data information. A data domain profile is used to define the data information. The data domain profile uses the Extended Backus-Naur Form (EBNF) to describe the data type definition and distribution. Structured data types will be defined based on other simpler or primitive data types. A tree structure can be used to depict the structure data type. Each node in the tree represents a data type. A primitive node is called a leaf node or a terminal. The portion of grammar that describes leaf nodes is:

```

type ::= [<continuous_type> | <discrete_type> |
<enumerate_type>]
leafnode ::= <type><ranges><resolution>
<exceptions><distribution><parameters>
                
```

In this grammar, a default value will be assigned when a pair of blank brackets is used at

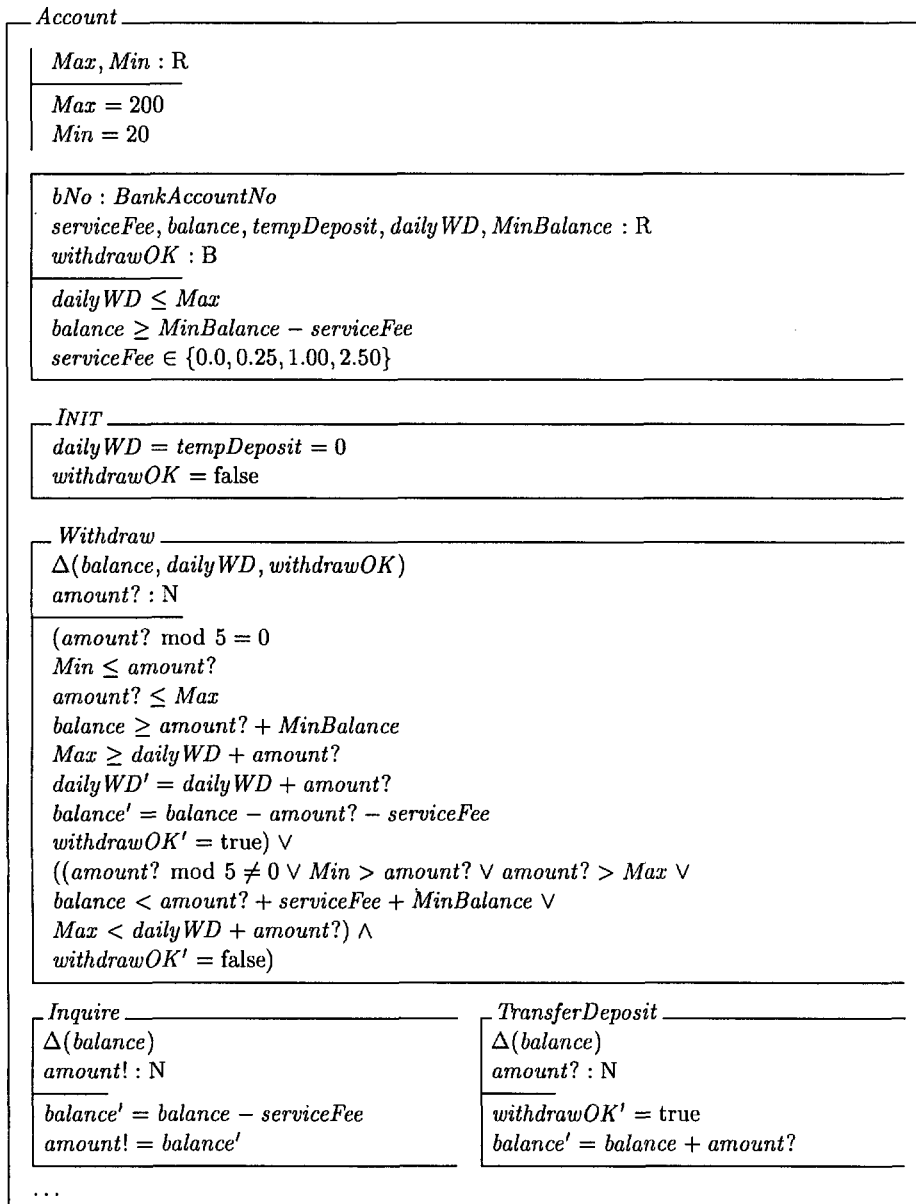


Figure 4: *Account* class in Object-Z

the position of any tuple. During testing, the data type is used to generate appropriate types of test data for variables. The test data generator can thus process the grammar description to generate any number of test cases according to the desired data distribution. The usage model and data domain profile allow the execution of test cases to emulate the actual application of software system.

4 An example

An ATM example is used to illustrate our approach. The top level state transition diagram of the ATM is shown in Figure 5, which represents the major operation states of the system. The arcs represent the events that cause state transitions. Since an emergency can happen in any state, a separate node is used to prevent clustering of the diagram, especially at the lower levels. The arcs entering or leaving the emergency state are represented by dotted lines to distinguish them from normal state transitions. Based on the state transition diagram, the operation sequences of the ATM can be described. The idle state is *Startup* (S). When a customer inserts a card, it moves to *Validating User* (V). In V, the personal identification number (PIN) is entered and verified. If the verification succeeds, then the system moves to *Validating Transactions* (T), otherwise the user can make two more tries, or quit and return to Startup. In T, if the transaction request is valid then the system moves to *Processing Transactions* (P), and finally returns to Startup. A normal operation sequence like this will be represented by a sequence of states: (S)(V)(T)(P)(S) or be represented by a sequence of events (0)(1)(4)(6)(8). After refinement, these states can be decomposed into lower level states. Figure 6 shows the first decomposition of the Validating Transaction state.

Using an OOA decomposition approach, these states will eventually correspond to the class operations. A sequence of events with frequency information is then used as a test scenario. By incorporating the data profile represented in EBNF, test data can be generated. An example of the data profile is given in Table 1. When a structured data type is selected, the parsing will continue until it reaches primitive types, which are enclosed in a pair of brackets (e.g. WithdrawAmount). The values to be generated for WithdrawAmount must

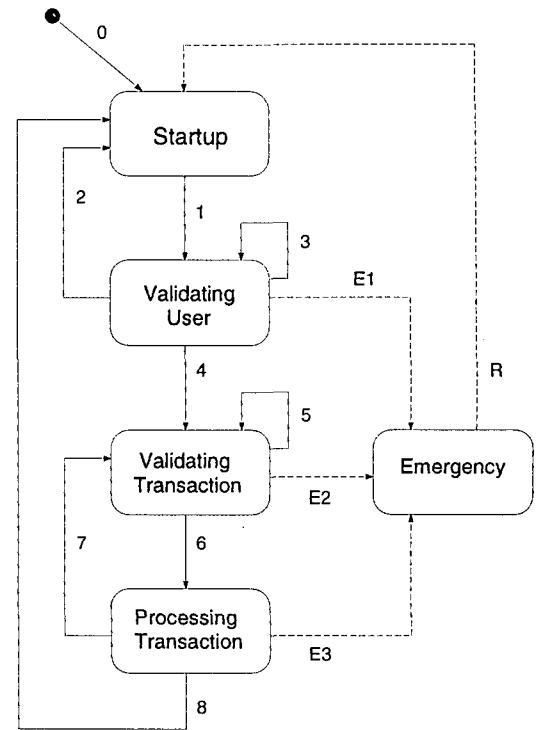


Figure 5: ATM top level STD

satisfy its definition, i.e., the range is greater than or equal to 20 and less than or equal to 200, the resolution is 5, no exception, and the distribution is a normal distribution type with mean = 86 and standard deviation = 42.06. Although the sample distribution may not be exactly the same as the actual distribution, the usage distribution provides a feasible way for the emulation of the usage.

In the EBNF, a default value is provided for each category. For example, the default distribution type is a uniform distribution and the default resolution for the integer type is 1. A data dictionary defined in the EBNF format would not only describe data distribution, but also provide a means in applying other test data selection criteria, e.g., boundary values and exception values. The test data generator can automatically generate test data when the test scenarios and EBNF definitions are provided.

Table 1: Example of ATM data dictionary

CustomerName	::= LastName + FirstName + MI
MI	::= [null letter]
LastName	::= Name
FirstName	::= Name
Name	::= letter(1..26)
letter	::= <char><[a..z,A..Z]>
RequestTypeInfo	::= [InquiryInfo TransferInfo DepositInfo WithdrawInfo]
InquiryInfo	::= AccountNo + CurrentBalance
WithdrawInfo	::= AccountNo + WithdrawAmount + CurrentBalance
AccountNo	::= BankNo + BankAccountNo
CardNo	::= [digit(10) digit(13) digit(16)]
BankNo	::= <string(6)><(100000..999999)>
BankAccountNo	::= <string(8)><(10000000..99999999)>
CurrentBalance	::= <float><minimalBalance..maximalBalance><0.01>
WithdrawAmount	::= <integer><[20..200]><5><><norm><86,42.06>

5 Conclusions

An usage-based testing framework is presented in this paper. Because of time and resource constraints, test selection should be done according to the importance of software components. With this philosophy, once the test is terminated, the most important and most frequently exercised parts of the software will have thus received most attentions and should be most reliable. Since testing will never be perfect except for very small programs, the errors detected based on usage will be in the order of its significance. This will directly contribute to the software's reliability. If the MTTF is larger than the span of the expected software life, the errors still contained in the software will have no effect to the end-users.

The objects and classes of the object-oriented model contain operations (member functions) and attributes that reflect the state of an object. The dynamic behavior of the object-oriented software system is described in EFSA to model its state. The EFSA also describes the usage of the system, as well as the dynamic features of the class function invocations. The usage of a system will be represented as the frequencies of operation sequences rather than the probabilities of modules exercised. The use of formal specification also facilitates the automatic generation of oracle. In general, a specification cannot be used to derive

the precise output values for specific test datum. Thus, mechanisms must be developed to derive invariant oracle information from the specification and use a corresponding oracle procedure to deal with dynamic assertions. In addition, oracle can not simply be done by comparing the expected outputs with execution results at the end. The intermediate states (or values) also need to be checked. The development of a tool to reason the formal specification is still under investigation.

Our current effort is to construct the EFSA grammar and the EBNF for the ATM system, and to implement the test data generator. While this research emphasizes on the testing aspect of object-oriented programs, ultimately, it can also lead to a systematic approach for object-oriented program development by integrating analysis, design, testing and verification.

References

- [1] G. Booch, *Object oriented design with applications, 2nd ed.*, Benjamin Cummings, 1994.
- [2] D. Carrington and P. Stocks, "A tale of two paradigms: formal methods and software testing", In *Workshops in Computing Series: Z User Workshop*, pages 51-68. Springer-Verlag, 1994.

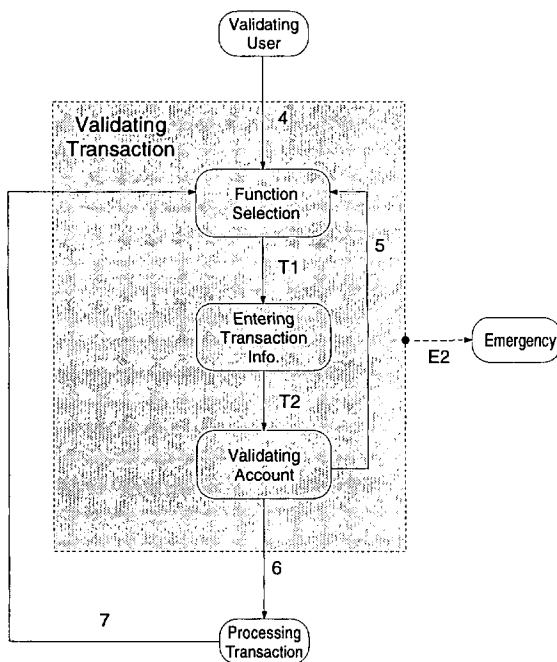


Figure 6: A Decomposed STD of the ATM

- [3] R.H. Cobb and H.D. Mills, "Engineering software under statistical quality control", *IEEE Software*, pages 44-54, November 1990.
- [4] A. Diller, *An introduction to formal methods*, John Wiley & Sons, 1990.
- [5] R. Doong and P. G. Frankl, "The Astoot approach to testing object-oriented programs", Technical Report pucs-104-93, Department of Computer Science, Polytechnic University, 1993.
- [6] R. Duke, P. King, G. Rose, and G. Smith, "The Object-Z specification language, Version 1", Technical Report 91-1, Software Verification Research Centre, Department of Computer Science, University of Queensland, May 1991.
- [7] M. Dyer, *The Cleanroom approach to quality software development*, John Wiley & Sons, 1992.
- [8] L.M.G. Feijs and H.B.M. Jonkers, *Formal specification and design*, Cambridge, 1992.
- [9] P. G. Frankl and S. N. Weiss, "An experimental comparison of the effectiveness of branch testing and data flow testing", *IEEE Trans. on Soft. Eng.*, 19(8):774-787, Aug. 1993.
- [10] M. J. Harrold, J. D. McGregor, and K. J. Fitzpatrick, "Incremental testing of object-oriented class structures", In *Proc. 14th International Conference on Software Engineering*, pages 68-80, May 1992.
- [11] S. Liao, *An Integrated Testing Approach for Object-Oriented Programs*, PhD Dissertation, Dept of Computer Science and Engineering, Auburn University, March 1997.
- [12] J. D. Musa, "Operational profiles in software reliability engineering", *IEEE Software*, pages 14-32, March 1994.
- [13] D. J. Richardson, S. L. Aha, and T. O. O'Malley, "Specification-based test oracles for reactive systems", In *Proc. 14th International Conference on Software Engineering*, pages 105-118, Melbourne, Australia, May 1992.
- [14] M. D. Smith and D.J. Robson, "A framework for testing object-oriented programs", *Journal of Object-Oriented Programming*, 5(3):45-53, June 1992.
- [15] J. M. Spivey, *The Z notation: a reference manual, 2nd ed.*, Prentice Hall, 1992.
- [16] S. Stepney, R. Barden, and D. Cooper (Eds.), *Workshops in computing series: object-orientation in Z*, Springer-Verlag, 1992.
- [17] P. A. Stocks and D.A. Carrington, "Test templates: a specification-based testing framework", In *Proc. 15th International Conference on Software Engineering*, pages 405-414, Los Alamitos, CA, 1993.
- [18] A. Wills, "Specification in Fresco", In *Workshops in Computing Series: Object-Oriented Orientation in Z*, pages 125-135. Springer-Verlag, 1992.

Call for Papers: Consciousness as Informational Phenomenalism

Special Issue of Informatica 21 (1997) No. 3

Informatica, an International Journal for Computing and Informatics, announces the Call for Papers for the issue of an interdisciplinary volume dedicated to the informational problems of consciousness.

The scientific program of the volume includes the following:

1. consciousness as an informationally emerging entity in events, processes and systems of understanding;
2. innovative formal symbolism for study, research and expression of dynamically structured and organized (arising, emerging, generic) events, processes, and systems of consciousness;
3. philosophical (existence, phenomenology), cognitive (intention, qualia, understanding), linguistic (semiotic, pragmatic), psychological (experience, feeling), physiological, neuronal (connectionist), cellular (biological), cybernetic (self-organized) and other views of consciousness as informational phenomenon;
4. physical (space-time, quantum, thermodynamical), chemical (molecular) and other natural models of consciousness as informational phenomena;
5. consciousness as learning, memorizing, associative, concluding, and intelligent processes of behavior;
6. classical, computational and artificial-intelligence approaches (stressing artificialness and constructivism) for understanding and modeling of the consciousness phenomenology;
7. emerging terminology and systematics (structure, organization) of consciousness.

Informatica 21 (1997) No. 3, in an enlarged volume, is fixed as the special issue.

The deadline for the paper submission in three copies is **May 15, 1997**. International refereeing will be performed according to the standard *Informatica* procedure. For more instructions see FTP `ftp.arnes.si` with anonymous login or URL: <http://www2.ijs.si/~mezi/informatica.html>.

Correspondence: E-mail addresses: `anton.p.zeleznikar@ijs.si`, `matjaz.gams@ijs.si`, and `mitja.perus@uni-lj.si`.

Printed-paper mail address: M. Gams, Jožef Stefan Institute, Jamova c. 39, SI-1111 Ljubljana, Slovenia.

Consciousness Scientific Challenge of the 21st Century

Edited by D. Raković and D. Koruga
Belgrade 1996

Published by the European Centre for
Peace and Development (ECPD) of the
United Nations University for Peace

The book (ISBN 86-7236-005-2) is a collection of papers presented by the Belgrade symposium *Consciousness: Scientific and Technological Challenge of the 21st Century*, held during 29-30 May 1995. It represents the activities of the Joint Laboratory of Belgrade scientific community.

The book is divided into four parts. The first part deals with *Phenomenology of Consciousness* and includes the following contributions: *Self-consciousness of the first civilization: The case of the divine Pelasgians of the Balkans* (L. Klakić); *The phenomenon of consciousness in philosophy* (V. Abramović); *Consciousness as a (psychological) function* (P. Ognjenović); and *Biological basis of consciousness* (V. Desimirović).

The second part, entitled by *Altered States of Consciousness*, begins with a survey of the structure of Universe, human selfhood, bodies, states of consciousness, psychic organization, and attainment of higher states of consciousness in esoteric practices [*States of consciousness in esoteric practice* (P. Vujićin)]. A further example in the direction of the altered states of consciousness is presented by the application of psychotherapeutic ritual in Amazon tribal societies, with shamanistic control and interpretation of hallucinogenic altered states of consciousness [*Psychotherapeutic ritual in Amazon tribal societies* (Č. Hadži-Nikolić & B. Petković)]. The last contribution of the second part deals with a survey of contemporary methods of neurolinguistic programming, including original integrative model for efficient hypnotherapeutic reprogramming of old behavioral models [*Neurolinguistic programming: An integrative model for states of consciousness* (G. Stanojević-Vitaliano)].

The third part encompasses *Electroencephalographic Correlates of States of Consciousness*. It

begins with a broad survey of phramacoelectroencephalography (PEEG), i.e. by an electroencephalographic study of drug effects, with significant clinical implications [*EEG studies of drugs acting on the central nervous system* (Ž. Martinić)]. It is followed by a detailed relationship between clinical neurophysiological polysomnographic data and different sleep disorders [*EEG and the sleep disorders* (N. Ilanković & A. Ilanković)]. Out of the new methods of EEG signal analysis, the application of the theory of deterministic chaos is given, illustrated in the cases of normal and pathological EEG [*Deterministic chaos in EEG signal* (V. Radivojević, M. Rajković, D. Timotijević & M. Car)]. The next contribution presents an original methodology and software environment for quantitative analysis of EEG activity in altered states of consciousness, with particular application on the monitoring of the healing process [*On methodology of EEG analysis during altered states of consciousness* (E. Jovanov)].

In Search of the New Paradigm is the title of the fourth part. It brings some original scientific approaches to the problem of consciousness, bearing characteristics of a new scientific synthesis. It deals with concepts of information physics as a synergetic theory of classical and quantum mechanics and theory of information, which relates consciousness with biology and physics, and searches the roots in biophysical cytoskeletal processes [*Information physics: In search of a scientific basis of consciousness* (D. Koruga)]. The second approach points out a universal Mind/Matter code starting from the unity of chemical and genetic codes, unifying global-integral introspective method of the East and single-partial empirical method of the West [*The universal consciousness and the universal code* (M. Rakočević)]. The original triunism concept is presented in *Brain and thought in neurobiological context* (L. Rakić). Finally, a new biophysical model of altered states of consciousness is presented in the view of electromagnetic field of brainwaves, dynamics of psychological processes, and bizarre transpersonal phenomena in transitional states of consciousness [*Brainwaves, neural networks, and ionic structures: Biophysical model for altered states of consciousness* (D. Raković)].

By A.P. Železnikar

Call for Papers

Parallel and Distributed Database Systems

Special Issue of Informatica

Parallel and distributed database technology is a core of many mission-critical information systems. New challenging problems are posed by the growing demand for large-scale, enterprise-wide, high-performance solutions. Innovative approaches and techniques are necessary to deal with the rapidly expanding expectations with regard to massive data volume processing, performance, availability, and solutions scalability.

The scope of this Special Issue includes all aspects of parallelism and distribution in database systems. The Issue will focus on design, development and evaluation of parallel and distributed database systems for different computing platforms and system architectures.

Original papers are solicited that describe research on various topics in parallel and distributed database systems including, but not limited to, the following areas:

- Distributed database modeling and design techniques
- Parallel and distributed object management
- Interoperability in multidatabase systems
- Parallel on-line transaction processing
- Parallel and distributed query optimization
- Parallel and distributed active databases
- Parallel and distributed real-time databases
- Multimedia and hypermedia databases
- Databases and programming systems
- Mobile computing and databases
- Transactional workflow control
- Parallel and distributed algorithms
- Temporal databases
- Data mining/Knowledge discovery

- Use of distributed database technology in managing engineering, biological, geographic, spatial, scientific, and statistical data

- Scheduling and resource management

Time Table and Contacts

Papers in 5 hard copies should be received by November 1, 1996 at one of the following addresses.

North & South America, Australia:
Bogdan Czejdo czejdo@beta.loyno.edu, Department of Mathematics and Computer Science, Loyola University, 6363 St. Charles Ave., New Orleans, LA 70118, USA

Europe, Africa, Asia:
Tadeusz Morzy morzy@pocz1v.put.poznan.pl, Institute of Computing Science, Poznan University of Technology, Piotrowo 3a, 60-965 Poznan, Poland

Silvio Salza salza@dis.uniroma1.it, Dipartimento di Informatica e Sistemistica, Universita di Roma La Sapienza, Via Salaria 113, I-00198 Roma, Italy

E-mail information about the special issue is available from the above guest editors.

Notification of acceptance will be sent before March 1, 1997. The special issue will be published in the middle of 1997.

Format and Reviewing Process

Papers should not exceed 5,000 words. Each paper will be refereed by at least three anonymous referees.

Call for Papers

Parallel Computing with Optical Interconnections

Special Issue of *Informatica*

<http://www.cps.udayton.edu/pan/info>

Communications among processors in a parallel computing system are always the main design issue when a parallel system is built or a parallel algorithm is designed. With advances in silicon and Ga-As technologies, processor speed will soon reach the gigahertz (GHz) range. Traditional metal-based communication technology used in parallel computing systems is becoming a potential bottleneck. This requires either that significant progress need to be made in the traditional interconnects, or that new interconnect technologies, such as optics, be introduced in parallel computing systems.

Fiber optic communications offer a combination of high bandwidth, low error probability, and gigabit transmission capacity and have been used extensively in wide-area networks. Advances in optical and optoelectronic technologies indicates that they could also be used as interconnects in parallel computers. In fact, many commercial massively parallel computers such as the Cray T3D use optical technology in their communication subsystems. Papers in this special issue will be selected to focus on the potential for using optical interconnections in massively parallel processing systems, and their effect on system and algorithm design.

The topics of interest include but are not limited to the following:

- Various optical interconnections,
- Optical pipelined buses,
- Multistage interconnection networks,
- Reconfigurable optical architectures,
- Embedding and mapping of applications and algorithms,
- Emulation of different models,
- Algorithms and applications exploiting parallel optical connections,
- Data distribution and partitioning,

- Task scheduling,
- Performance Evaluation,
- New analytical methods for optical interconnections,
- Scalability analysis,
- Computational and communication complexities.

Publication is scheduled for an issue in 1998. Four copies of complete manuscripts should be sent to one of the guest editors (see address below) by May 15, 1997. All manuscripts must conform to the normal submission requirements of *INFORMATICA*. Notification of acceptance will be sent by October 15, 1997.

Guest Editors of the Special Issue

North & South America, Australia

Professor Yi Pan, Computer Science Department, The University of Dayton, Dayton, OH 45469-2160, USA, Email: pan@hype.cps.udayton.edu

Professor Keqin Li, Dept. of Mathematics and Computer Science, State University of New York, New Paltz, New York 12561-2499, USA, Email: li@mcs.newpaltz.edu

Europe, Africa, Asia

Professor Mounir Hamdi, Department of Computer Science, Hong Kong University of Science and Technology, Clear Water Bay, Kowloon, Hong Kong, Email: hamdi@cs.ust.hk

Call for Papers (New Journal)**Intelligent Data Analysis—An International Journal**

An electronic, Web-based journal

Published by Elsevier Science

URL: <http://www.elsevier.com/locate/ida>,

<http://www.elsevier.nl/locate/ida>

Important e-mail addresses:

Editor-in-Chief: A. Famili

(famili@ai.iit.nrc.ca)

Editorial Office: Heather D. Joseph

(h.dalterio@elsevier.com)

Subscription Information:

USDirect@elsevier.com

Introduction

As science and engineering disciplines become more and more computerized, the volume and complexity of the data produced on a day-to-day basis quickly becomes overwhelming. Traditional data analysis approaches have proven limited in their ability to generate useful information. In a wide variety of disciplines (as diverse as financial management, engineering, medical/ pharmaceutical research and manufacturing) researchers are adapting Artificial Intelligence techniques and using them to conduct intelligent data analysis and knowledge discovery in large data sets.

Aims/Scope

The journal of Intelligent Data Analysis will provide a forum for the examination of issues related to the research and applications of Artificial Intelligence techniques in data analysis across a variety of disciplines. These techniques include (but are not limited to): all areas of data visualization, data pre-processing (fusion, editing, transformation, filtering, sampling), data engineering, database mining techniques, tools and applications, use of domain knowledge in data analysis, machine learning, neural nets, fuzzy logic, statistical pattern recognition, knowledge filtering, and post-processing. In particular, we prefer papers that discuss development of new AI architectures, methodologies, and techniques and their applications to the field of data analysis. Papers pub-

lished in this journal will be geared heavily towards applications, with an anticipated split of 70 oriented, and the remaining 30

Editor-in-Chief

A. Famili

National Research Council of Canada,
Canada

Editorial Board

Timothy Bailey, San Diego Supercomputer Center, USA

Francesco Bergadano, University of Torino, Italy

Pierre Boulanger, National Research Council of Canada, Canada

Pavel Brazdil, University of Porto, Portugal

Carla E. Brodley, Purdue University, USA

Paul R. Cohen, University of Massachusetts, USA

Luc De Raedt, Catholic University of Leuven, Belgium

Doug Fisher, Vanderbilt University, USA

Matjaz Gams, Jozef Stefan Institute, Slovenia

James Garrett, Jr., Carnegie Mellon University, USA

Larry Hall, University of South Florida, USA

Alois Heinz, Universitaet Freiburg, Germany

Achim G. Hoffmann, University of New South Wales, Australia

Jane Hsu, National Taiwan University, Taiwan

Scott Huffman, Price Waterhouse Technology Center, USA

Xiaohui Liu, University of London, UK

Ramon Lopez de Mantaras, Artificial Intelligence Research Institute, Spain

David Lubinsky, University of The Witwatersrand, South Africa

Nicolaas J.I. Mars, University of Twente, The Netherlands

Stan Matwin, University of Ottawa, Canada

Claire Nedellec, Universite Paris-Sud, France

Raymond Ng, University of British Columbia, Canada

Alun Preece, University of Aberdeen, UK

Lorenza Saitta, University of Torino, Italy

Alberto Maria Segre, The University of Iowa, USA

Wei-Min Shen, University of Southern California, USA

Evangelos Simoudis, IBM Research Almaden Research Center, USA

Stephen Smith, Carnegie Mellon University, USA

Tony Smith, University of Waikato, New Zealand

George Tecuci, George Mason University, USA

Richard Weber, Management Intelligent Technologies, GmbH, Germany

Sholom Weiss, Rutgers University, USA

Bradley Whitehall, United Technologies Research Center, USA

Gerhard Widmer, Austrian Research Institute for Artificial Intelligence, Austria

Janusz Wnek, George Mason University, USA

H.-J. Zimmermann, RWTH Aachen, Germany

Information for Authors

General

The journal of Intelligent Data Analysis invites submission of research and application papers within the aims and scope of the journal. In particular, we prefer papers that discuss development of new AI architectures, methodologies, and techniques and their applications to the field of data analysis.

Manuscript

The manuscript should be in the following format. The first page of the paper should contain the title (preferably less than 10 words), the name(s), address(es), affiliation(s) and e-mail(s) of the author(s). The first page should also contain an abstract of 200-300 words, followed by 3-5 keywords.

Submission

To speed up the production process, authors should submit the text of original papers in PostScript (compressed file), to the Editor-in-Chief (address below). Any graphical or tabular files should be sent in separate files in Encapsulated PostScript or GIF format. The corresponding author will receive an acknowledgement, by e-mail.

The standard format (Times Roman) is preferred. The Manuscript should not exceed 35-40 pages of text (or the compressed/uuencoded PostScript file should not be more than 1.0 Meg).

References

All references in the paper should be listed in alphabetical order under the first author's name and numbered consecutively by arabic numbers. The structure of the references should be in the following format:

(a) Example of journal papers: R.A. Brooks, Intelligence without Representation, Artificial Intelligence, 47 (1) (1991), 139-159.

(b) Example of monographs: A. Basilevsky, Applied Matrix Algebra in the Statistical Sciences, North-Holland, Amsterdam, (1983).

(c) Example of edited volume papers: J. Pan and J. Tenenbaum, An Intelligent Agent Framework for Enterprise Integration, in: A. Famili, D. Nau and S. Kim, eds., Artificial Intelligence Applications in Manufacturing, MIT Press, Cambridge, MA, (1992), 349-383.

(d) Example of conference proceedings papers: R. Sutton, Planning by Incremental Dynamic Programming, in: Proceedings of the 8th International Machine Learning Workshop, Evanston, IL, USA, Morgan Kaufmann, (1991), 353-357.

(e) Example of unpublished papers: C. H. Watkins, Learning from Delayed Rewards, Ph.D. Thesis, Cambridge University, Cambridge, England, (1989).

The Review Process

Each paper will be reviewed by at least two reviewers. The authors will receive the results of the review process through e-mail. The authors of conditionally accepted papers are expected to revise their papers within 2-3 months.

Proofreading

Authors will be responsible for proofreading. Final copies of papers will be made available to the author and should be verified by the author within three working days. No new material may be inserted in the text at the time of proofreading.

Final Manuscript

When paper accepted, the publisher requires an electronic copy of the paper in one of the following formats, along with the originals of figures and tables.

Papers can be submitted in any one of the following formats:

- FrameMaker,
- WordPerfect,
- MicroSoft Word or
- Postscript.

Graphical files must be submitted separately, in either PostScript or GIF formats. A paper copy original is also required for any graphical material.

Journal of Intelligent Data Analysis will be a fully electronic, refereed quarterly journal. It will contain a number of innovative features not available in comparable print publications. These features include:

- An alerting service notifying subscribers of new papers in the journal,
- Links to large data collections, including the U.C. Irvine Machine Learning Repository Database,
- Links to secondary collection of data related to material presented in the journal,
- The ability to test new search mechanisms on the collection of journal articles,
- Links to related bibliographic material.

If you are interested in submitting a paper to the Intelligent Data Analysis journal, please contact:

A. Famili, Ph.D. Phone: (613) 993-8554
 Editor-in-Chief, Intelligent Data Analysis Fax:
 : (613) 952-7151 Senior Research Scientist

email: famili@ai.iit.nrc.ca Institute for Information Technology <http://ai.iit.nrc.ca/> fazel National Research Council of Canada, Bldg. M-50, Montreal Rd. Ottawa, Ont. K1A 0R6 Canada

If you are interested in receiving further announcements or subscription information about the upcoming journal, Intelligent Data Analysis, please send e-mail to:

Heather D. Joseph
h.dalterio@elsevier.com

Correction and Comment

AI in Eastern and Central Europe

(Informatica Vol. 20, 223-229)

Among the mentioned AI systems developed in Slovenia, the system for constructing equations from data is called GoldHorn (not Golding as given in the paper).

As stated in the paper, only major research organisations in AI were included in the review, so in no case the review was meant to be exhaustive. It may be useful to add that in Slovenia, in addition to J. Stefan Institute and University of Ljubljana, AI-related work and applications are also carried out at the Institute of Chemistry in Ljubljana and at the University of Maribor.

Ivan Bratko and Matjaz Gams

ERK'97
Electrotechnical and Computer Science Conference
Elektrotehniška in računalniška konferenca
September 25–27, 1997

Conference Chairman

Baldomir Zajc

University of Ljubljana
Faculty of Electrical Engineering
Tržaška 25, 1001 Ljubljana, Slovenia
Tel: (061) 1768 349, Fax: (061) 1264 630
E-mail: baldomir.zajc@fe.uni-lj.si

Conference Vice-chairman

Jurij Tasič

University of Ljubljana
Faculty of Electrical Engineering
Tržaška 25, 1001 Ljubljana, Slovenia
Tel: (061) 1768 440, Fax: (061) 1264 630
E-mail: jure.tasic@fe.uni-lj.si

Program Committee Chairman

Saša Divjak

University of Ljubljana
Faculty of Comput. and Inform. Science
Tržaška 25, 1001 Ljubljana, Slovenia
Tel: (061) 1768 260, Fax: (061) 1264 647
E-mail: sasa.divjak@fri.uni-lj.si

Programme Committee

Tadej Bajd

Gerry Cain

Saša Divjak

Janko Drnovšek

Matjaž Gams

Ferdo Gubina

Marko Jagodič

Jadran Lenarčič

Drago Matko

Miro Milanovič

Andrej Novak

Nikola Pavešič

Franjo Pernuš

Borut Zupančič

Publications Chairman

Franc Solina

University of Ljubljana
Faculty of Comput. and Inform. Science
Tržaška 25, 1001 Ljubljana, Slovenia
Tel: (061) 1768 389, Fax: (061) 1264 647
E-mail: franc@fri.uni-lj.si

Advisory Board

Rudi Bric, Dali Djonlagić,

Karel Jezernik, Peter Jereb,

Marjan Plaper, Jernej Virant,

Lojze Vodovnik

Call for Papers

for the sixth **Electrotechnical and Computer Science Conference ERK'97**, which will be held on 25–27 September 1997 in Portorož, Slovenia.

The following areas will be represented at the conference:

- *electronics,*
- *telecommunications,*
- *automatic control,*
- *simulation and modeling,*
- *robotics,*
- *computer and information science,*
- *artificial intelligence,*
- *pattern recognition,*
- *biomedical engineering,*
- *power engineering,*
- *measurements,*
- ...

The conference is organized by the **IEEE Slovenia Section** together with the Slovenian Electrotechnical Society and other Slovenian professional societies:

- Slovenian Society for Automatic Control,
- Slovenian Measurement Society (ISEMEC '97),
- SLOKO–CIGRE,
- Slovenian Society for Medical and Biological Engineering,
- Slovenian Society for Robotics,
- Slovenian Artificial Intelligence Society,
- Slovenian Pattern Recognition Society,
- Slovenian Society for Simulation and Modeling.

Authors who wish to present a paper at the conference should send two copies of their final camera-ready paper to as. Dr. Andrej Žemva to Faculty of Electrical Engineering, Tržaška 25, 1001 Ljubljana. The paper should be max. four pages long. More information on <http://www.ieee.si/erk97/>

Time schedule: Camera-ready paper due: *July 22, 1997*
Notification of acceptance: *End of August, 1997*

THE MINISTRY OF SCIENCE AND TECHNOLOGY OF THE REPUBLIC OF SLOVENIA

Address: Slovenska 50, 1000 Ljubljana, Tel.: +386 61 1311 107, Fax: +386 61 1324 140.
WWW: <http://www.mzt.si>
Minister: **Lojze Marinček, Ph.D.**

The Ministry also includes:

The Standards and Metrology Institute of the Republic of Slovenia

Address: Kotnikova 6, 61000 Ljubljana, Tel.: +386 61 1312 322, Fax: +386 61 314 882.

Slovenian Intellectual Property Office

Address: Kotnikova 6, 61000 Ljubljana, Tel.: +386 61 1312 322, Fax: +386 61 318 983.

Office of the Slovenian National Commission for UNESCO

Address: Slovenska 50, 1000 Ljubljana, Tel.: +386 61 1311 107, Fax: +386 61 302 951.

Scientific, Research and Development Potential:

The Ministry of Science and Technology is responsible for the R&D policy in Slovenia, and for controlling the government R&D budget in compliance with the National Research Program and Law on Research Activities in Slovenia. The Ministry finances or co-finance research projects through public bidding, while it directly finance some fixed cost of the national research institutes.

According to the statistics, based on OECD (Frascati) standards, national expenditures on R&D raised from 1,6 % of GDP in 1994 to 1,71 % in 1995. Table 2 shows an income of R&D organisation in million USD.

Objectives of R&D policy in Slovenia:

- maintaining the high level and quality of scientific technological research activities;

	Basic Research		Applied Research		Exp. Devel.		Total	
	1994	1995	1994	1995	1994	1995	1994	1995
Business Enterprises	6,6	9,7	48,8	62,4	45,8	49,6	101,3	121,7
Government Institutes	22,4	18,6	13,7	14,3	9,9	6,7	46,1	39,6
Private non-profit Organisations	0,3	0,7	0,9	0,8	0,2	0,2	1,4	1,7
Higher Education	17,4	24,4	13,7	17,4	8,0	5,7	39,1	47,5
TOTAL	46,9	53,4	77,1	94,9	63,9	62,2	187,9	210,5

Table 3: Incomes of R&D organisations by sectors in 1995 (in million USD)

Total investments in R&D (% of GDP)	1,71
Number of R&D Organisations	297
Total number of employees in R&D	12.416
Number of researchers	6.094
Number of Ph.D.	2.155
Number of M.Sc.	1.527

Table 1: Some R&D indicators for 1995

	Ph.D.			M.Sc.		
	1993	1994	1995	1993	1994	1995
Bus. Ent.	51	93	102	196	327	330
Gov. Inst.	482	574	568	395	471	463
Priv. np Org.	10	14	24	12	25	23
High. Edu.	1022	1307	1461	426	772	711
TOTAL	1565	1988	2155	1029	1595	1527

Table 2: Number of employees with Ph.D. and M.Sc.

- stimulation and support to collaboration between research organisations and business, public, and other sectors;
- stimulating and supporting of scientific and research disciplines that are relevant to Slovenian national authenticity;
- co-financing and tax exemption to enterprises engaged in technical development and other applied research projects;
- support to human resources development with emphasis on young researchers; involvement in international research and development projects;
- transfer of knowledge, technology and research achievements into all spheres of Slovenian society.

Table source: Slovene Statistical Office.

JOŽEF STEFAN INSTITUTE

Jožef Stefan (1835-1893) was one of the most prominent physicists of the 19th century. Born to Slovene parents, he obtained his Ph.D. at Vienna University, where he was later Director of the Physics Institute, Vice-President of the Vienna Academy of Sciences and a member of several scientific institutions in Europe. Stefan explored many areas in hydrodynamics, optics, acoustics, electricity, magnetism and the kinetic theory of gases. Among other things, he originated the law that the total radiation from a black body is proportional to the 4th power of its absolute temperature, known as the Stefan-Boltzmann law.

The Jožef Stefan Institute (JSI) is the leading independent scientific research institution in Slovenia, covering a broad spectrum of fundamental and applied research in the fields of physics, chemistry and biochemistry, electronics and information science, nuclear science technology, energy research and environmental science.

The Jožef Stefan Institute (JSI) is a research organisation for pure and applied research in the natural sciences and technology. Both are closely interconnected in research departments composed of different task teams. Emphasis in basic research is given to the development and education of young scientists, while applied research and development serve for the transfer of advanced knowledge, contributing to the development of the national economy and society in general.

At present the Institute, with a total of about 700 staff, has 500 researchers, about 250 of whom are post-graduates, over 200 of whom have doctorates (Ph.D.), and around 150 of whom have permanent professorships or temporary teaching assignments at the Universities.

In view of its activities and status, the JSI plays the role of a national institute, complementing the role of the universities and bridging the gap between basic science and applications.

Research at the JSI includes the following major fields: physics; chemistry; electronics, informatics and computer sciences; biochemistry; ecology; reactor technology; applied mathematics. Most of the activities are more or less closely connected to information sciences, in particular computer sciences, artificial intelligence, language and speech technologies, computer-aided design, computer architectures, biocybernetics and robotics, computer automation and control, professional electronics, digital communications

and networks, and applied mathematics.

The Institute is located in Ljubljana, the capital of the independent state of Slovenia (or S^onia). The capital today is considered a crossroad between East, West and Mediterranean Europe, offering excellent productive capabilities and solid business opportunities, with strong international connections. Ljubljana is connected to important centers such as Prague, Budapest, Vienna, Zagreb, Milan, Rome, Monaco, Nice, Bern and Munich, all within a radius of 600 km.

In the last year on the site of the Jožef Stefan Institute, the Technology park "Ljubljana" has been proposed as part of the national strategy for technological development to foster synergies between research and industry, to promote joint ventures between university bodies, research institutes and innovative industry, to act as an incubator for high-tech initiatives and to accelerate the development cycle of innovative products.

At the present time, part of the Institute is being reorganized into several high-tech units supported by and connected within the Technology park at the Jožef Stefan Institute, established as the beginning of a regional Technology park "Ljubljana". The project is being developed at a particularly historical moment, characterized by the process of state reorganisation, privatisation and private initiative. The national Technology Park will take the form of a shareholding company and will host an independent venture-capital institution.

The promoters and operational entities of the project are the Republic of Slovenia, Ministry of Science and Technology and the Jožef Stefan Institute. The framework of the operation also includes the University of Ljubljana, the National Institute of Chemistry, the Institute for Electronics and Vacuum Technology and the Institute for Materials and Construction Research among others. In addition, the project is supported by the Ministry of Economic Relations and Development, the National Chamber of Economy and the City of Ljubljana.

Jožef Stefan Institute
Jamova 39, 61000 Ljubljana, Slovenia
Tel.:+386 61 1773 900, Fax.:+386 61 219 385
Tlx.:31 296 JOSTIN SI
WWW: <http://www.ijs.si>
E-mail: matjaz.gams@ijs.si
Contact person for the Park: Iztok Lesjak, M.Sc.
Public relations: Natalija Polenec

INFORMATICA

AN INTERNATIONAL JOURNAL OF COMPUTING AND INFORMATICS

INVITATION, COOPERATION

Submissions and Refereeing

Please submit three copies of the manuscript with good copies of the figures and photographs to one of the editors from the Editorial Board or to the Contact Person. At least two referees outside the author's country will examine it, and they are invited to make as many remarks as possible directly on the manuscript, from typing errors to global philosophical disagreements. The chosen editor will send the author copies with remarks. If the paper is accepted, the editor will also send copies to the Contact Person. The Executive Board will inform the author that the paper has been accepted, in which case it will be published within one year of receipt of e-mails with the text in Informatica L^AT_EX format and figures in .eps format. The original figures can also be sent on separate sheets. Style and examples of papers can be obtained by e-mail from the Contact Person or from FTP or WWW (see the last page of Informatica).

Opinions, news, calls for conferences, calls for papers, etc. should be sent directly to the Contact Person.

QUESTIONNAIRE

- Send Informatica free of charge
- Yes, we subscribe

Please, complete the order form and send it to Dr. Rudi Murn, Informatica, Institut Jožef Stefan, Jamova 39, 61111 Ljubljana, Slovenia.

Since 1977, Informatica has been a major Slovenian scientific journal of computing and informatics, including telecommunications, automation and other related areas. In its 16th year (more than five years ago) it became truly international, although it still remains connected to Central Europe. The basic aim of Informatica is to impose intellectual values (science, engineering) in a distributed organisation.

Informatica is a journal primarily covering the European computer science and informatics community - scientific and educational as well as technical, commercial and industrial. Its basic aim is to enhance communications between different European structures on the basis of equal rights and international refereeing. It publishes scientific papers accepted by at least two referees outside the author's country. In addition, it contains information about conferences, opinions, critical examinations of existing publications and news. Finally, major practical achievements and innovations in the computer and information industry are presented through commercial publications as well as through independent evaluations.

Editing and refereeing are distributed. Each editor can conduct the refereeing process by appointing two new referees or referees from the Board of Referees or Editorial Board. Referees should not be from the author's country. If new referees are appointed, their names will appear in the Refereeing Board.

Informatica is free of charge for major scientific, educational and governmental institutions. Others should subscribe (see the last page of Informatica).

ORDER FORM – INFORMATICA

Name:	Office Address and Telephone (optional):
Title and Profession (optional):
.....	E-mail Address (optional):
Home Address and Telephone (optional):
.....	Signature and Date:

Referees:

Witold Abramowicz, David Abramson, Kenneth Aizawa, Alan Aliu, John Anderson, Catriel Beeri, Fevzi Belli, Istvan Berkeley, Azer Bestavros, Balaji Bharadwaj, Jacek Blazewicz, Laszlo Boeszoermenyi, Ivan Bratko, Jerzy Brzezinski, Marian Bubak, Leslie Burkholder, Frada Burstein, Wojciech Buszkowski, Ryszard Choras, Jason Ceddia, Wojciech Chybowski, Andrzej Ciepielewski, Vic Ciesielski, David Cliff, Travis Craig, Tadeusz Czachorski, Milan Češka, Pavol Ďuriš, Sait Dogru, Georg Dorfner, Matija Drobnič, Maciej Drozdowski, Marek Druzdzel, Hesham El-Rewini, Pierre Flener, Terrence Forgarty, Hugo de Garis, Eugeniusz Gatnar, James Geller, Michael Georgiopolus, Janusz Gorski, Georg Gottlob, David Green, Herbert Groiss, Inman Harvey, Elke Hochmueller, Rod Howell, Tomáš Hruška, Ryszard Jakubowski, Piotr Jędrzejowicz, Eric Johnson, Li-Shan Kang, Roland Kaschek, Jan Kniat, Stavros Kokkotos, Kevin Korb, Gilad Koren, Henryk Krawczyk, Ben Kroese, Zbyszko Krolikowski, Benjamin Kuipers, Matjaž Kukar, Aarre Laakso, Phil Laplante, Bud Lawson, Ulrike Leopold-Wildburger, Joseph Y-T. Leung, Raymond Lister, Doug Locke, Matija Lokar, Jason Lowder, Andrzej Marciniak, Witold Marciszewski, Vladimir Marik, Jacek Martinek, Tomasz Maruszewski, Florian Matthes, Timothy Menzies, Dieter Merkl, Zbigniew Michalewicz, Roland Mittermeir, Madhav Moganti, Tadeusz Morzy, Daniel Mossé, John Mueller, Hari Narayanan, Jaroslav Nieplocha, Jerzy Nogieć, Stefano Nolfi, Tadeusz Pankowski, Warren Persons, Stephen Pike, Niki Pissinou, Gustav Pomberger, James Pomykalski, Gary Preckshot, Cveta Razdevšek Pučko, Ke Qiu, Michael Quinn, Gerald Quirchmayer, Luc de Raedt, Ewaryst Rafajłowicz, Wolf Rauch, Peter Rechenberg, Felix Redmill, David Robertson, Marko Robnik, Ingrid Russel, A.S.M. Sajeev, Bo Sanden, Iztok Sarnik, Wolfgang Schreiner, Guenter Schmidt, Heinz Schmidt, William Spears, Hartmut Stadtler, Przemysław Stpicznyński, Andrej Stritar, Maciej Stroinski, Tomasz Szmuc, Jiří Šlechta, Zahir Tari, Jurij Tasič, Piotr Teczynski, Ken Tindell, A Min Tjoa, Wiesław Traczyk, Marek Tudruj, Andrzej Urbanski, Kanonkluk Vanapipat, Alexander P. Vazhenin, Zyunt Vetulani, Olivier de Vel, John Weckert, Gerhard Widmer, Stefan Wrobel, Janusz Zalewski, Yanchun Zhang

EDITORIAL BOARDS, PUBLISHING COUNCIL

Informatika is a journal primarily covering the European computer science and informatics community; scientific and educational as well as technical, commercial and industrial. Its basic aim is to enhance communications between different European structures on the basis of equal rights and international refereeing. It publishes scientific papers accepted by at least two referees outside the author's country. In addition, it contains information about conferences, opinions, critical examinations of existing publications and news. Finally, major practical achievements and innovations in the computer and information industry are presented through commercial publications as well as through independent evaluations.

Editing and refereeing are distributed. Each editor from the Editorial Board can conduct the refereeing process by appointing two new referees or referees from the Board of Referees or Editorial Board. Referees should not be from the author's country. If new referees are appointed, their names will appear in the Refereeing Board. Each paper bears the name of the editor who appointed the referees. Each editor can propose new members for the Editorial Board or Board of Referees. Editors and referees inactive for a longer period can be automatically replaced. Changes in the Editorial Board and Board of Referees are confirmed by the Executive Editors.

The coordination necessary is made through the Executive Editors who examine the reviews, sort the accepted articles and maintain appropriate international distribution. The Executive Board is appointed by the Society Informatika. Informatika is partially supported by the Slovenian Ministry of Science and Technology.

Each author is guaranteed to receive the reviews of his article. When accepted, publication in Informatika is guaranteed in less than one year after the Executive Editors receive the corrected version of the article.

Executive Editor – Editor in Chief

Anton P. Železnikar
Volaričeva 8, Ljubljana, Slovenia
E-mail: anton.p.zeleznikar@ijs.si

Executive Associate Editor (Contact Person)

Matjaz Gams, Jožef Stefan Institute
Jamova 39, 61000 Ljubljana, Slovenia
Phone: +386 61 1773 900, Fax: +386 61 219 385
E-mail: matjaz.gams@ijs.si
WWW: <http://www2.ijs.si/~mezi/matjaz.html>

Executive Associate Editor (Technical Editor)

Rudi Murn, Jožef Stefan Institute

Publishing Council: Tomaž Banovec,
Ciril Baškovič, Andrej Jerman-Blažič,
Joško Čuk, Jernej Virant

Board of Advisors:

Ivan Bratko, Marko Jagodič,
Tomaž Pisanski, Stanko Strmčnik

Editorial Board

Suad Alagić (Bosnia and Herzegovina)
Shuo Bai (China)
Vladimir Bajić (Republic of South Africa)
Vladimir Batagelj (Slovenia)
Francesco Bergadano (Italy)
León Birnbaum (Romania)
Marco Botta (Italy)
Pavel Brazdil (Portugal)
Andrej Brodnik (Slovenia)
Ivan Bruha (Canada)
Hubert L. Dreyfus (USA)
Jozo Dujmović (USA)
Johann Eder (Austria)
Vladimir Fomichov (Russia)
Janez Grad (Slovenia)
Francis Heylighen (Belgium)
Hiroaki Kitano (Japan)
Igor Kononenko (Slovenia)
Miroslav Kubat (Austria)
Ante Lauc (Croatia)
Jean-Pierre Laurent (France)
Jadran Lenarčič (Slovenia)
Svetozar D. Margenov (Bulgaria)
Magoroh Maruyama (Japan)
Angelo Montanari (Italy)
Igor Mozetič (Austria)
Stephen Muggleton (UK)
Pavol Návrat (Slovakia)
Jerzy R. Nawrocki (Poland)
Marcin Paprzycki (USA)
Oliver Popov (Macedonia)
Luc De Raedt (Belgium)
Dejan Raković (Yugoslavia)
Jean Ramaekers (Belgium)
Paranandi Rao (India)
Wilhelm Rossak (USA)
Claude Sammut (Australia)
Walter Schempp (Germany)
Johannes Schwinn (Germany)
Branko Souček (Italy)
Oliviero Stock (Italy)
Petra Stoerig (Germany)
Jiří Šlechta (UK)
Gheorghe Tecuci (USA)
Robert Trappl (Austria)
Terry Winograd (USA)
Claes Wohlin (Sweden)
Stefan Wrobel (Germany)
Xindong Wu (Australia)

Informatica

An International Journal of Computing and Informatics

Contents:

Informational Supervenience	A.P. Železnikar	1
<hr/>		
A Situational Informatical Dynamics: The Case of Situation-contextual and Time-contextual Non-additive Influences	M. Maruyama	5
The Definition of an Integrated Software Life-Cycle Tool	R. Leonard	19
Qualitative Reasoning and a Circular Information Processing Algebra	F. Jović	31
Hierarchical Classification as an Aid to Browsing	J.R. Rose C.M. Eastman	49
Towards Recursive Models—A Computational Formalism for the Semantics of Temporal Presuppositions and Counterfactuals in Natural Language	S. Mizzaro	59
Informational Graphs	A.P. Železnikar	79
A Study in Generating Readable Modula-2 from Prolog	D. Resler D. Crookes	115
Design of Approximate Identity Neural Networks by using of Picewise-Linear Circuits	J. Kaderka	129
An Integrated Testing Framework for Object-Oriented Programs	S.-S. Liao K.H. Chang C.-Y. Chen	135
<hr/>		
Reports and Announcements		147