Volume 22 Number 4 December 1998 ISSN 0350-5596

Informatica

An International Journal of Computing and Informatics

Special Issue:
NLP & Multi-Agent Systems

The Slovene Society Informatika, Ljubljana, Slovenia

Informatica 22 (1998) Number 4, pp. 405–518

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 1998 (Volume 22) 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, Kranj, 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; Cognitive Science Society (Matjaž Gams)
Slovenian Society of Mathematicians, Physicists and Astronomers (Bojan Mohar)
Automatic Control Society of Slovenia (Borut Zupančič)
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, Computer Literature Index, Cur. Cont. & Comp. & Math. Sear., Current Mathematical Publications, Engineering Index, INSPEC, Mathematical Reviews, MathSci, 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 Ljubljana. Slovenia taxe Percue.

Intersecting Interests of Natural Language Processing and Multi-Agent Systems: Introduction to the Special Issue

Both Natural Language Processing (NLP) and Multi-Agent Systems (MASs) are quickly progressing branches of Computer Science. In the nineties, one has been able to observe the following main trends in the field of NLP. *Firstly*, the realization of the projects aimed at computer understanding of oral speech or at synthesis of oral speech. *Secondly*, the permanent growth of attention to the conceptual processing of real complicated discourses pertaining to law, medicine, business, technology, etc. This shift of accents in NLP to the analysis of real discourses has been possible due to the stormy progress of hardware and due to the experience accumulated in computational linguistics; an essential cause of this shift has been a huge amount of texts available on-line in Internet.

Both the first and the second trends imply the increase of interest to computational lexical semantics; i.e., to explicit representing and further use of semantic-syntactic information associated with lexical items. As for the first trend, the reason for such an interest is that, very often, the fragments of speech may be interpreted in different ways. Besides, it may be impossible to perceive some words or word combinations due to various noises. In such situations, each word which is perceived well may become a starting point for interpreting the surrounding words and finding the meaning of the utterance as a whole. Hence the role of lexical semantics in the understanding of oral speech is very high.

Its role in the understanding of written discourses is determined, in particular, by the following factors: (a) the existence of many meanings associated with one lexical items; (b) numerous references in discourses to the previously mentioned entities and to the meanings of phrases and larger parts of texts; (c) non-standard use of the words, metaphoric character of many fragments of discourses.

The third trend is the use of NL-analysers or NL-generators in combination with the other means of conveying information in multimedia systems.

The fourth trend is the growth of attention to the development and application of formal methods to the design of NL-processing systems. The *central idea* here is to build and use formal systems with the expressive possibilities being close to the possibilities of NL in order to construct underspecified and completely specified semantic representations (SRs) of NL-texts and to describe pieces of knowledge about the world. The systems of the kind are called NL-like formal systems of semantic and knowledge representations.

It appears that this idea (most important, together with formal means for its realization given by the theory of S-calcules and S-languages) was published in English in a widely distributed edition for the first time in [3]. However, this work outstripped the time of its publication and practically was not noticed by the artificial community in West. In the nineties, this idea has received a large popularity, especially due to the project Core Language Engine [1] and to the series of publications on Episodic Logic including [6 - 8]. One may assume that important prospects for computational semantics may be opened by the interchange of ideas between the studies on NL-like formal systems of SRs and such a part of logic programming which investigates the questions of processing NL.

Multi-Agent Systems have a much shorter history than the field of NLP. However, the progress of the first field in the recent years has been really overwhelming. The principal reason for this progress is as follows. One forecasts a very rapid development in the nearest future of electronic commerce, or E-commerce, based on largest possibilities of Internet [9], and MASs are considered as a key enabling technology for electronic commerce systems [5].

Analysing the state-of-affairs in NLP and MASs, it is possible to distinguish three points where the interests of NLP and MASs are intersecting:

1. The background of formal and computational modeling of NL-communication includes the theory of speech acts; it was born in the 1960s in the works of J.L. Austin, P.F. Strawson, and J.R. Searle.

   But speech acts theory gave an initial collection of ideas for developing in the 1990s agent communication languages (ACLs). The essence of the problem is as follows. Numerous intelligent agents (constructed by different centres, using different hardware and software) can effectively interact while solving various tasks only on condition that they possess a common communication language and follow some generally accepted rules of communication. The main idea of ACLs may be metaphorically described as forwarding each message in a special envelope where the intention of the message's sender is explicitly indicated.

   Quite recently, the international Foundation for Intelligent Physical Agents (FIPA) was created and registered in Geneva. FIPA is a non-profit association and includes 35 corporate members representing 12 countries from all over the world (according to the data of 1997). In October of 1997, FIPA suggested a standard for ACLs called below FIPA ACL [2]. This language includes 20 special

informational items called performatives and designating various communicative acts ("confirm", "propose", "agree", "refuse", etc.).

2. A very considerable part of projects in the field of MASs is motivated by the opportunities afforded by Internet: that applies also to Electronic Commerce. Internet provides a lot of textual information on-line, and this information must be analysed by NL-processing systems.

3. There is the opinion that the technology of MASs opens large prospects for constructing the computer systems realizing a NL-dialogue with artificial intelligent agents or with casual users or NL-processing systems dealing with real discourses pertaining to such domains as law, medicine, business, technology, etc.

The destination of this special issue is, in particular, *to establish a new connection* between NLP and MASs and, as a result, to give a new, powerful and flexible theoretical instrument for constructing ACLs. The starting idea is very simple. NL-texts include various forms of the verbs "to inform", "to confirm", "to order", etc. Suppose that we have a mathematical theory enabling us to effectively describe structured meanings of arbitrary NL-texts or of a very large spectrum of NL-texts including such verbs. Then a theory of the kind may be interpreted as a universal or widely-applicable formal tool for designing ACLs. The role of such a theory is played by the theory of restricted K-calcules and K-languages published in "Informatica" in 1996, No. 1.

This special issue includes seven papers selected as a result of rigorous reviewing. They reflect important tendencies in the development of NLP and MASs, and one of them establishes a new connection between these two fields of Computer Science.

The first group of papers is prepared by the authors from France, U.S.A., Canada, and United Kingdom and pertains to the field of NLP. P. Saint-Dizier (France) investigates the questions of computational lexical semantics, paying a particular attention to a conceptual analysis of the polysemic behaviour of adjectives and verbs and to the study of metonymical sentences.

A. Linninger and G. Stephanopoulos (U.S.A.) demonstrate the practical advantages of using a professional natural language by chemists for describing the recipes of batch operating procedures. A precious feature of this work is that the readers can get a clear understanding of the role of a NL approach in the complete computer-aided process for the production of pharmaceuticals.

The article of V. Dahl, P. Tarau, P. Accuosto, S. Rochefort, and M. Scurtescu (Canada) describes a new kind of logic grammars - assumption grammars - and shows their usefulness for studying a number of tasks associated with the design of knowledge bases (in particular, being components of NL-processing systems). R. Sahandi, D.S.G. Vine, and J. Longster (UK) give a substantial review of speech synthesis and facial animation techniques and describe a text-driven visual speech synthesis system designed at the Bournemouth University. In this system, a 2D cartoon face is synchronised with synthetic speech.

The paper of V.A. Fomichov (Russia) establishes a new connection between the fields of NLP and MASs. On the one hand, it gives the specialists on MASs a comprehensive and flexible mathematical framework for constructing ACLs. To this end, a new angle of look at the theory of restricted K-calcules and K-languages, or the RKCL-theory, [4] is set forth. It is shown that the RKCL-theory allows us both to describe contents of messages and to represent communicative acts. The main advantage of the suggested approach in comparison with FIPA ACL is the possibility to build arbitrary complicated goals (or to indicate complicated actions) and to represent contents of messages corresponding to arbitrary NL-texts.

On the other hand, the designers of NL-processing systems may consider this article as an introduction to a theory opening a lot of new prospects for formalizing arbitrary aspects of computational semantics. It may be noted that the expressive possibilities of restricted standard K-languages considerably exceed the possibilities of the formulas regarded in Episodic Logic, Discourse Representation Theory, Montague Grammars, Theory of Conceptual Graphs, and Theory of Semantic Structures suggested by R.S. Jackendoff.

The last group consists of two papers submitted by the authors from Switzerland and Italy, respectively. M. Magnanelli, A. Erni, and M. Norrie describe an intelligent agent extracting information from Web documents. It is the information needed for academic contacts: various addresses and affiliations of the researchers and the data about their publications and projects.

P. Baroni, D. Fogli, G. Guida, and S. Mussi suggest a new architecture of MASs; it is based on the so called active mental entities. The authors illustrate the stated ideas on the example concerning the behaviour of a department mail delivery robot. A prototype programming environment for the development of the suggested agent and multi-agent architecture was developed with the help of the language C++. It appears that the suggested architecture of MASs opens precious perspectives for constructing not only autonomous robots but NL-processing systems too.

On technical reasons, the eighth paper which was accepted for this special issue will be published only in the next volume. It is a paper written by V. Dahl, P. Tarau, S. Rochefort, and M. Scurtescu (Canada)

and devoted to designing a Spanish Interface to the Netscape subsystem of Internet. This interface is constructed under the framework of assumption grammars - a new kind of logic grammars described in this issue.

Acknowledgements

We are grateful both to the researchers submitted papers for this special issue and to the referees for their helpful remarks and prompt communication.

References

[1] ALSHAWI, H., Ed. 1992). The Core Language Engine. A Bradford Book. The MIT Press. Cambridge MA, London.

[2] FIPA 1997. The Foundation for Intelligent Physical Agents. FIPA'97 Specification. Part 2—Agent Communication Language. The text refers to the specification dated 10th October 1997. Geneva.

[3] FOMITCHOV, V.A. 1984. Formal systems for natural language man-machine interaction modeling. *In* Artificial Intelligence. Proc. of the IFAC Symposium. Leningrad, USSR, 4–6 October 1983. V.M. Ponomaryov, Ed. IFAC Proc. Series. 1984. No. 9:203–207. Oxford, UK. New York, etc. Pergamon Press.

[4] FOMICHOV, V.A. 1996. A mathematical model for describing structured items of conceptual level. Informatica 20:5–32.

[5] GUILFOYLE, C., J. JEFFCOATE & H. STARK. 1997. Agents on the Web: Catalyst for E-Commerce. Ovum Ltd. London.

[6] HWANG, C. H. & L.K. SCHUBERT. 1993a. Meeting the interlocking needs of LF-computation, deindexing, and inference: An organic approach to general NLU. Proc. 13th Int. Joint Conf. on AI (IJCAI'93). Chambery, France. 28 Aug.–3 Sept. 1993.

[7] HWANG, C.H. & L.K. SCHUBERT. 1993b. Episodic Logic: a comprehensive, natural representation for language understanding. Minds and Machines 3:381–419.

[8] SCHUBERT, L. K. & C.H. HWANG. 1989. An episodic knowledge representation for narrative texts. Proc. 1st Int. Conf. on Principles of Knowledge Repres. and Reasoning (KR'89) 444-458. Toronto, Canada.

[9] THOME, R., & H. SCHINZER. 1998. Market survey of electronic commerce. Informatica 22:11–19.

By Vladimir A. Fomichov and Anton P. Železnikar

On the Polymorphic Behavior of Word-Senses

Patrick Saint-Dizier
IRIT-CNRS,
118, route de Narbonne
31062 Toulouse Cedex France
stdizier@irit.fr

Keywords: Lexical semantics, meaning variations

Edited by: Anton P. Železnikar
Received: June 2, 1998 **Revised:** October 21, 1998 **Accepted:** October 30, 1998

In this paper, we firstly outline some elements related to sense variation and to sense delimitation within the perspective of the Generative Lexicon. We then develop the case of adjectival modification and several forms of selective binding and metonymies for verbs and show that, in some cases, the Qualia structure can be combined with or replaced by a small number of rules, which seem to capture more adequately the relationships between the predicator and one of its arguments. We focus on the Telic role of the Qualia structure which seems to be the most productive role to model sense variations.

1 Introduction

Semantics is probably one of the major corner stones of natural language processing. It is still a largely open field, compared to morphology, syntax and parsing, for example. A number of formalisms for representing meaning have been proposed around primitives, nets, graphs and frames. Some of the major difficulties of semantics are the links with syntax and morphology on the one hand, and the links with interpretation and pragmatics on the other hand. With respect to interpretation, one major problem is the taking into account of sense variations, e.g. metaphors, metonymies and many forms of slight variations which make understanding an utterance a delicate task. This paper presents some elements of a model to deal with sense variations within the Generative Lexicon perspective.

Investigations within the generative perspective aim at modelling, by means of a small number of rules, principles and constraints, linguistic phenomena (either morphological, syntactic or semantic) at a high level of abstraction, level which seems to be appropriate for research on multi-linguism and language learning. These works, among other things, attempt to model a certain form of 'creativity' in language: from a limited number of linguistic resources, a potentially infinite set of surface forms can be generated.

Among works within the generative perspective, let us concentrate on the Generative Lexicon (Pustejovsky 91, 95), which has settled in the past years one of the most innovative perspective in lexical semantics. This approach introduces an abstract model radically opposed to 'flat' sense enumeration lexicons. This approach, which is now well-known, is based (1) on the close cooperation of three lexical semantic structures: the argument structure (including selectional restrictions), the aspectual structure and the Qualia structure, (2) on a detailed type theory and a type coercion procedure and (3) on a refined theory of compositionality. The Generative Lexicon (GL) investigates the problem of the multiplicity of usages of a sense of a lexeme and shows how these usages can be analyzed in terms of possible type shiftings w.r.t. the type expected by a usage of that sense defined as the core usage. Type shifting is modelled by a specific inference mechanism: type coercion. The GL shows very clearly the inter-dependence between arguments and predicates.

Our strategy has here been to limit the complexity of semantic representation formalisms to the elements we want to show. It is clear, e.g. that the LCS is not sufficiently expressive for a number of verbs, the primitive GO been too general or vague. The extensions (e.g. ontologies or rules for metaphors) we propose are also minimal, but are sufficiently expressive w.r.t. our purpose.

In our perspective, we are not only interested in deciding whether an expression is an acceptable argument for a predicate and for what reasons, as it is also the case in the GL, but we want to be able to 'reconstruct' or to infer the meaning of the proposition from its parts (the predicate and its arguments), and possibly also from the implicit semantics conveyed by the syntactic form (Goldberg 94). We assume that the impossibility of building a semantic representation for a proposition entails that it is semantically ill-formed w.r.t. the grammar, lexicon and composition rules. This view is very common in formal semantics.

In this paper, the following points are addressed:

- The Qualia structure is a complex structure, quite difficult to describe, in spite of evidence of its existence, in particular the Telic role, (explored e.g. in the EuroWordNet project, the European WordNet). Qualias are well-designed and useful for nouns, but look more artificial for other lexical categories. We show that it is the telic role of nouns which is the most useful, and that the internal structure of this role can (1) be made more precise and its use more reliable and accurate by means of types (see also preliminary results of the EEC SIMPLE project (http at cnr at Pisa)) and (2) be partitioned by means of types into ontological domains for modelling some forms of metaphors. This restriction on the Telic role makes it easier to use and to specify Qualia structures since it is more restricted.

- Types are not sufficiently 'constrained' to account for the constraints holding, for each predicate, on the different sense/usage variations they may be subject to. We show that an underspecified Lexical Conceptual Structure (LCS) (Jackendoff 90), where conceptual variables are typed, is more appropriate because of its ability to represent the polymorphism of senses in the GL.

- Elements of the Qualia structure can be incorporated into semantic composition rules to make explicit the semantics of the combination predicate-argument, instead of developing lexical redundancy rules.

- We contrast a rule-based approach (also used by other authors such as (Copestake and Briscoe 95), (Ostler and Atkins 92), (Nunberg and Zaenen 79)) with the Qualia-based approach to deal with sense shiftings and in particular selective binding, metaphors (which the GL cannot resolve a priori) and metonymies. Rules seem to be more precise via the specification of constraints, whereas type shifting is a general, highly abstract operation. Rules can also be activated at different levels in the parsing process (at lexical insertion level or in semantic composition rules), whereas type coercion occurs only at a certain point, when parsing the item. Another view is presented in (Jackendoff 97) with the *principle of enriched composition*, which is in fact quite close to our view, but restricted to a few specific coercion situations (aspectual, mass-count, picture, begin-enjoy).

- The rules for type shifting we present here are not lexical rules, as in (Copestake and Briscoe 95), but they are part of the semantic composition system, which makes them more general.

To illustrate this study, we firstly survey the different senses and sense variations of one of the most polysemic French adjectives: *bon* (good), which covers most of the main sense variation situations that adjectives may undergo. Additional adjectives, often cited in the GL literature such as *rapide, triste* or *facile* (fast, sad and easy) behave similarly. In fact, we observed many behavior similarities within semantic families of adjectives (e.g. evaluative, locational, aspectual adjectives). Next, we present for verbs and VPs the case of selective binding and the treatment of some forms of metonymies related to the use and to actions on objects. In order to give more strength to our study, we focus on a few verbs while showing that the results may be extended to the verbs of the same semantic class.

2 The semantic environment

In this section, for the sake of readability, we briefly present the foundations of WordNet (and EuroWordNet, which is based on the same ideas, but is designed for a number of European languages) and of the Lexical Conceptual Structure (LCS).

2.1 An Overview of WordNet

WordNet is a large project in English, developed at Princeton University, which started in the early eightees at the initiative of G. Miller and C. Fellbaum (Miller and Fellbaum 91). WordNet is deliberately 'limited' to paradigmatic relations (although it introduces in its last version frame-based descriptions). Relations structure the different senses of a lexical item. The WordNet WWW interface is available at: *www.cogsci.princeton.edu.*

2.1.1 Main Organization

WordNet makes detailed and accurate sense distinctions for a given lexeme, based on usage, resulting sometimes in more than 20 senses for a verb (e.g. *give* has 27 senses). This reflects in a quite extensional way the complexity and the creativity of language. In WordNet, the different syntactic categories are studied independently, for technical and methodological reasons. The authors are of course aware of that sooner or later the relations between the different categories will have to be studied and developed.

Conventional dictionaries as well as most thesauri are built around words. In WordNet, the smallest unit is called a *synset*, which is a set that contains all the senses of different lexical items that express the same 'concept' or 'notion', e.g. [give, offer]. The notion of concept is of course not an absolute one and may receive different levels of granularity. However, in a synset, all the words, with the sense considered

(and often paraphrased by a natural language expression in the synset definition to make it explicit), behave identically in natural language utterances. If a word is polysemous, then it appears in several synsets. Consequently, distinctions and synsets are based essentially on usage, not (fully) on the lexicographer's intuition. Lexical semantics relations are established between synsets. About 17,000 verbs have been studied in WordNet, which is very large. About 11,500 synsets have been defined.

Verb semantic classifications have received little attention in the past, compared to the efforts devoted to nouns. However, a first partition of verbs into large 'families' (or semantic domains, also called "files" in WordNet, but this term is vague) is necessary to allow for a better and more 'modular' analysis of senses and relations between senses. A verb may appear in several families, if it is polysemous or if has different 'facets' that naturally fall into different classes.

The analysis of verbs by family is based on the observation that most, if not all, relations operate within quite precise semantic domains. In (Fellbaum 93, 98), verbs are first divided into states (this set is relatively small) and actions. This latter set is divided into 14 families, which may be quite large, but are rather consensual about researchers in cognitive science. These classes are the following: *motion, possession, perception, contact, communication, competition, change, cognition, consumption, creation, emotion (or psychological), bodily care and function, social behavior and interactions, and expression of weather.* These main classes are then further divided into subclasses, according to different dimensions, properties and domains (e.g. family, law, education etc.). It is clear that, for each level of classification, there are verbs which are more prototypical than others, around which others may aggregate. However, once again, in this domain much work has been carried out for nouns, but very little for verbs.

Families are headed by 'unique beginners', whose definition and justification is not an easy task, WordNet includes e.g.: *act, move, get, become, be, make.* To some extent, one may consider that these unique beginners may correspond to primitives, as those developed in the Lexical Conceptual Structure approach (Jackendoff 90). In that case, primitives could be combined with semantic domains (e.g. possession, space), corresponding to the fields of the Lexical Conceptual Structure.

A number of experiments have been carried out to explore the psychological validity of the relations used in WordNet. Results are good and they confirm on a larger scale the intuitions at the origin of WordNet. Interestingly enough, troponymy and opposition turned out to be the most frequently considered, indicating therefore a certain psychological salience of these relations. The results also suggest that there is a kind of 'homomorphism' between the verb organizations in human's mind and in WordNet.

2.1.2 A few limitations

WordNet and its more recent and still ongoing European equivalent EuroWordNet are two structured lexicographic and conceptuals ystems of much importance for processing language. The detailed organization of the system, the ontological classification about which there is a large consensus, and the base-type and multilingual system of EuroWodNet are extremely useful in a number of applications and as a basis for research on language.

Limitations would rather beon the way synsets are constructed and the strategy behind them. It is in fact quite difficult to have a clear strategy for sense delimitation for a given lexeme. WordNet makes sense distinctions which are very subtle, often base on usage. From our point of view, these distinctions are not really semantically motivated, but appear rather as slight sense variations or metaphors. metaphors are not clearly accounted for in WordNet.

2.2 The Lexical Conceptual Structure

Let us now introduce the Lexical Conceptual Structure (LCS), which is an elaborated form of semantic representation, with a strong cognitive dimension. The LCS came in part from the Lexical Semantics Templates (see above) and from a large number of observations such as those of (Gruber 67). The present form of the LCS, under which it gained its popularity, is due to Jackendoff (Jackendoff 83, 90). The LCS was designed within a linguistic and cognitive perspective, it has some similarities, but also major differences, with approaches closer to Artificial Intelligence such as semantic nets or conceptual graphs (which are very similar). The LCS is basically designed to represent the meaning of predicative elements and the semantics of propositions, it is therefore substantially different from frames and scripts, which describe situations in the world like going to a restaurant or been cured from a desease.

2.2.1 Main Principles and characteristics

The LCS is mainly organized around the notion of *motion*, other semantic/cognitive fields being derived from motion by analogy (such as change of possession, change of property). This analogy works perfectly in a number of cases, but turns out to be unnatural in a number of others. From that point of view, the LCS should be considered both as a semantic model providing a representational framework and a language of primitives on the one hand, and as a methodology on the other hand, allowing for the introduction of new

primitives to the language, whenever justified. Another important characteristics of the LCS is the close relations it has with syntax, allowing the implementation of a comprehensive system of semantic composition rules. From that point of view one often compares the LCS with a kind of X-bar semantics.

In the LCS, a single level of semantic representation is postulated: the conceptual structure, where all types of relevant information are represented.

2.2.2 The constituents of the LCS

Let us now introduce the different elements of the LCS language. They are mainly: conceptual categories, semantic fields and primitives. Other elements are conceptual variables, semantic features (similar to selectional restrictions, e.g. such as eatable entity, liquid), constants (representing non- decomposable concepts like e.g. money, butter) and lexical functions (which play minor roles).

A. Conceptual Categories

(Jackendoff 83) introduces the notion of conceptual constituent defined from a small set of ontological categories (also called conceptual parts of speech), among which the most important are: *thing, event, state, place, path, property, purpose, manner, amount, time.* These categories may subsume more specific ones, e.g. the category *thing* subsumes: *human, animal, object.* These categories may be viewed as the roots of a selectional restriction system.

The assignment of a conceptual category to a lexical item often depends on its context of utterance, for example the noun *meeting* is assigned the category *time* in:
after the meeting
while it is assigned the category *event* in:
the meeting will be held at noon in room B34.
There are constraints on the types of conceptual categories which can be assigned to a lexical item. For example, a color will never be assigned categories such as event or distance.

Conceptual categories are represented as an indice to a bracketed structure:
[<*semantic category*>]
where the contents of that structure has the type denoted by the semantic category. Here are a few syntactic realizations of conceptual categories:
[$_{thing}$ *Mozart*] *is* [$_{property}$ *famous*].
He composed [$_{amount}$ *many* [$_{thing}$ *symphonies*]].
[$_{event}$ *The meeting*] *starts at* [$_{time}$ *2 PM*].
Ann switched off the electricity [$_{purpose}$ *to prevent a fire*].

B. Conceptual primitives

The LCS is based on a small number of conceptual primitives. The main ones are BE, which represents a state, and GO, which represents any event. Other primitives include: STAY (a BE with an idea of duration), CAUSE (for expressing causality), INCH (for inchoative interpretations of events), EXT (spatial extension along something), REACT, EXCH (exchange), ORIENT (orientation of an object), etc. Their number remains small, while covering a quite large number of concepts. A second set of primitives, slightly larger (about 50) describes prepositions: AT, IN, ON, TOWARD, FROM, TO, BEHIND, UNDER, VIA, etc. These primitives are 'lower' in the primitive hierarchy and their number is *a priori* fixed once for all.

C. Semantic Fields

The LCS uses some principles put forward in (Gruber 65), namely that the primitives used to represent concepts of localization and movement can be transposed to other fields by analogy, and generalized. This statement has some obvious limits, but it has been used as far as possible in the LCS. It is clear that there are many similarities across semantic fields, for example, similar prepositions can be used for the expression of movement, localization, time and possession.

The main fields considered in the LCS are the following: localization (+loc), time (+temp), possession (+poss) and expression of characteristics of an entity, its properties (+char,+ident) or its material composition (+char,+comp). (Pinker 93) introduces additional fields such as: epistemic (+epist) and psychological (+psy).

Primitives can then be specialized to a field, e.g. GO_{+loc} describes a change of location, GO_{+temp} a change of time, GO_{+poss} a change of possession, and $GO_{+char,+ident}$ a change in the value of a property (e.g. weight, color). Similarly, BE_{+loc} describes a fixed localization, and BE_{+poss} the possession of something by someone (ommitting the arguments). All combinations of a primitive with a semantic field are not relevant.

2.2.3 Construction of LCS representations

LCS representations are constructed formally as follows:

- (1) $p \in P$, $v \in Var$, $semantic-field \in S$, $p_{+semantic-field}$ are well-formed LCS representations,
- (2) $p_{+semantic-field}(C_1, C_2, ..., C_n)$ is a wff LCS if the C_i are wff LCS representations (semantic fields may be underspecified or absent),
- (3) [$_e$ $C_1, C_2, ..., C_n$] is a wff LCS if e is a conceptual category ($e \in T$) and if the C_i are defined as in (2).

Condition (2) describes the instantiation of arguments by variable substitution while condition (3) describes the creation of the most external structure of an LCS

and the treatment of modifiers by concatenation (or adjunction).

The most important structures are the following (where semantic fields have not been included for the sake of readability since there are several possible combinations):

1. $PLACE \rightarrow [_{place}\ PLACE-FUNCTION([_{thing}\ ])]$
2. $PATH \rightarrow [_{path}\ TO/\ FROM/\ TOWARD/\ AWAY-FROM/\ VIA([_{thing/place}\ ])]$
3. $EVENT \rightarrow [_{event}\ GO([_{thing}\ ],[_{path}\ ])]\ /\ [_{event}\ STAY([_{thing}\ ],[_{place}\ ])]\ /\ [_{cause}\ CAUSE([_{thing/event}\ ],[_{event}\ ])]$
4. $STATE \rightarrow [_{state}\ BE([_{thing}\ ],\ [_{place}\ ])]\ /\ [_{state}\ ORIENT([_{thing}\ ],\ [_{path}\ ])]\ /\ [_{state}\ EXT([_{thing}\ ],\ [_{path}\ ])]$

PLACE-FUNCTIONS are symbols such as ON, UNDER, ABOVE, related to the expression of localization. Note that STAY is treated as a form of event.

2.2.4 A few limitations

It is clearly a huge enterprise to develop a formalism and a set of data sufficient to represent the meaning of a large number of words. The LCS is a vast system, with a limited number of primitives. The complexity of the system and its expressiveness lies in the complex interactions between its different constituents.

The primitive system of the LCS being initially based on movement verbs, it is clear that there are several families of verbs whose semantics cannot be reduced, even metaphorically, to a GO and a path or a place. For example, cognition verbs or psychological verbs are such families. Other verbs can be represented by the LCS, but their representation is not very natural and straightforward. We believe that the LCS framework is a formalism that has a good expressive power to represent the lexical semantics of words. However, it is necessary to slightly extend its set of primitives, in a principled way, in order to improve its coverage.

For the purpose of this paper, and w.r.t. the examples chosen, the LCS, in its current form, is sufficient to show how sense variations can be dealt with. We will however evoke some useful extensions, at various levels.

2.3 The Generative lexicon: main concepts

The approach developed by J. Pustejovsky in a number of articles (including (Pustejovsky 95)) is based on the notion of type coercion, which is a specific form of inference. We present here its main distinctive feature, the Qualia structure associated with the type coercion operation. Other aspects include: argument structure, event structure an dthe inheritance structure.

2.3.1 The Qualia structure

The Qualia structure is the most innovative structure of the GL. It describes most of the essential attributes of an entity. Although any lexical item may be given a Qualia structure, it is mainly designed for nouns and nominal structures and allows a more powerful taking into account of the semantics of nouns in verb phrases and in propositions. The Qualia structure is a convenient way of implementing the compositional relations between a predicate and its arguments. It does not define the denotational space of the entity in the world, but rather, and broadly, its interpretative space.

The Qualia structure is subdivided into four roles:

- **The formal role**, that distinguishes the object from those which are more generic. Globally, this role includes descriptions close to the *isa* relation, but probably with more flexibility. In particular, we may find indications about orientation, size, form, shape, color, and position.
- **The constitutive role**, which, essentially, describes the parts of the object. Again, this role has a wide coverage and may require some subdivisions to be usable. The GL distinguishes material, weight, parts, etc.
- **The telic role**, which defines the purpose and the goal of the object. All the actions which can be realized on the object are listed at this level (e.g. for book: read, copy-edit, bind, etc.).
- **The agentive role** describes the actions which allow the creation of the entity. For example, we have: an agent/creator, a causal chain or the natural type. Book will then be associated with actions such as write. Publish or copy-edit may also be candidates. Similarly, under this role also fall actions that describe the destruction of the object.

We have in the Qualia on the one hand two roles which describe the structure of the entity considered: the formal and the constitutive roles, and on the other hand two roles of a very different nature which are closer to the event structure: the telic and the agentive roles. The elements in the roles are predicates or names of entities denoting sets, which point to other lexical entries. For example, the Qualia structure of *novel* is the following:

Novel: noun, novel(Y)
constitutive: paper, cover, chapters, references,
formal: book(Y),
telic: read(X,Y), print(X,Y), etc...
formal: write(X,Y).

2.3.2 The generative operations

The GL essentially considers 3 generative operations: selective binding, co-composition and type coercion.

The two first operations have been presented above, let us now concentrate on type coercion, which is in fact the most important generative operation in the GL.

2.3.3 Type coercion

Type coercion is based on the notion of type shifting (Pustejovsky 91), which is a relatively well-known operation in formal linguistics and in computer science. Type shifting allows the creation of new types from subcategorization frames when the type expected for an argument does not correspond to the type found for the realized argument. Let us assume that there exists a certain set Σ of shifting operations defined on a given lexicon L, which can change the type of a lexical item. The *Function Application with Coercion* (FAC) procedure can then be defined as follows. Let α be of type c and β be of type $< a, b >$, then:

(i) if the type of c = a, then $\beta(\alpha)$ is of type b,

(ii) if there exist $\sigma \in \Sigma$ so that $\sigma(\alpha)$ results in an expression of type a, then $\beta(\sigma(\alpha))$ is of type b,

(iii) otherwise there is a type mismatch and the sentence is ill-formed.

For example, the type of a VP headed by *begin* is $<$ *event*, *event* $>$, which is for example realized in:
begin to read a novel.
In a sentence such as:
begin a novel,
there is a type mismatch since *a novel* is of type physical-object. At this stage, the Qualia of *novel* can be used to derive a new type which would meet the verb's requirement. Indeed, in the telic or in the agentive roles of the Qualia of novel, there are predicates such as write(X,Y), or read(X,Y) which refer respectively to the lexical entries write and read, both of type event. Therefore, the type of write and read, which is the type expected by the verb begin, can be 'pumped' and the VP *begin a novel* is well-formed. It is ambiguous and may mean begin reading or writing a book, but of course, the GL is not designed to resolve ambiguities. All the types which can in this manner be derived in one or more steps from the Qualia of novel define the *generative expansion* of the type of novel via type coercion and Qualia structure.

In the sentence:
Milano won the cup,
There is also a type mismatch at the level of the subject of *win* since the type of the expected subject is human, not a town or a location. Now, if, via the contitutive role of Milano, we know that it is a town (via the inheritance structure) composed of buildings, football teams, etc... and then, if we select football team, and consider again its constitutive role, we notice that a football team is composed of humans. Then the expected type is found. From the observation of a number of utterances, we believe that the depth of type pumping, to be psychologically realistic, should not exceed two levels.

2.3.4 A few limitations

The Qualia structure, in spite of some evidences of its 'existence', in particular for the Telic role, is particularly complex to describe, and open-ended. Also, it is postulated that sense variations can be resolved only by means of types and type shifting operations, which is not obvious. In fact, we demonstrate in this paper that it is not the case, e.g. for metaphors.

The GL system is very large and powerful and certainly overgenerates. It is necessary to introduce additional constraints. One possible solution we think would be useful is to consider, for each verb semantic class, which roles of the Qualias, for each argument, are the most relevant. For example, for 'aspectual' verbs such as *begin*, coercion for the object is mainly concerned with agentive and telic roles. Formal and constitutive roles do not lead to any viable solution. This short example shows that verb semantic classes can be used to organize and constrain the power of type coercion associated with Qualia structures.

3 A conceptual analysis of the polysemic behavior of adjectives

In this section we investigate the semantics of adjectives and their polysemic behavior within a GL perspective. A model based on rules and on underspecified LCS forms is given, and various forms of metaphors and metonymies are studied. We focus on the adjective *bon* (good), which is one of the most polysemic adjectives in French. The study of adjectives such as *rapide, difficile, triste* (fast, difficult, sad), which appear quite frequently in the GL litterature, shows many similarities and confirm the hypothesis presented here.

The syntax and the semantics of a number of adjectives has been investigated in (Bouillon 96, Bouillon 97), in a relatively strict GL framework. Our approach is more independent from the GL technical elements (e.g. the exact contents of the Qualia structure) and investigates in more depth the computation of the semantic representation of the compound adjective + noun. We also explore the modelling of metonymies and selection, and develop some sense variation rules.

3.1 Meanings of *'bon'*

Let us consider the adjective *bon*, which is one of the most polysemic adjective: 25 senses identified in WordNet (e.g. (Fellbaum 93)). In fact, *bon* can be combined with almost any noun (except color names)

in French, and as (Katz 66) pointed out, *good* would need as many different readings as there are functions for objects. These functions can, however, be found in the Telic role of the object argument.

We have identified the following senses and sense variations (metaphors and metonymies in particular, expressed as in (Lakoff 80)), for which we give some examples:

1. Idea of a good working of a concrete object w.r.t. what it has been designed for: *un bon tournevis, de bons yeux* (*good screw-driver, good eyes*). Metaphors abound: e.g.: 'communication acts as tools': *une bonne plaisanterie/mise au point* (a good joke), 'function for tool' (*un bon odorat*), 'paths as tools' (*a good road*). Metonymies are rather unusual since if X is a part of Y, a good X does not a priori entail a good Y [1].

2. Positive evaluation of moral, psychological, physical or intellectual qualities in humans: *bonne personne, bon musicien,* (*good person, good musician*). The basic sense concerns professions and related activites or humans as a whole: it is the ability of someone to realize something for professions, and for humans in general the high level of their moral qualities (qualities are more global in that case).

   This second sense could be viewed as a large metaphor of the first, with a structure-preserving transposition to a different ontology: from tools to professional or moral skills.

   There are some 'light' metaphors such as: 'social positions or ranks as professions' (*a good boss/ father / friend / citizen*), and a large number of metonymies: 'image for person, image being a part of a person' (*a good reputation*), 'tool for profession' (*a good scalpel*), 'place for profession' (*a good restaurant*). These metaphors have a good degree of systematicity, but need some semantic and pragmatic restrictions to avoid overgeneration.

3. Intensifier of one or more properties of the noun, producing an idea of pleasure and satisfaction (this is different for sense 5) [2]:
   noun(+edible): *good meal/dish/taste* = tasty, with metonymies such as 'container for containee' (*a good bottle/ glass*),
   noun(+fine-art): *good film/book/painting* = valuable, with metonymies such as 'physical support for contents' (*good CD*),
   noun(+smelling): *good odor* (this is less frequent for other senses)
   noun(+psycho): *good relation/ experience*
   noun(+human relations): *good neighbours.*
   Note that *bon* can only be used with neutral or positive nouns, we indeed do not have in French *good ennemies, *good humidity with the sense outlined here.

4. Quantification applied to measures or to quantities: *a good meter, a good liter, a good amount/salary, a good wind.* In this case, good means a slightly more than the unit/measure indicated or above the average (for terms which are not measure units such as wind or salary). This sense being quite different since it is basically a quantifier, it won't be studied hereafter.

5. Idea of exactness, accuracy, correctness, validity, freshness, etc.: *un bon raisonnement/calcul* = exact, accurate (*a good deduction/computation*), *good note/ticket* = valid, *a good meat* = fresh or eatable, *a good use* = appropriate, *good knowledge* = efficient, large and of good quality. The meaning of *bon* is therefore underdetermined. Depending on the noun, the semantics of *bon* is slightly different, this is not really a case of co-composition. It is the semantic type of the noun and that of the selected predicate in the telic role of the noun which determine the meaning of the adjective in this particular NP. We call this phenomenon, by comparison with selective binding, **selective projection**, because the meaning is projected from the telic role.

Bon appears in a large number of fixed or semi-fixed forms such as: *le bon goût, le bon sens, le bon temps, une bonne giffle.*

Almost the same behavior is observed for all evaluative adjectives such as excellent, terrific, bad or lousy in French. For example, for *mauvais* (bad), senses 1, 2 and 3 are identical, sense 4 is only applicable to amounts (*mauvais salaire*), not to units and sense 5 is almost identical, it conveys the idea of erroneous deduction, invalid ticket, bad use and rotting meat. Note that in WordNet, *bad* has only 14 senses, whereas good has 25 senses, with no clear justification.

3.2 Generative Devices and Semantic Composition

Let us now analyze from a GL point of view the meanings of the adjective *bon.*

In (Pustejovsky 95), to deal with the compound adjective+noun, a predicate in the telic of the noun is considered. For example, *fast*, modifying a noun such as typist, is represented as follows:

[1] This needs refinements: there are some weak forms of upward inheritance in the part-of relation: e.g. if the body of a car is red, then the car is said to be red.

[2] Norms are being defined for about 600 top-most nodes of a general purpose ontology in different projects and research groups (e.g. NMSU, ISI, Eagles EEC project), they will be used as soon as available.

$\lambda e\ [type'(e,x)\ \wedge\ fast(e)]$
where e denotes an event. This formula says that the event of typing is fast. A similar representation is given for *long*, in *a long record*. This approach is appropriate to represent temporal notions in a coarse-grained way, i.e. the event is said to be fast (with potential inferences on its expected duration) or long. But this approach is not viable for *bon*, and many other adjectives with little or no temporal dimension. In:
$\lambda e\ [type'(e,x)\ \wedge\ good(e)]$
it is not the typing event which is 'good' but the way the typing has been performed (certainly fast, but also with no typos, good layout, etc.). A precise event should not be considered in isolation, but the representation should express that, in general, someone types well, allowing exceptions (some average or bad typing events). This involves a quantification, more or less explicit, over typing events of x. Finally, *bon* being polysemous, a single representation is not sufficient to accomodate all the senses.

The semantic representation framework we consider here is the LCS, it has some obvious limitations, but seems to be appropriate for our current purpose. It is associated with a typed λ-calculus.

Below, we study in detail senses 1, 2 and 5 of *bon* to illustrate the above proposal.

3.2.1 sense 1: 'Bon' = that works well

This first sense applies to any noun of type tool, machine or technique: *a good car, a good screw-driver*. The semantic representation of *bon* requires a predicate from the telic role of the Qualia structure of the noun. It is the set (potentially infinite) of those predicates that characterizes the polymorphism. We have here a typical situation of *selective binding* (Pustejovsky 91), where the representation of the adjective is a priori largely underspecified. Let us assume that any noun which can be modified by *bon* has a telic role in which the main function(s) of the object is described (e.g. execute programmes for a computer, run for a car [3]), then the semantics of the compound adjective + noun can be defined as follows:
Let N be a noun of semantic type α, and of Qualia: [..., Telic: T, ...]
where T denotes the set of predicates associated with the telic role of the noun N. Let Y be the variable associated with N and let us assume that T is a list of predicates of the form $F_i(Y,X)$. Then the LCS-based representation of *bon* is:
$\lambda\ X,\ Y:\alpha,\ \lambda\ F_i(Y,X),$
$[_{state}\ BE_{+char,+ident}([_{thing}\ Y\ ],$
$[_{+prop}\ ABILITY-TO(F_i(Y,X)) = high\ ])]$.
which means that the entity denoted by the noun works well, expressed by the evaluation function ABILITY-TO and the value 'high'. This type of low-level function abounds in the LCS, this principle is introduced in (Jackendoff 97).

The Qualia allows us to introduce in a direct way **a pragmatic or interpretative dimension** via the instanciation of $F_i(Y,X)$.

From a compositional point of view, the combination Adjective + Noun is treated as follows, where R is the semantic representation of the adjective, T, the contents of the telic role of the Qualia of the noun N of type α, τ, a particular element of T and Y, the variable associated with the noun:
sem-composition(Adj(R),Noun(Qualia(T)) =
$\lambda\ X,\ Y:\alpha,$
$\exists F_i(Y,X) \in T,\ (N(Y)\ \wedge\ R(Y)(F_i(Y,X)))$.

The open position in R(Y) is instanciated by β-reduction. The selection of F_i is simple: for basic tools, there is probably only one predicate in the Qualia (screw-driver $\rightarrow$ screw), for more complex nouns, there is an ambiguity which is reflected by the non-deterministic choice of F_i, but probably organized with preferences, which should be added in the Qualia. It is the constraint on the type of Y that restricts the application of that semantic composition rule. This notation is particularly simple and convenient.

Metaphors are treated in a direct way: the constraint on the type of Y can be enlarged to:
$\lambda Y:\beta\ \wedge\ metaphor(\beta,\alpha)$
and the remainder of the semantic composition rule and semantic formula remains unchanged. We have, for example:
$metaphor(communication-act, tool)$ (joke).
$metaphor(communication-path, tool)$ (road).
The function F_i selected is again a predicate in the telic of the noun, with no interpolation towards a more normalized terms, as done in the GL for type shifting, e.g. we use: make-laught(X,Y) for *joke* and drive-on(X,Y) for *road*.

We have evaluated that, in French, there are about 12 frequent forms of metaphors for this sense. The study of this first sense suggests that the introduction of a hierarchy of preferences would be a useful extension to the Telic role, reflecting forms of prototypicality among predicates.

3.2.2 Sense 2: 'Bon' restricted to cognitive or moral qualities

Another sense of *bon* modifies nouns of type profession or human. The treatment is the same as in the above section, but the selection of the predicate(s) $\tau = F_i(X,Y)$ in the telic of the noun's qualia must be restricted to properties related to the moral behavior (makes-charity, has-compassion, has-integrity) when the noun is a person, or to some psychological attitudes and cognitive capabilities when the noun

[3] Less prototypical predicates can also be considered, e.g. comfort or security for a car, which are properties probably described in the constitutive role of the Qualia of car.

denotes a profession (e.g. *a good composer*). Alternatively, some of these properties could be found in the constitutive role (approximately the part-of relation), if properties can be parts of entities.

The typing of the predicates in the Qualia roles can be done in two ways, (1) by means of labels identifying the different facets of a role, but these facets are often quite ad'hoc and hard to define, or (2) by means of types directly associated with each predicate. These types can, for example, directly reflect different verb semantic classes as those defined in (Levin 93) or (Saint-Dizier 96) on a syntactic basis, or the major ontological classes of WordNet and their subdivisions. This solution is preferable, since it does not involve any additional development of the Telic role, but simply the adjunction of types from a separate, pre-defined ontology.

An LCS representation for this sense of *bon* is, assuming the following types for F_i:
$\lambda\ X,\ Y\ :\ human,\ \lambda\ F_i\ :\ action - related - to - profession \vee moral - behavior,\ \lambda\ Y : \alpha,$
$[_{state}\ BE_{+char,+ident}([_{thing}\ Y\ ],$
$[_{+prop}\ ABILITY - TO(F_i(Y,X)) = high\ ])].$

When several predicates are at stake, a set of $F_i(Y,X)$ can be considered in the representation, or the statement is ambiguous. The similarity of this representation with case (1) suggest that (2) is a metaphor of (1), as will be seen below.

Metonymies such as *a good scalpel* are resolved by the general rule: 'tools for professions'. This information could be in a knowledge base or, alternatively, it can be infered from the Telic role of the tool: any instrument has a predicate in its telic role that describes its use: the type of the first argument of the predicate is directly related to the profession that uses it. For example, scalpel has in its telic role:
$cut(X : surgeon \vee biologist,\ Y : body)$.
When the profession is identified, the standard procedure for determining the meaning of the compound can be applied. Metonymies using the part-of relation are quite simple to resolve using the constitutive role, as in the GL.

If this sense is viewed as **a metaphor of the first sense**, i.e. cognitive capabilities as practical/professional capabilities, then we can have here a **direct modelling of metaphors**, as fomulated in (Lakoff 80): a transposition of the current meaning to a distinct, but compatible, ontology. We can assume that structured parts of qualia roles in the Qualia structure correspond to different ontologies and to actions related to different ontological domains. These parts which structure a role (mainly the formal and telic ones), as advocated above, can be identified by means of types, such as those associated with the WordNet verb classes (and subclasses). Then, switching from one part of a role to another is a switch from an ontological domain to another. Very briefly, the telic role of a tool (sense 1) is of the form:

```
tool: Qualia: [ ..., Telic: P(X,Y):
                    technical-action, ...]
```

The type technical-action is e.g. a subtype of the WordNet verb class; 'verbs of creation, destruction and use'. Then, a profession and a person (sense 2) have the following generic Qualia elements of interest to us:

```
profession: [ ..., Telic: ... P(X,Y):
                      technical-action, ...]
human:  [ ..., Telic: ..., P1(X,Y):
                      psy-verb, ...].
```

We have here a direct transposition from the Telic of a tool (1) to the telic of a profession (different ontological domains are considered because the two lexical entries, tool and profession, belong to two different domains) and where a predicate P(X,Y) of the same type is considered, or (2) to the telic of a human, where a predicate of another ontology (human psychology) is considered. In this latter case, the difference in ontological domains is characterized by both the type of the lexical entry and the type of the predicate in the Qualia roles considered. The predicate is no longer P but P1. This behavior cannot be made systematic: such shiftings need to be explicitly specified, but it is nevertheless of much interest to use the descriptive power of the Qualia roles to model some metaphors.

3.2.3 Sense 5: 'Bon' = exact or correct

We have here a situation of *selective projection*: the exact meaning of *bon* is projected from the type of the modified noun and the type of the predicate selected in the noun's Telic role.

For example, if the noun is of type $bank - note \vee ticket$ and the type of the predicate selected in the noun's Telic role is $pay \vee give - access - to$, then the meaning of *bon* is 'valid':
$\lambda X : bank - note \vee ticket,$
$[_{state}\ BE_{+char,+ident}([_{thing}\ X\ ],$
$[_{place}\ AT_{+char,+ident}([_{+prop} valid(X)])])].$
The constraint on the type of the telic role is stated in the semantic composition rule:
sem-composition(Adj(R),Noun(X,Qualia(T)) =
$\lambda X : bank - note \vee ticket,$
$\exists F_i(\_,\_) : pay \vee give - access - to \in T,$
$(N(X) \wedge R(X))$. [4]

It is necessary to have both a constraint on the noun and on the predicate(s) in the telic role: (1) the type of the predicate in the telic role is certainly not a sufficient constraint , e.g. every noun's telic role in which there is the predicate *pay* cannot be combined with *bon* with sense 5; (2) the constraint on the type of the noun is also not sufficient, e.g. a medecine is a kind of food, but we don't eat it.

[4]The representation of the adjective in the LCS is adjoined to the main representation, the formalism is here simplified for readability.

3.3 Long-distance compositionality

The NP *a good meat* is related to senses 2 or 5. Instead of choosing one solution (a generate and test strategy), a set can be provided (as in constraint programming, see section 7). Now, if we have an NP of the form: *une viande bonne à consommer*, then the parsing of *consommer* will provoke the selection of sense 5 (and subsense 'fresh/consumable' via selective projection) because of the type of *consommer*. If, conversely, we have *une viande bonne à déguster*, then, since déguster is of type 'eat.enjoy' (a dotted type in the GL), sense 2 is selected. The space of meanings is restricted when additional information is found.

A second case involves default reasoning (as in (Pernelle 98)). In *un bon couteau pour sculpter* (a good knife to carve), by default, the action that the knife performs well is that protypically found in its telic role. But, if a less prototypical action is found explicitly in the sentence, then this latter is prefered and incorporated into the semantic representation instead of the default case. Indeed, the telic role describes prototypical actions, since the others are often unpredictable. The default meaning of *bon* is kept and 'frozen' until the whole sentence has been parsed. If there is no contradiction with that sense, then it is assigned to the adjective, otherwise, it is discarded in favor of the sense explicitly found in the sentence.

Finally, we consider the expressions *Y makes a good X, Y is a good X* as collocations where *good* is not fully treated compositionally.

4 Selection and Dimensions of Interpretation for Verbs

We hypothesize that a verb sense receives a single LCS representation, possibly largely underspecified, and a list of instanciations constrained by the nature of the arguments and also possibly by pragmatic factors. This 'polymorphic' representation is the representation of the verb. Usage variations entailed by metaphors or metonymies do not alter the meaning of the verb, but they are provoked by the juxtaposition of a verb and an argument. It is therefore the meaning of the VP or of the proposition which is not 'standard'. In a selection situation, the verb meaning becomes more specialized (a definition of subsumption in LCS is given in (Dubois and Saint-Dizier 96)). In the Generative Lexicon (Pustejovsky 91, 95), selection is treated by selective binding, which is an operation entirely based on type concordance and type subsumption. No attempt seems to be made to construct a meaning representation, which is not in fact the main goal of the GL.

Let us now present a few examples. Note that the verb classes considered here are those we defined for French (Saint-Dizier 96), they do not necessarily overlap with those defined in (Levin 93) or with those of WordNet.

4.1 The case of Construction verbs

The construction verb class includes verbs like *construire, bâtir, édifier, réaliser, composer*, (build, construct, realize, compose), etc. Let us concentrate on the verb *construire*, which includes usages such as:
construire une maison / un cercle / un projet / une relation.
(to build a house / a circle / a project/ a relation).
The sense variation goes from a central meaning with a concrete, physical object to an abstract object. The general representation of this verb is:
$\lambda J, I, [_{event}\ CAUSE([_{thing}\ I\ ],$
$[_{event}\ GO_{+char,+ident}([_{path}\ FROM_{+char,+ident}($
$[_{+prop}\ EPS(J) = non - exist\ ]),$
$TO_{+char,+ident}([_{+prop}\ EPS(J) = exist\ ]),$
$FROM_{+char,+ident}(part - of(J)),$
$VIA_{+char,+ident}(definition - constitutive(J))])]$.
which describes the coming into being of J, EPS means, roughly 'epistemic or existencial status of'. Two functions, related to lexical data, are used: part-of(J) which gets the parts of J, and definition-constitutive(J) which gets the definition of J (e.g. a circle is a set of points equidistant from a particular point: the center). If this definition is not available in the lexical entry corresponding to the lexeme J, then the function remains as it is, just stating that J has a certain constitutive definition.

Construire is probably the generic element of the class. If we consider the following sense of the verb *composer*, which is more specific, as in: *composer une sonate* (to compose a sonata), which is basically restricted to musical pieces (imposed by constraints proper to the verb), we get exactly the same phenomena and restrictions. Note that this verb has metaphorical extensions such as *composer un menu / une salade* (to compose a menu, a salad) where the property outlined is that the menu or the salad is going to look like a piece of art. These extensions are treated exactly as above. The form *se composer un visage* (to compose one's face = to hide his opinions/feelings) is also metaphorically derived from the sense considered here, but is rather a semi-fixed form since it is quite remote from the original sense and weakly compositional.

4.2 The Sell verbs

The 'sell' verb class introduces a simple default representation. Let us consider the verb *vendre* (sell), generic element of the class. Its basic argument is a physical object (which has an intrinsic value). Besides this usage, we have slightly more metaphorical ones,

such as:
vendre des rêves / des illusions (to sell dreams / illusions).
If we assume that, in this latter case, a dream has no intrinsic value, it is its association with vendre which makes emerged the idea of value via the expectations on the argument. We also have expressions like *vendre quelqu'un* = to betray someone. These usages define the possible sense variations of the verb sell. We can then say that these objects, in association with verbs of the 'sell' class (and a few other classes as well), get e.g. a fictive value, represented by the function: FICTIVE-VALUE(J), and there is also a type shifting on J.

The basic representation of *sell* is the following:
$\lambda I, J, K$, $[_{event}\ CAUSE([_{thing}\ I\ ]$,
$[_{event}\ GO_{+poss}([_{thing}\ J\ ]$,
$[_{path}\ FROM_{+poss}([_{thing}\ I\ ]),\ TO_{+poss}([_{thing}\ K\ ])])$,
$GO_{+poss}([_{thing}\ P\ ]$,
$[_{path}\ FROM_{+poss}([_{thing}\ K\ ]),\ TO_{+poss}([_{thing}\ I\ ])])])]$
$\wedge DEFAULT(P,\ VALUE-OF(J),\ J,$
$TYPE(J) = physical\text{-}object, COERCED\text{-}TYPE(J) = none)$.
where P is the anchoring point for the default, activated when J, the variable concerned, is of type physical-object. In this case, which is the standard one, J need not be coerced to any other type. The default representation represents the basic usage, for the other cases, this default is not used and other types of representations are anchored at P.

The general form of a default is then:
DEFAULT(anchor, representation, variable concerned, expected type for argument, coerced type if appropriate).
When the type is not physical object, then a different value is anchored to the position P, as explained above. The other possible values may equivalently (1) be specified in the representation of the verb, similarly to the default, but not with the status of a representation by default, and associated with constraints of use, or (2) by means of a rule. If the first case is chosen (with constraints on the type of the object), then it has the following form:
OTHER-REPT(P, FICTIVE-VAL(J), J, TYPE(J)= abstract artefact, COERCED-TYPE(J) = phys-obj).

4.3 The Measure verb class

The measure verb class includes verbs such as: *évaluer, mesurer, apprécier, explorer*, etc. (evaluate, measure, appreciate, explore). They can be represented by an LCS form, but this form needs to be paired with additional information. Of interest is, for example, the quality or certainty of the measure, which can be best represented by a non-branching proportional series (Cruse 86) where the scale orders verbs by increasing precision of the measure.

The object argument J of these verbs may be very diverse. It has at least one measurable dimension, which is probably given in the Constitutive role of its Qualia structure [5]. In the following LCS, we introduce the conceptual category 'epistemic', as defined in (Pinker 93). We first have the extraction of the property being measured, then state that it becomes known, and finally indicates that the value becomes known to the subject I. The property considered in J is given by the function PROPERTY-OF(J), which extracts a property in J (these functions are advocated in very recent works by Jackendoff (Jackendoff 97)). The representation is the following:
$\lambda I, J, P$, $[_{event}\ CAUSE([_{thing}\ I\ ]$,
$[_{event} ACT_{+epist}([_{thing}\ I\ ]$,
$[_{place}\ ON_{+epist}([_{prop}\ PROP(J)])]$,
$FOR_{+epist}([_{event}\ GO_{+epist}($
$[_{thing}\ VALUEOF(PROP(P))]$,
$[_{path}\ TOWARD_{+epist}([_{thing}\ known\ ])])])])])]$.

The primitive FOR indicates the goal. It seems that the object must have at least one direct measurable dimension, we have not identified any metaphorical use.

5 Metonymies related to uses and to actions on objects

Let us now investigate, for several classes of verbs, metonymies related to uses and to actions on objects. The elements used to treat these metonymies could be close to those found in the telic and agentive roles of the Qualia structure, they may be more vague and may also have a larger scope. This entails that the distinction between the agentive and the telic roles is weakened (e.g. the bringing about or creation, destruction (also in the Formal role), and the use of an object may overlap). By the implicit focus imposed by the semantics of the verb on a certain property of the argument, this property imposes its type to the argument, producing a kind of type shifting.

The observations we made tend to show that there are several regularities in argument shiftings, over sets of verbs, often over verb semantic classes as those constructed for English in WordNet or in (Levin 93) or for French, as those we have defined in (Saint-Dizier 96). However, there are several restrictions, in particular those naturally imposed by each verb on their arguments. There are also idiosyncracies and cases where argument shiftings are more or less acceptable in NL utterances.

A recurrent problem is the description of the properties of the objects in the lexicon. Although there

[5] We use the term 'probably' because the Constitutive role is not defined precisely enough from that point of view, but this is the most probable place to specify this type of property.

are emerging trends towards a normalization of lexical descriptions and the use of ontologies [6], there are still many significant differences in the nature of the representations and in their granularity. The lexical data we use here are made as precise and concrete as possible, but it is clear that they need a reformulation to be consistent with a particular lexical system.

5.1 The object to event metonymy

Let us consider first the treatment of famous cases of the GL where a type 'object' is coerced into an event (e.g. the famous 'begin a book'). If we have a general common-sense rule that says that *'any physical object has been created, may be used or deleted'* , then the type alternation object → event can be accounted for directly. Besides physical objects, we also have products of the human activity such as projects, debates, ideas, etc. which can become events. They are in general elements with a certain idea of duration and of production of a result.

Thus, in the sentence:
to begin a novel,
a novel is a physical object. The verb *begin* selects an object of type event. The type shifting rule $physical-object \rightarrow event$ can be applied. The ambiguity of the sentence is left unresolved. In fact, many types of events can be associated with this construction, such as: *read, write, summarize, classify, index, analyze, edit, copyedit, cover*, etc. These 'interpretations' may be felt to be part of a pragmatic component. Similarly to adjectives, preferences among predicates can be given.

Besides the verb *begin*, the above common-sense rule is also used for most verbs specifying actions: starting, creation, realization, fabrication, completion, destruction, maintenance verb semantic classes. Most verbs of creation, consumption and destruction in WordNet also accept this argument shifting:
to industrialize a product = to industrialize the manufacturing of a product, where industrialize selects an object of type 'procedure' (a property of a product is that it is e.g. produced or manufactured).

From these examples, we can somewhat generalize the above rule as follows:
$physical-object \vee intellectual-activity \rightarrow event$
This rule is valid for object arguments of the verb classes mentioned above. For the subject argument phenomena are different and somewhat more restricted. There are, in particular, many metonymies. There are, of course, other semantic types of objects, not presented here, which can undergo this type shifting.

[6]Norms are being defined for about 1200 top-most nodes of a general purpose ontology in different projects and research groups (e.g. NMSU, ISI, Eagles EEC project).

5.2 The metonymy: objects for their value

We now introduce type shifting operations on other properties of objects, which are not metaphors in the constructions we consider. These properties seem to be often measurable, such as the monetary value, the intensity of a smelling, etc.

If a common-sense rule states that any physical object may have a certain monetary value, we then have the type shifting:
$physical-object \rightarrow monetary-value$, as in:
payer une maison (to pay a house) = *to pay the value of the house* since *pay* selects an object of type 'monetary value' (as in *to pay a salary*).

Another set of properties are those related to measures:
augmenter, monter/baisser une note, une action (increase/decrease a mark, a share) = to increase the value of a mark, of a share (a mark has the property of being characterized by a value of type real or integer), or related to odors, as in:
sentir une fleur (to smell a flower) = to smell the perfume of a flower.
For this latter example, we have a shifting rule of the form:
$object(+has-smell) \rightarrow smelling$,
and the verb *to smell* selects an object of type smelling/odor. We think that the odor of a plant may not be in the constitutive of the word plant, since it is not a real part.

5.3 Metonymies related to functionalities or to the use of an object

Argument shifting related to the functionalities or to the use of an object goes beyond the alternation object → event presented above, while remaining in the same spirit. Some general and relatively frequent metonymies are related to functions, to the use, to the emergence, to the transformation or to the access of/to the object. We have, for example, the following cases, where, besides introducing a rule modelling the metonymy, we give a list of some of the most prominent verbs which accept this type shifting on their object argument. These examples need, obviously, to be further analyzed and categorized, but they nevertheless give a good view of what the type shiftings is:
— *automatiser / amorcer une procédure* (to automate/start a procedure) = to automate the running of a procedure, assuming that *automatiser* selects an object of type process. Most 'starting' verbs may be subject to this type shifting. The rule is then, roughly:
$phys-object(+procedure) \rightarrow process$.
— *arreter / hater / accellerer / activer / ajourner / retarder / repousser / un projet - un modèle de voiture*

(to start, ... , a project - a car model) = to stop the course of the production of a car model, the course of a project. Verbs like arreter and in fact most 'aspectual verbs', select an object of type event and may undergo the above type shifting, characterized roughly by the following shifting rule:
phys − object(+artefact) → process.
The use of the term 'process' is still approximate and it may be necessary to define a more precise typology of events. Below, events will be made more precise by means of feature, this is a temporary notation.
— *annoncer / communiquer / rapporter / révéler un secret - de la neige* (to announce ... snow) = to announce the creation, the existence, the coming into being, the arrival or the end of something. These communication or report verbs select an object of type event. The general shifting rule is, roughly:
phys − object → event(+coming − into − being).
— *défendre / interdire / autoriser un jeu - un jouet - un lieu - une sortie - un spectacle* (to forbid ... a game, a toy, a place, a show) = to forbid / allow the access to a place, the use of a toy, the attendance to a show. The allow/forbid verb class selects objects of type event expressing uses, accesses, attendance, movement, etc. The general shifting rule is, roughly:
phys − object(place ∨ toy ∨ ...) →
event(+use ∨ +access ∨ ...)

The argument shifting phenomena on the subject are slightly different and seem to be less complex:

- *Jean amuse / charme / captive / divertit / plait / déplait ...* (John amuses...) = the actions or the behavior(+verbal ∨ +physical) of John amuses. Most of the verbs of the *amuse* class are subject to this shifting, which we can summarize as follows: *human → event(+behavior ∨ +actions)*
- Similarly, but on a larger scale, we have: *Les syndicats influencent la stratégie de l'entreprise* (the trade-unions influence the company's strategy) = the behavior(+verbal ∨ +physical) of the trade-unions (of type 'human-organization') influences the company's strategy. Most verbs of the transformation class in French undergo this shifting, such as: *changer, métamorphoser, transformer, affermir, amoindrir, améliorer, redresser, etc.*, with their own additional, more or less idiosyncratic constraints on the subject. Other subjects related to natural or to mechanical activities are also subject to this shifting. It is interesting to note that this construction corresponds to a context (nb. 151, equivalent to a syntactic alternation) in French (Saint-Dizier 96). About 9.6% of the French verbs may undergo it, and are quite diverse.

The general form of a type shifting rule is then, roughly, the following:
< *Verb class* >, < *argument involved* >, < *constraints* >, *Type1* → *Type2*.
In forthcoming work, we will see that the syntactic form is part of the constraints of application, since some syntactic forms block type shifting.

The examples above show a quite good systematicity, and let us say that there is sufficient evidence, in spite of necessary exceptions, to motivate a treatement at this level. The Qualia structure turns out not to be necessary to determine acceptability of the sentences. The Qualia is useful to interpret them, but this interpretation is rather pragmatic, and we think that the ambiguity should be left unresolved.

6 Conclusion

In this paper, we have presented an analysis of adjectival modification and several cases of selective binding and metonymies on semantic classes of verbs withing the GL perspective. We have proposed several extensions to the Telic role to be able to account for the representation of the different forms of sense variations. In particular, we have shown how types can be added, and how predicates from the telic participate in the construction of the semantic representation of the compound noun + adjective and in the verb-argument relation. We have also shown how Telic roles contribute to the modelling of metaphors.

Coercions and the treatment of metaphors and metonymies are generally assumed to be general principles, however, they are in fact more specialized than they seem at first glance (e.g. *une bonne toque/ plume* = a good cook/ writer is quite specific, or very constrained). It is probably necessary to introduce narrower selectional restrictions on their use. Finally, of much interest is to know how much these rules are subject to linguistic variation.

In terms of implementation, our first experiments tend to show that a constrained-based approach is the best suited to deal with complex sets of potential solutions. Constraint programming allows indeed the management of large sets of potential solutions without assigning them one by one to variables as in Prolog's generate and test strategy. However, improvements are still necessary to better organize the management of constraints (simplifications and detection of incoherences) and the overall efficiency.

Acknowledgements

I would like to thank Alda Mari for working jointly with me on preliminary studies from which this work emerged. I also thank James Pustejovsky, Federica Busa, Mouna Kamel and Françoise Gayral for discussions which greatly helped to improve this work.

References

[1] Benhamou, F., Colmerauer, A.,(eds.), (1993), *Constraint Logic Programming*, MIT Press.

[2] Bouillon, P., Mental State Adjectives: the Perspective of Generative Lexicon, in Proc. *Coling'96*, Copenhague, 1996.

[3] Bouillon P., Polymorphie et sèmantique lexicale, Thèse de troisième cycle, Université de Paris VII, 1997.

[4] Busa, F., (1996), *Compositionality and the Semantics of Nominals*, PhD. Dissertation, Brandeis University, MA.

[5] Copestake, A., Briscoe, T., (1995), Semi-Productive polysemy and sense extension, *Journal of Semantics*, vol. 12-1.

[6] Cruse, A., (1986), *Lexical Semantics*, Cambridge University Press.

[7] Dorr, B. J., Garman, J., and Weinberg, A., (1995), From Syntactic Encodings to Thematic Roles: Building Lexical Entries for Interlingual MT, *Machine Translation*, 9:3-4, pp.71-100.

[8] Dorr, B., Jones, D., (1996), Role of Word Sense Disambiguation in Lexical Acquisition: Predicting Semantics from Syntactic Cues, in Proceedings of *Coling 96*, Copenhaguen.

[9] Dubois, D., Saint-Dizier, P., (1996) Construction et représentation de classes sémantiques de verbes: une coopération entre syntaxe et cognition, in Proc. RFIA96, Rennes, France.

[10] Fellbaum, C., (1993), "English Verbs as Semantic Net", *Journal of Lexicography.*

[11] Fellbaum, C.(1998), A Semantic Network of English Verbs, in C. Fellbaum (ed.) *WordNet: An Electronic Lexical Database*. Cambridge, MA: MIT Press.

[12] Goldberg, A., (1994), *Constructions: A Construction Grammar Approach to Argument Structure*, University of Chicago Press.

[13] Gruber, J., (1967), *Studies in Lexical Relations*, MIT doctoral dissertation and in *Lexical Structures in Syntax and Semantics*, North Holland (1976).

[14] Jackendoff, R. (1983), *Semantics and Cognition*, MIT Press, Cambridge.

[15] Jackendoff, R., (1990), *Semantic Structures*, MIT Press.

[16] Jackendoff, R., (1997), *The Architecture of the Language Faculty*, MIT Press.

[17] Katz, G. (1966), *The Philosophy of Language*, Harper and Row, New-York.

[18] Lakoff, G., Johnson, M. (1980), *Metaphors we Live By*, University of Chicago Press.

[19] Levin, B., (1993), *English verb Classes and Alternations: A Preliminary Investigation*, Chicago Univ. Press.

[20] Mari, A., (1997), Une analyse de la générativité en sémantique lexicale utilisant la structure lexicale conceptuelle, research report université de Lausanne and IRIT.

[21] Mari, A., Saint-Dizier, P., (1997), Générativité: au-delà d'une théorie des types, in Proc. RFIA, Grenoble, IMAG.

[22] Martin, R., (1979), Revue de Linguistique, vol. 17. Paris.

[23] Moravcsik, J.M., (1975) Aitia as a Generative Factor in Aristotle's Philosophy, *Dialogue*, 14.

[24] Nunberg, G.D., Zaenen, A., (1992), Systematic Polysemy in Lexicoilogy and Lexicography, Proc Euralex92, Tampere, Finland.

[25] Ostler, N., Atkins, S., (1992), Predictable Meaning Shifts: some lexical properties of lexical implication rules, in J. Pustejovsky and S. Bergler (eds.) *Lexical Semantics and Knowledge Representation*, Springer-Verlag.

[26] Pernelle, N., (1998), *Raisonnement par défaut et lexique génératif*, PhD dissertation, LIPN, Paris.

[27] Pinker, S., (1993), *Learnability and Cognition*, MIT Press.

[28] Pustejovsky, J., (1991), The Generative Lexicon, *Computational Linguistics*, vol 17-4.

[29] Pustejovsky, J., (1995), The Generative Lexicon, MIT Press.

[30] Pustejovsky, J., Bouillon, P., (1995), Logical Polysemy and aspectual coercion, Journal of Semantics, 12, pp. 133-162.

[31] Raskin, V., Nirenburg, S., (1995) Lexical semantics of adjectives, a micro-theory of adjectival meaning, MCCS report 95-288.

[32] Saint-Dizier, P. (1996) A Logic Programming interpretation of Type Coercion in the generative lexicon, in Proc. NLULP'96, Lisbon.

[33] Saint-Dizier, P., (1996), Verb semantic classes based on 'alternations' and on WordNet-like semantic criteria: a powerful convergence, in *Proc. Predicative Forms in Natural language and in lexical knowledge bases*, IRIT, Toulouse, to appear in a volume published Kluwer Academic (P. Saint-Dizier, ed.).

[34] Winston, M.E., Chaffin, R., Hermann, D., (1987), A taxonomy of part-whole relations, *Cognitive Science*, vol. 11.

A Natural Language Approach for the Design of Batch Operating Procedures

Andreas Linninger
Department of Chemical Engineering, University of Illinois at Chicago
Chicago, IL 60607, U.S.A.
Phone: +001 312 996 2581, Fax: +001 312 996 0808
E-mail: linninge@uic.edu
AND
George Stephanopoulos
Department of Chemical Engineering, Massachusetts Institute of Technology
Cambridge, MA 02139, U.S.A.
E-mail: geosteph@mit.edu

Keywords: Batch design, Natural Language, Task-Oriented Design

Edited by: Vladimir A. Fomichov
Received: August 8, 1998 **Revised:** November 12, 1998 **Accepted:** November 16, 1998

A computer-aided environment for the synthesis of processing schemes for the production of pharmaceuticals is presented. Human developers are guided through the innovative development process through a hierarchy of decision-levels following the principle of hierarchical decomposition. Progress of the development is organized through a knowledge-based model of process chemicals and their transformations involving quantitative and qualitative knowledge. Process chemists can directly implement their processing ideas by using a "natural" language used for the description of lab recipes. The computer-aided environment maintains consistency with first principles, offers assistance for crucial design decisions and solves autonomously specific sub-problems. The presented methodology demonstrates an example for the successful man-computer interaction in the innovative task of process synthesis.

1 Introduction

Process development of pharmaceuticals is a creative process which involves the concerted collaboration of experts in chemical science, chemical and process engineering. Efficient development of new chemical products secures the competitiveness of significant sectors of the pharmaceutical and specialty chemical industry. It is quite common that half of their current product palette did not exist on the market ten years ago. This rapid evolution is necessary despite long product development times in the order of three to ten years. The traditional design procedure consists of three major stages: (i) product discovery, (ii) product development and (iii) manufacturing.

In *product discovery* organic chemists systematically test new substances and/or modify properties of existing molecules to promote desired properties. The result of their research is a chemical recipe, i.e. the chemical reaction pathway including required raw materials, intermediate product and the desired final product as well as their stoichiometric relations.

In *process development*, chemical engineers depart from the laboratory recipe found in phase one to produce the substance at a pilot plant level. Their task is to identify the best set of chemical and physical transformations for raw material preparation, chemical reaction and product separation.

In the final *manufacturing phase*, which may only be reached after several years of research and development, the pilot plant results must be scaled and adjusted for large volume manufacturing. There are several shortcomings in the traditional design procedure which has been recognized by the industry [Dechema, 1990]: (i) Excessively long development times. (ii) Lack of a methodology for systematic process development covering all design phases. (iii) Missing strategies for finding the best trade-off between conflicting objectives such as economic versus environmental performance

1.1 Previous approaches in process development.

In order to overcome the drawbacks of the traditional process development procedure, computer-aided design methodologies have been considered. The following review briefly references previous work in the area of process design as well as AI work with relevance

for the subject. Short descriptions as well as possible limitations will be discussed.

Mathematical approaches. Several researchers have successfully applied purely mathematical approaches for synthesis tasks, [e.g., Grossman, 1996; Charalambides *et al.*, 1993]. Direct mathematical methods usually involve rigorous optimization of large superstructures using combinatorial or mixed continuous and binary variables, i.e. parameters and decisions respectively. Obviously only "solutions" anticipated within these superstructures can be identified. It is not clear though, how superstructures for sufficiently complex problems can be found without running into a combinatorial explosion.

Rule-based expert system approaches. Purely rule-based or expert systems were applied for the knowledge-based selection of separation systems [Barnicki and Fair, 1990; Wahnschafft, 1991] and the selection of solvents [Linninger *et al.*, 1996a]. While these systems have proven their success in diagnostic tasks or single decisions, they cannot accurately track quantitative system transition in response to their reasoning.

Planning. Another attractive synthesis strategy comes from applied artificial intelligence. A fundamental paper on planning theory was presented by Chapman [1987]. Planning for conjunctive goals exhibits strong problem solving capabilities in well-defined problem spaces [Lakshmanan, R. and G. Stephanopoulos, 1988a, 1988b; Currie and Tate, 1991; Fikes and Nilson, 1971] . The difficulty of domain-independent planning lies in its inability to deal with quantitative information. On the other hand, technical design decisions are generally based on relatively strict quantitative limits. Some researchers have proposed ways to circumvent the problem associated with the gap between fact-based reasoning and the necessity of quantitative information, e.g. Qualitative Process Theory [Forbes, 1984], Qualitative Simulation [Kuipers, 1986].

Several systematic approaches were developed by Siirola [Siirola and Rudd, 1971] and Douglas [1988]. Douglas' pioneering research led to the first systematic methodology for conceptual design of continuous chemical processes. His work motivated the hierarchical decomposition principle to divide the design problem into smaller well-defined decision levels.

1.2 Scope

In the subsequent following section, we will briefly outline the methodology of the BatchDesign-Kit (BDK), a computer environment for the development of conceptual process designs with ecological considerations. The main theme will evolve around the sub-problem of the design methodology of BDK which deals with the generation of operating procedures for batch pharmaceutical and specialty chemical manufacturing. An operation-centered language will be presented for the rapid ad-hoc generation of batch recipes. Section three will show the building blocks of the natural language already in use by chemists for the description of lab operating procedures. In section 4, the vocabulary and scope of the design language will be presented. The advantages from the use of a language approach for tackling open-ended design problems will be shown. We will briefly investigate the problem complexity and tractability of the combinatorial aspects of the synthesis problem in Section 5. Section 6 will present preliminary results from the application of the design language in the industrial practice.

2 The Decision Hierarchy.

In BDK, the design procedure is broken into four decision-making layers following the principle of hierarchical decomposition. Each layer departs form a well-defined initial information, i.e. preconditions. The objective at each stage is to elaborate the desired goal state through a series of design decisions, i.e. postconditions. The satisfaction of these stage goals can be made exclusively by the human and/or delegated to the synthesis agents of the environment. Although decomposing the chemical process design procedure is domain-specific, its study can serve as an example for computer-aided systems in other innovative design situations and search problems in open-ended design spaces.

Initialization. All information concerning a new technology is organized within the scope of a *project.* Project initialization requires the definition of the final design objective, i.e. specification of the desired production level of the target molecule. Addition input concerns documentation of the project start, developers, etc.

Level 1 - Reaction Network. Synthesis of a chemical substance usually involves numerous chemical reactions. A network of reactions leading to the final product is called a route. Each route is composed of several steps, where at each step only one controlled reaction may take place to produce a single step intermediate. The purified intermediate serves as the raw material for the subsequent step.

Level 2 - Step Refinement. The next phase aims at developing a production recipe for each step. This involves the exploration for the type of operations and their sequencing to carry out the reaction schemes defined at the route level. This is a search for a sequence of transformations like charge, heat, mix, react, filter, distill, etc, to do

- preparation of raw materials for reaction
- their chemical conversion to yield the desired intermediate and finally

• purification of the stage intermediate for its subsequent use in the next step.

Level 3 - Waste Treatment. Execution of the step production recipe leads to the final product in desired quantity and purity as well as large amounts of by-product streams. All the waste streams in violation of existing regulations need to be recycled or treated. Only benign residuals may be released into the atmosphere, the sewer or the deposited on landfill.

Level 4 - Equipment Allocation. With the final production flowsheet including recycling and waste treatment options, the conceptual recipe can be mapped into an existing plant facility. It is possible that one recipe produces distinct operating scheme in plant locations with dissimilar equipment types and capacities.

3 The Building Blocks of the Design Language.

A discussion of each of the four levels is beyond the scope and objective of this article. Therefore we will limit the discourse to the language-based approach for the interactive development of operational procedure (Level 2 - Task Refinement). Fig. 1 shows a typical stage batch recipe used in the pharmaceutical industry. In its entirety, the synthesis of the intermediate *Acetate-Salt* from the raw material *N-Hydroxy* involves 60 operational steps such as charge, heat, react. The system state at each stage is determined by the thermodynamic state of process streams. Physical and chemical operations cause state transitions in order to convert the reactants into the intermediate product in desired quantity and purity. The objective of the design is to identify the best sequence of operational tasks that can economically transform the initial state into the desired goal state. Therefore, generation of batch operating schemes requires (i) a model for the description of process streams and (ii) models for the operational tasks that reflect the system transition.

3.1 Representation of Material Properties

The thermodynamic state S of process mixtures is determined by amount, m, temperature, T, pressure, P, and composition vector $\underline{X}$, i.e. S = f (m, T, P, $\underline{X}$). All other properties of interest for design are functions of these state parameters. Ideally, a computer-aided design environment should incorporate a complete model for all chemical substances, their mixtures and thermodynamic properties. In the AI literature, integration of quantitative knowledge as opposed to mere factual propositions is termed *deep knowledge* [Currie and Tate, 1991]. Consistent state representation of processing streams requires deep knowledge. In this section we will present, how a formal representation, entitled the *Material_Model*, emulates context-specific description of material mixtures and their thermodynamic properties.

Changing instance behavior. It is important to recognize that computations of material properties are strongly context specific. This means that attributes such as density, total enthalpy depend on the state parameters, e.g. S = f (m, T, P, $\underline{X}$) amount, temperature, pressure as well as composition. Different values of the state parameters may also require consideration of distinct computational procedures. Even class membership is a function of state parameters. As an example consider a multi-component mixture at temperature T. If T is above the dew point, the mixture is a gas. If T is above the mixture' s bubble point, but below its dew point, the mixture is characterized by two phases, i.e. a liquid and a vapor phase. If T is below the bubble point, it is a single-phase liquid mixture. If is cooled further it may solidify and could be described as a solid.

Traditional concepts of object-oriented technology require a-priory assertion of class membership. This means that each instance must belong to a class at creation and persevere uninterrupted class membership throughout its lifetime. However, when specifying a new process stream at state S1 ($m_1, T_1, P_1, \underline{X}_1$), it may not be known to which category it belongs, e.g. single-phase, binary, etc. This state-dependent instance behavior calls for non-conventional class-instance relationships. The Material_Model only recognizes a single meta-class, whose attributes store the thermodynamic state parameters. All process mixture are instances of this class. Specific instance properties are generated dynamically subject to the state S via a Model Selection Mechanism.

Model Selection Mechanism (MSM). In the Material_Model traditional method execution is replaced by a state dependent model selection mechanism. The assessment of physical phases, i.e. *AssessPhase method*, demonstrates that point. *AssessPhase* analyzes the state parameters of the individual components of a mixture and returns symbolic results like 'liquid', 'solid'. Depending on this categorization, the evaluation of property functions, e.g. total enthalpy, invokes distinct procedures resonating qualitative differences among process streams or phases. MSM ensures execution of appropriate methods without the need for static object membership as propagated by conventional class inheritance. Specific object behavior does not have to be enforced by class-membership to a specialized class, e.g. solid mixture is a member-of Class Solid_Mixtures. The state and number of phases is dynamically assessed and property calls are routed to appropriate procedures. Property evaluation relies on a knowledge-based model selection mechanism embedded in the rules of the Property Manager. It acts as the

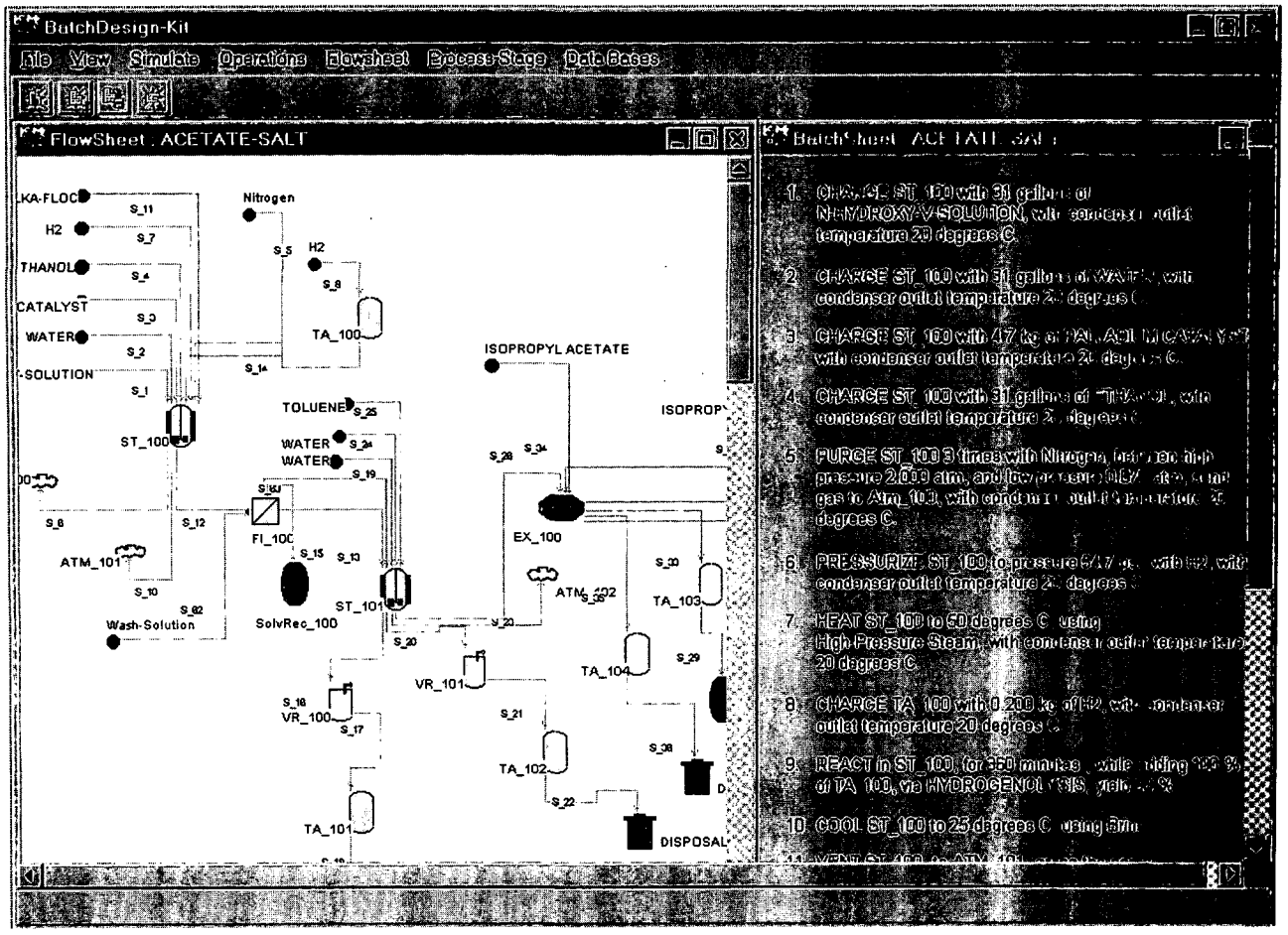


Figure 1: Typical Batch Recipe

agent in charge of finding appropriate data sources and selecting computational procedures. Property queries appear in functional, e.g. context-specific procedures for the calculation of mixture properties, as well as in static form, e.g. the molecular weight. Inclusion of new phenomena can therefore conveniently be implemented by adding a "feature" to the property manager, without changing the class design of the material model.

Dynamic linkage of external databases. The large number of relevant data for decision-making in design makes it necessary to resort to a host of external databases. These independent databases which may be distributed over the net [Krendl, 1998] may overlap in their content or cover complementary expertise. Dynamic property management shields the Material_Model from future changes of the database format or content.

The Material_Model's growing thermodynamic knowledge from the BDK project as well as earlier applications [e.g., DynEAF, Linninger et. al., 1995] is competent about 1300 organic and inorganic molecules and their properties between 200 K to about 1600 K at moderate pressures. Current expansions include:

- Refinement of rules for the model selection of the Property Manager [Lichtenberger, 1997]
- Health and safety related databases [Linninger, 1997]
- Augmentation of the scope to include ionic species as well as a submolecular and atomic perspective [Chakraborty and Linninger, 1998].

Through changing instance behavior and MSM, the Material Model offers consistent representation of pure chemical species multi-component, and multi-phase mixtures and safeguards context-specific state description of batch processing streams for a large range of state parameters.

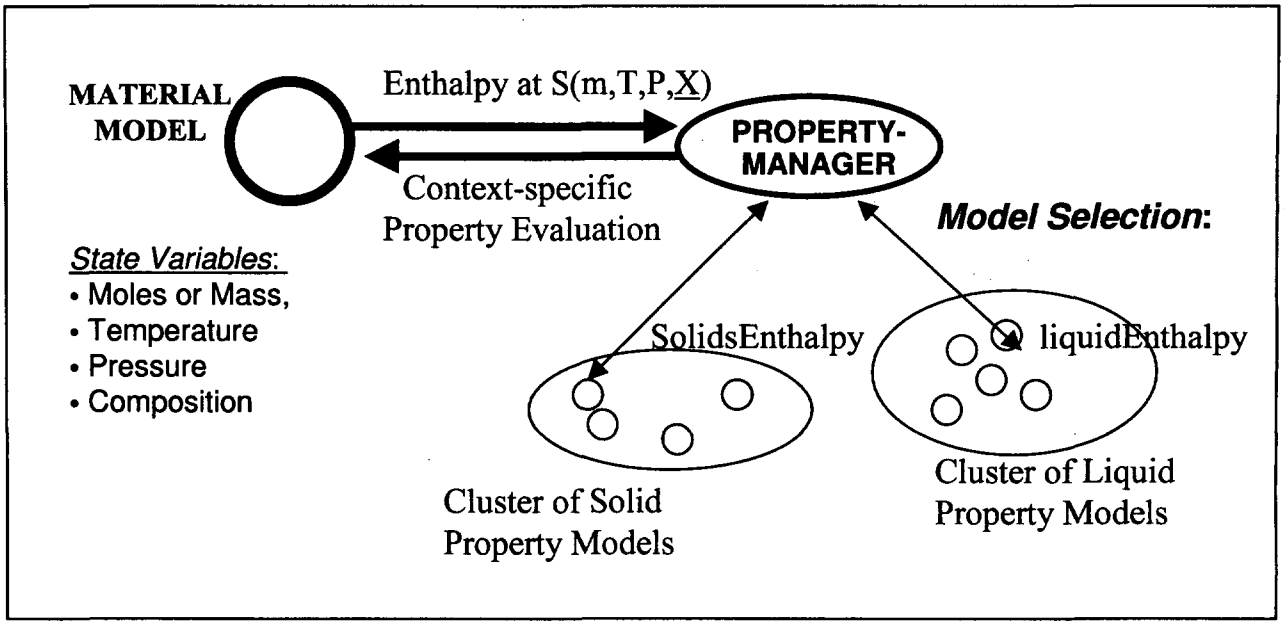


Figure 2: Context-specific Model Selection

3.2 Models for state transitions.

State transitions in batch processes involve physical and chemical transformations of the process streams described in section 3.1. In a computer-assisted design environment such as BDK, it is important to ensure that only technically feasible and consistent transforms are permitted by the system. These transformations like *material mixing, chemical reaction, or separation of species* alter state parameters of a material mixture in agreement with the laws of nature. These first principles like the conservation of mass and energy, thermodynamics and kinetics of reacting systems, or phase equilibrium relations in product separation steps can be expressed mathematically as systems of nonlinear algebraic, ordinary differential equations, or a combination of the above, differential-algebraic equations. The next paragraph will outline the modeling aspects of consistent state transitions of batch operations.

3.2.1 Consistency maintenance of state transitions.

Consistent state transitions caused by batch operations also known as tasks are constrained by *first principles*, which manifest themselves as algebraic relationships among state parameters in the initial state and the final state. A task is an operator that converts the initial material mixture state S_1 into a new state S_2 subject to a set of equations that describe physical or chemical phenomena, e.g cooling leads to a temperature decrease and possibly to the phase change freezing. In addition, some operations require conditions on the initial state S_1 in order to be applicable. As an example take the operation, distill, which requires that the mixture at S_1 is in the liquid state. This approach leads to operation models as action operators [Russel and Norvig, 1995] composed of two elements: (i) a set of *preconditions* that the initial state S_1 has to obey to and (ii) an *effect* that transfers S_1 into a new state S_2.

Preconditions. Before an operation can be applied, the initial material state is examined. Typical preconditions include (i) existence checks, (ii) qualitative conditionals such as number and composition of phases and (iii) quantitative constraints for state parameters or state functions. If any precondition is violated, the intended operation cannot be executed and the precondition violation is flagged to the user.

Preconditions may be eliminated through (i) insertion of another operation that remedies the precondition violation or (ii) retracting of the "illegal" task which is in violation with the system state.

Operation Effects. The effects serve the purpose of predicting the outcome of each task after application, i.e. the state S_2. Consistency maintenance for the transition requires two steps: (i) model building and (ii) constraint satisfaction depicted in Fig 3.

Model Building Mechanism (MBM). Model building automatically generates an appropriate set of consti-

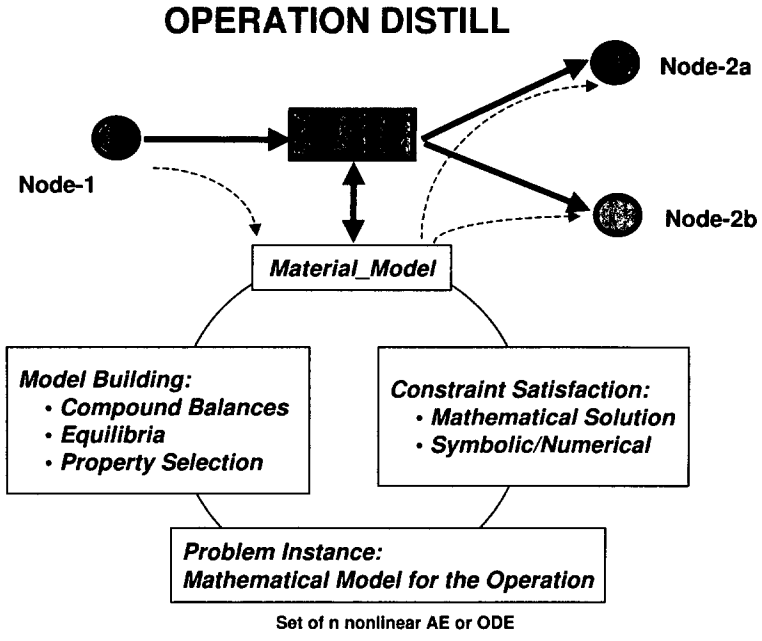


Figure 3: State Transition - Model Building and Constraint Satisfaction

tutive equations like thermodynamic constraints that describe the phenomena pertaining to an operational task. This phase involves the interpretation of desired operation and its parameters and the initial system state, i.e. the properties of the material mixture. The Model Building Mechanism interacts closely with the Model Selection Mechanism described for the Material_Model, since it involves not only qualitative statements about the actual state, but projects into properties of the goal state. In combination, MBM leads to a mathematical model composed of an appropriate set of thermodynamic or kinetic constraints.

Constraint Satisfaction. In a seconds step, the mathematical problem formulation of phase one needs to be solved in order to produce a consistent new state, i.e. the new properties of distillate and still. This step is done by delegating the problem formulation, i.e. a set of non-linear AE, to a mathematical agent for numerical solution. A more detailed discussion of the mathematical aspects of constraint satisfactions of problems formulated in the building phase can be found in [Linninger, 1993; and Linninger et al. 1995]. The separation of problem formulation and mathematical solution is the subject of active systems research in academia [Jarke and Marquardt, 1996; Westerberg, 1998, Linninger et al., 1998]. Progress in mathematical agents involving symbolic and numerical expertise is reported elsewhere [Linninger *et al.*, 1996b].

3.2.2 An exemplar operation model.

. As an example, we want to consider the case of a concentrate step of a ternary liquid organic mixture composed of three compounds A, B, C. The state is given by $S_1(m_1, T_1, P_1, X_1(x_{A1},x_{B1},x_{C1}))$. Concentrate the mixture at S_1 until the final concentration of compound A reaches x_{A2}. Clearly before the concentrate operation can be executed the precondition must be satisfied. These include the following propositions:

- T_1 is below the bubble point of mixture at S_1, i.e. the concentrate operation is only feasible for liquid mixtures.
- The concentration of A at S_2, x_{A2}, is lower (higher) than x_{A1}, if A is one of the volatile (heavy) compounds.
- Compounds A, B and C form a ternary liquid mixture, with positive nonzero vapor pressures

If these preconditions hold batch concentration is applicable. For calculating the effects of the operation, i.e. the sate parameters at S_2, the following constraints have to be satisfied:

- Component balances for each species in the vapor and the liquid phase
- Equality of the fugacity property of all species in all phases
- Enthalpy balance to calculate the required amount of heating medium, i.e. steam.
- Compatible property models used in the above equations as described in the Material_Model.

Fig. 4 summarizes the describing equations in model building phase in mathematical notation. Depending on the number of components, this simple concentration process could lead to several dozens of nonlinear

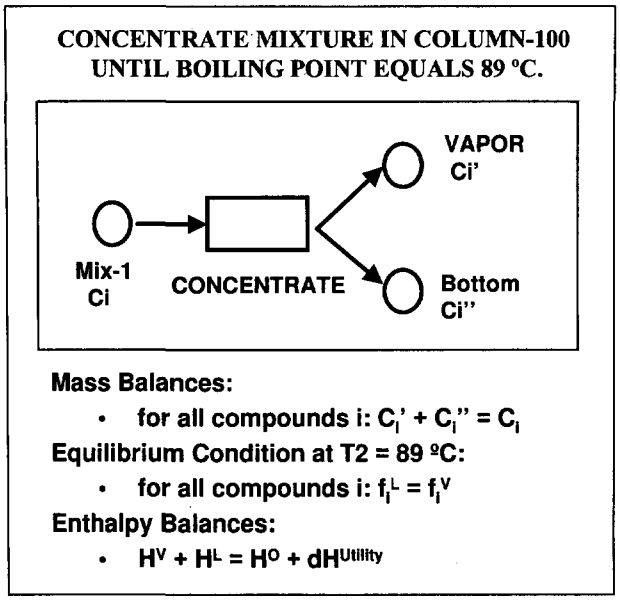


Figure 4: Overview of mathematical constraints generated by *concentrate* operations

algebraic constraints. Similar mechanisms are in place to safeguards consistent state transitions for the batch operational tasks.

4 Using the Operation-centered Design Language

4.1 Motivation

In the previous section, the relevant design concepts for the generation of operational procedures have been presented. This section will demonstrate the use of a *natural design language* for the interactive generation of batch recipes. This language mimics the terminology of chemists' laboratory procedures. The use of a design language is very appealing, since it allows designers to focus on the creative aspects of process development. Selection and evaluation of adequate operations in purely mathematical form would be very time consuming and absorb the designers' innovative energy. The evolution of conceptual process design concepts can be conducted in this *virtual laboratory* due to the consistency mechanisms discussed in the previous chapters. The subsequent section will discuss the main elements of this design language.

4.2 The vocabulary of the design Language.

BDK offers a high-level man-machine interface through which process chemists can directly implement their processing ideas by using their "natural" language. The vocabulary for the design language was designed according to the natural language used by chemists for the description of laboratory procedures [Kull and Hsu, 1991]. Currently, the BDK design language is composed of more than 40 operational tasks for material and heat transfer, reactions, operations on solids and gases, liquid and column separations as listed in Table 1.

Creating a Reaction Operation. As an example for the execution a chemical transformation consider the situation depicted in Fig. 5. Note that this `REACT` operation is using an auxiliary concept entitled a Reaction Template. It allows the definition of reaction information of arbitrary complexity by specifying the reaction stoichiometry. Using the reaction template of Fig. 5, designers can generate the following sentence in their natural chemist's language:

REACT IN REACTOR-1, ISOTHERMALLY, FOR 120 Minutes, WHILE ADDING 100

The textual input is interpreted by the BDK system in terms of the building blocks described in section 3. A 'reaction' in the virtual laboratory is carried out as follows: The REACT operation carried out in REACTOR-1 to convert the reactants in REACTOR-1 and additional feed from ST-200 into the appropriate reaction products as described by the stoichiometric information found in the "MY-REACTION" `REACTION OBJECT`. The keyword, isothermally, specifies that the operation does not change the temperature of the resulting reaction mix. That requires that possible reaction heat effects, exothermic reaction, must be absorbed by a cooling medium. The amount of cooling medium results form an enthalpy balances executed by the operation model. As described in the previous section. The reacting mixture instance within REACTOR-1 will be added the content of ST-200, followed by the calculation of the amount of products and remaining reactants. After execution of the react operation, the REACTOR-1 contains the reaction products as well as unreacted reactants.

4.3 Processing of the Chemists' Language Input.

Designers can compose batch recipes using the language constructs similar to the reaction case of the prior paragraph. All operations depicted in Table 1 offer input templates that facilitate the construction of syntactically correct statement. If the input parser identifies an input error, the statement is not accepted and an interactive message suggests a remedy to the problem, e.g. *The mole-fraction must be between 0 and 1.*

In addition to these syntactic rules, the system is capable of examining the semantic content of each op-

Group of Operation	Name of Operation
Material Transfer	Charge, Charge-Partially-from-Recycle, Recycle, Transfer, Transfer-through-Heat-Exchanger, Transfer Intermediate
Heat Transfer	Age, Cool, Heat, Heat and Reflux
Operations on Gases	Pressurize, Purge, Vacuum, Vent, Sweep
Operations on Solids	Centrifuge, Crystallize Continuously, Dry, Filter, Filter Continuously, Filter in Place, Wash Cake
Liquid Separations	Concentrate, Distill, Distill Continuously, Extract, Extract Continuously, Decant
Column Operations	Elusion, Loading, Regeneration
Reactions	React, React-in-CSTR, ph-Adjustment, Quench
Synchronization	Begin Parallel Operations, End Parallel Operations, Begin Sequential Operations, End Sequential Operations
Notes	Note, Safety Instruction

Table 1: The BDK operation-based language

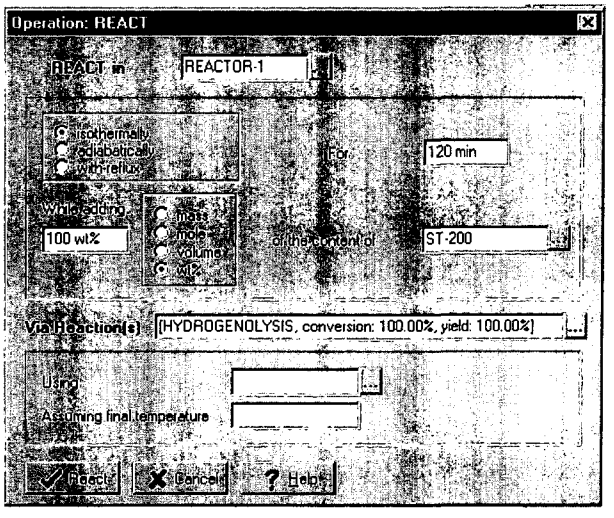


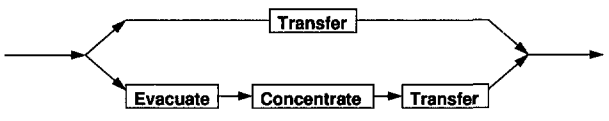
Figure 5: React Operation - Input Dialog

eration. BDK would also check for (i) Existence of referenced items: *Exists unit ST-200, its content is not empty;* and (ii) Operational Consistency, i.e. necessary preconditions of operational tasks. The react operation requires a all reactants to be present in the mixture before task execution.

Each operation offers a compulsory part and an optional part. The *words* within each operation sentence are instances of the following concepts: (i) Stock materials or previously defined process mixtures, i.e. instances of the Material_Model (ii) Specific equipment types and (iii) Quantifiers for the specification of state parameters, e.g. temperature.

Synchronization of Operations. The discussion so far focused on the sequential application of operations. In practice, some tasks in a batch plant have to be carried out in parallel. To account for these situations, the BDK operation-centered language provides the synchronization of multiple tasks. At this point a comment on the interpretation of the timing of the operations is in place. Each operation allows either for (i) explicit statement of the duration by the user or (ii) estimation of the duration for each batch task based on engineering heuristics [Linninger et al., 1997]. An agent, called the *BDK Scheduler*, keeps track of the evolution of the global process time by tracking the absolute system type starting from the first operational step until the last task. In a practical manufacturing site, a more sophisticated timing of the task network allowing for parallel operations and sequences of operations within parallel branches is desirable. The time information monitored by the *BDK Scheduler* at the conceptual design level, will enable the BDK environment to produce Gantt Charts and to optimize the process flowsheet, i.e. cycle time allocation and batch sequencing, at the preliminary design level.

Parallel Operations. Within the bracket of a `Begin Parallel Operations`, i.e. to initiate, and `End Parallel Operations`, i.e. to close, an arbitrary number of parallel tasks can be performed. All operations within parallel statements are assigned the same starting time. The BDK also al-



(a) Operational flow

1. Begin Parallel Operations
2. Transfer...
3. Begin Sequential Operations
4. Evacuate...
5. Concentrate...
6. Transfer...
7. End Sequential Operations
8. End Parallel Operations

(b) BDK syntax using Synchronization Commands

Figure 6: Syncronization of the operations

lows for inserting of a sequence of operations defined with the `Begin Sequential Operations - End Sequential Operations` idiom, within such a parallel parenthesis. Deeper nesting is not supported with this BDK release, although any number of parallel actions can be used consecutively in a stage.

Fig. 6 (a) illustrates where a situation where a *Transfer* operation has to be carried out in parallel with a sequence of three simultaneous operations. After completion of the parallel tasks, the subsequent operations continue from the point where the longer branch in parallel ends. Fig. 6 (b) demonstrates the syntax of operations in the BDK language using the synchronization commands. Each operation is abbreviated by its name.

4.4 Evolving the process design using the design language.

Each stage stores the tasks, i.e. the operations for the processing of the chemicals. The interactive evolution of the design will evolve graphically on the flowsheet and in textual form on the associated Batch Sheet. Each operation is assigned an index according to its creation on the Batch Sheet. The example of Table 2 shows an entire stage consisting of 32 operational steps. Step-1 to Step-10 are dedicated to the preparation of the raw materials, Step-11 involves the reaction and Step-12 through Step-32 serve the task of product separation. The last operation formally terminates the stage definition through a `Transfer-Intermediate` operation.

The chemists recipes generated with the operation-centered design language are visualized on the (i) Batch Sheet, which holds a textual description of the operational step, (ii) the Process Flowsheet, and (iii) the Process Sequence Diagram, which illustrates the timely evolution of the material streams and the derived functional and capacity requirement. Fig.1. shows a schematic of the BatchSheet and the Process FlowSheet in the BDK environment.

The unique operation-centered design language capability enables process chemists to do interactive evolution and refinement of their detailed chemical recipes. They can immediately observe the effects of different choices of operational tasks as state changes to the process mixtures in this virtual laboratory.

5 System Theoretical Approach to the Design Problem

The amount of possible combinations of operational sequences as well as the associated chemicals make task selection for batch recipes an extremely complex and open-ended problem. In this section, we will analyze the complexity of the problem and assess the effectiveness of the language-based approach from a system theoretical point of view. The analysis will include a quantitative evaluation of the complexity and tractability of this synthesis problem. This study will also provide qualitative arguments of the validity of the proposed problem formulation and its new operation-centered approach. We will offer a new paradigm for the interactive design of chemical recipes using a natural design language whose vocabulary emulates laboratory procedures used by chemists.

5.1 Tractability and Complexity.

Let edge-branching factor (e) be defined as the average number of operators applicable to a given state. An unconstrained purely mathematical approach to process design spans an indefinitely large design space of "moves", i.e. $e \rightarrow \infty$. Using the BDK operation based language, the edge-branching factor is equal to the cardinality of the set of operators, $e = 40$. If in addition task selection is made by an experienced human designer as proposed in the interactive design mode of BDK, his or her interpretation of the current situation may narrow the number of meaningful operators to a few alternatives. For the question of which operation to use for the separation of a product form the solvent rich reaction mix, a designer may name only a short list of possible options: Crystallization, Extraction, Extractive Distillation. In this situation the edge-branching factor (e) is equal to three.

The BDK operations can be classified into three categories according to the state change they produce: (i) neutral (ii) converging and (iii) branching operators. Neutral operations modify one or more state attributes of a single mixture, e.g. pressurize increases the pressure. Converging operations join more than one material node to produce a single new node. Branch-

Step-1 BEGIN PARALLEL OPERATIONS
Step-2 CHARGE ST-100 with 120.5 kg of acetic-acid, with condenser outlet temperature 20 degrees C
Step-3 BEGIN SEQUENTIAL OPERATIONS
Step-4 NOTE: THF must be sieve-dried
Step-5 CHARGE ST-101 with 204 kg of tetra-hydro-furan, with condenser outlet temperature 20 degrees C
Step-6 CHARGE ST-101 with 13 kg of potassium-butoxide, with condenser outlet temperature 20 degrees C
Step-7 CHARGE ST-101 with 23 kg of hydroxamineIV, from Drum-101, with condenser outlet temperature 20 degrees C
Step-8 AGE ST-101 for 30 minutes , with condenser outlet temp of 20 degrees C
Step-9 END SEQUENTIAL OPERATIONS
Step-10 END PARALLEL OPERATIONS
Step-11 REACT in ST-100, for 120 minutes, while adding 100 % of ST-101, via reaction RING-CLOSURE
Step-12 CHARGE ST-101 with 37 kg of tetra-hydro-furan, with condenser outlet temperature 20 degrees C
Step-13 CHARGE ST-101 with 37 kg of acetic-acid, with condenser outlet temperature 20 degrees C
Step-14 TRANSFER 100% contents of ST-101 to ST-100, with condenser outlet temp 20 degrees C
Step-15 AGE ST-100 for 30 minutes , with condenser outlet temp of 20 degrees C
Step-16 FILTER batch from ST-100, in FI-100, separating solids [100.0wt% , potassium-butoxide][100.0wt% , NaCl] [100.0wt% , potassium-acetate] as SOLID, lod of cake 30 %, sending mother liquor to ST-102, giving the name Mother-Liquor, operating time 240 minutes, with outlet temperature 20 degrees C
Step-17 WASH CAKE in FI-100 with 10 gallons tetra-hydro-furan, sending wash to ST-102, name it Spent-wash, lod of cake 20%, number of wash 1, operating time 90 hours per wash
Step-18 BEGIN PARALLEL OPERATIONS
Step-19 TRANSFER 100% contents of FI-100 to Disposal-100, the transferred portion is named THF-Wet-Cake, with condenser outlet
Step-19 EVACUATE ST-713 to 50 mmHg, operating time 15 minutes, with condenser outlet temp of 20 degrees C
Step-20 BEGIN SEQUENTIAL OPERATIONS
Step-21 EVACUATE ST-102 to 50 mmHg, operating time 15 minutes, with condenser outlet temp of 20 degrees C
Step-22 CONCENTRATE batch in ST-102 until final volume equals 30 gallons, through condenser CN-100, and receiver VR-100 , name the distillate Distillate, with condenser outlet temp of 20 degrees C
Step-23 TRANSFER 100% contents of VR-100 to TA-100, with condenser outlet temp 20 degrees C
Step-24 TRANSFER 100% contents of TA-100 to SolvRec-100, the transferred portion is named THF-ACOH-Dist, with condenser outlet temp 20 degrees C
Step-25 END SEQUENTIAL OPERATIONS
Step-26 END PARALLEL OPERATIONS
Step-27 CHARGE ST-102 with 241 kg of acetic-acid, with condenser outlet temperature 20 degrees C
Step-28 EVACUATE ST-102 to 15 mmHg, operating time 15 minutes, with condenser outlet temp of 20 degrees C
Step-29 CONCENTRATE batch in ST-102 until final volume equals 30 gallons, through condenser CN-100, and receiver VR-101, name the distillate Distillate, with condenser outlet temp of 20 degrees C
Step-30 TRANSFER 100% contents of VR-101 to TA-101, with condenser outlet temp 20 degrees C
Step-31 TRANSFER 100% contents of TA-101 to Disposal-101, the transferred portion is named ACOH-Dist, with condenser outlet temp 20 degrees C
Step-32 TRANSFER contents of ST-102 to Drum-100, name transferred material N-HYDROXY-SOLUTION

Table 2: Operating Steps for Case Study - Stage 3

ing nodes separate a single node into multiple children nodes. Typically branching nodes involve process separations such as decantation, which split a multiphase mixture into an aqueous and organic phase. The node branching factor (b) is defined as the number of new states generated by the application of an operator. Converging nodes reduce the number of states, i.e. negative node-branching factor. For branching operators, the branching factor (b) varies from one to three for complex separations, e.g. phase spilt plus emission model.

The depth of the problem space can be defined as the number of operators required to reach the goal. Analysis of the case studies of batch processes in the pharmaceutical industry [Linninger et al, 1997] reveals that 30 to 60 operations per stage are typical. Therefore the search depth (d) can be stated as 30 to 60. When introducing intermediate objectives, i.e. chemical reaction, phase split, product purification through techniques like island search [Chakrabarti, 1986], (d) can be even lowered further.

For a depth-first searching algorithm, problem representation using the operation-based language gives rise to a complexity measure of the design space in the order $O(e^d)$ [Korf, 1992]. In case of blind task selection this number 40^{30} is still a fairly big number. For human assisted task selection with an edge-branching factor of three we obtain 3^{30}. This is figure is an upper bound of the problem complexity, since branching into three alternatives does not occur at each step.

The considerations above are only of semi-quantitative matter, since at each node there is an infinite number of operating states stemming from continuous temperature or pressure ranges. Nevertheless, they also illustrate that the operation-centered design language narrows the structural complexity of batch recipes considerably. Automatic synthesis methods for batch operating procedure are under development, e.g. [Ali et al, 1998]. Methods exploiting the operation-centered concepts for the automatic synthesis of waste treatment flowsheets are demonstrated in [Chakraborty and Linninger, 1998].

6 Applications and Results.

The BDK system is currently tested or used by several pharmaceutical companies and commercialized by HYPROTECH, Inc. The information integration requirements make BDK attractive as the main computer-aided process development environment. Results from several workshops with industrial participation also revealed the following conclusions and potential improvements to the traditional chemical process development procedure: Application of the system for industrial case studies showed a potential for drastic reductions in the product development cycle of pharmaceutical and specialty chemicals. Language-based design helps chemists and chemical engineers to focus on design decisions. Numerous alternatives processing schemes were easily generated. Prior to the use of BDK, developers preferentially only considered one option, because the detailed evolution of a production recipe was extremely time consuming. Electronic process development facilitates multidisciplinary collaboration among different teams involved in the process design procedure. Shared process knowledge, i.e. the BDK project concepts and databases, advances *concurrent engineering*. Adverse ecological decisions through improper material selection can be assessed in the crucial early decision phases. Changes of the basic chemical recipe may lead to effective implementation of pollution prevention strategies such as hazardous solvent replacement. Automatic process assessment allows to consider many different environmental as well as health and safety related aspects simultaneously. Finding the best trade-off between economic and ecological performance can thus be objectified.

Summary and Significance.
A comprehensive design methodology for the interactive computer-aided development of chemical batch processes for the manufacturing of pharmaceuticals was presented. Human developers can rapidly generate and evolve conceptual processing schemes through a hierarchical decision-making framework. The computer environment tracks project progress and offers knowledge-based assistance for particular subproblems like material and waste treatment selection. The "natural" language in use by chemists for the description of lab recipes was modeled to create an intuitive and expeditious man-machine interface. Consistency of chemical and physical operations was maintained by an "intelligent" model for process mixtures and their transformations combining quantitative and qualitative information. Use of an operation-centered design language drastically reduced the combinatorial complexity of the open-ended synthesis problem and opens new avenues for systematic computer-aided synthesis methodology for the generation of batch operating procedures.

References

[1] Ali et al, 1998: Synthesis of Batch Processing Schemes Based on a Means-Ends Analysis Nonmonotonic Planning Approach, Paper 10a02, to be presented at the AICHE Annual Meeting, 1998.

[2] Barnicki and Fair , 1990: A knowledge-based approach, Ind. Eng. Chem. Res., 29, 421-432, 1990.

[3] Chakrabarti, 1986: "Heuristic Search Through Islands," Artificial Intelligence 29, 339-34, 1986.

[4] Chakraborty and Linninger, 1998: *"Chemical and Destructive Treatment Options"*, University of Illinois at Chicago, CRB -Proposal, Chicago, IL, 1998.

[5] Chapman, 1987: Planning for conjunctive goals, Artificial Intelligence, 32 (3), 333-377, 1987.

[6] Charalambides *et al.*, 1993: C.C. Optimal Batch Process Synthesis, presented at the AIChE Annual Meeting, St. Louis, Missouri, USA, 1993.

[7] Currie and Tate, 1991: O-Plan – the open planning architecture, Artificial Intelligence, 52, p49, 1991.

[8] Dechema, 1990: Produktionsintegrierter Umweltschutz in der Chemischen Industrie, Frankfurt, 1990.

[9] Douglas, 1988: *Conceptual Design of Chemical Processes*, McGraw-Hill, USA (1988).

[10] Fikes and Nilson, 1971: "STRIPS: A new Approach to the application of theorem proving to problem solving", Artificial Intelligence, (2), 198, 1971.

[11] Forbes, 1984: Qualitative Process Theory, Artificial Intelligence, 24, 85-168, 1984.

[12] Grossman, 1996: I. E. "Mixed-Integer Optimization Techniques for Algorithmic Process Synthesis", in J. L. Anderson, editor, Process Synthesis, Advances In Chemical Engineering, Vol 23, Academic Press, 1996.

[13] Jarke and Marquardt, 1996: W. Design and Evaluation of Computer-Aided Process Modeling Tools, Aiche Symposium Series, Vol. 92 (312), pp 97- 109. 1996.

[14] Korf, 1992: "Linear-Space Best-First Search: Summary of Results," AAAI, 533-538, 1992.

[15] Kuipers, 1986: Qualitative Simulation, Artificial Intelligence (29), 289,1986.

[16] Kull and Hsu, 1991: PROVAL User's Manual, Merck & Co, Inc.

[17] Krendl, 1998: TechTool , Progress Report – Fall 1998, VAI, Linz Austria, 1998.

[18] Lichtenberger, 1997: "MCL Plus", M.SC. Thesis, Fachhochschule Hagenberg, Austria, 1997.

[19] Linninger, 1993: M-Project – A Material-Based Approach for the Solution of Industrial Design Problems, presented on the Conference of Mathematical Modelling and Computer Simulation of Industrial Processes, Steyr, Austria, 1993.

[20] Linninger et. al., 1995: *"DynEAF – A dynamic modeling tool for integrated electric steelmaking"*, Iron and Steel Engineer, 72, 43-53, 1995.

[21] Linninger *et al.*, 1996a: "BatchDesign-Kit – A Comprehensive Computer-aided Design Framework for Batch Process Development with Ecological Considerations", Paper 146h, *1996 AICHE Annual Meeting*, Chicago, IL, Nov, 1996.

[22] Linninger et al., 1996b: *"M-PROJECT – Organizing problem representation and modeling of steady state and dynamic processes"*, Comp. Chem. Eng., 20, S425 – 430, 1996.

[23] Linninger and Stephanopoulos, 1996: "Computer-Aided Waste Management of Pharmaceutical Wastes", Paper 23a , *AIChE Meeting*, New Orleans, LA, 1996.

[24] Linninger et al., 1997:*"BatchDesign-Kit* – A System for Integrated Develoment and Design of Batch Prcesses for Pharmaceuticals, Agricultural and Specialty Chemicals,", LISPE-MIT, Cambridge, MA, 1997.

[25] Linninger, 1997: *"Treatment Technology Knowledge"*, University of Illinois at Chicago, CRB – Proposal, Chicago, IL, 1997.

[26] Linninger et al, 1998: An Initiative for Integrated Computer-aided Process Engineering, *Proc. FOCACPO '98*, Snowbird, UT, July, 1998.

[27] Siirola and Rudd, 1971: "Computer-Aided Synthesis of Chemical Process Designs", Ind. Eng. Chem. Fund. 10, 353, 1971.

[28] Lakshmanan, R. and G. Stephanopoulos, 1988a: "Synthesis of Operating Procedures for Complete Chemical Plants – I. Hierarchical Structured Modeling For Nonlinear Planning", *Comp. Chem. Eng.*, 12, 985, 1988a.

[29] Lakshmanan, R. and G. Stephanopoulos, 1988b: "Synthesis of Operating Procedures for Complete Chemical Plants – II. A Non-linear Planning Topology", *Comp. Chem. Eng.*, 12, 1003, 1988b.

[30] Russel and Norvig, 1995: Artificial Intelligence – A Modern Approach. Prentice Hall, 1995.

[31] Wahnschafft, 1991: Split – A Separation Process Designer; Comp. Chem. Eng,. 15 (8).; pp 565 – 581, 1991.

[32] Westerberg, 1998: 'ASCEND – Chemical Engineering Modeling Environment', *http : //www.cs.cmu.edu/afs/cs.cmu.edu/project/ascend/home/ascend_at_cmu.htm*, 1998.

Assumption Grammars for Knowledge Based Systems

Veronica Dahl, Pablo Accuosto, Stephen Rochefort and Marius Scurtescu
School of Computing Science
Simon Fraser University
Burnaby, B.C., Canada V5A 1S6
Email: {eronica,srochefo,mas}@cs.sfu.ca, accuosto@chasque.apc.org
AND
Paul Tarau
Department of Computing Science
University of Moncton
Moncton, NB, Canada E1A 3E9
Email: tarau@info.umoncton.ca

Keywords: Logic grammars, natural language processing, intuitionistic and linear implication

Edited by: Vladimir A. Fomichov
Received: June 15, 1998 **Revised:** November 9, 1998 **Accepted:** November 12, 1998

In this paper[1] we examine some knowledge base uses of a recently developed logic grammar formalism, Assumption Grammars, particularly suitable for hypothetical reasoning. They are based on intuitionistic and linear implications scoped over the current continuation, which allows us to follow given branches of the computation under hypotheses that disappear when and if backtracking takes place. In previous research on using Assumption Grammars [TDF96] for processing natural language [DTL97], we showed that they allow abstracting time and consumer/producer relationships in complex natural language processing (relatives, anaphora), therefore resulting in more readable programs; and that they offer the flexibility of switching between data-driven or goal-driven reasoning, at no overhead in terms of either syntax or implementation. In this paper we argue that Assumption Grammars are also useful for knowledge-based systems, and that some of the techniques developed for natural language processing can naturally and usefully be transferred to knowledge based systems applications. Surprisingly, the technique may also be extended to coordination languages (Linda) with similar results.

1 Introduction

A logic grammar has rewriting rules, just as a context-free grammar, but its symbols are logic terms, i.e., they may include arguments. For instance:

```
noun(maria) --> [mary].
```

expresses that the word "mary" (square brackets denote a word, and their absence, a non-terminal grammar symbol) is to be analysed as a noun with representation "maria". Logic grammars can be conveniently implemented in Prolog: grammar rules are translated into Prolog rules which can then be executed for either recognition of sentences of the language specified, or (with some care) for generating sentences of the language specified.

Different types of logic grammars have evolved through the years, motivated in such concerns as ease of implementation, further expressive power, a view towards a general treatment of some language processing problems, such as coordination [DTMP95], or towards automating some part of the grammar writing process, such as the automatic construction of parse trees and internal representations. Generality and expressive power seem to have been the main concerns underlying all these efforts.

Assumption Grammars are a recently developed logic grammar formalism which we believe to be the best compromise to date between expressive and linguistic power. Their natural language applications, in particular to three crucial problems in Computational Linguistics (namely, free word order, anaphora and coordination) have been studied in [DFRT96, DTL97]. Two surprising results have been examined there: a) Assumption grammars allow a direct and efficient implementation of link grammars which are context-free like formalisms developed independently from logic grammars; and b) they offer the flexibility of switching between data-driven or goal-driven reasoning, at no overhead in terms of either syntax or implemen-

[1]This article is a revised version of an article in the Proceedings of the Third International Workshop on Applications of Natural Language to Information Systems, Vancouver, June 1997.

tation. The use of Assumption Grammars for natural language interfaces has been exemplified in [DTA+99].

Some of the problems for which Assumption Grammars have been used in natural language processing share features with some problems typical of those knowledge bases which must combine components into configurations, from stored rules on how to combine them, plus specific user requests. For instance, for checking that a proposed combination of modules is legal, we must allow the user to describe the modules in any order (rather than imposing the use of a predefined order). This is a similar problem, as we shall see, as that of free word order in natural languages.

Assumption Grammars are logic grammars augmented with a) hidden multiple accumulators, useful in particular to make the input and output strings invisible, and b) linear and intuitionistic implications scoped over the current continuation (i.e., over the remaining AND branch of the resolution), based on a variant of *linear logic* [Gir87] with good computational properties [Kop95], *affine* logic [TDF96].

The paper is organized as follows: Section 2 gives the background on logic grammars, Section 3 describes Assumption Grammars and presents some variants of linear and intuitionistic logic which we denote Timeless Assumptions; Section 4 presents some simple formal language examples treated in terms of Assumption Grammars, in order to make it easier to follow the knowledge base examples to be shown later; Section 5 discusses some uses of assumptions for constructive knowledge bases; Section 6 extends our operations in the context of distributed programming such as Linda programming [CG89].

2 Background on Logic Grammars

Logic grammars originated with A. Colmerauer's Metamorphosis Grammars [Col78]. They consist of rewriting rules where the non-terminal symbols may have arguments, and therefore rule application may involve unification. They can be considered a notational variant of logic programs, in which goal satisfaction is viewed as acceptance of a string by a grammar, and where string manipulation concerns are hidden from the user.

Extraposition Grammars (XGs) [Per81] allow the interspersing of skips on the left hand side, and these are routinely rewritten in their sequential order at the rightmost end of the rule, e.g.[2]:

```
rel_marker, skip(X), trace -->
    rel_pronoun, skip(X).
```

[2]We use our notation for consistency. Pereira's notation for skip(X) is written "..." on the left hand side and simply left implicit on the right.

In an XG rule, symbols on the left hand side following skips represent left-extraposed elements (e.g., "trace" above marks the position out of which the "noun_phrase" category is being moved in the relativization process).

XGs allow us to describe left-extraposition phenomena powerfully and concisely, and to arrange for the desired representations to be carried on to the positions from which something has been extraposed. Here is for instance Pereira's extraposition grammar for the language $\{a^n b^n c^n\}$:

```
s --> as, bs, cs.

as --> [].
as, skip(X), xb --> [a], as, skip(X).

bs --> [].
bs, skip(X), xc --> xb, [b], bs, skip(X).

cs --> [].
cs --> xc, [c], cs.
```

Discontinuous Grammars (DGs) [Dah89] generalize and include both metamorphosis and extraposition grammars, by allowing for skips to be arbitrarily rearranged (or duplicated, or deleted) by a rewrite rule. They have been used in particular for implementing adaptations of Chomskyan theories [Cho82]. Here is for instance a discontinuous grammar equivalent to the above extraposition grammar:

```
s --> as, bs, cs.

as --> [].
as --> xa, [a], as.

bs --> [].
xa, skip(X), bs --> skip(X), [b],
    bs, xb.

cs --> [].
xb, skip(X), cs --> skip(X), [c], cs.
```

In the first grammar, symbols such as xb can be considered as marks for b's which are being left-extraposed. In the second grammar, such marks can be seen as right-extraposed. While in this particular example our choice may just be a matter of personal preference, there may be naturalness reasons to prefer a right-extraposing formulation: some movement phenomena in natural language are more naturally viewed as right rather than left-extraposition, although they could perhaps be forced into left-extraposing formulations. Even when this forcing can take place, the resulting impossibility to distinguish between left and right movement creates some theoretical problems (e.g. in the more strict bounding of rightward, as opposed to leftward, movement– see [Ros67]). There may

also be efficiency reasons to prefer a right-extraposing formulation: in our implementation of DGs, the DG above works faster than the XG shown.

The need to refer to skipped substrings explicitly (whether in the XG original notation "...", or in the DG notation skip(X)) can be avoided altogether when using Assumption Grammars, which we introduce next.

3 Assumption Grammars

3.1 Description of the Formalism

Assumption Grammars are logic grammars augmented with a) linear and intuitionistic implications scoped over the current continuation, and b) hidden multiple accumulators, useful in particular to make the input and output strings invisible.

3.1.1 Linear and intuitionistic implications

Implications are additional information which is only available during the continuation, i.e., the remainder of the current proof. If declared to be intuitionisic (noted `assumei`), they can be used (noted `assumed`) an indefinite number of times. In contrast, linear implications (noted `assumel`) can be used at most once, and then they dissapear.

For instance, the Prolog query:

```
?- assumei(p(5)), assumed(p(X)),
    assumed(p(Y)).
```

instantiates both X and Y into 5; whereas the query:

```
?- assumel(p(5)), assumed(p(X)),
    assumed(p(Y)).
```

fails after instantiating X to 5, since p(5) is no longer available.

Both types of implication vanish upon backtracking, and both have a scoped version which will not concern us here.

The fact that our linear assumptions are usable *at most once* (i.e., *weakening* is allowed), together with implicit scoping over the current continuation, distinguishes *affine* linear logic from Girard's original framework where linear implications should be used *exactly* once.

We can see the `assumel/1`[3] and `assumei/1` built-ins as linear affine and respectively intuitionistic implication scoped over the current AND-continuation, i.e. having their assumptions available in future computations on the *same* resolution branch.

Notice that to disallow weakening, we can simply specify that the set of left over assumptions should be empty (this is easy through negation as failure, as our first example in Subsection 4.1 shows).

[3] /1 means that this predicate has one argument

3.1.2 Multiple Accumulators

Typically, logic grammars such as DCGs must be preprocessed into an equivalent logic program in which each grammar symbol transforms into a Prolog predicate with two additional arguments: one for carrying the input string (the sentence or sentence fragment yet to be analysed), and one for getting the output string (the substring that remains after the current analysis has consumed some of it while analysing it). These extra arguments can be thought of as accumulators. The idea has been extended to accomodate multiple such accumulators [Van89], for various uses. For the purposes of this article, only those accumulators used for the input and output string are relevant. In our implementation these are hidden, as in DCGs, which allows us to disregard the input and output string arguments except upon first call, but unlike DCGs, we do not need to preprocess the grammar. The input and output strings are accessible through a set of five (BinProlog) built-ins, allowing us to define a 'multi-stream' *phrase*/3 construct,

```
dcg_phrase(DcgStream, Axiom, Phrase)
```

that switches to the appropriate `DcgStream` and uses `Axiom` to process or generate `Phrase`. We refer to [TDF96] for their specification in term of linear assumptions.

3.2 Timeless Assumptions

¿From our work on natural language processing using Assumption Grammars, the notion of "timeless" assumptions evolved. Timeless assumptions allow the flexibility of consuming an assumption even if it has not yet been made (through either waiting for it to be made, or adding it as necessary). For instance, we can assume that noun phrases we analyse are potential antecedents for pronouns, and upon finding a pronoun, try a compatible (e.g., with same gender and number) potential antecedent as its referent. But in "Near her, Alice saw a butterfly", where the "antecedent" actually appears after its corresponding pronoun, we need to consume an assumption before it has been made.

We next examine several types of timeless assumptions, some of which, as we shall show, have interesting applications to knowledge-based systems.

3.2.1 Find-or-Wait Timeless Assumptions

Consuming this type of assumption can be done either by finding a matching (linear) assumption if it has been made, or if not, by waiting until it has been made. The definition, in terms of linear assumption and consumption, follows:

```
% Assumption:
```

```
% the assumption being made was expected
% by a previous consumption
+X:-assumed(waiting(X)), !.
% if there is no previous expectation of
% X, assume it linearly
+X:-assumel(X).

% Consumption:

% uses an assumption, and deletes it
% if linear
-X:- assumed(X), !.
% if the assumption has not yet been
% made, adds its expectation as an
% assumption
-X:- assumel(waiting(X)).
```

With these definitions, assumptions can be consumed after they are made, or if the program requires them to be consumed at a point in which they have not yet been made, they will be assumed to be "waiting" to be consumed (through "waiting(X)"), until they are actually made (at which point the consumption of the expectation of X amounts to the consumption of X itself). Terminal symbols will be noted as: `#word`. The usual operator definitions are left implicit.

3.2.2 Find-or-add Timeless Assumptions

The above definitions implement a "find now or later" interpretation of timeless assumptions. An alternative interpretation, "find-or-add", seems more useful for knowledge-based systems, in which rather than waiting for another part of the program to add (if not yet there) an assumption we want to consume, the assumption is added directly by the consuming primitive.

In this interpretation, assumption and consumption can be collapsed to a single operation, which can be implemented as follows:

```
% if X has been assumed, consume it
% and remove it
=X:- assumed(X), !.
% if not, assume X now
=X:- assumel(X).
```

It turns out that similar constructs can be used in the context of unification based Linda programming (see section 6).

3.2.3 Intuitionistic vs. Linear Timeless Assumptions

Both types of assumptions, find-or-wait and find-or-add, were exemplified above in terms of linear assumption primitives. Intuitionistic counterparts, in which assumptions can be consumed more than once, are readily implementable as well. Section 4 will show some of their uses. The intuitionistic counterparts are noted with "*" preceding the operator. The code follows.

```
% Intuitionistic Find-or-wait

% the assumption was expected, and is
% still assumed for future re-use
*+X:- assumed(waiting(X)), !,
    assumei(X).
% if there is no previous expectation
% of X, assume it intuitionistically
*+X:- assumei(X).

% Consumption:

% uses an assumption, which remains
% available for re-use
-X:- assumed(X), !.
% if the assumption has not yet been
% made, adds its expectation as a
% linear assumption
-X: assumel(waiting(X)).

% Intuitionistic Find-or-add

% if X has been assumed, use it
*=X:- assumed(X), !.
% if not, assume X intuitionistically
% now
*=X:- assumei(X).

% Linear Find-or-add

% if X has been assumed, unify with it
+=X:- assumed(X), !, assumel(X).
% if not, assume X linearly now
+=X:- assumel(X).
```

Note that only linear find-or-add assumptions are suitable for bi-directional information exchange through unification, as their intuitionistic counterpart *= has a different (copying) semantics corresponding to the usual implementation of intuitionistic universal quantification [Mil89, HM94].

4 Some Formal Language Examples

We shall now present some formal language examples of the use of Assumption Grammars, to pave the way to understanding the transfer of this paradigm to the Knowledge Based systems area. The examples show how assumption programming can result in higher level formulations which do not need to resort to extraneous concepts such as markers.

4.1 $a^n b^n c^n$ revisited

The following AG for the language $\{a^n b^n c^n\}$ is basically the same as the DG shown in Section 2, but does not need to refer to skips explicitly. Markers are now treated as linear assumptions.

```
s:- as, bs, cs, all_consumed.

as :- #a, assumel(xa), as.
as.

bs:- #b, assumed(xa), assumel(xb), bs.
bs.

cs:- #c, assumed(xb), cs.
cs.

all_consumed:- \+xa, \+xb.
```

Notice that a more declarative programming style results, in that we no longer need to refer to procedural notions such as left or right extraposition. If a marker xa has been assumed, then it can be consumed upon encountering a corresponding terminal symbol #b.

4.2 A more interesting example — scrambled $a^n b^n c^n$

If we want our strings to retain the same number of a's, b's and c's, but in any order, we can use linear assumptions in a data driven formulation, as follows:

```
a:- #a, assumed(as), !.
a:- #a, assumel(bs), assumel(cs).

b:- #b, assumed(bs), !.
b:- #b, assumel(as), assumel(cs).

c:- #c, assumed(cs), !.
c:- #c, assumel(as), assumel(bs).
```

To query, we state for instance:

```
go:- assumel(as), assumel(bs),
     assumel(cs), (b,a,c,b,a,c),
     all_consumed.
all_consumed:- \+as, \+bs,\+cs.
```

Thus, we start with only one assumption for each of as, bs, cs. Encountering the corresponding terminal symbol (respectively, a, b, c) results in deleting that expectation. But if a terminal, say a, is encountered after its assumption has been consumed, this signals the need to expect a corresponding b and c to appear, so we add the assumptions : assumel(bs), assumel(cs). Therefore the input sequence itself triggers the firing of the rules. In other words, we achieve data-driven behavior. It is interesting to note that this example could be rewritten using find-or-wait assumptions provided that we rewrite all_consumed/0 to ensure that no "waiting(X)" assumptions still existed (i.e., $\backslash{+}waiting(as)$, $\backslash{+}waiting(bs)$, $\backslash{+}waiting(cs)$).

Here is an even simpler AG formulation using find-or-wait assumption, also data driven:

```
a:- #a, +count.

b:- #b, -count, +next.

c:- #c, -next.

all_consumed:- \+count, \+next,
    \+waiting(count), \+waiting(next).

go:- (a,b,a,b,c,c), all_consumed.
```

The first assumption, *count*, is used to match a's with b's, while *next* matches b's with c's, to ensure the same number of each. Notice that by allowing weakening (i.e., by not requesting that all assumptions be consumed at the end) we can obtain a subset of the language, in which for instance the following query succeeds:

```
go:- (a,a,b,b).
```

5 Using Assumptions for Constructive Knowledge Bases

Some knowledge based systems can be viewed as sets of specifications for combining different components into desired configurations or structures. These are constructed on demand, according to the combination rules stored and the particular requirements in a user's query. A well-known example of such a system is R2 [McD82], which configures computer systems automatically, using production rules plus extensive specialized knowledge that results in a deterministic, bottom-up system. In this section we shall examine some interesting uses of linear and intuitionistic timeless assumptions for this type of knowledge bases: how to allow the user to propose a set of components for a configuration *in any order*, and ask the computer to check whether it is a legal one; how to update and retrieve data using reversible dictionary lookups, also taking integrity constraints into consideration; how to generate components around a basic configuration plus changing user's requirements; how to communicate knowledge through "timeless associative search", and how to provide flexible answers by relaxing some null-value provoking queries.

5.1 Assumption Grammars for Verifying Possible Configurations

Here we shall examine the problem of checking whether a configuration proposed by the user is a legal one or not. This particular problem can be expressed in grammatical terms, with the configuration in question being represented as a string of elements, or computer modules (forming a sentence), and an Assumption Grammar in charge of accepting the sentence as belonging to the language.

Let us suppose a configuration is made up of n occurrences of modules each named m1, m2, and m3, with $n \geq 1$, and of any occurrences of a set of optional components (say, o1 and o2). Then "legal" configurations can be described by strings such as:

```
m1,m2,o1,m2,m3,o2,m3,o1,m1
```

The same configuration is described by any permutation of the string, so the user should have the flexibility of entering it in any order. The following Assumption Grammar, modeled after our formal example in Subsection 4.1 , accepts such strings:

```
m1:- -m1s, !.
m1:- +m2s,+m3s.

m2:- -m2s, !.
m2:- +m1s, +m3s.

m3:- -m3s, !.
m3:- +m1s, +m2s.

o1.
o2.
```

To query, we state for instance:

```
go:- +m1s, +m2s, +m3s,
    (m1,m2,o1,m2,m3,o2,m3,o1,m1),
    all_consumed.
all_consumed:- \+m1s, \+m2s,\+m3s.
```

5.2 Timeless Assumptions for Update and Retrieval

Of course, for *generating*, rather than verifying, an acceptable configuration, the bottom-up style of grammar shown above, in which terminals drive the parse, is not possible. In this section we shall see that even for generation we can still fruitfully exploit the methodology of timeless assumptions. In the next section we shall examine a knowledge base system application of intuitionistic find-or-add assumptions. This application was also inspired in natural language processing techniques- in this case, dictionary lookup.

5.2.1 A folklore example: reversible dictionary lookup

An interesting implementation of reversible dictionary lookup (due to Alain Colmerauer) is possible through incomplete terms (i.e., terms containing variables). A dictionary is defined as a term with pairs of, say, French and English words, and ends with a variable, e.g.:

```
dictionary([[etoile,star],[lune,moon],
  [soleil,sun]|X]).
```

This notation allows us the following definition of "find":

```
find(F,E,[[F,E]|_]):- !.
find(F,E,[[F1,E1]|D]):- find(F,E,D).

find(F,E):-dictionary(D),find(F,E,D).
```

which will either:

a) find the English equivalent of a French word, e.g. by querying:

```
?- find(etoile,E).
```

b) find the French equivalent of an English word, e.g. by:

```
?- find(F,moon).
```

c) add a new French-English pair to the dictionary, e.g. by:

```
?- find(oiseau,bird).
```

Consultation is exemplified by:

```
?- dictionary(D), find(soleil,X,D),
    find(Y,moon,D),
    find(bonheur,happiness,D).

X=sun, Y=lune, D=[[etoile,star],
                  [lune,moon],
                  [soleil,sun],
                  [bonheur,happiness]|D]
```

Although this implementation is quite clever and concise, it requires carrying a dictionary explicitly, and recursively searching through the tree representing it. The state of the dictionary, moreover, needs to be explicitly passed on from call to call as an argument.

By using intuitionistic find-or-add assumptions, on the other hand, we can express the same idea as follows:

```
% Initialization
dictionary:- *= word(etoile,star),
    *= word(lune,moon),
    *=word(soleil,sun).

% Consultation
?- dictionary,*=word(soleil,X),
    *=word(Y,moon),
    *=word(bonheur,happiness).
```

5.2.2 Timeless assumptions for Database Consultation/Update

We can now adapt this idea for database consultation and update, this time taking integrity constraints into consideration. Notice that incomplete terms can be used also to update a database even when information is not complete. In the sample program below, for instance, we add a record with all the information known about musician "Koichi". The missing information can be completed (or not) at any time, simply by another timeless assumption containing it (e.g., =r(musician,koichi,cello,_)). In what follows, "r" stands for "record".

```
%Database initialization:

database:- *=r(musician,alan,piano,40),
   *=r(musician,jacques,violin,36).

% Consultation:

?- database,
   consult(r(musician,alan,Instrument,_)),
   consult(r(musician,koichi,_,35).


consult(r(musician,N,I,A)):-
  check(instrument,I), check(age,A),
  *=r(musician,N,I,A).
```

The integrity constraint checker must now be defined, taking care to check those values which are known and to postpone such a check for unknown values (by succeeding in the last clause of "check"), until when and if they are entered:

```
check(instrument,I):- nonvar(I), !,
    member(I,[violin,piano,cello]).
check(age,A):- nonvar(A),!,
    between(A,2,100).
...
check(_,_).
```

This formulation may, however, be too powerful in some cases. For instance, if the record with information about musician Koichi is incomplete as above, and someone asks for a musician playing zamponia aged 35, this retrieval attempt may result in incorrectly updating the record for Koichi so that he appears to play zamponia. To correct such side effects we can either use names as keys and ensure that the definitions check groundness of keys before proceeding, or we may want to clearly separate the updating function from the retrieving one, so that no careless retrieval attempts result in inadvertent and undesired updates as a side effect [4]

5.3 Find-or-add Assumptions for Module Generation

We can describe a (oversimplified, for explanatory purposes) basic computer configuration as a teletype, a screen and a central unit, the types of which can be either proposed non-deterministically by the program, or chosen explicitly by the user in his/her query. Intuitionistic find-or-add assumptions can represent this process very cleanly:

```
basic_configuration:- teletype(_),
    screen(_), central_unit(_).

teletype(X):- +=a(teletype(X)),
    member(X,[t1,t2]).
    % --order becomes important to
    % avoid spurious assumptions!
    % --i.e. member should come second
    % chooses a type of teletype or
    % checks a proposed one, then adds
    % it as an assumption

screen(X):- +=a(screen(X)),
    member(X,[s1,s2,s3]).

central_unit(X):- +=a(central_unit(X)),
    member(X,[cu1,cu2]).

print_configuration:-
  all_bound,
```

[4] Notice that undesirable side effects are not the consequence of using timeless assumptions but of having multiple uses for the same predicate. The same side effects can be seen in the pure Prolog formulation of the previous section. However, the multiple use version may be useful in some cases, for which the Assumption grammar formulation is, as seen, more convenient.

```
  +=a(teletype(T)), write('teletype= '),
  write(T), nl,
  +=a(screen(X)), write('screen= '),
  write(X), nl,
  +=a(central_unit(U)), write('CU= '),
  write(U),nl.

% makes sure only what's eventually
% assumed, counts at the end
all_bound:- \+ (assumed(a(X)), var(X)).
```

A user can now explicitly request, say, a teletype t2, and leave the remaining choices to the basic configuration primitive, e.g.:

```
?- +=a(teletype(t2)),
   basic_configuration,
   print_configuration.
```

5.4 Intuitionistic Assumptions for Progressive Update

Let us now suppose we are configuring computer systems such that a minimal memory size, say 2 MB, is included in any configuration, and that according to whether an optional module is of type o1, o2 or o3, say, the minimum memory requirement grows to 4, 5 or 6 MB respectively.

This situation can easily be represented through incomplete terms plus find-or-add intuitionistic assumptions. The notion of "at least X" can be represented through "successor" notation, i.e., numbers will have the form X, where X is a variable, or s(X), where X is in turn a number in successor notation. Thus the minimum requirement of 2MB will be expressed by the term s(s(X)), and whenever an o1 module is requested, it will "grow" to "at least 4", simply by unification of X with s(s(Y)). When all modules and their constraints on memory size have been added, the final number obtained can be "closed" by transforming its inner variable into 0, and printing it in ordinary notation.

The following program exemplifies. Definitions for "teletype" etc. are taken to be the same as before.

```
requirements(optional(o1)):-
 +=memory(s(s(s(s(_))))).        % 4 MB
requirements(optional(o2)):-
 +=memory(s(s(s(s(s(_)))))).     % 5 MB
requirements(optional(o2)):-
 +=memory(s(s(s(s(s(s(_))))))). % 6 MB

add_optional(X):-
  member(X,[o1,o2,o3]),
  +=optional(X),
  requirements(optional(X)).
```

A user's query can then be compiled into "at least X" notation, for instance if at least 3MB are required by the user, the compiled query might look like:

```
?- +=memory(s(s(_))),
   +=memory(s(s(s(_)))),
   add_optional(o2),
  +=memory(M).
```

This will result in the memory module first becoming "at least 2" (from the basic configuration requirements), then being updated to "at least 3" (from the user's explicit requirement), and finally to "at least 5" i.e. s(s(s(s(s(_))))) (from the implicit requirement attached to the user's request for o2). This number can then be closed and transformed to regular notation, also invisibly to the user.

6 Communicating knowledge through 'timeless associative search'

A frequent situation in which abstracting temporal order simplifies programming dramatically is using our concepts to express how communicating agents are able to exchange information, without necessarily taking producer/consumer roles.

With +X interpreted as an out/1[5] Linda transaction and -X interpreted as an non-blocking in/1[6] transaction, =X can be seen as a communication port found dynamically through associative search (see [DBT96] for an example of unification based associative blackboard system).

If the life-span of the two 'agents' intersects then they should 'connect' through this 'port' by proceeding to unification of the two assumptions and therefore information exchange takes place. This is very much what would happen if they were the same, globally visible shared logical variable. This concept is however stronger than global logical variables as *associative search* takes place on both sides as a prelude to the actual information exchange. At implementation level, the first who will not find the assumption (with the non-blocking in/1 operation), will add it, so that the second will consume it, provided, of course, that their life-span intersects. Note that our concepts are scalable to the case when the agents are on different computers - as implemented in the LogiMOO system [TDB96]. The resulting =/1 operation is implemented as:

```
% non-blocking variant of in/1,
% fails if not found
```

[5] puts a term on the shared blackboard

[6] retrives a matching term from the blackboard or fails otherewise

```
=X :-cin(X),!.
% puts term X to the shared blackboard
=X :-out(X).
```

To fully emulate =/1 in this context, each process would have to spawn a thread looping over a blocking rd/1 operation[7] which will have to update/propagate further instantiations of the same assumption.

7 Using Assumptions for More Flexible Answers

Assumptions can provide a flexible way of relaxing some null-value provoking queries in view of providing more flexible replies than those allowed by a third logical value.

Typically, when the extension of a reply is empty (e.g. "Which blue unicorns won the race?" with respect to a world in which all unicorns are green), a third logical value distinct from true and false (e.g. "undefined") is associated with the answer, so that the corresponding natural language answer ranges from a terse "There are no entities satisfying your query" to a more informative "There is a false presupposition in your query: there are in fact no blue unicorns", according to how sophisticated the presupposition-detection mechanisms in the particular system might be.

Interesting as is the three-valued logic that arises from the interaction of failed presuppositions with other parts of the sentence, in some cases the user might feel too rigidly treated- why does the system simply not check which green unicorns won the race instead of just pointing out the user's mistake about the colour?".

To provide more flexibility with such cases, we may include in the parser the ability to assume a relaxed version of the failed presupposition in its stead, and see whether with this relaxed version it can get the answer to be non-empty. In our example, the parser can assume, after having failed to find blue unicorns in order to check whether they won the race or not, that the question simply asks which unicorns won the race. Then when green ones are obtained as an answer, say, it can inform the user: "No blue unicorns exist in this world, but do you want to have the list of the green ones which won the race? In general, dropping qualifiers (adjectives, relative clauses) would be the candidate operations to try to elicit a non-empty, perhaps also interesting, response. By including the amended noun phrase as an assumption, the DB evaluator can easily identify what replies should be given as firm and what replies should be given as alternative answers to guessed sub-texts in the query.

[7] see [DBT96] for description of various unification based Linda operations

8 Conclusion

We have shown some typical cases where time has been abstracted as a parameter of change. This has narrowed the gap between their operational and their declarative semantics. As expected they have resulted in simpler and more readable programs as our operators have helped to express their 'knowledge content' more declaratively.

We have also shown how natural language processing techniques can fruitfully be transferred to intelligent database systems. In some cases, such as when verifying possible configurations within constructive worlds, this correspondence is quite direct: the problem domain can be formulated as a grammar, and problem instances can be expressed as queries to (automatically) prove that a proposed configuration is a sentence in the language described by that grammar.

In other cases, such as that of dictionary consultation/update, this correspondence is obtained by transferring into data base applications (via assumption operations, with the consequent gain in expressiveness) techniques that were originally developed for natural language processing.

Cross-fertilization goes the other way as well: in trying to apply Assumption Grammar methodologies to the database field, the need manifested itself for interesting new variants of hypothetical reasoning tools which had not been necessary so far for natural language applications alone.

Other than the variety of tools thus obtained in our incarnation of linear/intuitionistic logic, it must be noted that with respect to previous incarnations, ours is portable and runs on top of generic Prolog systems. This is a practical advantage over systems like Lolli or λProlog.

Other knowledge based applications can benefit from our approach, e.g. the combination of Assumption Grammars and linear/intuitionistic timeless assumption is a practical basis for building semantically clean object oriented extensions on top of Prolog. With this article we hope to stimulate further research into database applications of timeless assumption methodologies.

References

[CG89] N. Carriero and D. Gelernter. Linda in context. *CACM*, 32(4):444–458, 1989.

[Cho82] N. Chomsky. Lecture notes on government and binding, the Pisa lectures, 2nd (revised) ed. *Foris Publications*, 1982.

[Col78] A. Colmerauer. Metamorphosis grammars. In *Lecture Notes in Computer Science*, New York, N. Y., 1978. Springer-Verlag.

[Dah89] V. Dahl. Discontinuous grammars. *Computational Intelligence*, 5:161–179, 1989.

[DBT96] K. De Bosschere and P. Tarau. Blackboard-based Extensions in Prolog. *Software — Practice and Experience*, 26(1):49–69, January 1996.

[DFRT96] Veronica Dahl, Andrew Fall, Stephen Rochefort, and Paul Tarau. A Hypothetical Reasoning Framework for NL Processing. In *Proc. 8th IEEE International Conference on Tools with Artificial Intelligence*, Toulouse, France, November 1996.

[DTA+99] Veronica Dahl, Paul Tarau, Pablo Accuosto, Stephen Rochefort, and Marius Scurtescu. A Spanish Interface to LogiMOO: Towards Multilingual Virtual Worlds. *Informatica. An International Journal of Computing and Informatics*, (2), June 1999.

[DTL97] Veronica Dahl, Paul Tarau, and Renwei Li. Assumption Grammars for Processing Natural Language. In Lee Naish, editor, *Proceedings of the Fourteenth International Conference on Logic Programming*, pages 256–270, MIT press, 1997.

[DTMP95] V. Dahl, P. Tarau, L. Moreno, and M. Palomar. Treating Coordination with Datalog Grammars. In *Proceedings of the Joint COMPULOGNET/ELSNET/EAGLES Workshop on Computational Logic For Natural Language Processing*, April 1995.

[Gir87] J.-Y. Girard. Linear logic. *Theoretical Computer Science*, (50):1–102, 1987.

[HM94] Joshua S. Hodas and Dale Miller. Logic Programming in a Fragment of Intuitionistic Linear logic. *Journal of Information and Computation*, 110(2):327–365, May 1994.

[Kop95] A. P. Kopylov. Decidability of linear affine logic. In *Proceedings, Tenth Annual IEEE Symposium on Logic in Computer Science*, pages 496–504, San Diego, California, 26–29 June 1995. IEEE Computer Society Press.

[McD82] Y. McDermott. R1: A rule based configurer of computer systems. *Artificial Intelligence*, (19):39–88, 1982.

[Mil89] D. A. Miller. Lexical scoping as universal quantification. In Giorgio Levi and Maurizio Martelli, editors, *Proceedings of the Sixth International Conference on Logic Programming*, pages 268–283, Cambridge, Massachusetts London, England, 1989. MIT Press.

[Per81] F. Pereira. Extraposition grammars. *American Journal for Computational Linguistics*, 7:243–256, 1981.

[Ros67] J.R. Ross. Constraints on variables and syntax. *Ph.D. Thesis, MIT*, 1967.

[TDB96] Paul Tarau and Koen De Bosschere. Virtual World Brokerage with BinProlog and Netscape. In Paul Tarau, Andrew Davison, Koen De Bosschere, and Manuel Hermenegildo, editors, *Proceedings 1st Workshop on Logic Programming Tools for INTERNET Applications*, Bonn, September 1996. http://clement.info.umoncton.ca/ lpnet.

[TDF96] Paul Tarau, Veronica Dahl, and Andrew Fall. Backtrackable State with Linear Affine Implication and Assumption Grammars. In Joxan Jaffar and Roland H.C. Yap, editors, *Concurrency and Parallelism, Programming, Networking, and Security*, Lecture Notes in Computer Science 1179, pages 53–64, Singapore, December 1996. Springer.

[Van89] Peter Van Roy. A useful extension to Prolog's Definite Clause Grammar notation. *SIGPLAN notices*, 24(11):132–134, November 1989.

Text-to-Visual Speech Synthesis

R. Sahandi, D.S.G. Vine and J. Longster
Bournemouth University, Fern Barrow, Poole, BH12 5BB, UK
Tel.: +44 1202 595258, Fax: +44 1202 595314
Email: rsahandi@bournemouth.ac.uk

Keywords: Synthetic speech, Facial animation

Edited by: Anton P. Železnikar

Received: September 12, 1998 **Revised:** November 8, 1998 **Accepted:** November 24, 1998

The development of interactive multimedia systems, coupled with the advances in computer technology and high-speed communication systems, has made it possible for information to be presented to users more effectively and efficiently. Man-machine communication can be enhanced via the use of synthesised speech and computer animation in multimedia systems.
Whilst synthetic speech is potentially a more natural communication medium, it can be improved by the addition of an animated human face synchronised with the synthetic speech. This facial display provides a number of visual cues relating to what the speaker is saying and the speaker's emotional state. This increases intelligibility if the synthetic speech is degraded with noise, and allows knowledge transfer for the hearing-impaired, through lip-reading.
This paper provides an overview of existing speech synthesis and facial animation techniques, and discusses the limitations of each. The paper concludes with a description of a visual speech synthesis system developed at Bournemouth University, and a discussion of the audio-visual synchronisation issues.

1 Introduction

The convergence of telecommunications, broadcasting and computer systems is providing a uniform platform for the dissemination of information locally, nationally and world-wide. Interactive multimedia systems, coupled with advances in computer technology and high-speed communication systems, make it possible to present information with added impact. The inclusion of synthesised speech and facial animation in multimedia systems has enhanced the effectiveness of man-machine communication, in particular by supporting users with impaired hearing or vision (Lavagetto, 1995). When synthetic speech is combined with an animated speaking mouth, a highly adaptable user interface is produced.

Speech synthesis technology potentially provides a natural communication interface between computers and humans (Edwards 1991, Rudnicky *et al.* 1994). Speech synthesis alone has been used to assist people with impaired sight or speech (Alm *et al.* 1993, Blenkhorn 1992, Demasco & Foulds 1982). However, in spite of its undoubted value in specific areas, synthetic speech is not yet widely used. This is partly due to a perceived lack of naturalness, and a general complaint that it can sound 'robotic' and even 'sinister' (Michaelis & Wiggins, 1982). In addition to this, the intelligibility of synthetic speech can be compromised under noisy conditions. This problem, however, can be partially alleviated by accompanying the synthetic speech with visual information.

When talking-face displays are used in conjunction with synthetic speech, the user's attention is captured more readily, compared to text-only displays. It is found that users also make fewer errors, offer more comments and take more time when interacting with a talking-face (Sproull *et al.*, 1996). Audio-visual speech perception relies on two perceptual systems co-operating with each other. Facial expressions can indicate the speaker's emotional state, produce contrast, and emphasise significant words (Cassell *et al.*, 1994). The speaker's mouth movements provide an improved method for transfer of information to the listener, aiding their understanding of an utterance. Whilst hearing people may subconsciously use some lip-reading skills to assist their understanding of speech, lip-reading is essential for people with impaired hearing. During lip-reading, the visual modality of perception combines with, or even substitutes for, the auditory modality. Visual information, such as lip aperture, helps to indicate the speech signal's power, and is a clue as to how the speech stream is segmented (Lavagetto, 1995). This is particularly important for applications in noisy environments, where the intelligibility of both natural and synthetic speech can be degraded (Le Goff *et al.*, 1995). The additional information provided by an animated face, synchronised with synthetic speech, can increase the overall

intelligibility by a mean benefit of 11dB (Summerfield *et al.*, 1989).

Speech can be incorporated into user interfaces in the form of text-to-speech (TTS) synthesis. This allow new messages to be input and synthesised in real time. TTS synthesisers work by transforming an input text into a series of phonetic symbols, representing speech sounds (Edgington *et al.*, 1996a). This process requires a large number of mapping rules (Witten 1986, O'Malley 1990). Aspects of speech such as phoneme duration, intonation, pauses and emotions must be deduced from the input text and applied (Edgington *et al.*, 1996b), in order to make the synthesised speech sound more natural.

Text-to-Visual Speech Synthesis (TtVSS) systems allow an audio-visual synthesis of new messages in real time. This ability makes possible a whole range of interactive applications such as virtual boardrooms, long distance learning, and interactive multimedia kiosks (Parke & Waters, 1996).

This paper provides an overview of techniques used in both speech synthesis and facial animation, and describes a typical TtVSS system. It concludes with a discussion of a visual speech synthesis system developed at Bournemouth University, which uses a speech synthesis technique based on concatenating segments of human speech, coupled with a 2D talking face.

1.1 Speech Synthesis Techniques

Synthetic speech signals can be physically produced using one of two main speech synthesis techniques: *formant synthesis* and *synthesis by concatenation.*

Formant synthesis uses a set of parameters (such as pitch and voice quality) to control the output speech, allowing its characteristics to be manipulated easily (Witten, 1986). Formant synthesisers use look-up tables, which contain parametric representations for each possible speech sound. In order to produce continuous speech, the parameters corresponding to the desired speech sound sequence are interpolated, resulting in smooth-sounding speech output. Formant synthesis allows the easy creation of new synthetic voices. However, formant synthesis requires a painstaking analysis of real speech in order to derive the appropriate parameters. Also, formant synthesis has reached a point where steady improvements are difficult to achieve, and yet still has a tendency to sound robotic (Rudnicky *et al.*, 1994).

Synthesis by concatenation involves joining together segments of recorded human speech to form continuous speech (Bhaskararao 1994, Ozum & Bulut 1994, Witten 1986). Concatenation systems such as BT's Laureate (Edgington *et al.*, 1996b) are based on recordings of actual human voices, and so have the potential to sound more 'natural' than formant synthesis systems (Kraft & Portele, 1995). With concatenative techniques, smoothing techniques are essential to make the transitions between speech segments less noticeable (Cowley & Jones, 1992). Some widespread concatenative synthesis techniques include Pitch-Synchronous Overlap-Add (PSOLA) and Linear Predictive Coding (Edgington *et al.*, 1996b, Markel & Gray 1976).

Synthetic speech can be produced with different emotions (Arnott *et al.*, 1993). This conveys a substantial clue as to the intentions of the speaker, and is particularly useful in communication prostheses for the non-vocal (Alm *et al.*, 1993). Some general effects on speech can be associated with the primary emotions (Murray & Arnott, 1993). For instance, anger has a slightly faster speech rate with a very high average pitch, whilst disgust exhibits a much slower speech rate and a lower average pitch.

1.2 Facial Animation Techniques

Facial animation involves both facial modelling and animation control. Facial modelling is concerned with specifying a 3D model of a human head. Animation control involves the specification and control of facial motions into and between lip and facial expressions.

1.2.1 Modelling: Polygonal and Parametric Surfaces

The modelling process involves determining mathematical geometric descriptions that accurately capture the shape and appearance of a real face. Many models, such as the Parke model (Parke, 1974) produce rendered images based on polygonal surfaces. The visible surfaces of the face are modelled as networks of connected polygons. The vertices of the polygons are then manipulated to achieve the required lip and facial expressions. Graphic workstations can display and update polygonal surfaces easily, producing near real-time animation.

Whilst a network of polygons can approximate a curved surface, obtaining a virtually smooth surface entails large amounts of data. This problem can be overcome using parametric surfaces, such as B-splines, Bezier patches and NURBS (non-uniform rational spline surfaces), which are based on curves, or splines (Bartles *et al.*, 1987). Control values modify the defined surface by altering the shape of the splines.

1.2.2 Animation Control

Facial animation requires a means of controlling the surface of the model over time, to produce lip positions and expressions that correspond to the speech content. This involves manipulating the control points or polygon vertices, which may be regarded as *parameters.* The common methods for achieving this are described below:

Interpolation is the earliest (Parke, 1972) and most widely used technique. At set time intervals, or frames, facial positions called "key poses" are specified. A computer algorithm then calculates the intermediate images by interpolating the surface vertices or control points. This technique has the disadvantage that the range of expressions is limited to the number of key poses available.

Direct Parameterization (Parke, 1974) largely solved the problem of limited expressions, encountered with interpolation. These models are controlled by sets of parameters, such as *Lip Opening, Eyelid Opening*, etc. The required key poses are produced by varying these parameters. Techniques such as local region interpolations are used to generate the in-between frames. This approach is simple and efficient, with low data storage requirements, providing that the appropriate control parameters can be determined.

Muscle Based - Several models are based on simplified facial anatomy of bone structure, muscles and skin tissues. Platt and Badler (1981) created the first muscle based model, using a spring and mass system to simulate facial muscles and skin connectivity. Waters (1987) and later, Terzopoulus and Waters (1990) developed more complex and realistic muscle-based models. In general, the muscle-based models achieve greater realism and provide natural control, although they are computationally quite expensive.

Using these techniques, the key speech and expression events can be mapped onto the facial model, and the in-between frames produced to give the animation.

1.3 TtVSS system components

A text-driven visual speech synthesis system requires a means of synthesising speech for any text, and a mechanism for computing appropriate lip shapes and facial expressions. These expressions can be mapped onto a 3D facial model and synchronised with the synthetic speech. The components required for a TtVSS system are described below.

A *speech generation module* produces a phonetic script from the textual input and synthesises the speech. Both text-to-pronunciation rules and text-to-prosody rules are included; these deduce a phonemic representation and predict aspects of speech, such as timing and pitch. Any desired emotions in the speech are added by 'tagging' the input text.

A *facial animation module* consists of two interrelated parts: one to animate the lips and the other to generate the facial expressions. Both the lip shapes and facial expressions are automatically computed from the timed and tagged phoneme script produced by the speech generation module.

Early speech animation systems used a library of mouth positions, known as *visemes*, which correspond to one or more speech sounds (*phonemes*). The term 'viseme' is derived from the expression 'visual phoneme'. A more common approach uses sets of parameters, such as *lip opening*. The visemes are produced by mapping the phonemes to the parameter value sets (Pearce *et al.*, 1986). Transitions between visemes are achieved by interpolating the parameter values, using phoneme timing information to control the speed of change.

Facial expressions change continuously in order to emphasise and clarify the intended message. Expressions fall into two categories: *emotional expressions*, which are linked to the speaker's current feeling (e.g. anger), and *non-emotional expressions*, which accompany the content of speech (e.g. raising the eyebrows on an accented syllable). In many facial animation systems, emotional expressions are specified in terms of elementary muscle actions, using the Facial Action Coding System (FACS) (Ekman & Friesen, 1978). Pelachaud *et al.* (*to be published*) implemented a system that computes both emotional and non-emotional expressions, linked to the speaker's emotions and the semantic content of the intended message.

2 Development issues in a TtVSS system

A TtVSS system has been developed at Bournemouth University, in which a 2D cartoon face is synchronised with synthetic speech (Sahandi & Vine, 1997). A cartoon face is used, since an exact representation of the real world is not always required for the purposes of improving communication; indeed, a cartoon-like animation allows facial features to be stylised and exaggerated if necessary (Patterson & Willis, 1994).

The speech synthesiser is based on the concatenation of digitally recorded segments of human speech, such as syllables, which can be used to make up any new word (Sahandi & Vine, 1997). This speech synthesis method makes use of an inventory of stored CV, VC, CC and CVC waveforms (where *C* and *V* refer to *consonant* and *vowel*, respectively). In order to synthesise a word, appropriate waveforms are chosen from the inventory and a segmental duration value is assigned to each waveform. A variable portion of each waveform is extracted and concatenated, governed by the segmental duration parameter. Each portion is taken from the start of a stored waveform and finishes at a point t milliseconds later, where t is the segmental duration.

For text-to-speech synthesis, appropriate speech segments are predicted using a *Text-to-Segment Conversion Module*, which initially converts text into phonemes. These are then grouped together into CV, VC, CC or CVC units. Where possible, CV-VC pairs are converted to CVC segments (see Sahandi & Vine, 1997). A *Timing Module* supplies a duration param-

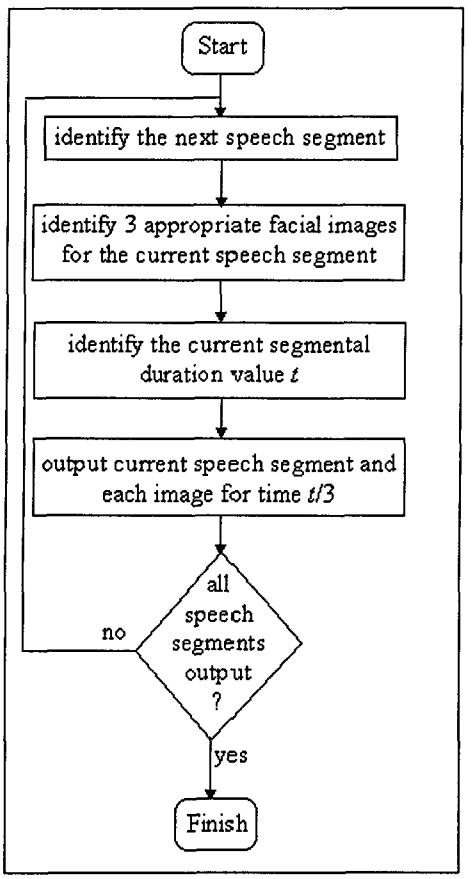


Figure 1: Synchronisation of animation with speech

eter for each speech segment, and this plays a large part in synchronising the facial images with the speech (Vine & Sahandi, 1997).

The animation consists of displaying a sequence of stored images, which must be synchronised with the speech output. Nine images are sufficient for animating lip movements for English words, since one image can correspond to several different sounds, e.g. the sounds /m/, /b/ and /p/ all share a similar mouth position (Benoit *et al.*, 1992). A sequence of images is pre-specified for each possible segment of speech. Since the duration for each speech segment can vary, the length of the image sequence displayed for each speech segment is adjustable. This is achieved by displaying a smaller or larger number of intermediate images as the speech segment is output. The number of intermediate images is proportional to the duration parameter that is supplied by the speech synthesiser module. At present, three lip images are specified for each speech segment, so that each different lip image is displayed repeatedly for a third of the specified duration (see Figure 1). The facial animation can be improved by increasing the number of different lip images used for each speech segment.

The current synchronisation system is somewhat inaccurate, because three different facial positions are always output per segment, regardless of the segmental duration value. However, in the particular speech synthesis method used, the segmental durations effectively control the sequence of phonemes that are extracted from each stored syllable and concatenated. For example, if the syllable /ca/ is assigned a longer duration (e.g. 220 ms), the entire /ca/ sound would form part of the final synthetic word. Alternatively, if /ca/ is given a shorter duration value (e.g. 45 ms), only the /c/ sound is included in the synthetic word. In this case, the pre-allocated viseme sequence /c/, /a/, /a/ would not be appropriate, because a visual /a/ would be displayed for two-thirds of the output time for /c/.

A potential solution under investigation is to label the start and end durations of every phoneme contained in each speech segment. For instance, for the stored speech segment /ca/, the /c/ sound could be labelled with a duration of 40 ms, whilst /a/ would have a duration of 140 ms. During the animation, if the duration assigned to /ca/ is less than 40 ms, only a visual /c/ should be displayed.

We hope that using the above synchronisation method, a more realistic synchronisation will be achieved between the synthetic speech and the facial animation.

3 Conclusion

Text-to-speech, when combined with facial animation, conveys messages more clearly and effectively. The perception of speech animation relies on the audio and visual perceptual systems co-operating with each other. This underlines the importance of accurate synchronisation between the audio and visual components. The facial animation can be exploited to some extent by the hearing-impaired, via lip-reading, whilst visually-impaired people benefit from the audio information. Whilst speech animation techniques are by no means perfect, they may soon become an indispensable part of multimedia interfaces.

References

[1] Alm, N., Todman, J., Elder, L. & Newell, A.F. (1993). Computer Aided Conversation for Severely Physically Impaired Non-speaking People. *In: Proc. INTERCHI '93, 24-29 April 1993,* p. 236-241.

[2] Arnott, J.L., Alm, N. & Murray, I.R. (1993). Enhancing a communication prosthesis with vocal emotion effects. *In: Proc. ESCA Workshop - Speech and Language Technology for Disabled Persons, KTH, Stockholm, Sweden, 31 May-2 June 1993,* p. 165-168.

[3] Bartles, R., Beatty, J., Barsky, B. (1987). *Introduction to Splines for Use in Computer Graphics and Geometric Modeling.* California, Morgan Kaufman.

[4] Benoit, C., Lallouache, T., Mohamadi, T. & Abry, C. (1992). A set of French visemes for visual speech synthesis. *In:* Bailly, G., Benoit, C. & Sawallis, T.R. (eds.) *Talking Machines - Theories, Models, and Designs*, London, Elsevier Science Publishers.

[5] Bhaskararao, P. (1994). Subphonemic Segment Inventories for Concatenative Speech Synthesis. *In:* Keller, E. (editor) *Fundamentals of Speech Synthesis and Speech Recognition.* Chichester, John Wiley & Sons, p. 69-85.

[6] Blenkhorn, P. (1992). A picture communicator for symbol users and/or speech-impaired people. *J. Medical Engineering & Technology*, 16, 6, p. 243-249.

[7] Cassell, J. *et al.* (1994). Animated Conversation: Rule-based Generation of Facial Expression, Gesture & Spoken Intonation for Multiple Conversational Agents. *ACM Computer Graphics 94*, p. 413-420.

[8] Cowley, C.K. & Jones, D.M. (1992). Synthesised or digitized? A guide to the use of computer speech. *Applied Ergonomics*, 23, 3, p. 172-176.

[9] Demasco, P. & Foulds, R. (1982). A New Horizon for Nonvocal Communication Devices. *BYTE*, September, p. 166-182.

[10] Edgington, M., Lowry, A., Jackson, P., Breen, A.P. & Minnis, S. (1996a). Overview of current text-to-speech techniques: Part 1 - text and linguistic analysis. *BT Technology Journal*, 14, 1, p. 68-83.

[11] Edgington, M., Lowry, A., Jackson, P., Breen, A.P. & Minnis, S. (1996b). Overview of current text-to-speech techniques: Part 2 - prosody and speech generation. *BT Technology Journal*, 14, 1, p. 84-99.

[12] Edwards, A.D.N. (1991). *Speech Synthesis - Technology for Disabled People.* London, Paul Chapman Publishing.

[13] Ekman, P. & Friesen, W.V. (1978). *Facial action coding system: A technique for the measurement of facial movement.* Consulting Psychologists Press, Palo Alto, Calif.

[14] Kraft, V. & Portele, T. (1995). Quality Evaluation of Five German Speech Synthesis Systems. *Acta Acustica*, 3, p. 351-365.

[15] Lavagetto, F. (1995). Converting Speech into Lip Movements: A Multimedia Telephone for Hard of Hearing People. *IEEE Trans. Rehabilitation Engineering*, 3, 1, p. 90-102.

[16] Le Goff, B., Guiard-Marigny, T. & Benoit, C. (1995). Read my lips...and my jaw! How intelligible are the components of the speaker's face? *In: Proc. Eurospeech '95*, p. 291-294.

[17] Markel, J.D. & Gray, A.H. (1976). *Linear Prediction of Speech.* New York, Springer-Verlag.

[18] Michaelis, P.R. & Wiggins, R.H. (1982). A human factors engineer's introduction to speech synthesisers. *In:* Badre, A. & Schneidermann, B. (eds.) *Directions in Human-Computer Interaction.*

[19] Murray, I.R. & Arnott, J.L. (1993). Toward the simulation of emotion in synthetic speech - A review of the literature on human vocal emotion. *J. Acoust. Soc. Am.*, 93, 2, p. 1097-1108.

[20] O'Malley, M.H. (1990). Text-To-Speech Conversion Technology. *IEEE Computer*, 23, 8, p. 17-23.

[21] Ozum, I.Y & Bulut, M.M. (1994). Knowledge Based Speech Synthesis by Concatenation of Phoneme Samples. *In: Proc. of the 7th Mediterranean Electrotechnical Conference, Antalya, Turkey, 12-14 April.* New York, IEEE, 1, p. 51-54.

[22] Parke, F.I. (1972). *Computer Generated Animation of Faces.* Master's thesis. University of Utah, UT. UTEC-CSc-72-120.

[23] Parke, F.I. (1974). *A Parametric Model for Human Faces.* PhD thesis, University of Utah, UT. UTEC-CSc-75-047.

[24] Parke, F.I. & Waters, K. (1996). *Computer Facial Animation.* Massachusetts, A K Peters Ltd.

[25] Patterson, J.W. & Willis, P.J. (1994). Computer Assisted Animation: 2D or not 2D? *The Computer Journal*, 37(10), p. 829-839.

[26] Pearce, A., Wyvill, B., Wyvill, G. & Hill, D. (1986). Speech and expression: A computer solution to face animation. *In:* Wein, M. & Kidd, E.M. (eds.) *Graphics Interface '86,* p. 136-140. Ontario, Canadian Man-Computer Communications Society.

[27] Pelachaud, C., Badler, N.I. & Steedman, M. (*to be published*). Generating facial expressions for speech. *Cognitive Science.*

[28] Platt, S.M. & Badler, N.I. (1981). Animating facial expressions. *Computer Graphics,* 15(3), p. 245-252.

[29] Rudnicky, A.I. *et al.* (1994). Survey of Current Speech Technology. *Communications of the ACM,* 37(3), p. 52-57.

[30] Sahandi, R. & Vine, D.S.G. (1997). A Concatenation Framework for Text-to-Speech Synthesis using CVC Syllables. *In: Proc. SALT Workshop on Evaluation in Speech and Language Technology, University of Sheffield, UK, June 17-18,* p.172-179.

[31] Sproull, L., Subramani, R., Kiesler, S., Walker J. & Waters, K. (1996). When the interface is a face. *Human Computer Interaction,* 11, p. 97-124.

[32] Summerfield, Q., MacLeod A., McGrath, M. & Brooke, M. (1989). Lips, teeth and the benefits of lipreading. *In:* Young, A.W. & Ellis, H.D. (eds). *Handbook of Research on Face Processing.* North-Holland, Amsterdam, Elsevier Science Publishers B. V.

[33] Terzopoulos, D. & Waters, K. (1990). Physically-based facial modeling, analysis, and animation. *Journal of Visualization and Computer Animation,* 1(4), p. 73-80.

[34] Vine, D.S.G. & Sahandi, M.R. (1997). Timing Databases for Concatenative Speech Synthesis. *In: Proc. DSP Scandinavia 97, Stockholm, Sweden, 3-4 June 1997,* p. 21-24.

[35] Waters, K. (1987). A muscle model for animating three-dimensional facial expressions. *Computer Graphics (SIGGRAPH '87),* 21(4), p. 17-24.

[36] Witten, I.H. (1986). *Making Computers Talk: An Introduction to Speech Synthesis.* London, Prentice-Hall International (UK) Ltd.

Theory of Restricted K-calcules as a Comprehensive Framework for Constructing Agent Communication Languages

Vladimir A. Fomichov
Faculty of Applied Mathematics, Moscow State Institute of Electronics and Mathematics
(Technical University), 109028 Moscow, Russia and
Department of Discrete Mathematics, Faculty of Mechanics and Mathematics
Lomonosov Moscow State University, 119899 Moscow, Russia
Tel: 007-095-930-9897; Fax: 007-095-939-2090
E-mail: vladfom@yahoo.com, vaf@nw.math.msu.su

Keywords: multi-agent system, communicative act, agent communication language, natural language, syntax, semantics, electronic commerce, Internet, FIPA ACL, integral formal semantics, restricted K-calculus, restricted standard K-language, ontology, semantic representation, structured meaning, discourse, knowledge representation, natural language processing, conceptual processing

Edited by: Anton P. Železnikar
Received: August 11, 1998 **Revised:** November 17, 1998 **Accepted:** November 24, 1998

For the first time, a comprehensive and flexible mathematical framework for constructing agent communication languages (ACLs) is suggested. That framework is the theory of restricted K-calcules and K-languages, or the RKCL-theory, published in Informatica, 1996, No. 1. The essence of the new framework is as follows. 10 operations on formal representations of conceptual structures are suggested. The contents of arbitrary messages and arbitrary communicative acts may be represented by the strings built out of primitive conceptual items by means of these 10 operations. The RKCL-theory possesses the expressive possibilities exceeding the expressive possibilities of the ACL published in October 1997 in Geneva by the international Federation of Intelligent Physical Agents (FIPA). The main advantages are, in particular, the possibilities to represent arbitrarily complicated goals (actions) and structured meanings of arbitrary discourses in natural language. The new framework enables us both to define syntax of ACLs and to formalize their semantics. The paper suggests also a little modified variant of the RKCL-theory without changing the expressive power of that theory.

1 Introduction

Multi-agent systems (MASs) is one of the most quickly progressing in the 1990s branches of Computer Science. The principal reason for permanently growing attention to this branch is as follows. One forecasts a very rapid development in the nearest future of electronic commerce, or E-commerce, based on largest possibilities of Internet (Thome & Schinzer, 1998), and MASs are considered as a key enabling technology for electronic commerce systems (Guilfoyle, Jeffcoate, & Stark, 1997).

In particular, one of the promising areas of using MASs is the travel industry. It is reasonable to construct intelligent agents (IAs) for this industry, because diverse implementations from various vendors (reserving airline tickets, booking rooms in hotels, renting cars, arranging for cultural programs) must interoperate and dynamically discover each other (FIPA, 1997).

It will be possible for numerous IAs (constructed by different centres, using different hardware and software) to interact effectively while solving tasks only in case when they will have a common communication language and proceed from generally accepted rules of communication. That is why one has developed in the recent years several standard agent communication languages (ACLs), first of all, the ACL developed in the ARPA Knowledge Sharing Effort (Genesereth, Fikes, et al., 1992; Finin et al., 1993; Genesereth, Singh, & Syed, 1994; Labrou, 1996; Finin, Labrou, & Mayfield, 1997; Labrou & Finin, 1997, 1998) and FIPA ACL (FIPA, 1997).

The ARPA ACL consists of three parts: a vocabulary, an "inner language" called KIF (or Knowledge Interchange Format), and an "outer language" called KQML (or Knowledge Query and Manipulation Language). KIF is a version of first order predicate language with various extensions for enhancing its expressiveness; it is used for representing diverse data and knowledge items. A message of ARPA ACL is an expression of the language KQML, where the "arguments" are terms or sentences in KIF formed by means of words from the ACL vocabulary (Genesereth, et al,

1994). The second language is a recent product of The Foundation for Intelligent Physical Agents being a non-profit association registered in 1995 in Geneva; its 35 corporate members represent 12 countries from all over the world (according to the data of October 1997).

A starting collection of ideas for developing ACLs was provided by the Theory of Speech Acts born in the works of J.L.Austin (Austin, 1962), P.F.Strawson (Strawson, 1964), J.R.Searle (Searle, 1969). These scholars essentially changed the angle of look in linguistics at sentences of natural language (NL). They contributed to a large recognition of the fact that it is unsufficient to be interested only in the truth values of assertions. People utter sentences having some beliefs about the conceptual systems of the dialogue partner and some intention concerning a change of the world's state (including a change in the conceptual systems of other people) as the result of uttering a sentence.

The ideas of the theory of speech acts became in the 1980s a part of the theory of natural-language-processing systems (NLPSs) and more general Artificial Intelligence (AI) theory. A considerable role in this process was played by the works (Perrault & Allen, 1980; Cohen & Levesque, 1985, 1990).

The main idea of ACLs may be figuratively described as sending each message in a special envelope where the intention of the message's sender is explicitly indicated. The role of such envelopes is played by special language constructions with a distinguished informational item (such as "confirm", "disconfirm", "propose", "request"). These informational items are called *performatives* and constitute *the library of communicative acts.* FIPA ACL contains 20 items of the kind. All intelligent agents in a multi-agent system (MAS) proceed from one reserved meaning of each performative and don't use these items in some other sense.

ACLs differ from the high-level programming languages in the following way: an ACL is not an interpreted or compiled language that is provided in some hardware platform or an abstract machine. The designers of intelligent agents are to elaborate the programs processing each construction with a performance.

In the end of the eighties and in the nineties, one has been able to observe one common tendency in several subfields of Computer Science: the attempts to find or to create an effective and comprehensive mathematical framework for the corresponding subfield. Let's mention, e.g., the works on extending the methods of logic and Situation Calculus for investigating the problems of robotics (Reiter, 1998; Sandewall, 1994, 1998) and the works on Episodic Logic (Hwang & Schubert, 1989; Hwang, 1992; Schubert & Hwang, 1993a-1993c) and Integral Formal Semantics of NL (Fomichov, 1988-1997) in the theory of NL processing systems.

But it appears that the field of designing ACLs possesses at the moment only an intermediate mathematical framework which doesn't give answers to several fundamental questions. The main reasons for this conclusion are indicated in the next section. First of all, the publications on ACLs indicate only very restricted formal means for expressing contents of messages transmitted by IAs. In many applications, IAs are to interchange messages representing meanings of NL-texts (business documents, malady records, contents of technical patents, etc.). However, the problem of formal representing meanings of arbitrary NL-texts isn't analysed. The purpose of this paper is to suggest an entirely new but natural and comprehensive mathematical framework for designing ACLs. It is the theory of restricted K-calculuses and restricted standard K-languages, or the RKCL-theory (Fomichov, 1996a, 1996b, 1997). The RKCL-theory determines a collection of such 10 operations on structured meanings (SMs) of NL-texts that, using primitive conceptual items as "blocks", we are able to build SMs of arbitrary NL-texts (including articles, textbooks, etc.) and arbitrary pieces of knowledge about the world.

The principal advantages of the new framework are as follows: (1) it enables us to build formal analogues of arbitrarily complicated goals and to represent knowledge about the world, in particular, terminological knowledge; (2) it provides the possibilities to reflect structured meanings of arbitrarily complicated NL-texts; (3) it allows us to represent communicative acts carried out by intelligent agents in the same way as it is done when one builds semantic representations (SRs) of texts mentioning such acts with the aid of the verbs "inform", "say", "write", think", etc.; (4) it is convenient for doing the same things as it would be convenient for the designers of MASs with the help of existing ACLs and, besides, a lot of other important things.

2 Task Statement

2.1 The Need of an Ontological Mathematical Framework for Constructing Agent Communication Languages

The analysis of (FIPA, 1997) shows that the current formal framework for constructing FIPA ACL consists of two components. Firstly, it is the theory of context-free grammars used for determining the syntax of messages. Secondly, it is a series of works on formal semantics of primitive communicative acts. These works stem from the Ph.D. dissertation (Sadek, 1991) on a first-order modal logic with identity, possessing many common features with the logical framework of Cohen and Levesque (1990).

Let's focus our discussion in this section on the first component - the formal framework for describing the structure of messages. It appears that the theory of context-free grammars provides only a *surface formal framework* for determining the structure of the ACL messages, and that the theory of agent communication acutely needs some other, *deep formal framework* for considering the same problem - how to describe messages of ACL.

That deep framework must be based on explicit enumerating all considered entities (both real and abstract) and on defining the admissible messages in a way enabling us to *formally distinguish* entities from basic considered classes. In other words, a new framework is to be based on *explicit ontology* of the considered world.

For instance, the ontology of first-order logic distinguishes three classes of entities: (a) various objects; (b) functions with arguments and values being such objects; (c) n-ary relationships ($n \geq 1$) on the universe of objects. Formulas of first-order logic describe the situations in the world with such ontology, but they are not the objects of that world. The ontology of terminological knowledge representation languages is larger and includes also a new class of entities - the class of concepts like "person", "computer", etc. Since at least 1980s, the use of the ontological approach has been a generally accepted practice in elaborating knowledge representation systems and formalisms destined for describing the structured meanings of NL-texts, i.e. for building semantic representations of NL-texts.

There are, in particular, the following main arguments in favour of elaborating a new, ontological mathematical framework for constructing ACLs.

Firstly, intelligent agents process the messages proceeding from the knowledge bases . *Secondly*, the messages may transfer the knowledge items from one agent to another. It may be repeated in this connection that one of the components of ARPA ACL is called Knowledge Interchange Format (Genesereth, Fikes, et al., 1992; Genesereth, Singh, & Syed, 1994; Finin, Labrou, & Mayfield, 1997).

Thirdly, one of the promising directions for designing multi-agent systems is the conceptual search and filtering of information available on Internet (Miyahara & Okamoto, 1997; Kang & Shi, 1997; Magnanelli, Erni, & Norrie, 1998). The ACL messages pertaining to this very large-scale application may be semantic representations (SRs) of NL-texts.

Fourthly, the messages being SRs of NL-discourses may include semantic items corresponding to the verbs "inform", "request", "ask", etc. These semantic items designate the communicative acts coinciding with or (more often) covering primitive communicative acts considered in KQML, FIPA ACL, etc.

Hence it would be convenient to represent in SRs of NL-texts the communicative acts meant by such verbs in the same form as the primitive communicative acts are represented in the *outer language* of an ACL.

2.2 The Task of the Research

As it was noted in Introduction, the impulse to the creation of the agent communication theory was given by the theory of speech acts being also a source of ideas for the theory of natural-language-processing systems (NLPSs). The progress in the design of NLPSs and in developing formal means for describing semantics of NL and modeling NL-dialogue led to raising the problem of building a mathematical theory of NL-communication, or mathematical linguocybernetics (Fomichov, 1988, 1992, 1993, 1994). According to (Fomichov, 1993a), a basic framework for constructing such a theory may be given by the theory of K-calcules and K-languages (1988, 1992, 1993b, 1994, 1995, 1996a, 1996b, 1997). The central part of the KCL-theory is the theory of restricted K-calcules and restricted standard K-languages, or the RKCL-theory (Fomichov, 1996a, 1996b, 1997). This theory provides large expressive possibilities of describing structured meanings both of sentences and of complicated discourses. It provides many essential advantages in comparison with the Discourse Representation Theory, Montague Grammar, Episodic Logic, Theory of Semantic Structures suggested by R.S.Jackendoff (Fomichov, 1996a, 1996b, 1997).

It appears that now it is an appropriate moment for the theory of NL processing to give a *second considerable impulse* to the development of the agent communication theory. The aim of giving this impulse is to introduce a widely applicable mathematical framework for constructing ACLs.

The purpose of this paper is to show that the RKCL-theory may be interpreted as a comprehensive and flexible mathematical framework for designing ACLs, because it enables us to represent: communicative acts; knowledge, beliefs, and intentions (goals) of intelligent agents; structured meanings of arbitrary messages in NL. Besides, this theory provides the possibility to realize and extend the known ways of formalizing semantics of ACLs.

Hence this paper provides a more extensive argumentation in favour of the conclusions drawn in (Fomichov, 1998).

This paper solves also some additional task: suggests a new variant of the RKCL-theory. For that, the rule with the number 9 (governing the use of universal and existential quantifiers) is stated in a new, simpler form. This doesn't influence the expressive power of restricted standard K-languages but makes easier the understanding of the rule.

3 Shortly about Restricted K-calcules and K-languages

3.1 Ontology of the RKCL-theory

The elaboration of the RKCL-theory was underlied by the idea to give the designers of NLPSs formal means being convenient for describing structured meanings (SMs) of practically arbitrary NL-texts and for representing knowledge about arbitrary application domains. That is why the ontology of the RKCL-theory is much richer than the ontology of first-order predicate logic or of terminological knowledge representation languages.

The RKCL-theory considers the following classes of basic entities:

1. The class of primitive objects, which includes, in particular, various things, events, situations, numbers, colours, logical connectives "not", "and", "or", universal and existential quantifiers. 2. The class of concepts qualifying various objects. 3. Semantic representations (SRs) of simple assertions (propositions) and of complicated narrative texts.

Compound entities are defined as ordered collections (systems, tuples) or sets consisting of primitive entities or of simpler compound entities. It is possible to consider relationships associating a finite number of primitive or compound entities. Functions with n arguments are interpreted as a subclass of the class of relationships being sets of $n+1$-tuples.

Each entity is associated with some string (a type) characterizing the class including this entity. For instance, the concept "a tourist group" may be associated with the type $\uparrow \{ins\}$, where the symbol *ins* is interpreted as the basic concept (a sort) "intelligent system", the symbol $\uparrow$ indicates that this entity is a concept, and $\{,\}$ tell that the concept qualifies the sets of intelligent systems.

Goals of intelligent systems are interpreted as a subclass of the class of all concepts. This approach will enable us to represent in similar forms, e.g., both the concept "an article on chemistry" and the goal "booking a single room in a three-star hotel" - as the strings

$$article * (Field, chemistry),$$

$$booking1 * (Object1, any\ room *$$

$$(Kind1, single)(Loc, any\ hotel *$$

$$(Kind2, three - star))).$$

3.2 The Structure of Simplified Conceptual Bases

The RKCL-theory makes the following discovery in linguistics and mathematical informatics: a system of such 10 operations on structured meanings (SMs) of NL-texts is found that, using primitive conceptual items as "blocks", we are able to build SMs of arbitrary NL-texts (including articles, textbooks, etc.) and arbitrary pieces of knowledge about the world. Such operations are called *quasilinguistic conceptual operations*. The formal side is stated very shortly below.

At the first step (consisting of a rather long sequence of auxiliary steps), the RKCL-theory defines a class of formal objects called *simplified conceptual bases (s.c.b.)*. Each s.c.b. B is a triple of the form

$$(S, Ct, Ql),$$

where S, Ct, Ql are some ordered collections (systems, tuples) of formal objects being mainly symbols or sets of symbols.

The system S is called a *simplified sort system (s.s.s.)*; it provides a set of symbols (sorts) denoting the most general concepts, distinguishes one of these sorts (called "sense of proposition"), and determines a hierarchy on the set of sorts.

The system Ct is called a *concept-object system*. It is destined for (a) indicating variables and basic informational items, (b) semantic classifying these variables and informational items, (c) distinguishing a subset of informational items designating various functions.

The system Ql is called a *system of quantifiers and logical connectives*. It formally distinguishes the sorts corresponding to logical connectives "not", "and', "or", logical quantifiers (universal quantifier and existential quantifier), and to the informational items associated with the words "some", "each", "all", "a few", etc. Besides, it distinguishes the informational item corresponding to the word "some" (when it is used for building word combinations in singular like "some house", "some method", etc.).

Each s.s.s. S is a triple of the form

$$(St, P, \leftarrow), \qquad (1)$$

where St is a finite set of symbols called sorts and designating the most general considered concepts; P is a distinguished sort "sense of proposition"; $\leftarrow$ is a binary relation on the set St interpreted in the following way: if s, t are sorts, and $s \leftarrow t$, then either s coincides with t or t designates a more general concept than s .

Example 3.1 Let

$St_0 = \{Prop, ins, event, st - sit, sit, nat, int, real, boolean, sp.ob, room.kind, hotel.kind, sint1, sint2, seq, snot, sbinlog, sext\}$,

where the elements of St_0 (considered as symbols) are interpreted as the designations, respectively, of the concepts "semantic representation of a declarative text", "intelligent system", "event" ("dynamic situation"), "static situation", "situation", "natural

number", "integer", "real number", "boolean value", "space object", "kind of a room", "kind of a hotel", "intensional quantifier of the first kind", "intensional quantifier of the second kind", "identity", "negation", "binary logical connective", "logical quantifier".

By intensional quantifiers of the first kind we'll understand the informational items corresponding to the meanings of the words "every", "any", "some", etc. in situations when they are parts of the word groups in singular ("each country", "some country", etc.). Intensional quantifiers of the second kind are informational items corresponding to the meanings of the words and expressions "all", "several", "a few", etc. A boolean value may be either "true" or "false".

Let's define the sets G_1, G_2, Gen as follows:
$G_1 = \{(s,s) | s \in St_0\}$,
$G_2 = \{(event, sit), (st-sit, sit), (nat, int), (nat, real), (int, real)\}$,
$Gen = G_1 \bigcup G_2$.

Then it is easy to show that the triple S_0 of the form $(St_0, Prop, Gen)$ is an s.s.s.

Each s.s.s. S determines some countable set of strings $Tp(S)$; these strings are called *types* and are used for qualifying entities of diverse kinds. The types with the beginning $\uparrow$ are associated with concepts. The types of the form $\{t\}$ correspond to sets, and the types of the form $(t_1, t_2, \ldots, t_n)$ characterize the n-tuples $< x_1, \ldots, x_n >$, where the entity x_i has the type t_i for $i = 1, \ldots, n$. Assume, e.g., that we consider the concept "a tourist group" and a concrete tourist Peter Jones. Then, taking into account the s.s.s. S_0, we would be able to associate with these two entities the types $\uparrow \{ins\}$ and ins, respectively. The set $Tp(S)$ includes three distinguished types: $[ent]$ ("entity" – the most general type), $[conc]$ ("concept"), and $[ob]$ ("object" as contrary to "concept").

A system Ct is called *a concept-object system (c.o.s.) for the s.s.s. S of the form* (1) iff Ct is a quadruple of the form

$$(X, V, tp, F), \tag{2}$$

where X is a countable set of strings (called *a primary universe*) used as elementary blocks for building knowledge modules and SRs of texts; St is a subset of X; V is a countable set of symbols called *variables;* F is a subset of X whose elements are called *functional symbols*; the component tp is a mapping from $X \bigcup V$ into the set of types $Tp = Tp(S)$, and several additional conditions are satisfied.

Example 3.2 Let's construct some c.o.s. Ct_0 for the s.s.s. S_0 determined in Example 3.1. Let N be the set consisting of all non-empty strings in the alphabet $\{'0', '1', \ldots, '9'\}$ with the beginning distinct from $'0'$. Let

$D_1 = \{person, scientist, tour-group, true, false, Slovenia, Ljubljana, Maribor, booking1, room, single, Kind1, hotel, three-star, Kind2, Cities, Number, Compos, Belong, Cause, Truth-value\}$,

$D_2 = \{::, every, any, all, \equiv, \neg, \wedge, \vee, \forall, \exists\}$,

$$X_0 = N \bigcup D_1 \bigcup D_2.$$

We will interpret the elements *Slovenia*, *Ljubljana*, *Maribor* as space objects; *booking*1 as a concept qualifying the events of booking the rooms in the hotels; the elements *room*, *hotel* as concepts designating space objects; *Kind*1 as a binary relation associating a room in a hotel and its category (single, double, etc.); *Kind*2 as a binary relation associating a hotel and its category (two-star, three-star, etc.). *Cities* is a designation of the function assigning to a country the set of all cities of that country; *Number* designates the function "The number of all elements of a finite set". *Compos* denotes a binary relation between a set of objects and a concept (primitive or compound) qualifying each element of that set; *Belong* is a designation of the relation "To belong to a set".

The element *Truth – value* corresponds to the function determined on SRs of simple assertions and declarative texts and assigning to such a SR the value "True" or "False". The element :: designates the informational item corresponding to the meaning of the word "some" in the word combinations in singular ("some person", "some tourist group", etc.); this item is called *the referential quantifier.* In our example, both this item and the items *every, any, all* form the set of *intensional quantifiers.* The items $\neg, \wedge, \vee$ are logical connectives; the elements $\forall, \exists$ are the universal and existential quantifiers.

Determine a mapping t_0 from X_0 into $Tp(S)$ by the following table:

z	$t_0(z)$
person, scientist	$\uparrow ins$
tour – group	$\uparrow \{ins\}$
true, false	*boolean*
Slovenia	*sp – ob*
Ljubljana, Maribor	*sp – ob*
*booking*1	$\uparrow$ *event*
room, hotel	$\uparrow$ *sp – ob*
single	*room – kind*
three – star	*hotel – kind*
*Kind*1	$\{(sp-ob, room-kind)\}$
*Kind*2	$\{(sp-ob, hotel-kind)\}$
Cities	$\{(sp-ob, \{sp-ob\})\}$
Number	$\{(\{[ent]\}, nat)\}$
Belong	$\{([ent], \{[ent]\})\}$
Cause	$\{(sit, sit)\}$
Compos	$\{(\{[ob]\}, [conc])\}$
Truth – value	$\{(Prop, boolean)\}$
::, *every, any*	*sint*1
all	*sint*2

$\equiv$	*seq*
$\neg$	*snot*
$\wedge, \vee$	*sbinlog*
$\forall, \exists$	*sext*
z from N	*nat*

Let $V_0 = \{x1, x2, x3, \ldots\}$, and for each v from V_0, $t_0(v) = [ent]$, where $[ent]$ is the most general type "entity". Let

$$F_0 = \{Cities, Number\},$$

$$Ct_0 = (X_0, V_0, t_0, F_0).$$

Then Ct_0 is a concept-object system.

If B is a s.c.b. of the form (S, Ct, Ql), S, Ct are the systems of the forms (1), (2), respectively, then the component Ql is called *a system of quantifiers and logical connectives (s.q.l.c.) for* S and Ct. The system Ql is some 7-tuple of the form

$$(int1, int2, ref, eq, neg, binlog, ext), \qquad (3)$$

where $int1, int2, eq, neg, binlog, ext$ are different sorts from St, ref is a distinguished element of the primary unverse X , and the interpretation of these elements is as follows. The sort $int1$ is the type of informational items coresponding to the words "each", "every", "some", "arbitrary", and so on, when these words are used for forming word expressions in singular ("each ship", "some ship", etc.). Similarly, $int2$ is associated as a type with informational items corresponding to the words "all", "several", "many", etc. The element ref corresponds to the meaning of the word "some" when it is used for building the word combinations in singular. The elements $eq, neg, binlog$ are to be considered as the types of the connectives "$\equiv$", $\neg$, $\wedge$, and $\vee$; the element ext is the type of the universal quantifier $\forall$ and existential quantifier $\exists$.

Example 3.3 The system Ql_0 of the form $(sint1, sint2, ::, seq, snot, sbinlog, sext)$ is a s.q.l.c. for the s.s.s. S_0 and the c.o.s. Ct_0 ; its referential quantifier is the symbol :: . As a result, the triple B_0 of the form (St_0, Ct_0, Ql_0) is a simplified conceptual basis (s.c.b.).

3.3 The Rules of Building Formulas

Each s.c.b. B determines three classes of formulas $Lrs = Lrs(B)$, Trs =Trs(B),Yrs=$Yrs(B)$ (l-formulas, t-formulas, anf y-formulas). The set $Lrs(B)$ is called the restricted standard K-language in the s.c.b. B. Its strings are convenient for building semantic representations (SRs) of NL-texts. Each formula from $Trs(B)$ has the form $d\,\&\,t$, where $d \in Lrs(B)$, $t \in Tp(B)$. The formulas from $Yrs(B)$ have the form $a_1\& \ldots a_n\&d$, where $a_1, \ldots, a_n, d \in Lrs(B)$, n is not the same for various d, and d is built out of $a_1, \ldots, a_n$ as out of "blocks" (some of these blocks may be slightly transformed) by applying only one time some inference rule. In order to determine for arbitrary s.c.b. B the classes of formulas Lrs, Trs, Yrs, a group of inference rules P[0], P[1], ... , P[10] is defined. The ordered pair $Krs(B) = (B, Rls)$, where Rls is the set consisting of all these rules, is called the restricted K-calculus in the s.c.b. B.

The rule P[0] provides an initial stock of l-formulas and t-formulas. Let $z \in X(B) \bigcup V(B)$, $t \in Tp(B)$, and $tp(z) = t$. Then, according to the rule P[0], $z \in Lrs(B)$, and the string of the form $z\&t \in Trs(B)$.

Let's regard (ignoring many details) the structure of l-formulas (called also K-strings) which can be obtained by applying any of the rules P[1], ... , P[10] at the last step of inferencing these formulas.

The rule P[1] allows us to build l-formulas of the form qc, where q is a semantic item corresponding to the meanings of such words and expressions as "some", "any", "arbitrary", "each", "all", "several", "many", etc. (such semantic items are called intensional quantifiers), and c is a designation (simple or compound) of a concept. Examples of l-formulas (K-strings) for P[1] as the last applied rule are as follows:

some person, some group $*$ $(Compos,$
$scientist)(Number, 12)$,
every person, any hotel $*$ $(Kind2,$
$three - star)(Loc, Ljubljana)$.

The rule P[2] is destined for constructing the strings of the form $f(a_1, \ldots, a_n)$, where f is a designation of a function, $n \geq 1$, $a_1, \ldots, a_n$ are l-formulas built with the help of any rules from the list P[0], ... , P[10]. The examples of l-formulas built with the help of P[2]:

$$Cities(Spain), Number(Cities(Spain)).$$

The rule P[3] enables us to build the strings of the form $(a_1 \equiv a_2)$, where a_1 and a_2 are l-formulas formed with the help of any rules from P[0], ... , P[10], and a_1 and a_2 represent the entities being homogeneous in some sense. Examples of K-strings for P[3]:

$(x1 \equiv Number(Cities(Spain))$,
$(Capital(Slovenia) \equiv Ljubljana)$.

The destination of the rule P[4] is, in particular, to build K-strings of the form $r(a_1, \ldots, a_n)$, where r is a designation of n-ary relation, $n \geq 1$, $a_1, \ldots, a_n$ are the K-strings formed with the aid of some rules from P[0], ... , P[10]. The examples of K-strings for P[4]:

$Belong(Maribor, Cities(Slovenia))$,
$Subset(Cities(Slovenia), Cities(Europe))$.

The rule P[5] allows us to construct the K-strings of the form $d : v$, where d is a K-string not including v, v is a variable, and some other conditions are satisfied. Using P[5], one can mark by variables in the SR of any NL-text: (a) the descriptions of diverse entities mentioned in the text (physical objects, events, concepts, etc.), (b) the SRs of sentences and of larger texts' fragments to which a reference is given in any part of a text. Examples of K-strings for P[5]:

$all\ scientist : Z1,$
$Less(Number(Cities(Belgium)),$
$1000) : P1.$

The rule P[5] provides the possibility to form SRs of texts in such a manner that these SRs reflect the referential structure of NL-texts. The examples illustrating this are considered below.

The rule P[6] permits to build the K-strings of the form $\neg d$, where d is a K-string satisfying a number of conditions. The examples of K-strings for P[6]:

$\neg scientist, \neg Belong(Bonn,$
$Cities(Belgium)).$

Here $\neg$ designates the connective "not".

Using the rule P[7], one can build the K-strings of the forms

$(a_1 \wedge a_2 \wedge \ldots \wedge a_n)$ or
$(a_1 \vee a_2 \vee \ldots \vee a_n),$

where $n > 1$, $a_1, \ldots, a_n$ are K-strings designating the entities which are homogeneous in some sense. In particular, $a_1, \ldots, a_n$ may be SRs of assertions (or propositions), descriptions of physical things, descriptions of sets consisting of things of the same kind, descriptions of concepts. The following strings are examples of K-strings (or l-formulas) for P[7]:

$(Finnland \vee Norway \vee Sweden),$
$(Belong((Namur \wedge Leuven \wedge$
$Gent), Cities(Belgium)) \wedge$
$\neg Belong(Bonn, Cities((Finnland \vee$
$Norway \vee Sweden)))).$

The destination of the rule P[8] is to build, in particular, K-strings of the form

$$c * (r_1, b_1) \ldots (r_n, b_n),$$

where c is an informational item from the primary universe X designating a concept, for $i = 1, \ldots, n$, r_i is a function with one argument or a binary relation,b_i designates a possible value of r_i for objects characterized by the concept c. The following expressions are examples of K-strings for P[8]:

$man * (First - name,' Peter')(Year - of -$
$studies, 4),$
$group * (Compos, student)(Number, 21),$
$turn * (Orientation, left).$

The rule P[9] enables us to build, in particular, the K-strings of the forms

$$\forall v(des)D \quad \text{and} \quad \exists v(des)D,$$

where $\forall$ is the universal quantifier, $\exists$ is the existential quantifier,des and D are K-strings, des is a designation of a prime concept ("person", "city", "integer", etc.) or of a compound concept ("integer greater than 200", etc.). D may be interpreted as a SR of an assertion with the variable v about any entity qualified by the concept des. The examples of K-strings for P[9] are as follows:

$\forall x1(nat) \exists x2(nat) Less(x1, x2),$
$\exists y(country * (Location, Europe))$
$Greater(Number(Cities(y)), 50).$

The rule P[10] allows us to construct, in particular, the K-strings of the form $\langle a_1, \ldots, a_n \rangle$, where $n > 1$, $a_1, \ldots, a_n$ are K-strings. The strings obtained with the help of P[10] at the last step of inference are interpreted as designations of n-tuples. The components of such n-tuples may be not only designations of numbers, things, but also SRs of assertions, designations of sets, concepts, etc. Using jointly P[10] and P[4], we can build the string

$Study1(\langle Agent1, some\ man *$
$(First - name,' Peter') \rangle, \langle Institution,$
$Moscow - State - University \rangle, \langle Enter - time,$
$\langle 1, September, 1995 \rangle \rangle),$

where the thematic roles $Agent1$, $Institution$, $Enter - time$ are explicitly represented.

Consider some expressive possibilities of restricted standard K-languages concerning representing knowledge: building formal definitions of concepts. Our examples pertain to biology and medicine. Suppose that E is a NL-expression, $Semp$ is a string of the restricted standard K-language in some s.c.b. and, besides, $Semp$ is a possible semantic representation (SR) of E. Then we'll say that $Semp$ is a K-representation (KR) of E.

Example 3.4 Let T1 = "All granulocytes are polymorphonuclear; that is, they have multilobed nuclei". Then T1 may have the following KR:

$(Property(anygranulocyte : x1,$
$polymorphonuclear) : P1 \wedge$
$Explanation(P1, If - then(Have1(\langle Agent1,$

$x1\rangle, \langle Object1, anynucleus : x2\rangle)),$
$Property(x2, multilobed)))).$

Here $P1$ is the variable marking the meaning of the first phrase of T1; the strings $Agent1, Object1$ designate thematic roles (or conceptual cases).

Example 3.5 If T2 = "Albumin - protein found in blood plasma" then T2 may have a KR of the form

$(albumin \equiv protein * (Location, some$
$plasma1 * (Location, some\, blood))).$

Example 3.6 Let T3 = "Type AB blood group - persons who possess types A and B isoantigens on red blood cells and no agglutinin in plasma". Then the following formula may be interpreted as a KR of T3:

$Definition(type - AB - blood - group, some\, set$
$*(Compos, person) : S1,$
$\forall x1(person) If - and - only - if(Belong1(x1,$
$S1), (Have1(\langle Agent1, x1\rangle,$
$\langle Object1, (some\, set * (Compos, type - A -$
$isoantigen) \wedge$
$some\, set * (Compos, type - B - isoantigen))\rangle,$
$\langle Location, some\, set * (Compos, cell1 *$
$(Part, some\, red - blood *$
$(Belong2, x1))))) \wedge$
$\neg Have1(\langle Agent1, x1\rangle, \langle Object1,$
$some\, set * (Compos, agglutinin)\rangle,$
$\langle Location, some\, set *$
$(Compos, plasma1 * (Belong2, x1)))))))).$

Here $\forall$ is the universal quantifier, the string *Compos* designates the binary relation "Qualitative composition of a set", the string *some* is interpreted in the RKCL-theory as the referential quantifier.

The considered examples show that restricted standard K-languages enable us, in particular, to describe the conceptual structure of texts with : (a) references to the meanings of phrases and larger parts of texts (Example 3.4), (b) compound designations of sets (Example 3.6), (c) definitions of terms (Examples 3.5, 3.6), (d) complicated designations of objects (Example 3.6), (e) generalized quantifiers ("arbitrary", "some", etc.). Besides, restricted K-languages provide the possibilities to describe the semantic structure of definitions, to build formal analogues of complicated concepts (Examples 3.5, 3.6), to mark by variables the designations of objects and sets of objects, to reflect thematic roles.

The advantages of the RKCL-theory in comparison with Montague Grammar, Discourse Representation Theory (Kamp & Reyle, 1993; van Eijck & Kamp, 1996), and Episodic Logic (Schubert & Hwang, 1989; Hwang, 1992; Hwang & Schubert, 1993a-1993c) are, in particular, the possibilities: (1) to distinguish in a formal way objects (physical things, events, etc.) and concepts qualifying these objects; (2) to build compound representations of concepts; (3) to distinguish in a formal manner objects and sets of objects, concepts and sets of concepts; (4) to build complicated representations of sets, sets of sets, etc.; (5) to describe set-theoretical relationships; (6) to describe effectively structured meanings (SMs) of discourses with references to the meanings of phrases and larger parts of discourses; (7) to describe SMs of sentences with the words "concept", "notion"; (8) to describe SMs of sentences where the logical connective "and" or "or" joins not the expressions-assertions but designations of things or sets or concepts; (9) to build complicated designations of objects and sets; (10) to consider non-traditional functions with arguments or/and values being sets of objects, of concepts, of texts' semantic representations, etc.; (11) to construct formal analogues of the meanings of infinitives with dependent words.

3.4 A New Formulation of the Rule of Using Logical Quantifiers

Let's consider a simpler formulation of the rule P[9] than its formulation in (Fomichov, 1996a). This rule determines the use of the universal quantifier and existential quantifier in the expressions of restricted standard K-languages. The difference of two formulations is as follows.

The formulation given in (Fomichov, 1996a) enables us to use variables of diverse types (more general and more concrete) in strings. E.g., let the considered s.c.b. B allows us to use the variables $sp1, sp2, \ldots$, denoting space objects (geographic objects, etc.) and the variables $n1, n2, \ldots$, denoting integers. If our basis B satisfies several additional requirements, then we can build a SR of the text T1 = "There is a country in Europe containing more than 50 cities" in the form

$$\exists sp1\,(country * (Loc, Europe))$$
$$\exists n1(integer) \qquad (4)$$
$$((Number(Cities(sp1)) \equiv n1)$$
$$\wedge Greater(n1, 50)).$$

The possibility to use the variables of diverse types is provided by the formulation of the rule P[9] which may seem to be rather complicated (because it is based on designations introduced before the definition of the rule P[1]).

The suggested new formulation of the rule P[9] allows us to employ only the most general variables with the universal and existential quantifiers. These are the variables of the type "entity" designated by the symbol [*ent*] . This doesn't restrict the expressive power of

the languages determined by the rules P[0], P[1], ... , P[9], P[10] but makes much simpler the formulation of the rule P[9].

The Definition 3.1 formulated below is destined for the joint use with several other definitions. One of them introduces a natural number i and a set $Numb$ of natural numbers. The number i is interpreted as the maximal ordered number of the considered rules. When we consider restricted K-calcules (as in this paper), 1 $\leq$i$\leq$ 10 Then $Numb$ is the intersection of the set $\{3,4,9\}$ and the set $\{1,\dots,i\}$. The set $Numb$ helps to select such formulas (corresponding to propositions) than it is possible to apply logical quantifiers to these formulas: the last rule used for constructing a formula of the kind must belong to the set $Numb$. $P(B)$ and $V(B)$ are designations of the distinguished sort "sense of proposition" and of the set of all variables of the s.c.b. B, respectively.

Definition 3.1 Denote by P[9] the assertion

"Let qex be the symbol $\forall$ or the symbol $\exists$, the string a belong to $L(B) \setminus V(B), P = P(B)$, k be a number from the set Numb , $a\&P$ belong to $T^k(B)$, v be a variable from $V(B), tp(v)$ be the most general type $[ent]$, a include the symbol v and don't include the substrings of the forms : $v, \forall v, \exists v$.
Besides, let the string des belong to $L(B) \setminus V(B)$, des don't include v, and there be such $m \in \{0,8\}$ and the type $u \in Tp(S)$ that the string $des\,\&\,u$ belong to $T^m(B)$.
Let B be the string of the form $qex\ v\ (des)\ a$. Then

$b \in L(B)$,
$b\&P \in T^9(B)$,
$qex\ \&\ v\ \&\ des\ \&\ a\ \&\ b \in Y^9(B)$ ".

Let's proceed below from a little modified variant of the RKCL-theory. It differs from the variant stated in (Fomichov, 1996a) only by the formulation of the rule P[9]: we'll use the Definition 3.1.

Example 3.7 There is such a s.c.b. B that it provides the variables $x1, x2, \dots$ of the type $[ent]$ and enables us to build with the help of the new rule P[9] and other rules the expression

$$\exists x1(country * (Loc, Europe))\exists x2\ (integer)((Number(Cities(x1)) \equiv x2) \wedge Greater(x2, 50)).$$

Obviously, this SR of the text T1 is equivalent to the SR of the form (4) .

4 Representing Contents of Messages by Means of Restricted Standard K-languages

The RKCL-theory possesses a lot of properties making restricted standard K-languages (RSK-languages) convenient for representing contents of arbitrary messages. Consider only some of them.

Property 1 It is possible to build formal representations of compound concepts. E.g., the string

$tour - group * (Number, 12)(Compos, scientist)$
may designate the concept (but not some concrete group) "a tourist group consisting of 12 scientists".

Property 2 RSK-languages provide large possibilities for representing knowledge items being definitions of concepts (see, in particular, Examples 3.5, 3.6).

Property 3 RSK-languages are convenient for building compound descriptions of various entities (things, events, sets). For this, one is to use the expresssions of the form

$$ref a * (r_1, b_1) \dots (r_n, b_n) : v,$$

where ref is a distinguished informational item called the referential quantifier and corresponding to the meaning of the word "some" (in the context of singular), a is a simple designation of a concept, $n = 1, 2, \dots, r_1, \dots, r_n$ are designations of binary relationships (in particular case, of functions), $b_1, \dots, b_n$ denote some objects or concepts, and v is a variable marking the described entity. E.g., a concrete tourist group consisting of 12 scientists may be represented by the string

$:: tour - group * (Number, 12)(Compos, scientist) : G1,$

where :: is the referential quantifier, and $G1$ marks that particular group.

Property 4 The property 2 allows us to model the use of the iota operator in the FIPA'97 ACL (FIPA, 1997). That language includes, in particular, the expression

$$(iota?x(UKPrimeMinister?x)),$$

denoting the person being the current Prime Minister of the United Kingdom. But the string

$$:: person * (Is, Prime - Minister(UK)) : x$$

of some RSK-language has the same meaning.

Property 5 RSK-languages enable us to build formal representations of simple and complicated goals. E.g., the goal "To book 12 single rooms in the three-star hotels of Ljubljana" may be represented by the string

$$Booking1 * (Object1, :: set*$$

$$(Number, 12)$$

$$(Compos, room * (Kind1, \quad (5)$$

$$single)(Loc, anyhotel * (Kind2,$$

$$three - star)(Loc, Ljubljana))))).$$

One can form complicated goals with the help of logical connectives "and", "or", "not". E.g., if $g1, g2, g3, g4$ are representations of simple goals like (5), then it would be possible to construct representations of compound goals

$((g1 \wedge g2) \vee (g3 \wedge \neg g4)),$
$(g1 \wedge g2 \wedge g3)$, etc.

For instance, the goal "To book 7 double rooms and 21 single rooms in the three-star hotels of Valencia and to rent a ship for a round travel of 35 tourists to the island Ibiza on September 23, 1999" may be formally represented as follows:

$(booking1 * (Object1, (:: set * (Number,$
$7)(Compos, room * (Kind1, double)(Loc,$
$any\ hotel * (Kind2, three - star)(Loc,$
$Valencia) : x1)) \wedge$
$:: set * (Number, 21)(Compos, room*$
$(Kind1, single)(Loc, x1))) \wedge$
$renting * (Object2, any\ ship : x2)$
$(Intention, round - travel * (Agent, :: set$
$*(Number, 35)(Compos, tourist))$
$(Instrument, x2)(Goal - space - object,$
$:: island * (Name,' Ibiza'))(Date,$
$(23, September, 1999))))).$

Property 6 Restricted standard K-languages allow us to reflect meanings of commands (orders) with the help of the strings of the form

$Order(A1, A2, g, t)$, where $A1, A2$ designate the intelligent agents giving an order (Sender) and receiving an order (Receiver), g is the representation of the goal to be achieved, and t denotes the moment of giving an order. It is possible, when, e.g., the primary unverse $X(B)$ of the considered s.c.b. B includes the sorts $Ins, event, moment$ ("intelligent system", "event", "moment of time") and the item $Order$, where $tp(Order) = \{(ins, ins, event, moment)\}$.

For example, the order of the agent $A1$ to the agent $A2$ to book 12 single rooms in the three-star hotels of Ljubljana may be represented by the K-string of the form

$Order(A1, A2, g1, t1)$, where $g1$ is the expression of the form (5), and $t1$ designates a corresponding moment.

Property 7 Suppose that the primary unverse $X(B)$ of a considered s.c.b. B includes the informational items $Prop, Question, Truth - value$, where $Prop$ is the distinguished sort "sense of proposition", $Question$ is associated with the type $\{([ent], Prop)\}$, and $Truth - value$ has the type $\{(Prop, boolean)\}$. Then the meanings of general questions (with the answer "Yes" or "No") may be represented in the form

$$Question(v, (v \equiv Truth - value(Form))),$$

where v is a variable, $Form$ is a K-string (l-formula) including the variable v . E.g., a possible SR of the question Q1 = "Is Maribor a city of Slovenia ?" is the string of some RSK-language

$Question(x1, (x1 \equiv Truth - value$
$(Belong(Maribor,$
$Cities(Slovenia))))).$

Property 8 For representing the meanings of special questions (including interrogative words), we can use the strings of the form

$Question(v_1, F(v_1, \ldots, v_n))$ and
$Question((v_1 \wedge \ldots v_n),$
$F(v_1, \ldots, v_n)),$

where the variables $v_1, \ldots, v_n$ denote the values to be found, and $F(v_1, \ldots, v_n)$ is a formula including $v_1, \ldots, v_n$ and representing an assertion. E.g., if Q2 = "What is the number of people in the tourist group from Gent ?" then the string

$Question(x2, (x2 \equiv Number(:: tour - group$
$*(Compos, tourist * (City, Gent)))))$

is a possible SR of Q2 in some RSK-language.

Property 9 It is the possibility and convenience of describing structured meanings (SMs) of arbitrary assertions. This possibility is substantially grounded in (Fomichov, 1992, 1994, 1995, 1996a). Besides, it was illustrated above. That is why consider only two more examples demonstrating the possibility to represent SMs of discourses with references to the meanings of the fragments being phrases or larger parts of the discourse.

Let T1 = "The president of the company, Mr. Smith believes that the firm A will sign this contract. And

the vice-president, Mrs. Jones, doesn't believe that". Choosing an appropriate s.c.b. B_1, we'll be able to build the following possible KR of T1:

$(Believe(\langle Agent1, :: man * (Isa,$
$president * (Organization, :: company1 :$
$x1))(Name,' Smith') : x2\rangle, \langle Time,$
$present\rangle, \langle Content, Sign1(\langle Agent2,$
$:: firm * (Name,' A') : x3\rangle, \langle$
$Object2, :: contract : x4\rangle, \langle$
$Time, future\rangle) : P1 \wedge$
$\neg Believe(\langle Agent, :: woman * (Isa,$
$vice - president * (Organization, x1))$
$(Name,' Jones') : x5\rangle,$
$\langle Content, P1\rangle, \langle Time, present\rangle)).$

If T2 = "Mr. Green had asked to send him 3 containers with ceramics. That request was fulfilled on June 10" then a possible KR of T2 may be constructed as follows:

$((Ask1(\langle Agent1, :: man * (Name,' Green')$
$: x1\rangle, \langle Request - role, sending * (Object1,$
$:: set * (Number, 3)(Compos, container*$
$(Contain, ceramics)) : x2\rangle, \langle Time, t1\rangle)$
$\wedge Before(t1, present)) : P1 \wedge$
$Fulfilled(:: request * (Description, P1) :$
$x3, t2) \wedge (t2 \equiv \langle 10, June, current-$
$year\rangle) \wedge Before(t2, present) \wedge$
$Before(t1, t2)).$

5 Representing Communicative Acts

The RKCL-theory provides large and flexible means for representing communicative acts. In particular, the structure of constructed strings may be very close to the structures used in the KQML and FIPA ACL. E.g., the request to tell who is the current Prime Minister of the United Kingdom may be represented in the FIPA ACL in accordance with (FIPA, 1997) by the expression

$(request : sender\, i : receiver\, j : content$
$(inform - ref : sender\, j : receiver\, i$
$: content\, (iota?x\, (UKPrimeMinister\, ?x))$
$: ontology\, world - politics : language\, sl)$
$: reply - with\, query1 : language\, sl).$

The same request may be represented by a string of some SRK-language as follows:

$:: communicative - act * (Kind,$
$request)(Sender, i)(Receiver, j)(Content,$
$Question(x1, (x1 \equiv$
$PrimeMinister(UK))))(Ontology, world-$
$politics)(Language1, RSK)$
$(Reply - with, query1)(Language2, RSK).$

It is easy to represent in the similar ways all communicative acts considered in (Finin et al., 1993; Genesereth, Singh, & Syed, 1994; FIPA, 1997; Labrou & Finin, 1998) with the help of SRK-languages.

6 Conclusions

The analysis indicates that the RKCL-theory may be interpreted as the first comprehensible and highly flexible mathematical framework for designing agent communication languages (ACLs). Its principal advantage consists in providing an effective universal approach to representing contents of messages corresponding to arbitrary NL-texts. Besides, the same language structure may be used for (a) forming goals and intentions, (b) representing communicative acts in the manners similar to the manners used in the KQML and FIPA ACL and (c) expressing knowledge about the world, in particular, as in terminological knowledge representation languages and as in KIF. It is easy to show that the RKCL-theory may be used also for formalizing semantics of ACLs in the same ways as indicated in (Labrou, 1996; Labrou & Finin, 1997, 1998; FIPA, 1997; Wooldridge, 1998).

Suppose that there are N different contents languages for each of N intelligent agents. Then we need N(N-1)/2 pairs of translators from one language to each other. But in case of using one Universal Contents Language (UCL), it is necessary to have only 2N translators from each particular contents language to the UCL and from the UCL to each of N particular contents languages. Hence the suggested mathematical framework promises to be economically attractive for constructing ACLs.

The analysis shows that the RKCL-theory provides a lot of new opportunities for the theory and design of natural-language-processing systems, in particular, as concerns: building underspecified semantic representations (SRs) and completely specified SRs of arbitrarily complicated sentences and discourses; formalizing lexical semantics; describing conceptual macrostructure of discourses (especially, in intelligent text summarization systems); developing interlingua; representing terminological knowledge, constructing ontologies of application domains; formalizing the processes of conceptual processing NL-discourses; NL-generation; modeling NL-dialogue in correspondence with the ideas of the speech acts theory.

McCarthy (1996) expressed the opinion that "human level intelligence requires reasoning about strategies of action, i.e. action programs". The RKCL-theory provides large opportunities for representing complicated sequences of actions depending on various conditions. That is why this theory may be useful for formalizing reasoning about strategies of action.

Acknowledgements

I am grateful to Dr. Michael Wooldridge for his help in getting the latest materials on agent communication languages. Thanks to the referees for the useful comments.

References

[1] Austin, J.L. (1962): *How to Do Things with Words.* Clarendon Press.

[2] Cohen, P.R. & Levesque, H.J. (1985): *Speech acts and rationality.* Proc. 23rd Annual Meeting of the Association for Computational Linguistics, Chicago, IL, 49-59.

[3] Cohen, P.R. & Levesque, H.J. (1990): *Intention is choice with commitment.* Artificial Intelligence, Vol. 42(2-3), 213-261.

[4] Cohen, P.R. & Levesque, H.J. (1995): *Communicative actions for artificial agents.* Proc. of the First International Conference on Multi-Agent Systems (ICMAS'95), San Francisco, California, 1995.

[5] Eijck, D.J.N. van & Kamp, H. (1996): *Representing Discourse in Context.* Amsterdam, The University of Amsterdam.

[6] Finin, et al. (1993): Finin, T., Weber, J., Wiederhold, G., Genesereth, M., Fritzson, R., McKay, D., Mc Guire, J., Pelavin, R., Shapiro, S., & Beck, C.: *Draft. Specification of the KQML. Agent-Communication Language.* The DARPA Knowledge Sharing Initiative, External Interfaces Working Group.

[7] Finin, T., Labrou, Y., & Mayfield, J. (1997): *KQML as an agent communication language.* In J. Bradshow (Ed.), Software Agents, MIT Press, Cambridge, 1997.

[8] FIPA (1997): *The Foundation for Intelligent Physical Agents. FIPA'97 Specification. Part 2 - Agent Communication Language.* The text refers to the specification dated 10th October 1997. Geneva, 1997.

[9] Fomichov, V.A. (1988):*Representing Information by Means of K-calcules.* Moscow: The Moscow Institute of Electronic Engineering Press (in Russian).

[10] Fomichov, V. (1992): *Mathematical models of natural-language-processing systems as cybernetic models of a new kind.* Cybernetica (Belgium), XXXV (1), 63-91.

[11] Fomichov, V.A. (1993a): *Towards a mathematical theory of natural-language communication.* Informatica. An Intern. J. of Computing and Informatics (Slovenia), 17(1), 21-34.

[12] Fomichov, V.A. (1993b): *K-calcules and K-languages as powerful formal means to design intelligent systems processing medical texts.* Cybernetica (Belgium), XXXVI (2), 161-182.

[13] Fomichov, V.A. (1994): *Integral Formal Semantics and the design of legal full-text databases.* Cybernetica, XXXVII (2), 145-177.

[14] Fomichov, V.A. (1995): *A variant of a Universal Metagrammar of Conceptual Structures. Algebraic systems of conceptual syntax (Invited talk).* In A. Nijholt, G. Scollo, R. Steetskamp (eds.), Algebraic Methods in Language Processing. Proc. of the Tenth Twente Workshop on Language Technology joint with First AMAST Workshop on Language Processing, University of Twente, Enschede, The Netherlands, December 6 - 8, 1995, 195 - 210.

[15] Fomichov, V.A. (1996a): *A mathematical model for describing structured items of conceptual level.* Informatica (Slovenia), 20(1), 5-32.

[16] Fomichov, V.A. (1996b): *An outline of a formal metagrammar for describing structured meanings of complicated discourses.* II Intern. Conf. on Mathematical Linguistics (ICML'96), Abstracts, Tarragona, 2 - 4 de maig de 1996. Grup de Recerca en Linguistica Matematica i Enginyeria del Llenguatge (GRLMC), Report 7/96, Universitat Rovira i Virgili, 1996, 31-32.

[17] Fomichov, V.A. (1997): *K-calcules and the problem of conceptual information retrieval in textual data bases.* Knowledge Transfer (Volume II). Edited by A.Behrooz. (Proc. of the Intern. Conf. "Knowledge Transfer - 1997 (KT97)", University of London, 14-16 July 1997), 52-58.

[18] Fomichov, V.A. (1998): *A comprehensive mathematical framework for designing agent communication languages.* Proc. of the International Conference "Information Society (IS'98)", Ljubljana, Slovenia, 6 - 7 October 1998, 81-84.

[19] Genesereth, M.R., Fikes, R.E., et al. (1992): *Knowledge Interchange Format Version 3 Reference Manual. Technical Report Logic-92-1,* Computer Science Department, Stanford University.

[20] Genesereth, M.R., Singh, N.P., & Syed, M.A. (1994): *Technical Report,* Computer Science Department, Stanford University.

[21] Guilfoyle, C., Jeffcoate, J., & Stark, H. (1997): *Agents on the Web: Catalyst for E-Commerce.* London: Ovum Ltd., April 1997.

[22] Hwang, C.H (1992): *A Logical Approach to Narrative Understanding. Ph.D. Dissertation,* U. of Alberta, Edmonton, Canada.

[23] Hwang, C. H. & Schubert, L.K (1993a): *Meeting the interlocking needs of LF-computation, deindexing, and inference: An organic approach to general NLU.* Proc. 13th Int. Joint Conf. on AI (IJCAI'93). Chambery, France, 28 Aug.-3 Sept. 1993.

[24] Hwang, C.H. & Schubert, L.K. (1993b): *Episodic Logic: a comprehensive, natural representation for language understanding.* Minds and Machines, 3, 381-419.

[25] Hwang, C.H. & Schubert, L.K. (1993c): *Episodic Logic: a situational logic for natural language processing.* In Aczel, Israel, Katagiri, & Peters (1993), 303-338.

[26] Kamp, H. & Reyle, U. (1993): *Introduction to Modeltheoretic Semantics of Natural Language, Formal Logic and Discourse Representation Theory.* Dordrecht: Kluwer Academic Publishers.

[27] Kang, X. & Shi, C. (1997): *MANIS - A multiagent system for network information services.* Informatica (Slovenia), 21(2), 193-200.

[28] Labrou, Y. (1996): *Semantics for an Agent Communication Language. Ph.D. thesis,* University of Maryland, Baltimore County, August 1996.

[29] Labrou, Y., & Finin, T. (1997): *Semantics and conversation for an agent comunication language.* Proc. of the 15th International Joint Conference on Artificial Intelligence (IJCAI-97), Nagoya, Japan, August 1997.

[30] Labrou, Y., & Finin, T. (1998): *Semantics for an Agent Communication Language.* In Agent Theories, Architectures, and Languages. Ed. by M.P.Singh, A.S.Rao, and M.J.Wooldridge. Lect. Notes in AI, Vol. 1365, Springer-Verlag, 209-214.

[31] McCarthy, J. (1996): *From here to human-level AI.* Proc. of the Fifth Intern. Conf. on Principles of Knowledge Representation and Reasoning (KR'96). San Francisco, California: Morgan Kaufman Publishers, Inc., 640-646.

[32] Magnanelli, M., Erni, A., & Norrie, M. (1998): *A Web agent for the maintenance of a database of academic contacts.* Special Issue on Natural Language Processing and Agent Communication. INFORMATICA. An International Journal of Computing and Informatics (Slovenia), Vol. 22, No. 4.

[33] Miyahara, K. & Okamoto, T. (1997): *Collaborative information filtering: towards distributed coopeartive work/learning environment.* Meeting the Challenge of the New Technologies. PEG97. Proceedings of the Eighth International PEG Conference, Sozopol, Bulgaria, May 30th - June 1st, 1997, 260-267.

[34] Perrault, C.R. & Allen, J.F. (1980): *A plan-based analysis of indirect speech acts.* Amer. J. Computational Linguistics, 6(3), 167-182.

[35] Reiter, R. (1998): *Sequential, temporal GOLOG.* Proc. of the Sixth International Conference on Principles of Knowledge Representation and Reasoning (KR'98). Edited by A.G.Cohn, L.Schubert, S.C.Shapiro. San Francisco, California: Morgan Kaufman Publishers, Inc., 547-556.

[36] Sadek, M.D. (1991): *Attitudes mentales et interaction rationelle: vers une theorie formelle de la communication. These de Doctorat Informatique,* Universite de Rennes 1, France.

[37] Sandewall, E. (1994): *Features and Fluents. A Systematic Approach to the Representation of Knowledge about Dynamical Systems.* Oxford University Press.

[38] Sandewall, E. (1998): *Logic based modelling of goal-directed behaviour.* Proc. of the Sixth Intern. Conf. on Principles of Knowledge Representation and Reasoning (KR'98). Edited by A.G.Cohn, L.Schubert, S.C.Shapiro. San Francisco, California: Morgan Kaufman Publishers, Inc., 304-315.

[39] Searle, J.R. (1969): *Speech Acts: An Essay in the Philosophy of Language.* Cambridge (England): Cambridge University Press.

[40] Schubert, L. K. & Hwang, C. H. (1989): *An episodic knowledge representation for narrative texts.* Proc. 1st Int. Conf. on Principles of Knowledge Repres. and Reasoning (KR'89), Toronto, Canada, 444-458.

[41] Strawson, P.F. (1964): *Intention and convention in speech acts.* The Philosophical Review, LXXXIII (4), 439-460.

[42] Thome, R., & Schinzer, H. (1998): *Market survey of electronic commerce.* Informatica (Slovenia), Vol. 22, No. 1, 11-19.

[43] Wooldridge, M. (1998): *Verifiable semantics for agent communication languages.* Proc. of the International Conference on Multi-Agent Systems (ICMAS-98), Paris, France, July 2-7, 1998, IEEE Press.

A Web Agent for the Maintenance of a Database of Academic Contacts

M. Magnanelli, A. Erni and M. Norrie
Institute for Information Systems, CH-8092 Zurich, Switzerland
E-mail: {magnanel, erni, norrie}@inf.ethz.ch

Keywords: agents, information extraction

Edited by: Vladimir A. Fomichov
Received: July 30, 1998 **Revised:** November 5, 1998 **Accepted:** November 8, 1998

We describe an Internet agent which continuously gathers information from Web documents in order to maintain a local database and ensure its currency. As a specific application, we detail an agent which maintains a database with information about academic contacts, their projects and publications. Agent operation is driven by a combination of an extraction profile which specifies what and how information is to be extracted from Web documents and a local database which specifies the particular contacts of interest. The agent detects both new and updated information and, when the confidence level is above a user-specified threshold, automatically updates the local database accordingly.

1 Introduction

The World Wide Web (WWW) has become a major source of information about all areas of interest. Users typically spend many hours searching not only for new Web documents, but also for updates to documents. For example, an academic may look for new technical reports, a financial analyst for new economic data and a computer enthusiast for new software products and versions. Further, it also requires significant time to download information and effort to organize it in a convenient form.

To assist users in the tasks of finding, fetching and working with information published in Web documents, we use an Internet agent to gather information and store it in a local client database, thereby allowing users to browse, query and process that information at their convenience. Agent operation is driven by a combination of an extraction profile specifying what and how information is to be extracted from Web documents and the local database specifying the particular entities of interest. Thus, the user accesses the local database system and it is the responsibility of the agent to maintain this database and ensure its currency.

While the approach is general and the agent dynamically configurable, here we use a specific application system, ACADEMIA, to describe the operation of the agent and the information extraction process. ACADEMIA is a system to support academics by automatically keeping track of contact information for other researchers – such as telephone numbers and email addresses – and also information on their projects and publications.

A problem of maintaining contact information on persons is keeping it current given that people change their jobs and occasionally organizations may change their telephone systems. While it is not essential that a user be notified of every change immediately, it would be useful if the next time a user wants to contact the person, the new information was available. One goal of ACADEMIA was therefore to provide a form of address book which is updated automatically based on information extracted from Web documents.

In the area of research, the Web is very important as a source of information on projects and publications. The immediacy of the Web as a publishing medium means that articles are often available on the Internet before they appear in conference proceedings, journals or books. Frequently, academics are interested in any new articles or reports by researchers in their field and may periodically go to their Web sites to check for updates to their publication lists and download new publications of interest. Therefore, a second goal of ACADEMIA was to maintain information on publications and projects of other researchers and update it automatically based again on information extracted from their Web documents.

A third goal of ACADEMIA was to maintain information on conferences of interest. This means that rather than going to Web sites with lists of conferences to look for information, a user can specify the names of conferences of interest and the agent will search for new Web pages for these conferences, extract the key information concerning the date, location and paper sub-

This paper is an extended version of the paper ACADEMIA: An Agent-Maintained Database based on Information Extraction from Web Documents, Proc. of EMCSR'98.

mission dates and store this information in the user's ACADEMIA database for later perusal.

The ACADEMIA agent runs in the background, periodically searching the Web. The frequency of the search is specified by the user. By creating an entry for each researcher of interest, the user effectively specifies the domain of interest and the agent uses this information to know who or what to search for.

The ACADEMIA agent is actually divided into two parts, the *person agent*, which looks for general information on people as well as their publications and projects and the *conference agent*, which looks for information about conferences.

The information extraction process is controlled by an *extraction profile* which specifies *how* information is to be extracted from Web documents based on a combination of keyword searches, term matching and proximity measures. Confidence measures are associated with the various extraction patterns, thereby allowing the agent to calculate reliability scores for extracted information items. These reliability scores, along with user-specified confidence thresholds, determine whether, for a given information item, the agent updates the database directly or consults the user.

ACADEMIA combines techniques developed in various research areas for extracting information from Web documents. In the database area, systems are being developed to allow querying over dynamically generated Web documents. For example, in (Hammer et.al. 1997), a language is proposed for specifying extraction patterns to enable structured objects to be constructed from information contained in HTML documents. These systems only work over fixed Web sites for which patterns have been specified. In contrast, our agent does not base extraction on fixed patterns and can extract information from any form of Web page.

Our agent does use pattern-based extraction mechanisms to extract information on publications and projects. However, the agent itself generates these patterns based on the structure of individual items found in repeating items such as HTML lists and tables. Similar techniques have been used, for example, in comparative shopping agents to extract information from specific sites of on-line stores (Doorenbos et.al. 1997). However, these agents use training keywords to learn the patterns of announced pages, while our agent finds pages by itself and does not need explicit training keywords.

Work such as (Menczer 1997) and (Armstrong et.al. 1995) use more complex retrieval functions, but focus mainly on presenting whole Web pages to the user. In our agent, the extraction profile drives retrieval by specifying how to find possible pages of interest and its main task is to then extract information from these pages.

Section 2 describes the components and operation of the ACADEMIA system and section 3 gives details of the extraction profile and the extraction process. Section 4 describes the specific process of extracting information on publications. Section 5 describes how confidence values are assigned to extracted facts. The forms of user interaction with the agent are discussed in section 6. Finally, concluding remarks are given in section 7.

2 ACADEMIA System

ACADEMIA is used to reduce the work of an academic in finding and updating information about other researchers. While we use this specific application to explain our general extraction mechanisms, we note that the general concepts of this system may be used in other applications and, with this aim in mind, the agent can be dynamically configured.

Figure 1 shows the components of the ACADEMIA system and the work flow between them.

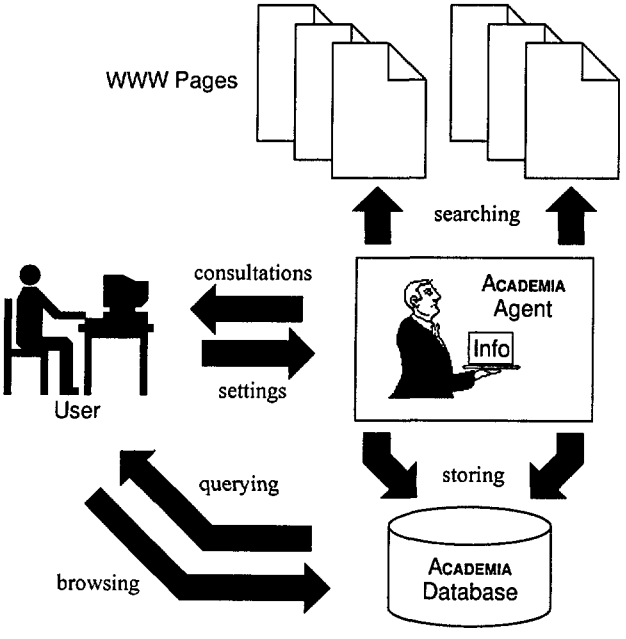


Figure 1: The components of ACADEMIA

The ACADEMIA database is implemented using the OMS object-oriented database management system (DBMS) described in (Norrie & Würgler 1997a, 1997b, Norrie 1993). The OMS system provides a graphical browser, a full query language and methods which support the various operations of the user such as the downloading of documents through ftp. Since the system also supports URLs as a base type, simple clicking on URLs can be used to view Web documents or send email through an Internet browser such as Netscape. Further, since a generic Web browser for OMS is available, the ACADEMIA database can also be accessed

through any general Internet browser such as Netscape Communicator or Internet Explorer.

The key contact information stored in the database consists of the name of a person and their WWW-address. The name is necessary to identify the person, while the address is used as a general starting point for the agent to search for updates. If the WWW-address of a person is unknown, the agent uses a search component which tries to find relevant Web documents for the person.

The database also stores general facts about persons such as title, address, photo and information about research activities including the titles of publications, URLs leading to abstracts or a publication file, project titles and URLs of pages which contain further information on such a project.

The user accesses the database directly to retrieve and process information on academic contacts. The ACADEMIA agent provides a *value-added* service by using information extracted from Web documents to maintain the database and ensure its currency. The agent may either update the database directly, or consult with the user as to whether or not it should perform the updates. The level of autonomy of the agent in terms of performing automatic updates is controlled by the user via a confidence threshold. If the confidence level of information found is above the specified threshold, the agent updates the database directly.

The confidence threshold is one of the settings specified by the user to control the agent. Other settings include specifying how often and when the agent should run. Clearly, this depends on the application in terms of the frequency of updates and how important it is that the database is current.

The extraction process of the agent is specified in terms of an extraction profile. For a given application system such as ACADEMIA, this profile is provided as part of the system. However, the user can adapt it to search for additional information. For example, if interested in *finger* information for a person, they could add an extraction pattern to the profile to look for this information. In section 3, the extraction profile is explained in detail.

An ACADEMIA agent runs in the background according to the periodicity specified by the user. It first reads the name and WWW-address of each person in the database to determine the search domain. If the agent does not find a WWW-address for a person, it tries to find it by using a search component which combines the MetaCrawler (Selberg & Etzioni 1997) and the Ahoy! Home Page Finder (Shakes et. al. 1997). In this case, the only search arguments are the first and last name of the person and, of course, it is not sure whether relevant documents will be found. The agent performs a search with the best pages returned by the search component and, in the case that information is found, later consults with the user who decides whether this information is reliable or not and should be stored in the database.

We note that we have experimented with various combinations of search engines. As the results were not very exact, we implemented our own search component which additionally tests the resulting pages for their usability.

Once one or more possible homepages have been located for a person, the agent starts to extract information from these pages and its referenced pages. Searching homepages is done in two basic ways – keyword-based and pattern-based search. In the case of keyword-based search, the agent searches for keywords as specified in the extraction profile. For each keyword, the user also defines a set of options, which tell the agent what information may be found in proximity to the keyword. For example, if a URL follows the keyword "www", it is likely to be a link to another homepage. Details of the extraction process and the format of the extraction profile are given in the next section. Although such keyword searching is relatively simple, it has proved to be effective and is used in ACADEMIA to find general information about a person and also potential links to pages containing publication lists or project descriptions.

Pattern-based search is used to find information about publications and projects. In most cases, this information is represented in lists and cannot be extracted by the keyword approach. For example, publications are frequently represented within Web documents as an HTML list with each item giving the authors, title, publication information and one or more URLs to download the document. The keywords "author" or "title" do not occur explicitly. Our agent therefore tries to detect a recurring pattern in the HTML page indicating the occurrence of such a list. This is based on HTML-commands around text items and the use of lists, tables and different fonts to structure information. Details of pattern-based search is given in section 4.

As mentioned, the agent starts searching from a potential home page and repeatedly follows interesting links to search for further information. Links are collected in a search list and can be of three types: links that likely lead to general information, to publications or to research topics. The agent searches each link using the corresponding search technique defined for each type of link. A future version of ACADEMIA will contain the possibility for the user to alter these search techniques.

After the search for one person, a confidence value (CV) is computed for each piece of information found based on the reliability of the extraction pattern used to find that item as specified in the extraction profile and the level of corroborating and/or contradictory information found. For example, if the same telephone number is found in several places, the level of confi-

dence will be high. However, if different phone numbers are found on different pages, the confidence will be low. These CVs can only be calculated at the end of the search since it is not possible to predict when and where items will be found.

Once the search is complete, the agent starts the interaction with the database. For every fact that has a CV greater than the user-specified threshold, the agent writes the fact in the database and records this action in a log which at any stage the user may access to examine the agent's actions. For facts which have CVs below the threshold, the agent will later consult the user who decides whether or not the fact will be stored. The agent stores the decisions of the user for future reference, thereby avoiding repeatedly asking the user the same questions. Whenever the user gains more confidence in the agent, he may reduce the threshold to give the agent greater autonomy.

Note that the agent stores in the database certain meta data about the Web documents searched such as the last update time and length of the page. The agent uses this meta data in future searches to detect whether or not a page has changed. Only when one or more pages related to a given person have changed, is it necessary to repeat the extraction process.

3 Extraction Profile

In this section, we describe the extraction profile in detail. To assist understanding, we explain some of the issues that led to our solutions by means of examples.

The profile consists of a set of extraction patterns each of which specifies a *keyword* to be searched for and also its significance in terms of the form of information to be extracted, the proximity of this information, any supporting keywords and an associated CV.

The general idea behind the extraction process based on these patterns is as follows. The agent first searches the page for a keyword of an extraction pattern. If a keyword is found, it indicates that information we seek may be located in the vicinity. There are several ways in which this information can be found. As an example, consider the case of looking for a person's phone number. The keyword "phone" indicates that a phone number may follow. Phone numbers usually consist of digits with a few special characters. Further, it is very likely that this number follows immediately after the keyword. With this background knowledge, it is not difficult to extract a string that is likely to be the phone number – if it exists. As another example consider finding the title of a person. If the keyword "prof" is found followed by the name of the person, it is likely to be the title of the person.

Such reasoning leads to the specification of the various extraction patterns. At any stage, the user can add new patterns or refine existing ones. Part of the extraction profile for the ACADEMIA agent is shown in table 1.

Each line in the profile corresponds to an extraction pattern which specifies a keyword along with additional information as to how an agent can determine if a fact of the appropriate form has been found and how to extract that fact. The extraction pattern is specified in terms of a number of attributes – some of which are optional. We start by explaining the attributes shown in table 1.

The first attribute after the keyword itself, *R*, specifies the type of information to be extracted. The agent distinguishes between two main categories of information – reference information and textual information. Textual information consists of facts to be extracted. It may be of several types. If `R=s`, a string value is to be extracted. If `R=b`, a Boolean value is returned indicating whether or not a specific term has been located. For example, in determining the title of a person, we simply want to know whether a specific designation such as "prof" appears in front of that person's name. Other types include email-addresses (`R=e`), dates (`R=d`) and images (`R=i`).

Reference information consists of links to other Web documents of interest and is used to direct the search. It can be one of the three main link types, general page (`l`), publication page (`p`) or research page (`r`), or it can be a link to a Web page where a "finger"-command is performed (`f`).

The next attribute, *in*, determines the position of the keyword in the Web document. The keyword may be found in usual text (`x`), in text belonging to a link (`k`), in the title of a page (`t`), in a header (`h`), generally inside of an HTML-command (`c`) or in a link reference (`l`). Thus, in row 3 of table 1, we specify that the keyword "publication" has to be found in a link – indicating that a reference to a Web document containing a list of publications has possibly been found.

D determines the locality of the information to be extracted in terms of the maximum distance (in characters) from the keyword. Positive numbers mean the information has to be searched after the keyword, while negative numbers mean the information has to occur before the keyword. 0 means the result can be in any distance from the keyword.

The attributes *N* and *FN* can be used to specify that the surname or forename of the person must appear in proximity to the keyword. For example, the first extraction pattern of table 1 specifies that the surname of the person must appear at most 10 characters from the keyword "prof". This is used to check that the designation belongs to the person whose information we are seeking and not some other person. The second extraction pattern specifies that the forename also occurs. A 0 for either of these attributes means that the corresponding name does not have to occur. *ML* and *XL* determine the minimum, respectively maximum,

Keyword	R	in	D	N	FN	ML	XL	C	Obj
prof	b	x	0	10	0	0	0	50	title
prof	b	x	0	24	10	0	0	100	title
publication	p	l	10	0	0	0	0	100	-

Table 1: Example 1 of the extraction profile

Keyword	R	in	D	CharSet	ML	XL	Obj
email	e	x	0	-	8	40	email
phone	s	x	6	+()0123456789/	8	22	phone
finger	f	k	0	-	0	0	finger

Table 2: Example 2 of the extraction profile

length of the resulting information for types string and email-address.

The CV associated with an extraction pattern is specified in attribute *C*. It is given in percent. More about CVs is given in section 5.

The last attribute, *Obj*, is used to tell the agent where the extracted information is to be stored in the database. In the case of the first extraction pattern of table 1, the extracted information (consisting of the keyword "prof") is stored as the `title` value of the appropriate person object. In the case of reference information, no information is stored and therefore *Obj* is unspecified.

In table 2, we show other optional attributes for specifying in more detail the format of values to be extracted. Note that we have omitted here the CVs which happen to all be 100.

CharSet specifies all possible characters allowed to occur in a string value. These are used in table 2 to specify expected forms of telephone numbers. Thus, to find a phone number, the agent looks for a string containing only those characters and starting within a distance of 6 characters after the keyword "phone". The keyword must occur in usual text. The result is a phone number with a length between 8 and 22.

In the case of email and finger information, the character set is unspecified, however the result types `e` and `f` indicate the format of values to be extracted. Thus, for an email address, the agent automatically looks for a string containing "@" and no spaces.

Another part of the extraction profile is given in table 3, showing how supporting keywords, and their required proximity to the main keyword, can be specified. *SK* is used to specify a second keyword that has to occur in the same Web document. *SKD* specifies the maximum distance of the second keyword from the first. 0 means any distance.

The first two extraction patterns in table 3 are used to get links to pages with general facts. The first specifies that keywords "home" and "page" must both occur in text that belongs to a link. The distance between them is not specified, but they have to occur in the same link text. The second line specifies that "work" should begin within 12 characters of "my" and both should appear in regular text. The extracted link to a further Web document of possible general interest has to be found within a maximum distance of 15 characters.

The third extraction pattern of table 3 is used to find a link to a page which may specifically contain information about projects. If keywords "project" and "lead" occur in usual text with "lead" appearing at most 20 characters before "project", a link within a distance of no more than 50 characters is assumed to be a possible link to a Web document listing projects. For example, a line of an HTML-page may contain the text: "Currently, I'm leading a project called Artemis". If, following Artemis, there is a link to the home page of this project, the extraction pattern would cause the agent to extract this link and search the resulting Web document for project information.

The specific values shown in the example tables were those which, during testing, led to good results. We chose them by analysing the forms of many Web pages containing relevant information and then adapting the proximity values based on experience. More detailed information about the extraction profile and the keyword-based extraction process can be found in (Magnanelli 1997).

The keyword-based approach led to good solutions for the extraction of general facts, but it was too weak to lead to many publications or research topics. The reason for this lies in the fact that, in most cases, information on publications, and possibly projects, is given in list form. These lists are structured, but the structure varies and therefore the agent must detect the structure dynamically. This was achieved through pattern-based extraction which is described in the next section.

Keyword	R	in	D	SK	SKD	ML	XL	C	Obj
home	l	k	0	page	0	0	0	100	-
my	l	x	15	work	12	0	0	100	-
project	r	x	50	lead	-20	0	0	100	-

Table 3: Example 3 of the extraction profile

4 Extraction of Publications

In this section, we describe pattern-based extraction by detailing how the agent extracts information about publications. We start by assuming that the agent has located a document (or part of a document) that is deemed likely to contain information on publications. The agent then looks for some form of pattern of repeated entries such as an HTML list or table structure.

If the agent detects a recurring pattern, it next tries to find the structure of the items. For example, assuming it finds what appears to be a list of items each of which gives information about a publication, it tries to determine where in the items information such as the author names or title appear. See figure 2 for an example of a publication list which, with respect to our agent operation, is ideal in terms of extracting information.

```
<H2>Object-Oriented Temporal Databases</H2>
<B>A. Steiner and M. C. Norrie.</B>
<I>Institute for Information Systems, ETH Zuerich.</I>
April 1997.<br>
Proc. 5th Int. Conf. on DASFAA'97, Melbourne, Australia
<br><br> Available files:
 [<a href="ftp:// ... /97c-dasfaa.abstract">abstract</a>]
 [<a href="ftp:// ... /97c-dasfaa.ps">postscript</a>]
<p>
<H2>New Programming Environment for Oberon</H2>
<B>J. Supcik and M. C. Norrie.</B>
<I>Institute for Information Systems, ETH Zuerich.</I>
March 1997.<br>
Proc. JMLC'97, Linz, Austria
<br><br> Available files:
 [<a href="ftp:// ... /97b-jmlc.abstract">abstract</a>]
 [<a href="ftp:// ... /97b-jmlc.ps">postscript</a>]
<p>
```

Figure 2: Part of an HTML publication list

Both entries shown contain the same structure of HTML-commands. We note that this case occurs seldomly as it may be that not all entries contain the same fields. For example, a particular publication entry may contain no date or proceedings. In fact, typically, the larger a pattern is, the more likely it is that there are small differences between several entries and our agent respects that.

Because of possible irregularities in items, we decided not to use every HTML-command to define the pattern of an entry. The tag "
", for example, never stands for a significant separation of two parts in an entry. Also the links beginning with "<a ...>" and ending with "" should not be used because not all publications may have referenced pages or postscript versions.

The agent first looks for the position of the name of the person in question. For example, in figure 2, we might look for publications of which Supcik is an author. "Supcik" is found between the HTML-tags <B> and </B>. The agent therefore assumes that, for every item, the names of the authors will be located in the corresponding part. If the person's name cannot be found, the agent assumes that the person may be the author of all entries. This situation arises relatively often when people give a list of publications on their home page without explicitly stating the authors. In this case, the confidence value will be lower, because there is no further evidence that this assumption is true.

The agent next tries to extract the title of the publication. For this, we used the observation that the title occurs towards the beginning of an item – either in the first or second position. Also, usually, the title contains at least twenty characters and seldomly contains punctuation characters such as commas. The agent uses these statistics to extract the titles from the entries. The associated CV will reflect the reliability that an extracted string is the title based on whether or not these various observations occur. Thus, if a title is less than twenty characters in length, it may still be extracted, but have a lower CV.

The agent also examines every link given in an entry. These links are also stored if they appear to be of interest to the user, for example postscript files or abstracts of the topic.

We note that, increasingly, HTML documents for publication lists are being generated automatically and this tends to produce more regular entries in terms of the HTML commands and this facilitates our extraction process and additionally leads to more reliable results.

Pattern-based extraction is also used to look for information on research projects. However, in this case, the ability of the agent to extract information is not as good as for publications. The main reason for this is that information given about research projects tends

to be less well-structured. In fact, it is often given as free text, without any heading or page name to indicate that it is indeed information about research topics or projects.

5 Confidence Values

Having described how the agent extracts information from Web documents, it is necessary to give some more information as to how the agent determines the reliability of this information.

As stated in the previous section, each extraction pattern has an associated CV which gives a measure of the reliability of an extracted information item in isolation. However, to compute an overall confidence measure of extracted information, the agent must consider the context in which the information was found and also the occurrence of any corroborating or conflicting information.

We refer to the CVs associated with extraction patterns as conditional possibilities, i.e.

$C(f|k)$ = the possibility that fact f occurs given a keyword k

The idea of CVs is adapted from certainty factors as defined in MYCIN (Buchanan & Shortliffe 1984). The main difference between their confidence values and ours is the range. Certainty factors in expert systems such as MYCIN normally range from -1 (complete uncertainty) to +1 (complete certainty). Our CVs range from 0 to infinity, because there exists no complete certainty whether a fact found is reliable. We let the user set a threshold which indicates the CV that a fact has to reach in order to seem reliable to the user.

Mathematically, there is no complete certainty but, in practice, we found many patterns which always led to reliable facts. Therefore, we decided to use percentage values for the CVs to indicate the reliability of the extraction pattern in terms of the number of cases in which the pattern leads to correct facts. For example, a value of 100 percent means that in all cases a fact extracted using that pattern is, in isolation, considered reliable. A value of 50 indicates that only in half of all cases does the associated extraction result in reliable information.

It is however not sufficient to consider only the effectiveness of a given extraction pattern in calculating the CV of a fact. For example, it may indeed be the case that an extraction pattern leads correctly to a phone number, but that we have a low confidence that the page being analyzed contains information about the person in question. Thus, the CV of a fact also depends on the CV associated with the context. Each page is therefore assigned a CV that indicates how likely it is that facts on this page belong to the processed person.

$C(p)$ = possibility that page p contains useful information on the focused person

An initial page obtained from a URL stored in the database is allocated a CV of 100 indicating that it is certain that this page contains information about the person in question.

To receive the final CV for an occurrence of a fact, we multiply the CV of the associated keyword extraction pattern with the one of the page in which it was found.

$$C(f) = C(p) \cdot C(f|k)$$

Of course, further pages that are processed as the search proceeds also get their CVs the same way: We multiply the CV of the keyword that led to the link to that page with the CV of the page where the link was found.

At the end of the search of all pages concerning one person, the same fact may be found more than once. In this case, the information is considered more reliable and as a CV for that fact we take the sum of the CVs of all equal facts found:

$$C(f) = \sum_i C(f_i)$$

Note that, with this rule, it is possible to get CVs above 100 percent.

Also, it is possible that similar, but unequal, facts may be found. In such a case, according to the similarity, we effectively merge the similar facts by selecting that with the highest CV.

$$C(f) = max_i C(f_i)$$

It can also be the case that certain facts are dependent on other facts. For example, the title of a publication and the title of an abstract associated with that publication should be the same. What happens if the extraction process yields unequal values? If the two values are similar, for example, the title of the abstract is a substring of that of the publication, the agent will assume that the title of the publication is the correct title and also associate that with the abstract. If the CV associated with the publication title is lower than that of the extracted abstract title, then the CV of the abstract title will also be updated. We introduce this example to show that the calculation of CVs where similar, but unequal, facts are found can be quite complicated and depends on many factors – including the context of the search and the type of the facts being extracted. It is beyond the scope of this paper to present all details of the confidence rules for ACADEMIA, however these are fully discussed in (Magnanelli 1997).

It is important to point out that we consider the information in the Web not only as free to use but also as true and updated. The agent is unable to detect that information is wrong in the case that the correct information is not available.

6 User Interaction

An important fact in using and trusting agents is the interaction with the user. It must be easy for the user to monitor the agent actions and control its level of autonomy in terms of automatically updating the database. We therefore provide a simple graphical interface through which the user may first install the agent and specify the various parameters such as the reliability threshold and the periodicity of execution. As the user gains more confidence in the agent, they may reduce the threshold value thereby giving the agent greater autonomy.

After the agent has executed, there may be some facts remaining that have a CV below the threshold. In this case, the agent must consult the user and it therefore opens an interaction window displaying the person name and the information found with its CV. The user then instructs the agent to either store or drop the fact. The agent records the decision of the user to avoid repeating the same questions in the future. This means that, initially, the user may be involved in a rather long and tedious interaction with the agent, but future consultations will be minimal and occur only when new people are entered in the database or Web pages are changed significantly.

In order that the user may inspect all actions taken by the agent, a detailed log-file is maintained. This log-file not only records all interactions with the database in terms of updates performed, but also keeps a record of the Web pages searched.

As it was mentioned previously, it is also possible for the user to inspect and update the extraction profile at any stage. This allows the user to manually change the CVs associated with extraction patterns or to add new extraction patterns so that the agent is able to find additional information. However, clearly, such direct updates are not generally recommended as they could have drastic consequences for the operation of the agent. Rather, updates of an extraction profile should generally be performed by an "application programmer" rather than an "end-user". We intend developing configuration tools which will assist users both in tailoring a specific application system such as ACADEMIA to their own requirements and in developing a new system for an entirely different application domain.

7 Conclusions

This work showed the important role that autonomous agents may take in the future in helping users to benefit from the wealth of information available on the Internet without requiring them to invest vast amounts of time and effort. Our experiments with ACADEMIA showed that it can handle a large amount of data in comparatively little time and produce good results. For example, in a test set of 26 persons, it extracted approximately 450 facts, of which more than 99 percent were correct. This involved the agent searching about 250 Web pages – a total of more than 2 megabytes.

In the future, we want to focus on improving the general operation of the agent by reducing the search time through better caching techniques for retrieved Web pages. Also, the system components have always to be improved in order to get better reliability of information extracted. One part will include more learning techniques to enable an agent to analyse and improve its performance. The agent must also be capable of analysing its own actions to give more feedback to the user. For example, the user will get information about keywords that seldomly lead to extracted items, and therefore, the user can adapt or even delete the keyword and its attributes from the extraction profile.

Additionally, we are currently generalizing the system to further support the rapid development of other applications systems through dynamic configuration. It is our aim to provide a system which can be configured for any type of search for information on the Web as well as on other resources such as databases or the Usenet.

References

[1] Armstrong R., Freitag D., Joachims T. & Mitchell T. (1995) WebWatcher: A Learning Apprentice for the World Wide Web. *Proc. Symp. on Information Gathering from Heterogeneous, Distributed Environments*, Stanford, California.

[2] Buchanan B.G. & Shortliffe E.H. (1984) *Rule-Based Expert Systems: The MYCIN Experiments of the Stanford Heuristic Programming Project*, Addison-Wesley, Reading, Massachusetts.

[3] Doorenbos R., Etzioni O. & Weld D. (1997) A Scalable Comparison-Shopping Agent for the World-Wide Web. *Proc. of the 1st Intl. Conference on Autonomous Agents*, Marina del Rey, California.

[4] Hammer J., Garcia-Molina H., Cho J., Aranha R. & Crespo A. (1997) Extracting Semistructured Information from the Web. Technical report, Department of Computer Science, Stanford University, Stanford, California.

[5] Magnanelli M. (1997) Maintenance of a Contact Database by an Internet Agent. Master's thesis, Institute for Information Systems, ETH Zürich.

[6] Menczer F. (1997) ARACHNID: Adaptive Retrieval Agents Choosing Heuristic Neighborhoods for Information Discovery. *Machine Learning: Proc. of the 14th International Conference*, San Francisco, California.

[7] Norrie M.C. & Würgler A. (1997a) OM Framework for Object-Oriented Data Management. *INFORMATIK Journal of the Swiss Informaticians Society*, June.

[8] Norrie M.C. & Würgler A. (1997b) OMS Object-Oriented Data Management System. Technical report, Institute for Information Systems, ETH Zürich.

[9] Norrie M.C. (1993) An Extended Entity-Relationship Approach to Data Management in Object-Oriented Systems. *Proc. of the 12th Intl. Conference on Entity-Relationship Approach*, Arlington, Texas.

[10] Selberg E. & Etzioni O. (1997) The Metacrawler Architecture for Resource Aggregation on the Web. *IEEE Expert*, 12(1), p. 8-14.

[11] Shakes J., Langheinrich M. & Etzioni O. (1997) Dynamic Reference Sifting: A Case Study in the Homepage Domain. *Proc. of the Sixth Intl. World Wide Web Conference*, Santa Clara, California.

A Multi-Agent Architecture Based on Active Mental Entities

Pietro Baroni, Daniela Fogli and Giovanni Guida
University of Brescia - DEA, Via Branze 38, I - 25123 Brescia, Italy
Phone: +39 30 3715455, Fax: +39 30 380014
E-mail: {baroni, fogli, guida}@ing.unibs.it
AND
Silvano Mussi
CILEA - Via Sanzio 4, I - 20090 Segrate (MI), Italy
E-mail: mussi@icil64.cilea.it

Keywords: distributed artificial intelligence, multi-agent systems, mental attitudes

Edited by: Vladimir A. Fomichov
Received: July 29, 1998 **Revised:** November 6, 1998 **Accepted:** November 10, 1998

In this paper, we propose an approach to the design of multi-agent systems based on an original model of the mental activity of single agents. In particular, we introduce the concept of active mental entity, as a new way of representing mental attitudes such as intentions and persuasions. The internal architecture of each agent is thus understood as a distributed system whose reasoning activity is determined by the interactions among active mental entities. Then, this architecture is extended in order to enable the agents to operate in a multi-agent context. A detailed description of the structure and operation of an agent and of a multi-agent system is thus provided. The implementation of the proposed paradigm is then illustrated and some performance examples are presented.

1 Introduction

A multi-agent system (MAS) can be regarded as a team of computational entities which have well-established competencies and expertise and which are able to cooperate and conflict according to some specific interaction protocols, in order to determine the overall operation of the system. However, a MAS can not simply be conceived as a traditional distributed system, namely as a collection of interacting components. In fact, agents taking part of a MAS are required to show a rational and autonomous behavior, that is they are expected to be capable of operating according to their own objectives, independently from human intervention.

Therefore, in a MAS, the overall system behavior emerges from the interaction of autonomous components, rather than from fixed pre-determined operation schemata. For this reason, one of the most important issues in the design of a MAS is the definition and implementation of its basic components, namely agents exhibiting an autonomous behavior.

One of the major issues in order to actually achieve a fully autonomous behavior is the definition of a proper architecture at the level of single agent. The main approaches to agent architecture design include the reactive approach and the deliberative approach. *Reactive* agents (Agre & Chapman, 1987; Brooks, 1991; Maes, 1995) are built according to a paradigm called *behavior-based*, since their architecture is understood as a hierarchy of task-accomplishing behaviors whose activation is determined by a fixed set of *stimulus-response* rules. Agent operation is relatively simple and guarantees a high level of reactivity; however, it is completely driven by the external stimuli rather than by its (implicit) internal goals. In a sense, the agent is at the mercy of stimuli: its behavior depends entirely on them. On the other side, the *deliberative* approach (Georgeff & Lansky, 1987; Bratman et al., 1988; Pollack, 1992) is based on the assumption that the exp! licit representation of mental activity should provide agents with high-level reasoning capabilities. In particular, agents are assumed to possess a set of *goals* and to produce and execute plans in order to fulfil such goals. External stimuli give rise to *beliefs* and agent operation includes specific mechanisms devoted to revise plans according to beliefs. In this approach, the behavior of an agent is fully regulated by its mental activity.

Mental activity is in turn modeled in terms of *mental attitudes*, namely mentalistic notions (such as knowledge, belief, intention, obligation, and so on) which usually are applied to humans in order to explain their behavior (Dennett, 1987; Wooldridge & Jennings, 1995). The notion of mental attitude can be extended, as a modeling tool, to software agents: mental attitudes support the design of agent internal structure and then determine agent behavior during operation.

While several theoretical works provide a sound logical foundation for modeling agent mental activity (see for example (Cohen & Levesque, 1990; Rao & Georgeff, 1991)), there is still a significant gap, however, between theoretical proposals and actual implementations of agents endowed with mental attitudes, as explicitly pointed out for example in (Wooldridge & Jennings, 1995). In this paper, starting from the basic ideas presented in (Cohen & Levesque, 1990), we propose an original approach to modeling agent mental activity which provides a sound basis for a practical implementation of agents and MASs.

2 Active mental entities

Cohen and Levesque's theory (Cohen & Levesque, 1990) is based on the notions of belief and goal. Intuitively, a belief is a proposition with an associated truth value, while a goal represents a desirable state of the world to be reached.

Some pragmatic questions, concerning both the representation and the manipulation of such mental attitudes, arise when an implementation has to be realized:

- How should beliefs and goals be represented?
- When, how and why do beliefs arise?
- Which is the mechanism ruling belief revision?
- When, how and why do goals arise?
- How is commitment to goals realized?
- Which is the mechanism ruling the selection and revision of action plans for pursuing adopted goals?

A typical answer to these questions is provided by a family of agent architectures (Georgeff & Lansky, 1987; Bratman et al., 1988; Pollack, 1992; Jennings, 1995) implementing the widely adopted BDI theory. Within such architectures, mental attitudes are conceived as data structures which are manipulated by a specific centralized process that constitutes the kernel of the agent. The design of such centralized mechanism is however very complicated and not transparent enough to guarantee a clear understanding of the issues listed above.

In order to overcome these difficulties, we start from an original point of view concerning the model of the mental activity of an agent. Our main ideas can be summarized as follows:

(i) the mental processes occurring inside an agent are the result of the cooperation and conflict between a set of mental attitudes;

(ii) since the interactions between attitudes are expected to produce a globally intelligent behavior, it is essential that mental attitudes are provided with individual and independent operation capabilities;

(iii) as a consequence of (i) and (ii), we represent mental attitudes as *active mental entities*, i.e. entities that can freely interact in a distributed context according to their own nature;

(iv) active mental entities give existence to dynamic processes that can be created and disposed when necessary.

In this paper we focus on two classes of active mental entities, namely *persuasions* and *intentions*, which are enough to show the potential of our approach.

2.1 Persuasion

A persuasion is meant to be an active mental entity definitely committed to ascribe and revise, whenever appropriate, a truth value V to a proposition P that represents a fact about the world, whose truth or falsity is of interest for the agent. In other words, the main task of a persuasion consists in finding and verifying elements that can justify the association of a truth value to a given interesting proposition. For this reason, a persuasion is generated when a new interesting proposition P is met and is dismissed when the interest in P ceases. In this sense a persuasion is persistent, that is it remains active until the proposition to which it refers is considered interesting; after a believed truth value has been determined, the persuasion has to monitor whether any change with respect to previous situation affects the currently believed truth value and, if appropriate, it has to revise such truth value accordingly.

In more precise terms, we can define a *persuasion* as a four-tuple P = ⟨ S, V, J, PM ⟩, where:

- S is the *subject* of the persuasion, namely a proposition whose truth or falsity is of interest for the agent;
- V is the currently most believed *truth value* of the subject S;
- J is the current *justification* that supports the assignment of the truth value V to the subject S. According to the nature of the information it is based upon, a justification has an associated *justification type* which may assume one of the following three values:
  - ◇ *agent knowledge*, i.e. knowledge available inside the agent;

◇ *sense data*, i.e. data collected from the external environment;

◇ *related proposition*, i.e. another proposition from which V can be derived, but which, differently from knowledge and sense data, needs in turn to be justified.

– PM is a set of *methods* used by the persuasion in its operation.

Initially, when a persuasion is generated, only S and PM are given; V and J have to be determined and then dynamically revised by the persuasion itself through its operation. Therefore, the set of methods PM must include all the procedural elements necessary for the persuasion to effectively carry out these tasks.

The set of methods PM can be represented as a five-tuple PM = ⟨ Gm, Sm, Vm, Rm, Cm ⟩ where:

– Gm is a *generation method*, i.e. a method for generating candidate activities for justification building, i.e. activities whose execution can provide evidence for the association of a truth value V to the subject S;

– Sm is a *selection method*, i.e. a method for the selection of candidate activities to exploit for building the current justification, i.e. for the determination of the current truth value V to associate to the subject S;

– Vm is a *verification method*, i.e. a method for the verification of the selected candidate activities, i.e. for determining whether one of the selected activities can actually lead to the construction of a justification and to the assignment of a truth value V to the subject S;

– Rm is a *revision method*, i.e. a method for the revision of the currently believed truth value V of the subject S when changes occur that affect the justification currently adopted for supporting the assignment of V to S;

– Cm is a *conflict resolution method*, i.e. a method for solving conflicts between persuasions.

2.2 Intention

An intention is meant to be an active mental entity committed to pursue a goal G, under the assumption that a given condition C holds, under which the goal is considered valid, namely pursuing such goal makes sense. Therefore, an intention is generated when a new goal G is identified and is dismissed when G is reached or the condition C ceases to hold.

An intention is pursued through a plan. A plan is understood as a sequence of actions which can be elementary, consisting in computations, sensorial acquisitions or actions on the environment; or non-elementary, whose accomplishment leads to the creation of new intentions. Each plan has an associated applicability condition representing the condition under which the plan can be executed, i.e. a condition that has to be true during plan execution.

After a viable action plan has been determined, the intention has to take care of its execution, to monitor whether any change affects the currently executed plan, and, if appropriate, it has to revise such plan accordingly.

In more precise terms, we can define an intention as a four-tuple I = ⟨ S, C, P, IM ⟩, where:

– S is the *subject* of the intention, namely a proposition describing a goal;

– C is the *validity condition* that enables the subject S to be a valid goal;

– P is the current *action plan* that is supposed to achieve the subject S;

– IM is a set of *methods* used by the intention in its operation.

Initially, when an intention is generated, only S, C and IM are given; P has to be constructed, executed and then dynamically revised by the intention itself during its operation. Therefore, the set of methods IM must include all the procedural elements necessary for the intention in order to effectively carry out these tasks.

The set of methods IM is in turn a five-tuple IM = ⟨ Gm, Sm, Em, Rm, Cm ⟩ where:

– Gm is a *generation method*, i.e. a method for generating candidate action plans capable of achieving the subject S;

– Sm is a *selection method*, i.e. a method for the selection of a candidate plan P to exploit for the achievement of the subject S;

– Em is an *execution method*, i.e. a method for the execution of the selected plan P;

– Rm is a *revision method*, i.e. a method for the revision of the currently active plan P when changes occur that affect the applicability condition of the currently active plan or of other candidate plans;

– Cm is a conflict resolution method, i.e. a method for solving conflicts between intentions.

Note that, since goals can be related either to permanent internal needs of the agent or to transient needs, related to the actual satisfaction of the permanent needs in a specific context, we can distinguish two kinds of intentions:

- *Primitive intentions* are created at the same time as the agent and are always active. A primitive intention is kept forever by the agent and does not depend on a specific achievement (its validity condition is always true). Primitive intentions represent therefore very general and fundamental objectives which are intrinsic to the existence of an agent and, in a sense, represent its basic *raison d'être*: "Preserving integrity" is an example of primitive intention.
- *Generated intentions* correspond to transient goals and are created by other mental entities. A generated intention is produced when the achievement of its subject is necessary for the achievement of the subject of another intention (either primitive or generated) or for determining the truth value of a persuasion. Intentions of this kind remain active only until their subject is achieved or their validity condition no more holds.

2.3 Relations between intentions and persuasions

Intentions and persuasions are strictly related to each other: in fact, in order to carry out its activity an active entity (either an intention or a persuasion) may need to generate some new active entities, devoted to solve sub-problems on behalf of their generator. According to this perspective, we define here the relationships concerning the dynamic generations of mental entities in our distributed mental activity model.

2.3.1 Generation of persuasions by intentions

Each intention is related to two propositions, namely its subject and its validity condition, whose truth values affect its operation. Therefore, when an intention is generated, such propositions become interesting propositions and, as a consequence, two persuasions are generated, having such propositions as subjects.

Persuasions are also involved in the phases of plan selection and execution by an intention. In fact, such phases are affected by the applicability conditions of the generated plans. As a consequence, a new persuasion is generated for each applicability condition, in order to allow both the initial plan selection and, possibly, subsequent plan revisions, if applicability conditions change.

2.3.2 Generation of intentions by intentions

Once an intention has selected a plan, it has to put it at work, by executing the various actions composing the plan. Each action can represent either an elementary activity (a computation, a sensorial acquisition, an action on the environment) which can be executed by a suitable operative component (see below), or a non elementary activity, whose accomplishment corresponds to generating a new intention whose subject represents the accomplishment of the action.

2.3.3 Generation of persuasions by persuasions

The justification for the truth value of the subject of a persuasion can be obtained through an inference step which uses as premise another proposition. In this case, the related proposition becomes the subject of a new persuasion.

2.3.4 Generation of intentions by persuasions

Each persuasion is in charge of searching for justifications in order to determine the association of a truth value to its subject. This activity may involve the generation of new intentions, if it encompasses the execution of non elementary tasks (for instance concerning acquisition and interpretation of sense data).

2.3.5 Operational constraints

The following operational constraints hold between a mental entity (intention or persuasion) and the mental entity that has generated it:

- a persuasion is in charge of notifying its generator when there is a change in its believed truth value;
- an intention is in charge of notifying its generator when its subject is achieved or when a definitive failure in its accomplishment is detected.

Finally, every mental entity that has generated another mental entity may decide to suppress it at any moment if the activity of the generated entity is no more considered useful, for instance due to a plan revision.

3 Solving conflicts between mental entities

One of the most significant parts of the cognitive activity of an agent is constituted by conflict resolution between mental entities. In fact, among the most important capabilities an autonomous agent should be endowed with there are:

- the ability to deal with different contrasting goals, which can be obtained by exploiting suitable methods for conflict resolution between intentions;
- the ability to cope with the uncertainty and ambiguity that affects the perception and representation of the external world, which can be obtained by exploiting suitable methods for conflict resolution between persuasions.

Due to space limitation we can give here only a brief account about this important topic (see (Baroni et al., 1997) for a more detailed description).

3.1 Conflict resolution between persuasions

The findings of different persuasions, which are based on different justifications and may have been acquired at different times, may lead to contradictory conclusions, so generating a conflict, which needs to be solved if such conclusions affect agent decisions.

Here we consider only a simple conflict resolution mechanism for persuasions, based on the comparison between the justification types of plans adopted by persuasions. In practice, we assume that a pre-defined priority order exists between the justification types "knowledge" and "sense data". In particular, the latter, being based on recent and up-to-date data acquisitions, is assumed to be stronger than the first one (of course this is a working assumption which might not be adequate in some specific situations, e. g. where sensory devices are particularly unreliable). Regarding the "related proposition" type, it has to be considered that this kind of justification in general brings about a chain of related propositions. At the root of this chain, a terminal node represents a justification which can be in turn "knowledge" or "sense data". Thus, each persuasion whose justification type is "related proposition" must search first for the justification type ascribed to the termina! 1 node of the chain of propositions supporting its subject and then determine its justification strength (for the sake of simplicity we do not consider here the case where multiple justification chains are available).

Therefore, conflict resolution is carried out by the involved persuasions by determining which one has the stronger justification. If both persuasions have the same justification type, a more articulated conflict resolution mechanism is necessary: its description is however beyond the limits of this paper.

3.2 Conflict resolution between intentions

A conflict between two intentions may arise when they concurrently try to assign contrasting tasks to a shared operative component. If the involved intentions are primitive, we assume that a *priority* attribute makes it possible to directly establish the prevailing intention. If one conflicting intention is primitive and the opponent one is generated, the latter refers to the priority of the primitive intention underlying it. Finally, if both intentions are generated, conflict resolution involves a more articulated interaction protocol. For the sake of simplicity, we will consider here only a simple example of protocol, where the conflict is solved by postponing the execution of the plan of a conflicting intention to that one of the opponent. In order to decide which plan should be postponed, each intention simulates the execution of a new compound plan, which is obtained from the previous one by hypothesizing to give priority to the opponent intention. If the conflict ca! nnot still be solved, i.e. both intentions are unable to accept to be postponed, the conflict is transferred at the level of primitive intentions, where it can be directly solved according to their priority attribute.

4 Overall agent structure

According to the proposal presented above, the overall structure of an agent A includes two parts:

- a *static part*, which is created at the moment an agent is defined and which remains unchanged during all its operational life;
- a *dynamic part*, which includes components that are generated and disposed during agent operation.

The static part S is a three-tuple S = ⟨ Ip, O, K ⟩ where:

- Ip is the set of *primitive intentions* of A;
- O is the set of *operative components* of A, in charge of performing elementary actions, either mechanical, concerning the interaction with external world through sensors and actuators, or symbolic such as inferential and computational activity;
- K is the set of *knowledge bases* available to A for problem-solving purposes. It includes:
  - ◇ *self knowledge*, that is knowledge concerning agent's own specific features and capabilities, used to decide whether a request arriving from outside is acceptable by the agent or not;
  - ◇ *domain knowledge*, that is knowledge concerning the agent competence domain, used by the agent components in order to carry out its activities.

The dynamic part *D(t)* is a time-variant pair D(t) = ⟨ P(t), Ig(t) ⟩ where:

- P(t) is the set of persuasions which are active in A at time t;
- Ig(t) is the set of generated intentions which are active in A at time t.

5 From agent architecture to multi-agent systems

After having proposed a model for the internal structure of an agent, we have to provide a methodology to build MASs based on such agent model: in this section, some extensions to the agent structure are introduced and interaction mechanisms between different agents are defined.

Agent structure is extended with a new component, called *interface*, in charge of managing interactions with other agents, and with a new knowledge base, called *mutual knowledge*, including information about the competencies and capabilities of the other agents belonging to the MAS. As far as interaction is concerned, each agent interface is able to interact with the other agent interfaces by addressing them requests concerning specific tasks accomplishment. At the level of a single agent, all the described interaction occurring among active mental entities remain valid.

Agent extensions and interaction mechanisms are analyzed in more detail in the following subsections.

5.1 Agent interface

In a multi-agent system (MAS), an agent is competent only about a particular portion of the world. Therefore, articulated tasks, involving different types of activities, may require the cooperation of different agents. So when an agent is unable to accomplish either the task of finding and verifying justifications for a given proposition or of achieving a given goal, it may resort to the external support of other agents. This requires that the agent in need of external support is able to send suitable problem-solving requests to other agents having competencies about the proposition of interest or the goal to be achieved.

Therefore, an additional active component, playing an intermediary role in communication between different agents, must be included in the agent structure. The main activities of this additional component, called agent *interface*, consist in:

- Examining problem-solving requests coming from the interface of another agent and addressing them to the proper internal component. As a consequence, if the request concerns a goal to be achieved, self knowledge is exploited to verify if an operative component able to satisfy the request exists. If such operative component is not available, but at least one plan can be identified for achieving the goal, a new intention is created within the agent. On the other hand, if the received request concerns a proposition whose truth value has to be determined, the creation of a new persuasion is carried out by the agent interface.
- Examining problem-solving requests coming from a component inside the agent (and that can not be addressed to another component of the same agent) and addressing them to the interface of another appropriate agent. To this purpose the interface exploits mutual knowledge (see below) in order to establish a matching between the problem to be solved and the agent having competencies about it.

5.2 Mutual knowledge

An agent belonging to a community of agents should be aware not only of its own competencies, but also, as far as possible, of the competencies of the other agents in order to be able to properly address problem-solving requests. For this reason, each agent of a MAS is endowed also with mutual knowledge.

Mutual knowledge of an agent includes a representation of competencies and capabilities of the other agents belonging to the architecture and is exploited to manage interaction and cooperation among different agents. In fact, when an agent has to accomplish an action which does not fall into its competencies, the interface, using mutual knowledge, finds the agent able to tackle the problem and therefore, sends the relevant request to it.

5.3 Agent operation

The activity of an agent is mainly guided by its primitive intentions: the permanent needs of satisfying user requests or of preserving its own integrity are examples of such kind of intentions. Primitive intentions try to achieve their subjects by determining the creation of new persuasions and intentions which in turn generate and dispose other mental entities. The overall interaction and cooperation of active mental entities determine the global agent behavior.

Moreover, in a multi-agent context, the activity of an agent also involves the problem-solving requests sent to and received from other agents.

Therefore, agent interface is in charge of receiving and processing external problem-solving requests. If the request concerns a goal to be achieved, the interface uses self knowledge in order to check whether the request can be accepted, i.e. if the incoming problem falls into the agent competence. If this is the case, a way to tackle the problem has to be found out. If the problem is elementary and an operative component able to solve it is available, the problem is directly addressed to it. Otherwise, if the incoming problem is more complex and can not be directly solved by a single operative component, a new intention is generated, whose subject coincides with the solution of the problem. The intention, using domain knowledge, is in charge of identifying a set of alternative plans for the solution of the problem at hand, of selecting one of

them and of putting it at work by resorting to the cooperation of other (mental or operative) components. Finally, the obtained result! s are returned to the agent which initially sent the problem-solving request.

On the other hand, when an intention realizes that some task included in its chosen plan does not fall into the competencies of the agent to which it belongs, such intention tries to resort to the cooperation of other agents. In practice, the intention asks the agent interface for identifying another agent having competencies about the task to be carried out. The agent interface, by exploiting the mutual knowledge, determines the identifier of such competent agent and forwards a request to it. When the allocated task has been completed, the intention waiting for it is notified; such intention is then allowed to keep on carrying out its problem-solving activity.

5.4 Architecture and operation of the multi-agent system

The agents composing the multi-agent architecture are able to accomplish specific tasks and are capable of cooperating according to a task-sharing approach. Each agent has its own local problem-solving and reasoning capabilities and is endowed with individual resources; it can autonomously deal with some part of the problem and can produce partial solutions. To carry out its job, an agent may resort to other agents, which are asked to carry out activities it can not carry out directly. For this reason, agents are assumed to be benevolent (Rosenschein & Genesereth 85), i.e. they are always available to comply with cooperation requests of other agents.

The overall operation of the architecture results from the autonomous operation of the agents. In practice, agents operate concurrently on different portions of the assigned problem. Therefore, the overall MAS behavior emerges from the behaviors of the agents composing it and from proper cooperation protocols ruling their interaction.

Each agent included in the MAS is endowed with primitive intentions which rule its individual operation. According to the circumstances, one or more agents will undertake actions for the accomplishment of their intentions or remain waiting for relevant events.

In particular, it may be hypothesized that a user is entitled to ask the MAS for carrying out a task. For this reason, a suitable agent inside the architecture should be devoted to the interaction with the users: such agent is capable of understanding user problem-solving requests and of forwarding them to the most appropriate agent which, by cooperating with the other agents, will carry out their resolution.

External events influence the operation of agents by determining changes of their internal structure (in practice, generation and deactivation of mental entities) and plan revisions. This may entail the need of facing new problems: their resolution may involve the assistance of other external agents and, for this reason, new conflict or cooperation relations among agents may arise. If a conflict between mental components of different agent occurs, the conflict resolution protocol defined for an individual agent remains valid for the multi-agent case. In particular, it is assumed that a global priority order holds among primitive intentions belonging to different agents. On the other hand, conflicts which occur between persuasions of different agents can still be solved on the basis of the justification types as explained in section 3.1. However, it is reasonable to think that more articulate conflict resolution mechanisms might be designed for MASs. For example, intentions belonging to different agents might be justified themselves, and the agents might engage a negotiation activity based on the justifications of the intentions every time a conflict must be solved. This leads to the issue of negotiation through argumentation (Parsons & Jennings, 1996): an important research topic which is however beyond the limits of the present work.

6 Implementation and experimentation

On the basis of the general approach introduced in the previous sections, a prototype programming environment for the development of agent and multi-agent architectures endowed with active mental entities has been developed. The implementation of such prototype has been written in C++. In particular, the *Coroutine Library* (Stroustrup & Shopiro, 1987) has been exploited, that contains basic facilities for multi-thread programming. In this environment, a thread can be implemented as an instance of a user-defined class derived from the basic class *task* and can be suspended and resumed when necessary. All components of an agent are thus implemented as classes derived from class task and an agent is realized as a multi-thread system, where each component is associated to a thread executed concurrently with respect to other ones. In particular, operative components and primitive intentions, which belong to the static part of an agent, correspond to threads starting thei! r operation at the moment of the creation of an agent, while persuasions and generated intentions, which belong to the dynamic part of an agent, correspond to threads which are dynamically created by other threads. Finally, generated threads may terminate spontaneously or may be killed by their generator, according to the circumstances.

The overall multi-agent architecture can be regarded as a multi-thread system obtained by the union of the

multi-thread systems composing the various agents, which interact by exchanging messages through the thread implementing their interface.

Communication among different agents and among agent components is carried out through a message passing paradigm by exploiting primitives available in the library.

The prototype has been experimented in the implementation, at a high level of abstraction, of the simulated control architecture of a department mail delivery robot. An example is presented below in order to give an idea of how the prototype works.

The example concerns a department mail delivery robot to which the user consigns an envelope to be delivered to Mr. X. The robot is conceived as a multi-agent system where each agent is able to perform a specific task such as managing a sonar sensor, managing a TV camera, controlling movement actuators, etc.

Let us suppose that the primitive intention whose subject is "obey-the-user" is active into the UI (User Interaction) agent of the robot. After receiving the request of delivering mail to Mr. X, a new intention has to be generated whose subject is "deliver-mail-to-Mr.X". However, since UI has no specific competence on mail delivering, it has to address the request of creating this new intention to another competent agent. By resorting to its interface, UI identifies the MD (Mail Delivery) agent and forwards the request to it. The new intention "deliver-mail-to-Mr.X" is therefore created within MD. This intention may then generate different plans for its achievement. For instance, a simple plan relying on the persuasion "Mr. X is in his office" may be:

Task 1: go to the office of Mr. X

Task 2: deliver the envelope to Mr. X

Task 1 is considered first: it still concerns a quite generic and high-level task and must therefore be associated to a new intention. A request of generating such an intention is therefore addressed by MD to MM (Movement Manager) agent.

While "go-to-the-office-of-Mr.X" is active within MM, other intentions are continuously looking for better plans and persuasions are continuously looking for new evidences. So, while the robot is moving towards the office of Mr. X, the intention "deliver-mail-to-Mr. X" of MD may elaborate the following alternative plan, relying on the persuasion "Mr. X seen in front of the robot":

Task 1: go near Mr. X

Task 2: deliver the envelope to Mr. X

The persuasion "Mr. X seen in front of the robot" is then activated and is in charge of finding support. For this reason, it requires the generation of intention "recognize-face-of-Mr.X" to the VC (Video Camera) agent, so creating a sort of mechanism of attention to the presence of Mr. X in the neighbourhood.

Let us now suppose that the robot, thanks to the activity of this intention, realizes that Mr. X is in front of it. Then, a conflict between the persuasions "Mr. X is his office" and "Mr. X seen in front of the robot" must be solved. Since the first one has justification type *knowledge* (Mr. X is an employee that is assumed to stay in his office) whilst the latter is justified by *sense data*, the conflict is solved in favour of the second persuasion. Accordingly, the plan relying on such persuasion is put at work, whereas the other plan is abandoned.

Let us now restart from the situation in which "go-to-the-office-of-Mr. X" was active within MM and let us suppose that, while the robot is moving towards the office of Mr. X, its energy reaches the minimum threshold. This fact is noticed by a persuasion active within the EC (Energy Control) agent having the purpose of constantly monitoring the energy level. The subject of this persuasion is "battery-is-drying-up" and represents the applicability condition of a plan of the primitive intention "preserve-energy-level" of the EC agent: the persuasion notifies the intention of the change occurred in its truth value and, as a consequence, the primitive intention undertakes the execution of the following plan:

Task 1: go to the recharging point

Task 2: wait for the complete battery recharge

Task 1 leads to the generation of the new intention "go-to-the-recharging-point" in the MM agent, which immediately starts its activity by carrying out a proper movement plan. Now, both intentions "go-to-the-office-of-Mr.X" and "go-to-the-recharging-point" need to resort to the movement actuators and, therefore, they may conflict one another. Let us suppose, for example, that the recharging point and the office of Mr. X are in opposite directions. Then, at a certain time, intentions "go-to-the-office-of-Mr.X" and "go-to-the-recharging-point" could have to send to the wheels the commands "turn-left" and "turn-right" respectively. Obviously, a conflict arises.

In order to solve the conflict, intentions operate as follows:

- "go-to-the-office-of-Mr.X" estimates the time for the execution of a (simulated) plan where reaching Mr.X is postponed to the completion of the plan to which the opponent intention belongs;
- "go-to-the-recharging-point" estimates the time for the execution of a (simulated) plan which delays the arrival at the recharging point after the completion of mail delivery.

Then, since generated intentions have an associate *deadline* attribute, they compare the obtained time estimations with such deadlines in order to decide an acceptable scheduling.

If both times of simulated plans are incompatible with the intentions deadlines, conflict resolution is delegated to primitive intentions "obey-the-user" and "preserve-energy-level" which solve the conflict according to their priorities. In this case, we can reasonably suppose that "preserve-energy" has more priority than "obey-the-user", so it wins the conflict and propagates this information to the other intention involved in the conflict, whose accomplishment will be necessarily postponed.

7 Discussion

As already stressed, the construction of a MAS involves the specification of computational units, namely agents, which may be mainly characterized in terms of autonomy. In this section, we outline how our approach based on the notion of active mental entities is definitely appropriate for the actual realization of autonomous agents operating in a multi-agent context.

A deep investigation about the concept of autonomy is carried out by Castelfranchi in (Castelfranchi, 1995). In particular, Castelfranchi deals with two different forms of autonomy, namely *cognitive autonomy* and *social autonomy*. In order to achieve cognitive autonomy, beliefs and goals are introduced in place of stimuli and reactions. On the other side, social autonomy concerns the relationship between the goals of different agents. In order to obtain this kind of autonomy each agent "is endowed with goals of its own, which it has not received from outside as contingent commands. And its decisions to adoption of others' goals are taken on the basis of these goals" (Castelfranchi, 1995). Moreover, Castelfranchi asserts that a key factor for cognitive autonomy is that "it is impossible to change automatically the beliefs of an agent", since it depends on the mechanism of belief updating rather than just on the fact that the agent has some internal beliefs. A similar ! consideration can be made for goals, since an agent should not be blindly available to adopt exogenous goals. Therefore, having an explicit representation of beliefs and goals is not sufficient to guarantee that an agent is cognitively and socially autonomous.

In order to achieve cognitive autonomy an agent should be also able to:

(a) select the information it is interested in, rather than collecting any stimulus provided by the external environment;

(b) search for interesting information when it is not immediately available;

(c) recognize when an information is no more interesting;

(d) solve conflicts between contradicting information;

(e) find appropriate plans to pursue its own goals (possibly by requiring the cooperation with other agents) by maintaining a commitment both to plans and to goals.

These capabilities are naturally encompassed by the approach based on active mental entities. In fact, active persuasions are generated by other mental entities and this guarantees that each agent focuses its attention only on the aspects of the world which are of some interest for it. Moreover, since persuasions are active entities, they do not simply acquire the readily available information from the environment, but may start also information acquisition activities through the generation of suitable intentions. Then, since persuasions are dynamic entities, they remains active only until the belief to which they refer is no more considered interesting.

The explicit representation of contradicting points of view is a key feature for acting in realistic contexts, and the solution of conflict which eventually arises from these contradictions may be carried out directly by the mechanism of interaction between persuasions, as explained in subsection 3.1.

Finally, active intentions allow the realization of the two identified forms of commitment with respect to goals and to plans. They are persistent and are capable of managing external events which could influence such commitments: the intention operation mechanism includes the capability of revising plans, when they are no more applicable, and of dropping goals, when they are no more achievable.

Turning to social autonomy, an autonomous agent should be able to:

(a) distinguish between endogenous and exogenous goals;

(b) mediate external requests with its pre-existing goals, by eventually solving conflicts between contrasting needs.

Both these properties are guaranteed by our notion of intention as active mental entity. The concept of primitive intention guarantees that any agent is endowed with an endogenous and permanent set of intentions, which represent the objectives that the agent is permanently committed to achieve.

Externally generated intentions (i.e. those ones deriving from the requests of other agents) can not overwrite pre-existing intentions, but rather have to cohabit with them. If an exogenous intention is not compatible with another pre-existent intention, a conflict arises and is solved through the internal conflict resolution mechanism.

8 Conclusion

In this paper, we have presented a new approach for designing and implementing multi-agent systems based on a new model of agent internal structure. We have remarked, as an important feature of the deliberative approach, the separation occurring between goals and actions and between beliefs and stimuli. Then, we have proposed, as a starting point of our modeling perspective, a new kind of separation: namely the separation between goals and intentions on one hand and between beliefs and persuasions on the other hand. In particular, intentions and persuasions are conceived as active mental entities, i.e. they are conceived as computational entities, able to carry out autonomous activities.

Starting from this standpoint, we have proposed a formalization of the structure and operation of persuasions and intentions and of the agent structure and architecture, conceived as a dynamic and distributed system. A simple extension of such architecture provides the basis for applying such approach to the construction of multi-agent systems.

A prototypical software implementation has been developed and some examples concerning the control architecture of a simulated mobile robot have been presented.

References

[1] Agre P.E. & Chapman D. (1987) Pengi: An Implementation of a Theory of Activity. *Proc. of the Sixth National Conference on Artificial Intelligence (AAAI-87)*, Seattle, WA, 268-272.

[2] Baroni P., Fogli D., Guida G., Mussi S. (1997) Active mental entities: a new approach to endow multi-agent systems with intelligent behavior. *Proc. of the Workshop on Multi-Agent Systems: Theory and Applications (MASTA'97)*, Coimbra, 1-15.

[3] Bratman M. E., Israel D. J., Pollack M. E. (1988) Plans and resource-bounded practical reasoning. *Computational Intelligence*, 4, 349-355.

[4] Brooks R. A. (1991) Intelligence without representation. *Artificial Intelligence*, 47, 139-159.

[5] Castelfranchi C. (1995) Guarantees for Autonomy in Cognitive Agent Architecture, M. J. Wooldridge, N. R. Jennings (eds.), *Intelligent Agents*, LNAI-890, Springer-Verlag, Berlin, 56-70.

[6] Cohen P. R. & Levesque H. J. (1990) Intention is choice with commitment.*Artificial Intelligence*,42(3),213-261.

[7] Dennett D. C. (1987) *The intentional stance.*, The MIT Press, Cambridge, MA.

[8] Georgeff M. P. & Lansky A. L. (1987) Reactive Reasoning and Planning. *Proc. of the Sixth National Conference on Artificial Intelligence (AAAI-87)*, Seattle, WA, 268-272.

[9] Jennings N. R. (1995) Controlling cooperative problem solving in industrial multi-agent systems using joint intentions. *Artificial Intelligence*, 75(2), 195-240.

[10] Maes P. (1995) Artificial Life Meets Entertainment: Lifelike Autonomous Agents, *Communications of the ACM*, 38(11), 108-114.

[11] Parsons S. & Jennings N.R. (1996) Negotiation through argumentation-a preliminary report, *Proc. of ICMAS 96, 2nd Int. Conf. on Multi Agent Systems*, Kyoto.

[12] Pollack M. E. (1992) The uses of plans, *Artificial Intelligence*, 57 (1), 43-68.

[13] Rao A. S. & Georgeff M. P. (1991) Modeling Rational Agents within a BDI-Architecture, *Proc. of KR&R-91 Int. Conf. on Knowledge Representation and Reasoning.* Cambridge, MA, 473-484.

[14] Rosenschein J. S. & Genesereth M. R. (1985) Deals among rational agents, *Proc. 9th IJCAI Int. Joint Conference on Artificial Intelligence*, Los Angeles, CA, 91-99.

[15] Stroustrup B. & Shopiro J. (1987) A set of C++ classes for Co-routine Style Programming, *Proc. of USENIX C++*, Santa Fe, 417-439.

[16] Wooldridge M. & Jennings N. R. (1995) Intelligent Agents: Theory and Practice. *Knowledge Engineering Review*, vol.10(2),115-152.

Applying MPI to Electromagnetic Field Calculations

Christos Christodoulou, Javier Gomez Tangle and Janusz Zalewski
Dept. of Electrical and Computer Engineering
University of Central Florida
Orlando, FL 32816-2450, USA
Phone: (407)823-6171, Fax: (407)823-5835
Email: jza@ece.engr.ucf.edu
AND
Marek Machura Dept. of Computing
Bournemouth University
Talbot Campus
Poole, Dorset BH12 5BB, UK

Keywords: Distributed Computing, MPI, Electromagnetic Field Calculations, FDTD

Edited by: Marcin Paprzycki
Received: July 1, 1998 **Revised:** October 5, 1998 **Accepted:** October 20, 1998

This paper describes an implementation of the Finite Difference Time Domain (FDTD) method using MPI (Message Passing Interface) library to the sample electromagnetic problem: calculating the input impedence of a microstrip antenna. Its objective is to demonstrate to engineers in disciplines other than computing, how easy it is to use contemporary software tools, such as MPI, to speed up engineering computations.

1 Introduction

Nowadays, parallel and distributed processing are becoming a common place in the field of computational analysis. Techniques such as Finite Difference Time Domain (FDTD) Method and Finite Element Method, among others, have already been implemented using powerful supercomputers that make use of various parallel processing architectures. Access, though, to these supercomputers and programming them are not easy. MPI (Message Passing Interface) provides a much simpler way for implementing distributed/parallel processing without resorting to the use of supercomputers. The objective of this paper is to demonstrate to engineers in disciplines other than computing, how easy it is to apply this software tool to parallelize computations.

MPI is a standard specification [1, 2] of a set of library calls for passing messages between computers interconnected via a data communication network. It is an application programming interface, portable across different platforms. Implementations of the MPI standard exist for all major architectures, from supercomputers, to all versions of Unix-based machines, to Windows NT. A full list of implementations is available from [3] and more information on MPI has been collected on the MPI web page [4]. The MPI standard defines interfaces (called bindings) to two langauges, C and Fortran (as well as C++ and Fortran 90). That is, each MPI function can be called either from a C program or from a Fortran program. The literature on using MPI is growing rapidly and currently includes several dozen articles and four major books [5, 6, 7, 8].

This growing popularity of MPI, as well as its simplicity and portability, made it attractive to engineers of all professions. In this paper, we are trying to evaluate the usefulness and ease of use of MPI in one important problem in electromagnetics. Any code for electromagnetic field calculations that can be broken into smaller parts is a very natural candidate to run on a number of separate processors. This is because various pieces of the electromagnetic system can be calculated independently and the master process can take individually computed results and wrap the work up. If the number of individual code pieces is n, and all of them can be executed on separate processors, then the total speed-up of computations, compared to the sequential execution of the same program, should be almost n.

Here, MPI is used in conjunction with the FDTD method. An example of a simple rectangular microstrip antenna, fed by a microstrip line, is presented and discussed. A sample code for the MPI implementation is included and its implementation for more complex structures is discussed.

2 MPI Tutorial

Since this article is addressed to engineers trying to use MPI for the first time, we present a brief tutorial using C bindings. To learn MPI one has to realize, first of all, that it is not a new programming language. It is just a set of function calls, just like a set of functions we usually use to perform I/O in C (`scanf()`, `printf()`, etc.), or a set of functions one calls to perform graphics operations in the X window system (`XOpenDisplay()`, etc.).

The only difficulty is then in realizing basic concepts of parallel and distributed computations. The primary concept necessary to grasp is that of a process. A process can be roughly defined as a unit of concurrency, for example, performing a selfcontained computation concurrently with other processes. Each process may be a separately compiled program, however, a single program may define many processes as well.

2.1 Send and Receive Functions

MPI has been designed to exchange data between processes via messages, so its basic functions include facilities for what is technically called point-to-point communication: exchanging messages between two parties (processes), a caller and a responder. Two basic functions to facilitate this kind of data exchange are `MPI_Send()`, to send data, and `MPI_Recv()`, to receive data. Their syntax is described below:

- `MPI_Send(void * message, int count, MPI_Datatype datatype, int dest, int tag, MPI_Comm comm)`

where

`message` is a pointer to the buffer containing the message
`count` is the length of (number of elements in) a message buffer
`datatype` is the MPI type of the individual elements of the message (for example, a C float type is designated as `MPI_float`)
`dest` is the rank, that is identity, of the receiver
`tag` is the type of the message (integer in the range 0..32767)
`comm` is a communicator that defines a collection of processes that can exchange messages.

- `MPI_Recv(void * message, int count, MPI_Datatype datatype, int source, int tag, MPI_Comm comm, MPI_Status status)`

where the meaning of all parameters is the same as in `MPI_Send`, except of

`source` – the rank (identity) of the sender
`status` – a structure that provides information on the data actually received.

2.2 Housekeeping Functions

There are several items (one directive and a number of function calls) each MPI program written in C must include:

- `#include "mpi.h"`, which includes definitions and declarations necessary for compilation
- `MPI_Init()` to be called before any other MPI functions can be called; allows for an MPI library to be used
- `MPI_Finalize()` performs clean-up after MPI, deallocates memory, etc.
- `MPI_Comm_size()` returns the rank (id) of a process in its second parameter
- `MPI_Comm_rank()` which finds out the number of processes involved in the execution of the program.

3 Electromagnetic Application

It was mentioned in the Introduction that any code that can be broken into smaller parts is a very natural candidate for MPI implementation. The success of the method depends on how the programmer breaks the code into separate, although possibly dependent, pieces. The more pieces a code can be broken into, the faster the entire code will run, since more processors can be used simultaneously and each processor will have less load to handle. As an illustration how to apply MPI, an FDTD code used for the calculation of the input impedence of a rectangular microstrip antenna is discussed below.

Figure 1 shows a flowchart of a sequential FDTD analysis code written for a microstrip antenna. The microstrip line, used as a feed, is first analyzed without the antenna attached to it. It is attached to a matched load and its frequency response is obtained once the FDTD run is completed (steps 2-4, in the flowchart). This part is used for calibration purposes. Next, the feed line is connected to the antenna, as shown in Figure 2. Then the second FDTD run takes place to obtain the frequency response of the entire structure (steps 5-7). It is used with the response of the feed line alone to obtain the input impedence of the antenna (step 8).

Figure 3 shows how the parallelization has been implemented. The original problem was divided into three parts:

- Master process (running on processor 0) to do initialization, start remote processes, receive data from them and calculate the final result
- Process #1 to calculate frequency response of the feed and deliver results to Master
- Process #2 to calculate frequency response of the antenna and deliver results to the Master.

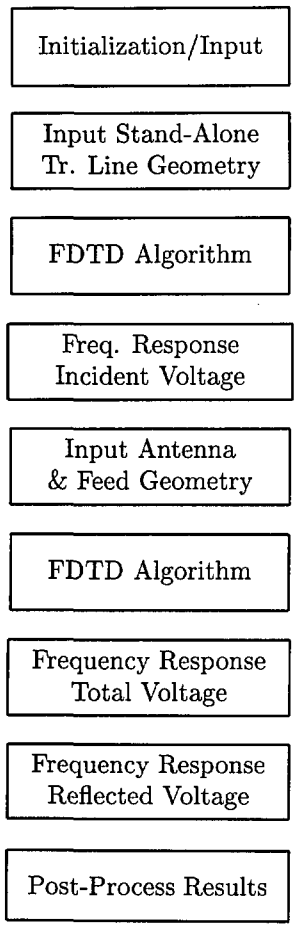


Figure 1: Sequential FDTD Code for a Microstrip Antenna (Top to Bottom).

The only changes in the original sequential code, to make it parallel, are in the `main()` function. They include additions of the necessary MPI function calls and the optional timing functions. The summary of changes is presented in Appendix. The full code is available via the Internet from the following URL: `http://www-ece.engr.ucf.edu/~jza`. For executions on UltraSparc 1, it was found that while the sequential version ran for over 5 hours, the distributed version ran a little bit more than a half of the sequential run time (that is, around 2.5 hours).

The same procedure is now being applied to a more complex structure, such as an array antenna. Similarly to the simple geometry from Figure 2, near-linear speedup is obtained for splitting the antenna into multiple elements. The larger number of elements in the array the more advantageous MPI can be. For example, for a 5-element linear array antenna, to calculate one specific parameter, say S, each element is fed with an input voltage while the other elements are connected to a matched load. MPI is utilized to run this procedure to obtain the values of S for all elements simultaneously in a single run, by using 6 processors (one master and one per each antenna element). The

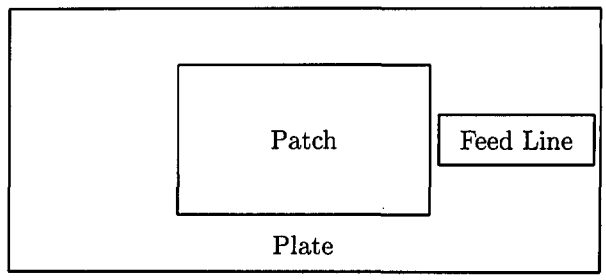


Figure 2: Geometry of Microstrip Antenna and Its Feed Line (Fixed Thickness of a Plate).

time savings become more significant when one has to model a larger linear array, or a two-dimensional array, or to calculate mutual coupling among all elements.

4 Conclusion

MPI can be a very powerful tool to distribute certain types of computations, when the program can be split into a number of relatively independent parts running on their own and communicating relatively rarely, for instance, only delivering final results to the master. In this paper, only a couple of MPI calls have been described and their use demonstrated, to show the attractiveness and simplicity of this tool for engineering calculations, especially to those who are not computer experts. In fact, there are many more extremely powerful MPI functions for use by programmers, in particular for collective communication, to broadcast and scatter data to multiple processes in one shot or to gather data from multiple processes by one or many processes.

References

[1] MPI Forum, *MPI: A Message Passing Interface Standard,* Version 1.1, June 1995, `http://www.mpi-forum.org`

[2] MPI Forum, *MPI-2: Extensions to the Message Passing Interface,* July 1997, `http://www.mpi-forum.org`

[3] MPI Implementations, Ohio Supercomputer Center, Columbus, Ohio, 1997, `http://www.osc.edu/mpi`

[4] MPI Home Page, Argonne National Laboratory, Argonne, Illilois, 1997, `http://www.mcs.anl.gov/mpi/index.html`

[5] Gropp W. et al., *MPI – The Complete Reference. Vol. 2. The MPI Extensions,* MIT Press, Cambridge, Mass., 1998

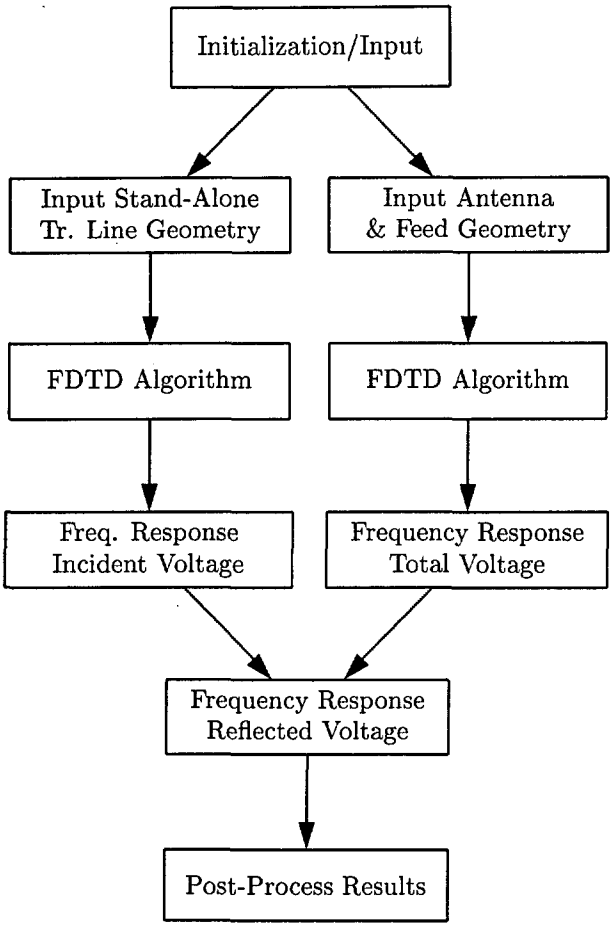


Figure 3: Distribution of FDTD Code for a Microstrip Antenna Using MPI.

[6] Gropp W., E. Lusk, A. Skjellum, *Using MPI: Portable Parallel Programming with the Message-Passing Interface,* MIT Press, Cambridge, Mass., 1994

[7] Pacheco P.S., *Parallel Programming with MPI,* Morgan Kaufmann Publishers, San Francisco, Calif., 1997

[8] Snir M. et al., *MPI – The Complete Reference. Vol. 1. The MPI Core. Second Edition,* MIT Press, Cambridge, Mass., 1998

Appendix. Outline of Changes in the main() Function

```
main(int argc, char * argv[])
{
    int my_rank, p;      // rank of process and number of processors
    int tag = 1, nameLength;     // message type and length of name
    char processor_name[MPI_MAX_PROCESSOR_NAME];
    MPI_Status status;
    FILE * fptr;         // file pointer
    double startTime, endTime;

    fptr = fopen("time.dat", "a");
    MPI_Init(&argc, &argv);
    MPI_Comm_rank(MPI_COMM_WORLD, &my_rank);
    MPI_Comm_size(MPI_COMM_WORLD, &p);
    MPI_Get_processor_name(processor_name, &nameLength);
    fprintf(stderr, "Process %d on %s\n", my_rank, processor_name);
                        // Ez = exp{-[(t-to)/ct]^2}
    ct  = 15e-12;       // pulse period
    to  = 3.5*ct;       // pulse start
    dt  = 0.44e-12;     // time increment
    tc1 = 2*to/dt;      // pulse cut-off time increment
    tco = (int)tc1;
    dz  = 0.000265;     // z-increment
    dx  = 0.000389;     // x-increment
    dy  = 0.0004;       // y-increment
    epsr= 2.2;          // Substrate relative permitivity
    mu  = PI*4e-7;      // Permeability

    substrate(); // define substrate configuration


    ////////////////////////////////////////////////////////////////////////
    // Code for worker process #1 starts here
    if (my_rank == 1) {
        startTime = MPI_Wtime();

        /* Analyze stand-alone transmission line */
        /* Initialize etx,ety matrices to non-conductor */
        init_non_cond();

        /* Define stand-alone transmission line */
        t_line_alone();

        /* Do time loop to find e-fields, h-fields and voltage */
        init_fields();
        Antenna = 0;
        find_fields();

        MPI_Send(ts2, lt+1, MPI_DOUBLE, 0, tag, MPI_COMM_WORLD);
    }
    // End of code for worker process #1
    ////////////////////////////////////////////////////////////////////////
```

```
    ////////////////////////////////////////////////////////////////////////
    // Code for worker process #2 starts here
    if (my_rank == 2) {
        startTime = MPI_Wtime();

        /* Analyze transmission line fed antenna */
        /* Initialize etx, ety matrices to non-conductor */
        init_non_cond();

        /* input transmission line geometry */
        tline_geom();

        /* input antenna geometry */
        ant_geom();

        /* initialize fields */
        init_fields();

        /* Do time loop to find e-fields, h-fields and voltage */
        Antenna = 1;
        find_fields();

        MPI_Send(ts1, lt+1, MPI_DOUBLE, 0, tag, MPI_COMM_WORLD);
    }
    // End of code for worker process #2
    ////////////////////////////////////////////////////////////////////////

    ////////////////////////////////////////////////////////////////////////
    // MPI part of code for the master process #0
    if (my_rank == 0) {
        MPI_Recv(ts2, lt+1, MPI_DOUBLE, MPI_ANY_SOURCE, tag, MPI_COMM_WORLD, &status);
        MPI_Recv(ts1, lt+1, MPI_DOUBLE, MPI_ANY_SOURCE, tag, MPI_COMM_WORLD, &status);

        /* Save time samples of the total, incident and reflected voltage */
        /* in the file voltage.txt */
        save_voltage();

        endTime = MPI_Wtime();
        fprintf(fptr, "Processor #0 clock time = %f sec\n", endTime - startTime);
    }

    fclose(fptr);
    MPI_Finalize();
    // End of code for the maste rprocess #0
    ////////////////////////////////////////////////////////////////////////

    return 0;
}
```

Mapping Complete Binary Tree Structures into a Faulty Supercube with Unbounded Expansion

Haun-Chao Keh and Jen-Chih Lin
Department of Computer Science and Information Engineering
Tamkang University, Tamsui, Taipei, Taiwan 251, R.O.C.
contact author E-Mail:g5190032@tkgis.tku.edu.tw

Keywords: parallel algorithm, fault-tolerance, supercube, embed

Edited by: Xindong Wu
Received: November 10, 1997 **Revised:** May 13, 1998 **Accepted:** June 21, 1998

We consider a new supercube architecture, a new interconnection network derived from the hypercube. The supercube retains the connectivity and diameter properties of the corresponding hypercube. The embedding of one interconnection network into another is a very important issue in the design and analysis of parallel algorithms. In this paper, the problem of embedding and reconfiguring complete binary tree structures is considered in a supercube with faulty nodes. We also propose a new method for embedding and reconfiguring complete binary trees in a faulty supercube. Furthermore, the results enable us to obtain a good method for embedding complete binary tree structures into a faulty supercube with n-expansion. The result enables us to obtain $O(n^2 - m^2)$ faults which can be tolerated, where $(n-1)$ is the dimension of a supercube and $(m-1)$ is the height of a complete binary tree.

1 Introduction

Hypercube, an interconnection architecture in parallel machines, has been the focus of researches in parallel computing due to its well-defined properties such as the modularity, the regularity, and the low diameter property, etc. These characteristics make it easy to design efficient parallel programs and share machines among users. Its important advantages are high data bandwidth and low message latency. Moreover, the hypercube may contain many other networks as its subgraphs such as rings, trees, meshes, etc. On the other hand, lots of interconnection networks can be mapped into the hypercube. It is apparent to demonstrate how all of the parallel algorithms, designed by those interconnection networks, can be directly implemented on the hypercube without significantly affecting the number of processors or the computation time.

¿From the computational perspective, the hypercube is quite powerful, but there are some imperfections to be an architecture for parallel computation. For example a hypercube has a vital disadvantage. That is the number of nodes must be equal to 2 to a power of n. In order to conquer the difficulties associated with hypercubes, several generalizations of the hypercube structures have been proposed during past years, which are discussed as follows.

(1) *Generalized Hypercube*[4] can be constructed with an arbitrary number of nodes. The structure has two major drawbacks. When the number of node is a prime number, it reduces to a completely connected graph. The second drawback is that significant changes have to be made whenever a new node is added.

(2) *Incomplete Hypercube*[7] can also be constructed with no limit of node number. Howerever, it has serious limitations in the connectivity. In some extreme situation, removing a single node may disconnect the whole graph.

A supercube[23] is one of the generalization of hypercube and does not have the drawbacks from the Generalized Hypercube and the Incomplete Hypercube. Speaking of its architecture, a supercube has the same connectivity and diameter as the corresponding hypercube. It is shown to have the following desirable characteristic: (1) the nodes connectivity of a N-node supercube is at least $\lfloor \log_2 N \rfloor$, (2)the node degree of a N-node supercube is between $(k-1)$ and $(2k-2)$, where $k = \lceil \log_2 N \rceil$, (3)adding a new node to an existing network is easy; it doesn't need reorganization of existing edges, and(4)the diameter of a N-node supercube is at most $\lfloor \log_2 N \rfloor$.

We map arbitrarily complete binary trees into a *faulty supercube* with a unit load. In a fault tolerance multicomputer system, two models need to be considered[12]: *total fault model* and *partial fault model.* The total fault model is defined as the computation and the communication fails in the system. The partial fault model is defined as the computation fails but the communication still works. This paper

proposes the mapping mechanism under the partial model.

An embedding of a guest graph G into a host graph H includes the mapping of nodes of G onto nodes of H and the mapping of edges of G into a path of H. The graph G and H are referred as the guest and host graphs respectively. Four costs associated with graph embedding are *dilation*, *expansion*, *load* and *congestion*. The dilation of an embedding is the maximum length of a path in the host graph, which is the mapping from an edge of the guest graph. The expansion is the ratio of the total number of nodes of G to the total number of nodes of H. The load of an embedding is the maximum number of nodes in the guest graph that are embedded into a node in the host graph. The congestion of an embedding is the maximum number of edges of the guest graph G that is embedded into any single edge of the host graph H.

We propose a mapping algorithm in this paper. The results enable us to obtain a good mapping method for embedding a complete binary tree into a faulty supercube with unbounded expansion such that, up to $O(n^2 - m^2)$ faults can be tolerated with congestion 1 and dilation 4, where $(n-1)$ is the dimension of a supercube and $(m-1)$ is the height of a complete binary tree.

The remainder of this paper is organized as follows. In the next section, some notations and definitions and how to map a complete binary tree into a supercube will be introduced. Section 3 describes how to map a complete binary tree into a hypercube and a faulty supercube with 2-expansion under partial fault model. Section 4 proposes a mapping method of an arbitrarily large complete binary tree into a faulty supercube under partial fault model. Finally, we conclude this paper and discuss further research problems.

2 Preliminary

The following formal definition of the supercube graph is from [15]. A supercube is constructed by any number of nodes and based on hypercube. A supercube, denoted by S_N, is defined as an undirected graph $S_N = (V, E)$, where V is the set of processors (called nodes in our discussion) and E is the set of bidirectional communication links between the processors (called edges). Assuming that Vcontains N nodes and each node can be numbered by an identical number in the range over $(0, N-1)$, in an $(n-1)$-dimensional supercube, each node can be expressed by an n-bit binary string because $2^{(n-1)} \leq N \leq 2^n$,where n is a positive integer.

Define 2.1[17] Suppose $S_N = (V, E)$ is an $(n-1)$-dimensional supercube, then the node set V can be divided into three subsets V_1, V_2, V_3, where

$V_3 = \{x |\ x \in V,\ x = 1u$, where u is a $(n-1) - bit$ sequence$\}$,

$V_2 = \{x |\ x \in V,\ x = 0u, 1u \notin V$, where u is a $(n-1) - bit$ sequence$\}$,and

$V_1 = \{x |\ x \in V,\ x = 0u, 1u \in V$, where u is a $(n-1) - bit$ sequence$\}$.

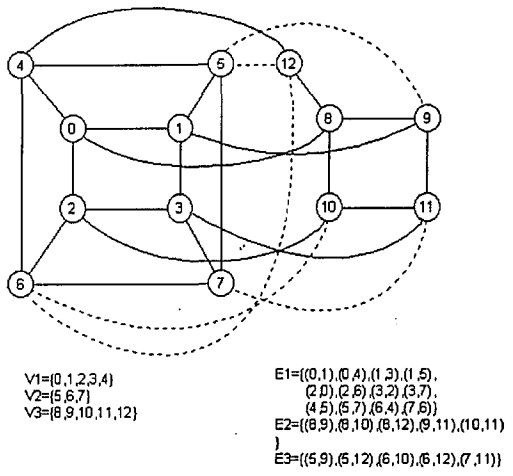


Figure 1:

Define 2.2 The *Hamming distance* of two nodes x and y is denoted by $H.D.(x, y)$ which defines as the number of different bits between the binary numbers of the nodes x and y. In other words, $H.D.(x, y)$ is the number of bits set in the resulting sequence of the bitwise XOR of x and y.

Define 2.3 The *dimension* of two nodes x and y is denoted by $Dim(x, y)$ and is equal to $(0, ..., i, ..., n)$ if and only if $H.D.(x, y) = n$ and $x_0 \neq y_0, ..., x_i \neq y_i, ...x_{n-1} \neq y_{n-1} (0 \leq i \leq (n-1))$.

Define 2.4[17] Suppose $S_N = (V, E)$ is an $(n-1)$-dimensional supercube, then the edge set E is the union of E_1, E_2, E_3 and E_4, where

$E_1 = \{(x, y) |\ x, y \in V,\ x = 0u,\ y = 0v$, where u, v are $(k-1)$-bit sequences and $H.D.(x, y) = 1\}$,

$E_2 = \{(x, y) |\ x, y$ in $V3,\ x = 1u,\ y = 1v$, where u, v are $(k-1)$-bit sequences and $H.D.(x, y) = 1\}$,

$E_3 = \{(x, y) |\ x$ in $V3,\ y$ in $V2, x = 1u,\ y = 0v$,where u, v are $(k-1)$-bit sequences and $H.D.(x, y) = 1\}$,and

$E_4 = \{(x, y) |\ x$ in $V3,\ y$ in $V1, x = 1u,\ \ y = 0u$, where u are $(k-1)$-bit sequences$\}$.

The supercube with 13-node is shown in Figure 1.

A mechanism to map a complete binary tree into a supercube with dilation 1 and load 1 is found in [2] .

Define2.5 If a complete binary tree is a rooted binary tree and each internal nodes contains two offspring nodes, then a complete binary of height h, denoted by T_h, contains $2^{h+1} - 1$ nodes.

Define2.6 A double-rooted complete binary tree is the complete binary tree with an extra node between the root and its right child.

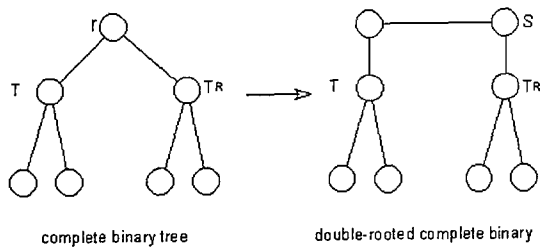


Figure 2:

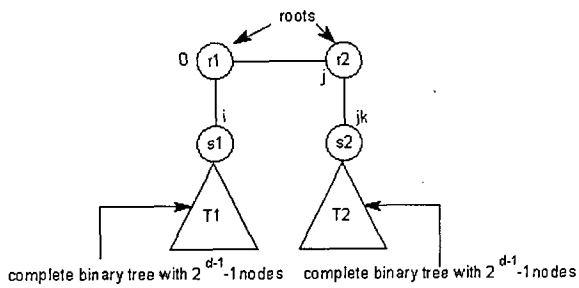


Figure 3:

The relationship between a complete binary tree and a double-rooted complete binary tree is shown in Figure 2.

Theorem2.7[1] The complete binary tree T_h can be mapped in $S_{2^{h+1}-1}$ with dilation 1.

3 Mapping a complete binary tree into a hypercube and a faulty supercube with 2-expansion under partial fault model

In [1], although a complete binary tree of height h can be mapped into a 2^{h+1}-node supercube with dilation 1 and load 1, the supercube can't find the replaceable node of the faulty node. Therefore, we considered how to map a double-rooted complete binary tree into a hypercube with dilation 2 and load 1[2], such that it can be mapped into a supercube.

We can obtain recursive construction of a mapping of a double-rooted complete binary tree into a hypercube as shown in Figure 3 to 7.

In Figure 3, a double-rooted complete binary tree with 2^d nodes is mapped into a d-cube. Node r_1 is mapped into (00...0); nodes s_1 and r_2 are mapped into the d-cube nodes with the i^{th} and j^{th} bits set, respectively; node s_2 is mapped into the node with the

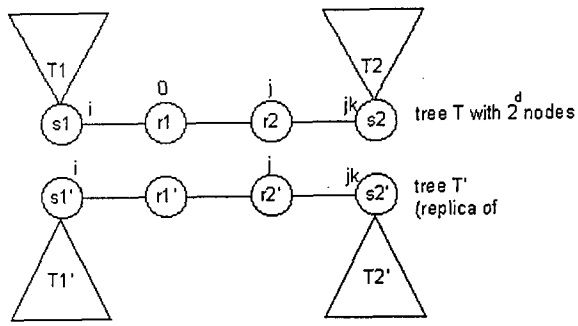


Figure 4:

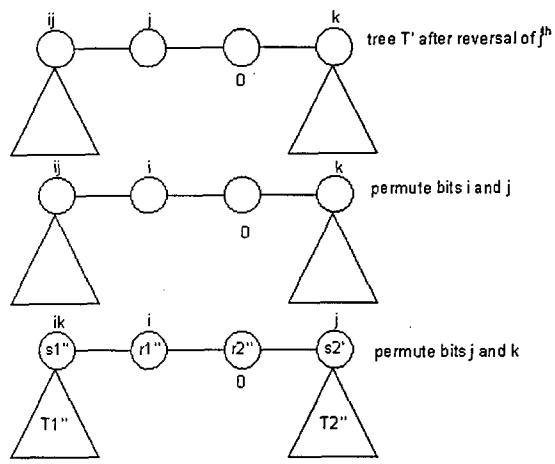


Figure 5:

j^{th} and k^{th} bits set. In Figure 4, we present a mapping of a double-rooted tree with 2^{d+1} nodes into the $(d+1)$-cube starting with two trees, T and T', which are mapped into the d-cube. Three transformations are applied sequentially to the node identity numbers of T'. All results yield the mappings of T' into the d-cube. These are reversal of the j^{th} bit, a permutation of the i^{th} and j^{th} bits, and a permutation of the j^{th} and k^{th} bits. The identity number for node r_2'' is now (00...0). The identities of node r_1'' and s_2'' are different from that of r_2'' in the i^{th} and j^{th} bit, respectively. Similarly, the identity of node s_1'' are different from that of r_2'' in the i^{th} and k^{th} bits. A leading zero (one) is next appended to the identity number of each node of T. We now introduce the links $(s1, r1'')$, $(r1, r2'')$ and $(r2, s2'')$ as partially shown in Figure 6. We also obtain a mapping of the double-rooted complete binary tree into the $(d+1)$-cube as shown in Figure 7 .

We illustrate an example as in Figure 8 to 13 by Figure 3 to 7.

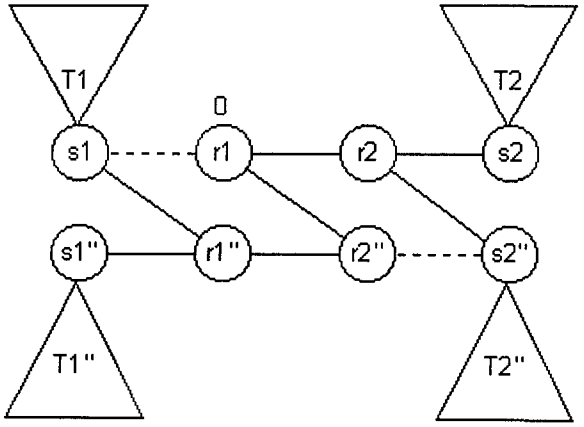


Figure 6:

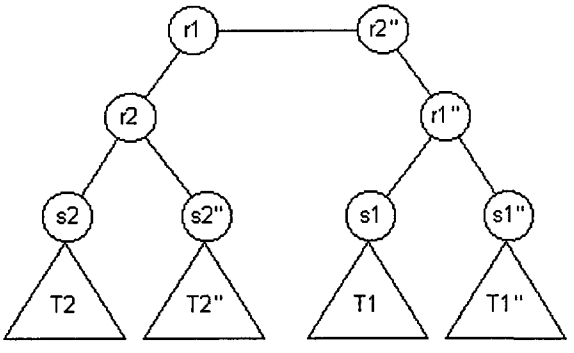


Figure 7:

Lemma 3.1 A double-rooted complete binary tree can be embedded in a hypercube (see Figure 7). The four nodes r_1,r_2,r_1'' and r_2'' are adjusted in Figure 8. A double-rooted complete binary tree can be constructed according to the four nodes in a hypercube[2].

Theorem 3.2[2] A double-rooted complete binary tree can be embedded in a hypercube with dilation 2 and load 1.

We infer the method of the embedding which maps a complete binary tree into a faulty supercube with 2-expansion. The method of the reonfiguring is proved by theorem 3.3.

Theorem 3.3[11] A complete binary tree of height h can be embedded into a $(2^{h+2}-2)$-node supercube with dilation 2, expansion 2 and load 1.

Proof. For finding the replaceable node of the faulty node, we assume the ratio of expansion is equal to 2. The total number of nodes of a complete binary

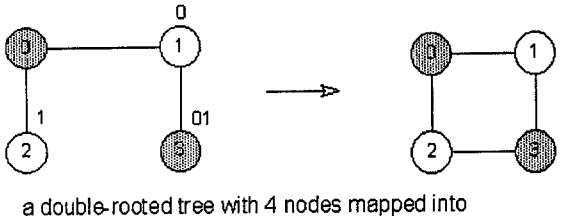


Figure 8:

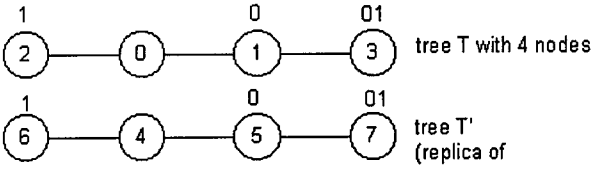


Figure 9:

tree T_h is $2^{h+1}-1$. The total number of nodes of a supercube is thus $2*(2^{h+1}-1)=2^{h+2}-2$. The node set V of the supercube is partitioned into three subsets V_1, V_2 and V_3. The set V', which is the union of V_1 and V_2, has 2^{h+1} nodes and a complete binary tree can be mapped into a supercube with 2^{h+1} nodes using a double-rooted complete binary tree. Therefore, a complete binary tree of height h can be mapped into a $(2^{h+2}-2)$-node supercube with dilation 2, expansion 2 and load 1.

Define 3.4 If a (2^h-x)-node supercube is lack of x nodes, we called these nodes *virtual nodes.*

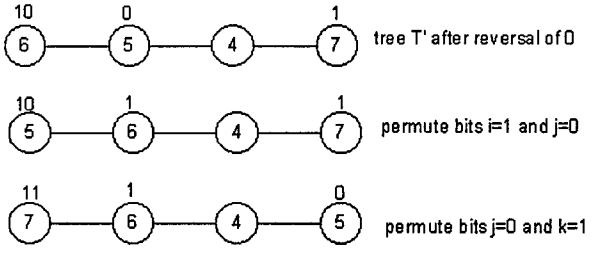


Figure 10:

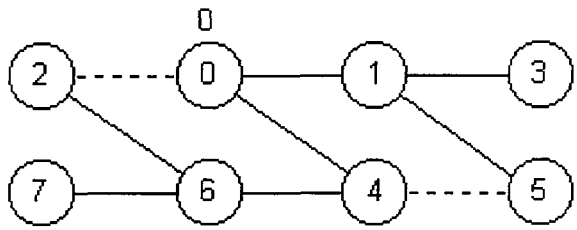


Figure 11:

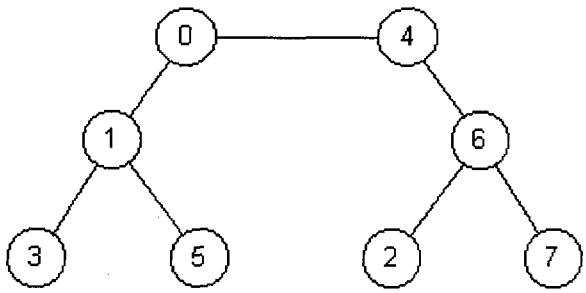
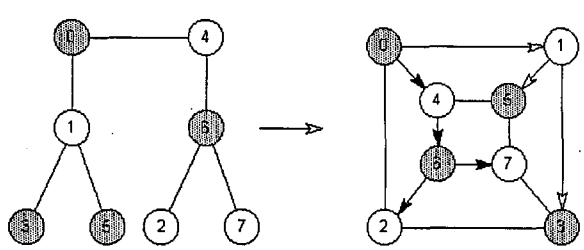


Figure 12:

4 Mapping any arbitrarily large complete binary trees into a faulty supercube

In this section, we eliminate the limitation of expansion. We assume that the total number of nodes of supercube S_N is N, $2^{n-1} \leq N \leq 2^n$ and the total number of nodes of a complete binary tree T_m of height $(m-1)$ is $2^m - 1$.



a double-rooted tree with 8 nodes mapped into a 3-cube

Figure 13:

Lemma 4.1 A complete binary tree of height $(m-1)$ can be embedded into a $(2^n - x)$-node supercube $(0 \leq x \leq 2^{n-1})$, with dilation 2 and load 1.

Proof. The result is trivial from theorem3.3.

Algorithm searching – ruleII :
1. *if* the root r is faultily *then*
1.1 search the spacer node S
1.2 *if* the spacer node S is faultily *then*
1.2.1 return the root r
1.2.2 $searching - pathII(r)$
1.3 *else*
1.3.1 node r is replaced by node S.
1.3.2 exit the *algorithm searching – ruleII*
2 *if* the other node p is faultily *then*
2.1 $searching - pathII(p)$

$searching - pathII(r)$
1 $i = 0; j = 0$
2 *while* $i \neq (n-m)$ *do*
2.1 we can search the node b
/* $b \in V$, $H.D.(r,b) = 1$,
$Dim(r,b) = m + i$*/.
2.2 *if* node b is not virtual node
and it is free then
2.2.2.1 node r is replaced by node b
2.2.2.2 remove all of nodes in a queue
2.2.2.3 exit the *while-loop*
2.3 $put(b, i+m-1)$ in a queue
2.4 $i = i+1; j = j+1$
3 *while* the queue is not empty *do*
3.1 remove the first pair (c,d) from the queue
3.2 *if* $c \in V_1$ *then*
3.2.1 $i = 0$
3.2.2 *while* $i \neq d$ *do*
3.2.2.1 we can search the node g
/* $g \in V$, $H.D.(c,g) = 1$,
$Dim(d,g) = i$*/.
3.2.2.2 *if* node g is not a virtual node
and it is free then
3.2.2.2.1 node r is replaced by node g
3.2.2.2.2 exit the *while-loop*
3.2.2.3 $i = i+1; j = j+1$
3.3 else $v - searching - pathII(c,d)$
4 *if* $j = [(n-m)(n+m+1)/2]$ then
4.1 declare the replaceable node of searching
is faultily.
4.2 exit the $searching - pathII()$

$v - searching - pathII(c,d)$
1 $i = 0$
2 *while* $i \neq d$ *do*
2.1 we can search the node k
/* $k \in V_3$, $H.D.(c,k) = 2$,
$Dim(c,k) = (d-1,i)$, $E(c,k) \in E_3$*/.
2.2 *if* node k is not a virtual node

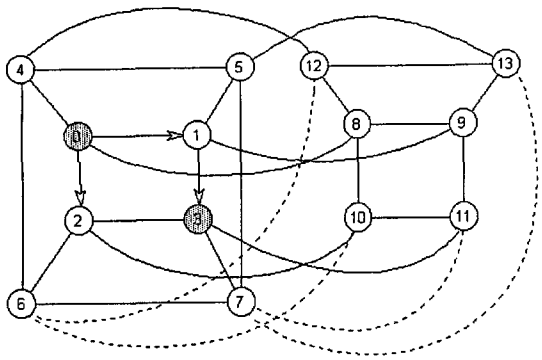


Figure 14:

and it is free *then*
2.2.1 node r is replaced by node k
2.2.2 exit the *while-loop*
2.3 $i = i + 1$; $j = j + 1$

By the *algorithm searching−pathII*, the searching path of the faulty node of a subtree is shown as follows.

node $0 = 0X_{n-2}X_{n-3}...X_mX_{m-1}...X_1X_0$
node $1 = 0X_{n-2}X_{n-3}...X'_mX_{m-1}...X_1X_0$
.
.
node $(n-m-1)$
$= 0X'_{n-2}X_{n-3}...X_mX_{m-1}...X_1X_0$
node $(n-m) = 1X_{n-2}X_{n-3}...X_mX_{m-1}...X_1X_0$
node $(n-m+1)$
$= 0X_{n-2}X_{n-3}...X'_mX_{m-1}...X_1X'_0$
node $(n-m+2)$
$= 0X_{n-2}X_{n-1}...X'_mX_{m-1}...X'_1X_0$
.
.
node $(n-m+m)$
$= 0X_{n-2}X_{n-1}...X'_mX'_{m-1}...X_1X_0$
node $(n-m+m+1)$
$= 0X_{n-2}X_{n-1}...X'_{m+1}X_mX_{m-1}...X_1X'_0$
.
.
node $[(n-m)(n+m+1)/2]$
$= 1X'_{n-2}X_{n-1}...X_mX_{m-1}...X_1X_0$

We illustrate an example of finding a replaceable node in Figure 14 to 16.

Theorem 4.2 The ending of searching path includes at least $\{[(n-m)(n+m+1)]/2 - x\}$ nodes.

Proof. By lemma4.1, we can embed a complete binary tree into a supercube from node 0 to node $(2^m - 1)$, which can be expressed by an m-bit binary string $i_{m-1}...i_0$, where $i_p \in \{0,1\}$. First, we can change a bit

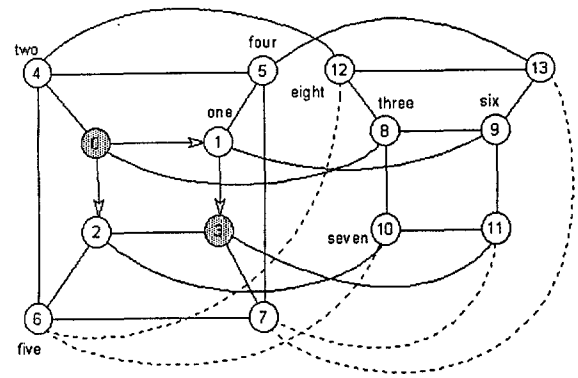


Figure 15:

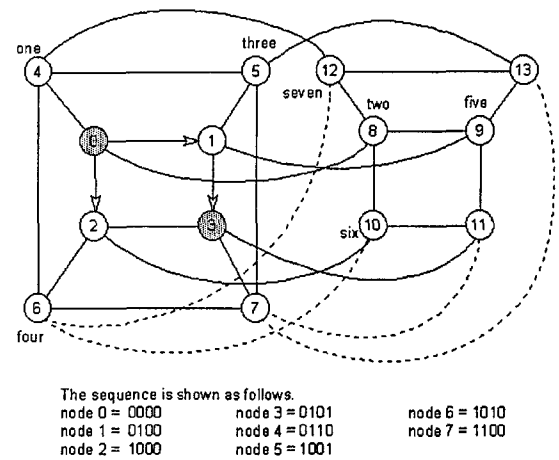


Figure 16:

in a sequence from bit m to bit $(n-1)$ and push the node in the queue. We can get $(n-m)$ different nodes. Second, we pop the node from the queue. From the first node we can change a bit in sequence from bit 0 to bit $(m-1)$, and we can get m different nodes. Then, we can change a bit in sequence from bit 0 to bit m from the second node, and we can get $(m+1)$ different nodes. Until the queue is empty, we can get the sum of searching of nodes, which is $[m+(m+1)+...+(n-1)]$. The ending of searching path includes $(n-m)+[m+(m+1)+...+(n-1)] = (n-m)+[(n-m)(n+m-1)]/2 = [(n-m)(n+m+1)]/2$ nodes. We assume we have x virtual nodes. Therefore, in the worst case we can search at least $\{[(n-m)(n+m+1)]/2-x\}$ nodes. By [13] and [17], we infer the edges of the searching-ruleII exist and none of the node has a duplicate searching.

Theorem 4.3 If the root of the tree is faultily and the number of faulty nodes is less than $\{[(n-m)(n+m+1)]/2+1-x\}$, we can find the replaceable node

of node after $\{[(n-m)(n+m+1)]/2+1\}$ times of search at the worst case.

Proof. We assume we can't find the replaceable node of the faulty node. That is, all of nodes on the searching path are already used or fault. By theorem 4.2, we can search the spacer node S of at least $\{[(n-m)(n+m+1)]/2-x\}$ nodes. In the worst case, the searching path includes at most x virtual nodes . Therefore, we can search $\{[(n-m)(n+m+1)]/2+1-x\}$ nodes in $[(n-m)(n+m+1)]/2+1]$ iterations. Because the number of faulty nodes is less than $\{[(n-m)(n+m+1)]/2+1-x\}$, we can find the replaceable node by *pigeonhole principle*. The originally assume is wrong. We can find the replaceable node of the root r in $\{[(n-m)(n+m+1)]/2+1\}$ iterations at most.

Theorem 4.4 If a node of a subtree is faultily and the number of faulty nodes is less than $\{[(n-m)(n+m+1)]/2-x\}$, we can find the replaceable node of faulty node in $\{[(n-m)(n+m+1)]/2\}$ iterations.

Proof. We assume that we can not find the replaceable node of the faulty node. That is, all of nodes on the searching path are already used or fault. By theorem 4.2, at least we can search the $\{[(n-m)(n+m+1)]/2-x\}$ nodes. In the worst case, the searching path is including x virtual nodes. Therefore, we can search $\{[(n-m)(n+m+1)]/2-x\}$ nodes after $[(n-m)(n+m+1)]/2]$ times of search. Because the number of faulty nodes is less than $\{[(n-m)(n+m+1)]/2-x\}$, we can find the replaceable node by *pigeonhole principle*. The originally assume is wrong. We can find the replaceable node of the root r in $\{[(n-m)(n+m+1)]/2\}$ times of iterations.

Theorem 4.5 There are $O(n^2-m^2)$ faults that can be tolerated in the algorithm of searching ruleII.

Proof. By theorem 4.3, there are $\{[(n-m)(n+m+1)]/2+1-x\}$ faults that can be tolerated. By theorem 4.4, there are $\{[(n-m)(n+m+1)]/2-x\}$ faults that can be tolerance. To sum up, we can show that $O(n^2-m^2)$ faults can be tolerated.

Theorem 4.6 The result holds dilation 4, congestion 1, and load 1.

Proof. We show that we can embed a complete binary tree of height m into a (2^n-x)-node supercube using nodes of $V_1 \cup V_2$ with dilation 2 by theorem 3.2.

Case 1. If a node p of a subtree is faultily, we can search the node b, $H.D.(p,b)=1$ by the $searching-pathII()$. If the node b is used or fault, we can search the other nodes g,$H.D.(b,g)=1$ by the $searching-pathII()$. We can get the dilation 2 in the worst case.

Case 2. If the root node r is faultily, we can search the spacer node S. If a node S of a subtree is faultily, we can search the node b, $H.D.(p,b)=1$ by the $searching-pathII()$. If the node b is used or fault, we can search the other nodes g, $H.D.(b,g)=1$ by the $searching-pathII()$. We can get the dilation 2 in the worst case.

Because every replaceable path is the only path by the *algorithm searching − ruleII* and[13] , we can get congestion 1 and load 1. Therefore, when the root node and spacer node are faultily, it is a worst case the dilation= $2+2=4$ at most, the dilation is $=1+2=3$ in other conditions. The other three costs associated with graph mapping are congestion 1 and $O(1)$ load.

5 Conclusion

In [11], we consider the problem of mapping a complete binary tree into a faulty supercube. Finding the replaceable node of the faulty node, we allow 2-expansion such that we can show that up to $(n-2)$ faults can be tolerated. The result implies that any complete binary tree can be mapped into a supercube with congestion 1 and dilation 4. That is, $(n-1)$ is the dimension of the supercube. After a complete binary tree can be mapped into a supercube with faulty nodes, we deduced our result to unbound expansion from 2-expansion such that it has the same cost and $O(n^2-m^2)$ faults can be tolerated. According to the result, we can embed the parallel algorithms developed by the structure of a complete binary tree into a supercube. These methods of reconfiguring enable extremely high-speed parallel computation.

After any arbitrarily complete binary tree structures can be reconfiguring in a supercube with faulty nodes, we are interesting the mapping of an arbitrary binary tree and multi-dimensional meshes into a supercube with faulty nodes.

References

[1] V. Auletta, A.A. Rescigno, and V. Scarano, "Embedding Graphs onto the Supercube," *IEEE Trans. on Computers*, Vol. 44, No. 4, pp. 593-597, April 1995.

[2] V. Auletta, A.A. Rescigno, and V. Scarano, "Fault Tolerant Routing in The Supercube," *Parallel Processing Letters*, Vol. 3, No. 4, pp. 393-405, 1993.

[3] Dimitri P. Bertsekas and John N. Tsitsiklis, "Parallel and Distributed Computation: numerical methods," *Prentice Hall*, Englewood Ciffs, New Jersey, 1989.

[4] L. Bhuyan and D.P. Agrawal, "Generalized Hypercubes and Hyperbus structure for a computer network," *IEEE Trans. Computers*, Vol. 33, pp. 323-333, 1984.

[5] Bethany M. Y. Chan, Francies Y. L. Chin, Senior member, IEEE, and C.-K. Poon, "Optimal Simulation of Full Binary Trees on Faulty Hypercubes, "*IEEE Trans. on parallel and distributed systems*, Vol. 6, No. 3, pp. 269-286, March 1995.

[6] Ching-Tien Ho, Ming-Yang Kao, "Optimal Broadcast in All-Port Wormhole-Routed Hypercubes," *IEEE Trans. on Parallel and Distributed systems*, Vol. 6, No. 2, pp. 200-204, February 1995.

[7] H.P. Katseff, "Incomplete Hypercubes," *IEEE Trans. on Computers*, Vol. 37, No. 5, pp. 604-608, May 1988 .

[8] A. Kanevsky and C. Feng, "On the embedding of cycles in pancake graphs," *Parallel Computing*, Vol. 21, pp. 923-936, 1995.

[9] Haun-Chao Keh, Po-Yu Chou, Tzong-Heng Chi and Tso-Chen Hsiung, "Multicast in Incrementally Extensible Hypercube (IEH) Graphs," *Proceedings of the Int'l Conf. on Parallel and Distributed Processing Techniques and Applications*, PDPTA '96, Vol. 1, pp. 454-465.

[10] Haun-Chao Keh, Po-Yu Chou and Jen-Chih Lin, "Embedding Hamiltonian Cycles on Incrementally Extensible Hypercube Graphs," To appear in *Third Joint Conference on Information Sciences* March 3, 1997.

[11] Haun-Chao Keh, Po-Yu Chou and Jen-Chih Lin, "Embedding a Complete Binary Tree into a Faulty Supercube," To appear in *IEEE Third International Conference on Algorithms and Architectures for Parallel Processing,* Dec 8, 1997.

[12] Foster J. Provost and Rami Melhem, "A Distributed Algorithm for Embedding Trees in Hypercubes with Modefications for Run-Time Fault Tolerance," *J. Parallel Distrib. Comput.*,Vol. 14, pp. 85-89, 1992.

[13] Y. Saad, and M. Schultz, "Topological properties of Hypercube," *IEEE Trans. on Computers*, Vol. 37, No. 7, pp. 867-871, July 1988.

[14] C. Seitz, "The Cosmic Cube," *Commun. ACM*, Vol. 28, pp. 22-33, 1985.

[15] A. Sen, "Supercube: An Optimally Fault Tolerant Network Architecture," *Acta Informatica*, Vol. 26, pp. 741-748, 1989.

[16] A. Sen, A. Sengupta, S. Bandyopadhyay, "On the routing problem in faulty supercubes," *Information Processing Letters*, Vol. 42, pp. 39-46, 1992.

[17] A. Sen, A. Sengupta, S. Bandyopadhyay, "Generalized Supercube: An incrementally expandable interconnection network," *Proceedings of the Third Symposium on Frontiers of Massively Parallel Computation-Frontiers'90*, pp. 384-387, 1990.

[18] H. Sullivan, T. Bashkow, "A large scale, homogeneous, fully distributed parallel machine, I," *in Proc. 4th Symp. Computer Architecture, ACM*, pp. 105-177, March 1977.

[19] Y.-C. Tseng and D.K. Panda, "A trip-based Multicasting for Wormhole-routed Networks with Virtual Channels," *IEEE Int'l Parallel Processing Symposium*, pp. 276-283, 1993.

[20] A.Y. Wu, "Embedding of tree networks into Hypercubes," *J. Parallel Distrib. Comput.*,Vol. 2, pp. 238-249, 1985.

[21] P.-J. Yang, S.-B. Tien, and C.S. Raghavendra, "Embedding of Rings and Meshes onto Faulty Hypercube Using Free Dimensions," *IEEE Trans. on Computers*, Vol. 43, No. 5 , pp. 608-618 , May 1994 .

[22] S.-M. Yuan, "Topological properties of supercube," *Information Processing Letters*, Vol. 37, pp. 241-245, 1991.

[23] S.-M. Yuan, and H.-M. Lien, "The Shortest algorithm for Supercube," *Proceedings of National Computer Symposium*, pp. 556-561, 1991.

[24] S.-M. Yuan, "Short communication: An efficient faulty-tolerant decentralized commit protocol," *Parallel Computing*, Vol. 20, pp. 101-114, 1994.

An Optical Interconnection Structure Based on the Dual of a Hypercube

Yueming Li and S.Q. Zheng
Department of Computer Science, Louisiana State University,
Baton Rouge, LA 70803, USA
AND
Jie Wu
Department of Computer Science and Engineering,
Florida Atlantic University,
Boca Raton, FL 33431, USA

Keywords: hypercube, hypergraph, hypernetwork, interconnection network, optical bus, optical interconnection, parallel and distributed computing

Edited by:

Received: July 10, 1997 **Revised:** December 3, 1997 **Accepted:** January 25, 1998

A new class of interconnection networks, the hypernetworks, have been proposed recently. Hypernetworks are characterized by hypergraphs. Compared with point-to-point networks, they allow for increased resource-sharing and communication bandwidth utilization, and they are especially suitable for optical interconnects. One way to derive a hypernetwork is by finding the dual of a point-to-point network. Hypercube Q_n, where n is the dimension, is a very popular point-to-point network. In this article, we consider using the dual Q_n^* of hypercube of Q_n as an interconnection network. We investigate the properties of Q_n^*, and present a set of fundamental data communication algorithms for Q_n^*. Our results indicate that hypernetwork Q_n^* is a useful and promising interconnection structure for high-performance parallel and distributed computing systems.

1 Introduction

Designing high bandwidth, low latency and scalable interconnection networks is a great challenge in the construction of high-performance parallel computer systems. Traditionally, interconnection networks are characterized by graphs. Network topologies under graph models have been extensively investigated. Many network structures have been proposed, and some have been implemented. Observed the improving electrical bus and switching technologies and maturing optical interconnection technologies, Zheng pointed out the conventional graph structure is no longer adequate for the design and analysis of the new generation interconnection structures and proposed a new class of interconnection networks, the hypernetworks [14].

The class of hypernetworks is a generalization of point-to-point networks, and it contains point-to-point networks as a subclass. In a hypernetwork, the physical communication medium (a hyperlink) is accessible to multiple (usually, more than two) processors. Optical fibers and devices are suitable for implementing hyperlinks. The relaxation on the number of processors that can be connected by a link provides more design alternatives so that greater flexibilities in trade-offs of contradicting design goals are possible. The underlying graph theoretic tool for investigating hypernetworks is hypergraph theory [2]. Hypergraphs are used to model hypernetworks. Hypernetwork designs have been formulated as an optimization problem of constructing constrainted hypergraphs. Interested readers may refer to [14, 15, 17] for more justifications, design issues and implementation aspects of hypernetworks.

Existing results in hypergraph theory and combinatorial block design theory, which is closely related to hypergraph theory, can be used to design hypernetworks. For example, in [15], Zheng introduced several low diameter hypernetworks based on the concept of Steiner Triple System. In [17], Zheng and Wu proposed a scheme for constructing a new hypernetwork from an existing one using the concept of dual graph in hypergraph theory. They showed that the dual H^* of a given hypergraph H is a hypergraph that have some properties related to the properties of H so that one can investigate the properties of H^* based on the properties of H. Since the structure of H and its dual H^* can be drastically different, finding hypergraph duals can be considered as a general approach to the design of new hypernetworks. They investigated the structure of the dual K_n^* of an n-vertex complete point-to-point network K_n.

Hypercube is a popular point-to-point network which has many desirable features such as small diameter, symmetry, and supporting a large class of efficient

parallel algorithms. In this article, we propose a class of hypernetworks, Q_n^* hypernetworks. Hypernetwork Q_n^* is the dual of the n-dimensional hypercube Q_n. We discuss the topological and fault tolerance aspects of Q_n^*, and present a set of parallel data communication algorithms for Q_n^*. Our results indicate that hypernetwork Q_n^* is a useful and promising interconnection network for high-performance parallel and distributed computing systems.

This article is organized as follows. In Section 2, we introduce several basic concepts of hypergraphs and hypernetworks. In Section 3, we discuss the relations between a hypergraph and its dual, and show that hypergraph duals can be used to derive new hypernetworks. We then introduce hypernetwork Q_n^*, and show that it possesses a set of desirable properties. In Section 4 we present a set of fundamental data communication algorithms for hypernetwork Q_n^*, and analyze their performances based on the bus implementation of hyperlinks. Finally, in Section 5, we discuss the generalizations and implications of this work.

2 Preliminaries

Hypergraphs are used as underlying graph models of hypernetworks. A *hypergraph* [2] $H = (V, E)$ consists of a set $V = \{v_1, v_2, \cdots, v_n\}$ of vertices, and a set $E = \{e_1, e_2, \cdots, e_m\}$ of hyperedges such that each e_i is a non-empty subset of V and $\{v | v \in e_i, 1 \leq i \leq m\} = V$. An edge e contains a vertex v if $v \in e$. If $e_i \subseteq e_j$ implies that $i = j$, then H is a *simple hypergraph*. In this article, we only consider simple hypergraphs. When the cardinality of an edge e, denoted as $|e|$, is 1, it corresponds to a selfloop edge. If all the edges have cardinality 2, then H is a graph that corresponds to a point-to-point network. A hypergraph of n vertices and m hyperedges can also be defined by its $n \times m$ incidence matrix A with columns representing edges and rows representing vertices such that $a_{i,j} = 0$ if $v_i \notin e_j$, $a_{i,j} = 1$ if $v_i \in e_j$.

For a subset E' of E, we call the hypergraph $H'(V', E')$ such that $V' = \{v | v \in e, e \in E'\}$ the *partial hypergraph of H generated by the set E'*. For a subset U of V, we call the hypergraph $H''(V'', E'')$ such that $E'' = \{e_i \cap U | e_i \cap U \neq \phi, 1 \leq i \leq m\}$ and $V'' = \{v | v \in e, e \in E''\}$ the *sub-hypergraph induced by the set U*. Note that such an induced sub-hypergraph may or may not be a simple hypergraph.

The degree $d_H(v_i)$ of v_i in H is the number of edges in V that contain v_i. A hypergraph in which all the vertices have the same degree is said to be *regular*. The *degree of hypergraph H*, denoted by $\Delta(H)$, is defined as $\Delta(H) = \max_{v_i \in V} d_H(v_i)$. A regular hypergraph of degree k is called *k-regular hypergraph*. The *rank* $r(H)$ and *antirank* $s(H)$ of a hypergraph H is defined as $r(H) = \max_{1 \leq j \leq m} |e_j|$ and $s(H) = \min_{1 \leq j \leq m} |e_j|$, respectively. We say that H is a *uniform hypergraph* if $r(H) = s(H)$. A uniform hypergraph of rank k is called *k-uniform hypergraph*. A hypergraph is *vertex* (resp. *hyperedge*) *symmetric* if for any two vertices (resp. hyperedges) v_i and v_j (resp. e_i and e_j) there is an automorphism of the hypergraph that maps v_i to v_j (resp. e_i to e_j).

In a hypergraph H, a path of length q is defined as a sequence $(v_{i_1}, e_{j_1}, v_{i_2}, e_{j_2}, \cdots, e_{j_q}, v_{i_{q+1}})$ such that (1) $v_{i_1}, v_{i_2}, \cdots, v_{i_{q+1}}$ are all distinct vertices of H; (2) $e_{j_1}, e_{j_2}, \cdots, e_{j_q}$ are all distinct edges of H; and (3) $v_{i_k}, v_{i_{k+1}} \in e_{j_k}$ for $k = 1, 2, \cdots, q$. A path from v_i to v_j, $i \neq j$, is a path in H with its end vertices being v_i and v_j. A hypergraph is *connected* if there is a path connecting any two vertices. We only consider connected hypergraphs. A hypergraph is *linear* if $|e_i \cap e_j| \leq 1$ for $i \neq j$, i.e., two distinct buses share at most one common vertex. For any two distinct vertices v_i and v_j in a hypergraph H, the distance between them, denoted by $dis(v_i, v_j)$, is the length of the shortest path connecting them in H. Note that $dis(v_i, v_i) = 0$. The *diameter* of a hypergraph $H = (V, E)$, denoted by $\delta(H)$, is defined by $\delta(H) = \max_{v_i, v_j \in V} dis(v_i, v_j)$. More concepts in hypergraph theory can be found in [2].

A *hypernetwork* M is a network whose underlying structure is a hypergraph H, in which each vertex v_i corresponds to a unique processor P_i of M, and each hyperedge e_j corresponds to a *connector* that connects processors represented by the vertices in e_j. A connector is loosely defined as an electronic or a photonic component through which messages are transmitted between connected processors, not necessarily simultaneously. We call a connector a *hyperlink*.

Unlike a point-to-point network, in which a link is dedicated to a pair of processors, a hyperlink in a hypernetwork is shared by a set of processors. A hyperlink can be implemented by a bus or a crossbar switch. Current optical technologies allow a hyperlink to be implemented by optical waveguides in a folded-bus using time-division multiplexing (TDM). Free-space optical or optoelectronic switching devices such as bulk lens, microlens array, and spatial light modulator (SLM) can also be used to implement hyperlinks. A star coupler, which uses wavelength-division multiplexing (WDM), can be considered either as a generalized bus structure or as a photonic switch, is another implementation of a hyperlink. Similarly, an ATM switch, which uses a variant TDM, is a hyperlink. In the rest of this article, the following pairs of terms are used interchangeably: (hyper)edges and (hyper)links, vertices and processors, point-to-point networks and graphs, and hypernetworks and hypergraphs.

The problem of designing efficient interconnection networks can be considered as a constrainted optimization problem. For example, the goal of designing point-to-point networks is to find well-structured

graphs (whose ranks are fixed, as a constant 2) with small degrees and diameters. In hypernetwork design, the relaxation on the number of processors that can be connected by a hyperlink (i.e. the rank of the hyperlink) provides more design alternatives so that greater flexibilities in trade-offs of contradicting design goals are possible. The detailed discussion is beyond the scope of this article.

3 Dual Hypernetworks and Q_n^* Hypernetworks

The *dual* of a hypergraph $H = (V, E)$ with vertex set $V = \{v_1, v_2, \cdots, v_n\}$ and hyperedge set $E = \{e_1, e_2, \cdots, e_m\}$ is a hypergraph $H^* = (V^*, E^*)$ with vertex set $V^* = \{v_1^*, v_2^*, \cdots, v_m^*\}$ and hyperedge set $E^* = \{e_1^*, e_2^*, \cdots, e_n^*\}$ such that v_j^* corresponds to e_j with hyperedges $e_i^* = \{v_j^* | v_i \in e_j \text{ and } e_j \in E\}$. In other words, H^* is obtained from H by interchanging of vertices and hyperedges in H. The incidence matrix of H^* is the transpose of the incidence matrix of H. Thus, $(H^*)^* = H$. The following relations between a hypergraph and its dual are apparent [17].

Proposition 1 *H is r-uniform if and only if H^* is r-regular.*

Proposition 2 *The dual of a linear hypergraph is also linear.*

Proposition 3 *A hypergraph H is vertex symmetric if and only if H^* is hyperedge symmetric .*

Proposition 4 *The dual of a sub-hypergraph of H is a partial hypergraph of the dual hypergraph H^*.*

Since $(H^*)^* = H$, all the above propositions still hold after interchanging H with H^*.

Proposition 5 $\delta(H) - 1 \leq \delta(H^*) \leq \delta(H) + 1$.

Propositions 1 - 5 show that some properties of the dual hypergraph H^* of a given hypergraph H can be derived from properties of H. For example, if H is a ring, then H^* is isomorphic to H. However, in general, the structures of H and its dual H^* can be drastically different. Finding hypergraph duals can be considered as a general approach to the design of new hypernetworks.

We consider using the dual Q_n^* of the hypercube Q_n as a hypernetwork. An n-dimensional hypercube Q_n consists of 2^n vertices, each being labeled by a unique n-bit binary number. Two vertices are connected by an edge if and only if their binary labels are distinct in one bit position. Properly labeling the vertices and hyperedges in Q_n^* can greatly simplify its use as a communication network. Vertex labels are used as processor addresses. Similarly, hyperedge labels are used as the unique names of hyperlinks.

Let I_n be the set of non-negative integers that can be represented by n-bit binary numbers. For $l, u \in I_n$, we use $d(l, u)$ to denote the number of different bits in the binary representations of l and u, i.e. $d(l, u)$ is the Hamming distance between the binary representations of l and u. We use a pair of integers to label a vertex in hypernetworkQ_n^*.

Definition 1 Let $N_n = n2^{n-1}$ for $n \geq 2$. The Q_n^* hypernetwork is a hypergraph with vertex set $\{\langle l, u\rangle | l, u \in I_n,\ l < u, \text{ and } d(l, u) = 1\}$ of N_n vertices and 2^n hyperlinks, $e_0, e_1, \cdots, e_{2^n-1}$. Each vertex $\langle l, u\rangle$ is contained in exactly two hyperedges e_l and e_u.

Example 1 *The incidence matrix A of Q_3^* is*

	e_0	e_1	e_2	e_3	e_4	e_5	e_6	e_7
⟨0,1⟩	1	1	0	0	0	0	0	0
⟨0,2⟩	1	0	1	0	0	0	0	0
⟨1,3⟩	0	1	0	1	0	0	0	0
⟨2,3⟩	0	0	1	1	0	0	0	0
⟨0,4⟩	1	0	0	0	1	0	0	0
⟨1,5⟩	0	1	0	0	0	1	0	0
⟨2,6⟩	0	0	1	0	0	0	1	0
⟨3,7⟩	0	0	0	1	0	0	0	1
⟨4,5⟩	0	0	0	0	1	1	0	0
⟨4,6⟩	0	1	0	0	1	0	1	0
⟨5,7⟩	0	0	0	0	0	1	0	1
⟨6,7⟩	0	0	0	0	0	0	1	1

A^T, the transpose of A, is

	$e_{0,1}$	$e_{0,2}$	$e_{1,3}$	$e_{2,3}$	$e_{0,4}$	$e_{1,5}$	$e_{2,6}$	$e_{3,7}$	$e_{4,5}$	$e_{4,6}$	$e_{5,7}$	$e_{6,7}$
v_0	1	1	0	0	1	0	0	0	0	0	0	0
v_1	1	0	1	0	0	1	0	0	0	0	0	0
v_2	0	1	0	1	0	0	1	0	0	0	0	0
v_3	0	0	1	1	0	0	0	1	0	0	0	0
v_4	0	0	0	0	1	0	0	0	1	1	0	0
v_5	0	0	0	0	0	1	0	0	1	0	1	0
v_6	0	0	0	0	0	0	1	0	0	1	0	1
v_7	0	0	0	0	0	0	0	1	0	0	1	1

Clearly, A^T is the incidence matrix of the hypercube Q_3. Figure 1 shows the bus implementation of hypernetwork Q_3^*, whose incidence matrix A is given above. Its corresponding hypercube, whose incidence matrix is A^T, is shown in Figure 2, where each edge is labeled by its two end vertices.

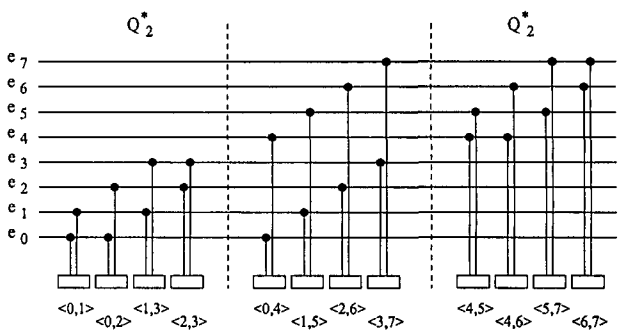


Figure 1: Bus implementation of Q_3^*.

□

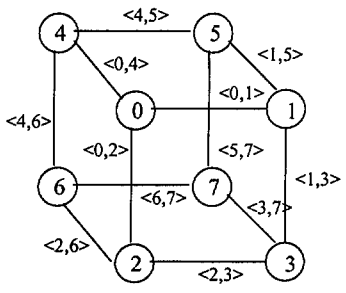


Figure 2: Hypercube Q_3 corresponding to Q_3^*.

The Q_n^* hypernetwork can also be defined in a recursive way. One can easily observe that Q_{n+1}^* can be constructed using two copies of Q_n^* and 2^n additional vertices (see Figures 1 and 2). For brevity, we omit the recursive definition of Q_n^*.

Based on the properties of hypercube Q_n and Propositions 1 to 4, we have the following fact:

Fact 1: Q_n^* is 2-regular, n-uniform, linear, and vertex and hyperedge symmetric.

Since the diameter of Q_n is n, Property 5 indicates that the diameter of Q_n^* is at most $n+1$. We show that the diameter of Q_n^* is also n. Let $dis(\langle l,u\rangle, \langle l',u'\rangle)$ denote the distance between vertices $\langle l,u\rangle$ and $\langle l',u'\rangle$ in Q_n^*.

Lemma 1 *For any two vertices $\langle l,u\rangle$ and $\langle l',u'\rangle$ in Q_n^*, $dis(\langle l,u\rangle, \langle l',u'\rangle) = \min\{d(l,l'), d(l,u'), d(u,l'), d(u,u')\} + 1$.*

Proof. We view $\langle l,u\rangle$ and $\langle l',u'\rangle$ as two edges in Q_n. The minimal path connecting these two edges is one from one end node of $\langle l,u\rangle$ to one end node of $\langle l',u'\rangle$. Therefore, the distance is the length of a minimal path between two end nodes plus one. □

Lemma 2 *For any two vertices $\langle l,u\rangle$ and $\langle l',u'\rangle$ in Q_n^*, $dis(\langle l,u\rangle, \langle l',u'\rangle) \leq n$.*

Proof. For two hyperedges $\langle l,u\rangle$ and $\langle l',u'\rangle$ in Q_n^*, if $d(l,l') = n$ then $d(l,u') < n$. By Lemma 1, $d(\langle l,u\rangle, \langle l',u'\rangle) = \min\{d(l,l'), d(l,u'), d(u,l'), d(u,u')\} + 1 \leq \min\{d(l,l'), d(l,u')\} + 1 \leq (n-1)+1 \leq n$. □

Theorem 1 *The diameter of Q_n^* is n.*

Proof. From lemma 2, we know that the diameter is less than or equal to n. All we need to do is to prove that there are two vertices $\langle l,u\rangle$ and $\langle l',u'\rangle$ such that $d(\langle l,u\rangle, \langle l',u'\rangle) = n-1$. Actually, it is easy to see that vertices $\langle 00...0, 100...0\rangle$ and $\langle 111...1, 011...1\rangle$ meet the above condition. □

In Q_n^*, the number of processors is $n/2$ times the number of the hyperlinks. Each processor is attached to exactly two hyperlinks, and this simplifies the processor interface circuit design. Each hyperlink connects n processors. Suppose that a 10×10 crossbar switch is implementable and cost effective, then Q_{10}^* of 5,120 processors can be implemented using 1,024 such switches as hyperlinks.

Consider the fault tolerance aspect of hypernetwork Q_n^*. We say that a hypernetwork H is *x-processor fault-tolerant* (resp. *y-hyperlink fault-tolerant*) if it remains connected when no more than any x processors (resp. y-hyperlinks) are removed. We have the following claim.

Theorem 2 *Q_n^* is $(2n-3)$-processor fault-tolerant and 1-hyperlink fault-tolerant.*

Proof. Consider hypercube Q_n. After taking an edge e and deleting all edges that share a vertex with e, the resulting graph becomes disconnected. This implies that it is possible to disconnect hypernetwork Q_n^* by removing $2(n-1)$ processors. However, removing any less than $2(n-1)$ processors from Q_n will not make the remaining part of Q_n disconnected. Hence, Q_n^* is $(2n-3)$-processor fault-tolerant. Since Q_n^* is 2-regular, removing any two hyperlinks that share a vertex v will disconnect processor v. □

4 Data Communication Algorithms for Q_n^*

In this section, we use the vertex and hyperedge labels to design data communication algorithms for hypernetwork Q_n^*. For simplicity, we assume bus implementation of hyperlinks. In the electronic domain, the bus load, i.e. the number of processors that can be connected by a bus, is limited. Using optical fibers to implement a bus, the bus load can be increased significantly. Recently, optical bus architectures have received considerable attention (e.g. [5, 8, 9, 10, 12, 16]). Since a bus is shared by all its connected processors, the performance of a bus depends on the way it is accessed by processors. For example, one way for processors to share a bus is to use time-division multiplexing (TDM), which allocates time slots to processors so that they can only access the bus during their slots. Another way is to let processors compete for bus tenure, and use an arbiter to grant bus tenure in an on-line fashion.

We assume a synchronous mode communication. Bus allocations, although operated dynamically, are predetermined by an off-line scheduling algorithm. This bus operational mode has been used in [3] for analyzing a multiple-bus interprocessor connection structure. Assume that all messages are of the same length. The communication performance is measured in terms of parallel message steps. We adopt these assumptions for two reasons. First, under these assumptions,

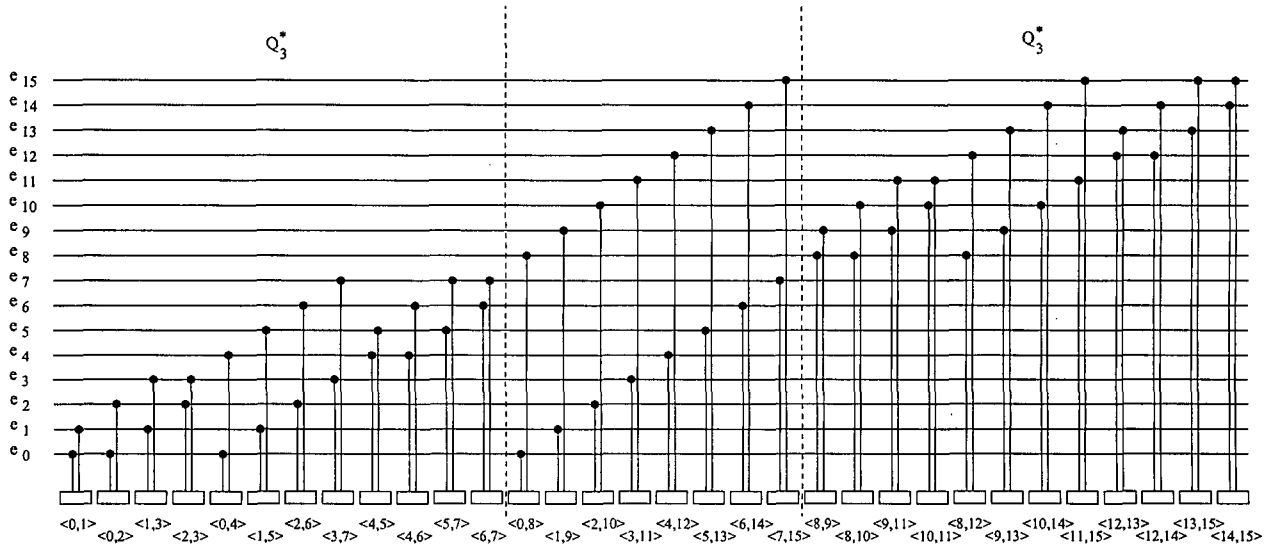


Figure 3: Bus implementation of Q_4^*.

it is easier to assess the capability and limitation of the proposed hypernetwork structure. Secondly, the performance results obtained can be easily used to measure other bus communication method by either adding additional overheads, which may incur in TDM transmission and asynchronous bus allocation, or deducting latency saving due to pipelining effect of a pipelined optical bus.

We discuss four types of communication operations: one-to-one communications, one-to-many communications, many-to-one communications and many-to-many communications. For each type, we present an algorithm for a representative communication operation. These communication algorithms constitute a useful set of tools for designing parallel algorithms on hypernetwork Q_n^*.

4.1 One-to-One Communications

We consider shortest path routing between two processors. We use $\langle l, u\rangle$ and $\langle l', u'\rangle$ to represent the source processor and the destination processor, respectively. The idea of the shortest path routing is as follows. If the two processors share one hyperlink, the message is transmitted through that hyperlink. Otherwise, the transmission is done from hyperlink e_a toward hyperlink e_b such that $a \in \{l, u\}$, $b \in \{l', u'\}$, and $d(a, b)$ is the minimal. The source processor sends the message to the processor $\langle a, c\rangle$ through hyperlink e_a such that $d(c, b) = d(a, b) - 1$. This process is recursive; that is, the processor $\langle a, c\rangle$ will then relay the message toward the processor $\langle l', u'\rangle$. The following is the shortest path routing algorithm.

procedure $ROUTE(\langle l, u\rangle, \langle l', u'\rangle)$
begin
  Let $a \in \{l, u\}$, $b \in \{l', u'\}$ such that
    $d(a, b) = dis(\langle l, u\rangle, \langle l', u'\rangle) - 1$;
  if $a = b$ **then** processor $\langle l, u\rangle$ sends the
    message to $\langle l', u'\rangle$ using e_a
  else begin
    Select c such that $d(a, c) = d(a, b) - 1$;
    processor $\langle l, u\rangle$ sends the message to
      $\langle a, c\rangle$ using e_a;
    $ROUTE(\langle a, c\rangle, \langle l', u'\rangle)$
    end
end

Theorem 3 *For any given pair of processors $\langle l, u\rangle$ and $\langle l', u'\rangle$ in hypernetwork Q_n^*, algorithm ROUTE routes a message from $\langle l, u\rangle$ to $\langle l', u'\rangle$ along a shortest path in $dis(\langle l, u\rangle, \langle l', u'\rangle)$ message steps.*

Proof. The theorem directly follows from Lemma 1 and Lemma 2. □

4.2 One-to-Many Communication

We consider broadcasting a message from any processor $\langle l, u\rangle$ to all other processors in Q_n^*, By Proposition 3, we only need to consider the case that $\langle l, u\rangle = \langle 0, 1\rangle$.

Let $Q_k = \underbrace{0..0}_{n-k-1}\ 0\ \underbrace{*..*}_{k}$ and $Q_k' = \underbrace{0..0}_{n-k-1}\ 1\ \underbrace{*..*}_{k}$ denote the k-dimensional subcube of Q_n induced by all vertices whose left $n-k$ bits are $\underbrace{00..0}_{n-k-1}$ 0 and $\underbrace{00..0}_{n-k-1}$ 1, respectively. Here, an $*$ in a bit position stands for "don't care". We use $Q_k + Q_k'$ to denote the $(k+1)$-dimensional subcube of Q_n induced by vertices in Q_k and Q_k'. As a shorthand, we write $Q_{k+1} = Q_k + Q_k'$. We explain the broadcasting algorithm in Q_n^* using an n-dimensional hypercube (Q_n) by interchanging the

role of vertices and edges. The idea behind our broadcasting algorithm is as follows. Assuming that initially the edge connecting vertices $00\ldots00$ and $00\ldots01$ in Q_n is colored, and all other edges in Q_n are not colored. We want to edge traversing algorithm A which systematically traverses all edges in Q_n in n steps. In the k-th step, algorithm A select a subset E_k of edges in Q_n that satisfy the following conditions: (1) all edges in E_k are not previously traversed, (2) each edge in E_k has at least one end vertex that is an end vertex of a previously colored edge, and (3) for each edge e in in E_k assign a direction it is traversed: let u and v be the two end vertices of e, and suppose that u is an end vertex of a previously traversed edge, then traverse e from u to v. By Theorem 1, (1), (2), and (3) all edges of Q_n are guaranteed to be traversed in n parallel steps. Obviously, such an algorithm A corresponds to a broadcasting algorithm A^* for Q_n^*. By Theorem 1, (1) and (2) all edges of Q_n guaranteed to be colored in n parallel steps.

Now, let us describe our algorithm A. Starting from Q_1 (which corresponds to the edge connecting vertices 00...0 and 00...1 in Q_n, and the source vertex $\langle 00...0, 00...1\rangle$ in Q_n^*), increase the dimension of the cube by one in each step. For convenience, we use $Q'_{k,0} = \underbrace{00..0}_{n-k-1}\ 10\underbrace{*\,*\,..*}_{k-1}$ and $Q'_{k,1} = \underbrace{00..0}_{n-k-1}\ 11\underbrace{*\,*\,..*}_{k-1}$ to denote the two $(k-1)$-dimensional subcubes of Q'_k induced by vertices whose left $n-k+1$ bits are $\underbrace{00..0}_{n-k-1}\ 10$ and $\underbrace{00..0}_{n-k-1}\ 11$, respectively.

In the first step, we consider $Q_2 = Q_1 + Q'_1$, the two edges connecting vertices in Q_1 and Q'_1 are colored. In the second step, the edge connecting $Q'_{1,0}$ and $Q'_{1,1}$ is traversed in the direction from $Q'_{1,0}$ to $Q'_{1,1}$, and the edges connecting Q_2 and Q'_2 are traversed in the direction from Q_2 and Q'_2. Assume that after k, $1 \le k \le n-2$, steps, all the edges in $Q_k = \underbrace{00..0}_{n-k-1}\ 0\underbrace{*\,*\,..*}_{k}$ and the ones connecting Q_k and $Q'_k = \underbrace{00..0}_{n-k-1}\ 1\underbrace{*\,*\,..*}_{k}$ have been traversed, but all the edges in Q'_k have not been traversed. In step $k+1$, we traverse all the edges in Q'_k in the direction from $Q'_{k,0}$ to $Q'_{k,1}$, and the edges connecting Q_{k+1} and Q'_{k+1}, where $Q_{k+1} = \underbrace{00..0}_{n-k-2}\ 0\underbrace{*\,*\,..*}_{k+1}$ and $Q'_{k+1} = \underbrace{00..0}_{n-k-2}\ 1\underbrace{*\,*\,..*}_{k+1}$, in the direction from Q_{k+1} to Q'_{k+1}. In step n, we only need to traverse all the edges in Q'_{n-1} in the direction from $Q'_{n-1,0}$ to $Q'_{n-1,1}$. Let $b_{n-1}b_{n-2}\cdots b_0$ be the binary representation of b. We use $b^{(i)}$ to represent the binary number (and its corresponding decimal value) obtained by complementing the ith bit, b_i, of the binary representation of b. Translating the above hypercube edge traverse algorithm into an algorithm for traversing vertices in Q_n^*, we obtain the following algorithm.

procedure *BROADCAST*($\langle 0,1\rangle$)
begin
  for $k = 1$ **to** $n-1$ **do**
    for all $\langle a,b\rangle$ where $a \in \underbrace{00..0}_{n-k}0\underbrace{*\,*\,..*}_{k-1}$ and $b \in \underbrace{00..0}_{n-k}1\underbrace{*\,*\,..*}_{k-1}$ **do in parallel**
      processor $\langle a,b\rangle$ sends the message to $\langle a, a^{(k+1)}\rangle$ using e_a;
      processor $\langle a,b\rangle$ sends the message to $\langle b, b^{(k+1)}\rangle$ using e_b;
      processor $\langle a,b\rangle$ sends the message to $\langle b, b^{(i)}\rangle$ using e_b, if $a \ge 2^{k-1}$, $b \ge 2^{k-1}$, and $b^{(i)} > b$ for $i \in \{0,1,...,k-1\}$
    endfor
  endfor
  for all $\langle a,b\rangle$ **do in parallel**
    if $a \ge 2^{n-1}$, $b \ge 2^{n-1}$ and $b < b^{(i)}$ for $i \in \{0,1,...,k-1\}$ **then** processor $\langle a,b\rangle$ sends the message to $\langle b, b^{(i)}\rangle$ using e_b
  endfor
end

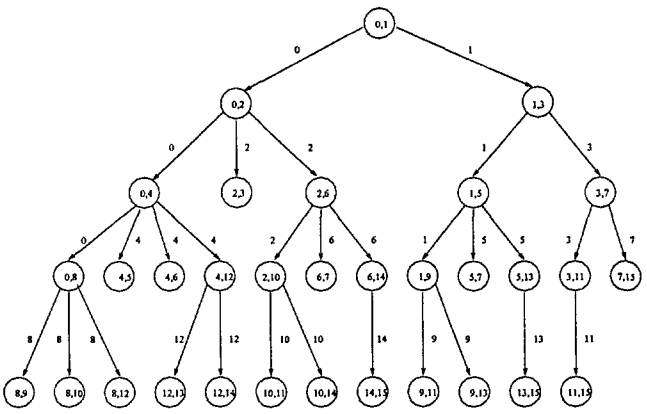


Figure 4: Data communication pattern for broadcasting from $\langle 0,1\rangle$ in Q_4^*.

In Figure 4, we show the broadcasting tree for Q_4^*, whose bus implementation is shown in Figure 3. In Figure 4, a circle labeled by a pair of integers a and b represents a processor $\langle a,b\rangle$. A directed edge labeled by an integer c from $\langle a,b\rangle$ to $\langle a',b'\rangle$ indicates that the message is transmitted from $\langle a,b\rangle$ to $\langle a',b'\rangle$ using hyperlink e_c.

Theorem 4 *Assuming bus hyperlinks of Q_n^*, algorithm BROADCAST broadcasts a message from processor $\langle 0,1\rangle$ to all other processors in n parallel message transmission steps.*

Proof. The theorem follows directly from a simple induction based on the discussion predeeding *BROADCAST*. □

4.3 Many-to-One Communication

A reduction (or census, or fan-in) function is defined as a commutative and associative operation on a set of values, such as finding maximum, addition, logic or, etc. It can be carried out using a many-to-one communication operation. We only consider the case that the specified reduction operation is addition. The same algorithm can be slightly modified to perform other reduction operations.

We present an algorithm that can be used to perform a summation on a set of N_n values stores in the A registers of processors, one per processor, and putting the final result in processor $\langle 0,1\rangle$. That is, the algorithm computes $\sum_{\langle a,b\rangle \in Q_n^*} A_{\langle a,b\rangle}$, and putting the final result in $A_{\langle 0,1\rangle}$ of processor $\langle 0,1\rangle$ We assume that each processor $\langle a,b\rangle$ has a working register $B_{\langle a,b\rangle}$.

Summation is done in two phases. In the first phase, a set of 2^{n-1} partial sums are obtained and stored in processors $\langle z, 2^{n-1}+z\rangle$, $0 \le z \le 2^{n-1}-1$. The second phase computes the sum of these partial sums and stored the final result in $\langle 0,1\rangle$.

procedure *REDUCTION*($\langle 0,1\rangle$)
begin
/* *phase 1* */
for $i = 1$ **to** $n-1$ **do**
for all $\langle a,b\rangle$ **do in parallel**
if $a_i a_{i-1} = 00$ and $b_i b_{i-1} = 01$ **then**
processor $\langle a,b\rangle$ sends $A_{\langle a,b\rangle}$ from $\langle a,b\rangle$ to $\langle a,(b^{(i)})^{(i-1)}\rangle$ using e_a;
if $a_i a_{i-1} = 10$ and $b_i b_{i-1} = 11$ **then**
processor $\langle a,b\rangle$ sends $A_{\langle a,b\rangle}$ from $\langle a,b\rangle$ to $\langle (a^{(i)})^{(i-1)},b\rangle$ using e_b;
if $\langle a,b\rangle$ received a value **then** store this value in $B_{\langle a,b\rangle}$ and perform
$A_{\langle a,b\rangle} := A_{\langle a,b\rangle} + B_{\langle a,b\rangle}$
endfor
endfor
/* phase 2 */
for $i = n-1$ **down to** 1 **do**
for all $\langle a,b\rangle$ **do in parallel**
if $a = \underbrace{00\ldots0}_{n-i-1}\,00\,\underbrace{**\ldots*}_{i-1}$ and
$b = \underbrace{00\ldots0}_{n-i-1}\,10\,\underbrace{**\ldots*}_{i-1}$ **then**
processor $\langle a,b\rangle$ sends $A_{\langle a,b\rangle}$ from $\langle a,b\rangle$ to $\langle a,(b^{(i)})^{(i-1)}\rangle$ using e_a;
if $a = \underbrace{00\ldots0}_{n-i-1}\,01\,\underbrace{**\ldots*}_{i-1}$ and
$b = \underbrace{00\ldots0}_{n-i-1}\,11\,\underbrace{**\ldots*}_{i-1}$ **then**
processor $\langle a,b\rangle$ sends $A_{\langle a,b\rangle}$ from $\langle a,b\rangle$ to $\langle (b^{(i)})^{(i-1)},a\rangle$ using e_a;
if $\langle a,b\rangle$ received two values **then**
store one value in $A_{\langle a,b\rangle}$ and the other in $B_{\langle a,b\rangle}$, and perform
$A_{\langle a,b\rangle} := A_{\langle a,b\rangle} + B_{\langle a,b\rangle}$
endfor
endfor
end

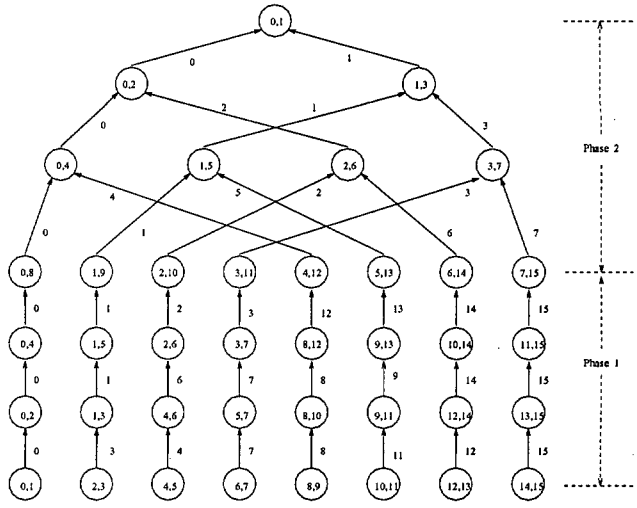


Figure 5: Data communication pattern for reduction in Q_4^*.

Theorem 5 *Assuming bus hyperlinks of Q_n^*, algorithm REDUCTION carries out a reduction operation in $2(n-1)$ parallel message transmission steps.*

Proof. First, we claim that at the end of the first phase the sum of the partial sums stored in processors $\langle z, 2^{n-1}+z\rangle$, $0 \le z \le 2^{n-1}-1$, is the final sum. It is easy to verify that the claim is true for $n = 2$ and $n = 3$. Suppose that the claim is true for $n = k$, and consider the case $n = k+1$. By the algorithm, any processor $\langle a,b\rangle$ such that $a_{k+1}a_k = 00$ and $b_{k+1}b_k = 01$ has not sent and receive any value before the k-th step (iteration). Furthermore, by the hypothesis, the sum of the partial sums stored in processors $\langle z, 2^{k-1}+z\rangle$, $0 \le z \le 2^{k-1}-1$, is the sum of the values originally stored in sub-hypernetwork Q_k^* induced by all vertices $\langle a,b\rangle$ in Q_{k+1}^* such that $a < 2^k$ and $b < 2^k$. Consider one more step (i.e. the k-th iteration) of the first phase. In this step the partial sum stored in $\langle z, 2^{k-1}+z\rangle$, $0 \le z \le 2^{k-1}-1$, is sent to processor $\langle z, 2^k+z\rangle$, $0 \le z \le 2^{k-1}-1$, using hyperlink e_z. By the symmetry, the sum of the partial sums stored in processors $\langle 2^k+z, 2^k+2^{k-1}+z\rangle$, $0 \le z \le 2^{k-1}-1$, is the sum of the values originally stored in sub-hypernetwork Q_k^* induced by all vertices $\langle a,b\rangle$ in Q_{k+1}^* such that $a \ge 2^k$ and $b \ge 2^k$ and after the k-th step, the partial sum stored in

$\langle 2^k+z, 2^k+2^{k-1}+z\rangle$, $0 \le z \le 2^{k-1}-1$, is sent to processor $\langle 2^{k-1}+z, 2^k+2^{k-1}+z\rangle$, $0 \le z \le 2^{k-1}-1$, using hyperlink $e_{2^k 2^{k-1}+z}$. Therefore, after performing parallel additions in processors $\langle z, 2^k+z\rangle$, $0 \le z \le 2^k-1$, we obtain the claimed 2^k partial sums for Q^*_{k+1}. This completes the induction.

Now, we claim that the second phase computes the sum of the partial sums stored in processors $\langle z, 2^{n-1}+z\rangle$, $0 \le z \le 2^{n-1}-1$, and store the final result in $\langle 0,1\rangle$ of Q^*_n. This claim is true for $n=2$ and $n=3$. Suppose that the claim is true for $n=k$, and consider the case $n=k+1$. In the first iteration ($i=k$), each processor $\langle z, 2^{k-1}+z\rangle$, $0 \le z \le 2^{k-1}-1$, receives two partial sums, one from $\langle z, 2^k+z\rangle$ via e_z, and the other from $\langle 2^{k-1}+z, 2^k+2^{k-1}+z\rangle$ via $e_{2^{k-1}+z}$. After additions, 2^{k-1} partial sums are obtained and stored in processors $\langle z, 2^{k-1}+z\rangle$, where $0 \le z \le 2^{k-1}-1$. Then, the induction hypothesis guarantees that after $k-1$ more steps the final result will be stored in processor $\langle 0,1\rangle$. This completes the proof of the claim for phase 2, and the proof of the theorem. □

In Figure 5, we show the communication pattern used by *REDUCTION* on Q^*_4, whose bus implementation is shown in Figure 3. As Figure 4, in this figure, a circle labeled by a pair of integers a and b represents a processor $\langle a,b\rangle$. A directed edge labeled by an integer i from $\langle a,b\rangle$ to $\langle a',b'\rangle$ indicates that the message is transmitted from $\langle a,b\rangle$ to $\langle a',b'\rangle$ using hyperlink e_i.

4.4 Many-to-Many Communication

We consider a general case, the all-to-all communication. In an all-to-all communication, each processor sends a message to all the other processors. It is also called the *total exchange* operation.

We can obtain a total exchange communication algorithm by modifying algorithm *REDUCTION*. The operator used is set union instead of addition. After applying *REDUCTION*, all messages are collected at processor $\langle 0,1\rangle$. Then, by applying *BROADCAST*, processor $\langle 0,1\rangle$ broadcasts the N_n messages to all remaining processors of Q^*_n. By Theorem 5, the second phase alone takes nN_n parallel message steps. Note that the lower bound for the time of a total exchange operation on Q^*_n is $\Omega(N_n)$, and clearly, this algorithm is not efficient.

If what follows, we present an algorithm *TOTAL_EXCHANGE* which takes $O(N_n)$ message steps to perform the total exchange operation on Q^*_n. Algorithm *TOTAL_EXCHANGE* is an all-port algorithm, i.e. the two I/O ports of each processor may participate in a message transmission step. However, each port performs either a send operation or receive operation, but not both. This algorithm can be easily converted to a single-port algorithm with the same communication complexity.

For convenience, we define that a processor $\langle i,j\rangle$ is of dimension k, $0 \le k \le n-1$, if $j-i=2^k$. It is easy to verify the following facts: (i) There are exactly 2^{n-1} processors of dimension k, $0 \le k \le n-1$, in Q^*_n; (ii) There is exactly one processor of dimension k, $0 \le k \le n-1$, attached to each hyperlink in Q^*_n; and (iii) Any two processors of the same dimension are not attached to the same hyperlink in Q^*_n.

```
procedure TOTAL_EXCHANGE
begin
  for all hyperlinks e_h do in parallel
    All processors attached to e_h sends its
    message to the processor of dimension 0
    that is attached to e_h using e_h;
  endfor
  Let the set of messages received by each
  processor ⟨i,j⟩ be denoted by M_⟨i,j⟩;
  for k = 0 to n − 1 do
    for all processors ⟨i,j⟩ of dimension k
    do in parallel
      Broadcast M_⟨i,j⟩ to all processors
      attached to hyperlink e_i and e_j
    endfor
    Let the set of messages received by each
    processor ⟨i,j⟩ be denoted by M_⟨i,j⟩;
  endfor
end
```

The correctness of this algorithm can be verified by the following induction. It is easy to see that the algorithm is correct for Q^*_4. Suppose that the algorithm is correct for $n=m$, and consider the case of $n=m+1$. After m iterations of the **for** loop, the total exchange operations are performed with respect to the subhypergraph of Q^*_{m+1} induced by vertices $\langle i,j\rangle$ such that $i < 2^m$ and $j < 2^m$, and the subhypergraph of Q^*_{m+1} induced by vertices $\langle i',j'\rangle$ such that $i' \ge 2^m$ and $j' \ge 2^m$. In addition, dimension m processors have received all messages in Q^*_{m+1}. In one additional iteration, each processor $\langle a,b\rangle$ of dimension m broadcasts all its received messages to processors attached to hyperlinks e_a and e_b. Then, by (i), (ii) and (iii), the total exchange operation is performed with respect to Q^*_{m+1}.

Now, let use analyze the performance of *TOTAL_EXCHANGE*. In our algorithm, we assume that we a processor broadcasts a set of messages, it broadcasts all messages it received in the previous step. As a consequence, duplicated messages are broadcast. We show that even with duplicated messages, the performance of *TOTAL_EXCHANGE* is within a constant factor of the optimal. In the first **for** statement, $2(n-1)$ messages are collected by each dimension 0 processor $\langle a,b\rangle$ using two hyperlinks e_a and e_b, and this takes $(n-1)$ message steps. Consider the **for** loop. It has n iterations. In the first iteration, $2n$ messages are broadcast from each dimension 0 proces-

sor $\langle a, b\rangle$ to all processors attached to e_a and e_b. In the second iteration, $4n$ messages are broadcast are broadcast from each dimension 1 processor $\langle a, b\rangle$ to all processors attached to e_a and e_b. In general, in the iteration with $k = m$, $2^{m+1}n$ messages need to be broadcast by each dimension m processor $\langle a, b\rangle$ to all processors attached to e_a and e_b. By (ii) and (iii) above, the iteration with $k = m$ takes no more than $2^{m+1}n$ message steps. Therefore, *TOTAL_EXCHANGE* requires no more than $(n-1)+(2+4+8+\cdots+2^n)n = (2^{n+1}-1)n-1$ parallel message steps. We summarize this analysis by the following claim.

Theorem 6 *Assuming bus hyperlinks of Q_n^*, algorithm TOTAL_EXCHANGE carries out a total exchange operation in $4N_n - n - 1$ parallel message transmission steps.*

5 Discussions

We proposed a new class of hypernetworks baseed on the duals of hypercubes. The structures of Q_n^* and Q_n are quite different, but as we showed, many properties of Q_n^* can be directly derived from the properties of Q_n. The Q_n^* hypernetwork is suitable for exploiting the high bandwidths provided by new interconnection technologies such as optical fiber or devices. We presented a set of basic data communication algorithms for Q_n^* based on bus implementation of hyperlinks. Algorithms *ROUTE* and *BROADCAST* are optimal, and algorithms *REDUCTION* and *TOTAL_EXCHANGE* are optimal within a constant factor.

Our algorithms are closely related to the ideas behind their corresponding algorithms on the hypercube network. This leads us to a pose an open problem: is there a simulation scheme that can be used to simulates Q_n by Q_n^* efficiently? If such a scheme can be found, then all previously known hypercube algorithms can be automatically translated to algorithms for a machine using Q_n^* as the interconnection network.

Using the hypergraph dual concept, one can obtain another class of hypernetworks that contains the duals of the star graphs. The n-star graph S_n (refer to [1] for its definition) is a point-to-point network that has $n!$ vertices, $n(n-1)!/2$ edges, and its degree and the diameter are $n-1$ and $\lfloor 3(n-1)/2 \rfloor$, respectively. S_n is vertex and edge symmetric. Therefore, S_n^* has $n!(n-1)/2$ vertices and $n!$ hyperedges; S_n^* is 2-regular, $(n-1)$-uniform, linear, and vertex and hyperedge symmetric; and the diameter of S_n^* is no greater than $\lfloor (3n-1)/2 \rfloor + 1$, respectively. Both diameter and degree of S_n^* are sub-logarithmic functions of the number of processors and the number of hyperlinks in S_n^*. Compared with Q_n^*, S_n^* has some advantages. The topological and communication aspects of S_n^* hypernetworks deserve further investigations.

References

[1] S. Akers, D. Harel, and B. Krishnamurthy, The Star Graph: an Attractive Alternative to the n-Cube, *Proceedings of 1987 International Conference on Parallel Processing*, pp. 393-400, 1987.

[2] C. Berge *Hypergraphs*, North-Holland, 1989

[3] O.M. Dighe, R. Vaidyanathan, and S.Q. Zheng, The Bus-Connected Ringed Tree: A Versatile Interconnection Network, to appear in *Journal of Parallel and Distributed Computing.*

[4] P. E. Green, Jr., *Fiber Optical Networks*, Prentice Hall, 1993.

[5] Z. Guo , R. Melhem, R. Hall, D.Chiarulli and S. Levitan, Array Processors with Pipelined Optical busses, *Journal of Parallel and Distributed Computing*, **12**(3), pp. 269-282, 1991.

[6] J. Jahns and S.H. Lee (editors), *Optical Computing Hardware*, Academic Press, Inc., 1994.

[7] F. T. Leighton, *Introduction to Parallel Algorithms and Architectures: Arrays · Trees · Hypercube* , Morgan Kaufmann Publishers, Inc., 1992, pp. 78-82, 239-244.

[8] Y. Li, Y. Pan and S.,Q. Zheng, A Pipelined TDM Optical Bus with Conditional Delays", to appear in *Optical Engineering.*

[9] R. Melhem, D. Chiarulli, and S. Levitan, Space Multiplexing of Waveguides in Optically Interconnected Multiprocessor Systems, *Computer Journal*, **32**(4), pp.362-369, 1989.

[10] Y. Pan and K. Li, Linear Array with a Reconfigurable Pipelined Bus System – Concepts and Applications, *Proceedings of 1996 International Conference on Parallel and Distributed Processing Techniques and Applications*, pp. 1431-1442, 1996.

[11] C. Qiao and R. Melhem, Time-division Optical Communications in Multiprocessor Arrays, *IEEE Trans. on Computers*, **42**(5), pp. 577-590, 1993.

[12] C. Qiao, R. Melhem, D. Chiarulli and S. Levitan, Optical Multicasting in Linear Arrays, *International Journal of Optical Computing*, **2**(1), pp. 31-48, 1991.

[13] C. Partridge, *Gigabit Networking*, Addison-Wesley, 1994.

[14] S.Q. Zheng, Hypernetworks - A Class of Interconnection Networks with Increased Wire Sharing: Part I - Part IV, Technical Report #94-008, Department of Compute Science, Louisiana State University, Baton Rouge, LA 70803, Dec., 1994.

[15] S.Q. Zheng, Sparse Hypernetworks Based on Steiner Triple Systems, *Proc of 1995 International Conference on Parallel Processing*, pp. I.92 - I.95, 1995.

[16] S.Q. Zheng and Y. Li, A Pipelined Asynchronous TDM Optical Bus, Technical Report #97-006, Dept. of Computer Science, Louisiana State Univ., April, 1997.

[17] S.Q. Zheng and J. Wu, Dual of a Complete Graph as an Interconnection Network, *The Proceedings of 8th IEEE Symposium on Parallel and Distributed Processing*, pp. 433-442, 1996.

Call For Papers

8th International Conference on Computer Analysis of Images and Patterns
CAIP'99

Ljubljana, Slovenia, 1-3 September 1999

Conference Cochairs

Franc Solina, Aleš Leonardis
University of Ljubljana
Faculty of Comput. and Inform. Science
Tržaška 25,
1001 Ljubljana, Slovenia
Tel:386 61 1768 389,
Fax:386 61 1264 647,
E-mail:
franc.solina,ales.leonardis@fri.uni-lj.si

Program Committee

S. Ablameyko, Belarus
J. Arnspang, Denmark
R. Bajcsy, USA
I. Bajla, Slovakia
A. M. Bruckstein, Israel
V. Chernov, Russia
D. Chetverikov, Hungary
A. Del Bimbo, Italy
J. O. Eklundh, Sweden
V. Hlaváč, Czech Republic
J. Kittler, United Kingdom
R. Klette, New Zealand
W. Kropatsch, Austria
A. Leonardis, Slovenia
R. Mohr, France
M. Schlesinger, Ukraine
W. Skarbek, Poland
F. Solina, Slovenia
G. Sommer, Germany
L. Van Gool, Belgium
M. A. Viergever, Netherlands
S. W. Zucker, USA

Call for Papers

The CAIP conference is a traditional Central European Conference devoted to all aspects of computer vision, image analysis, pattern recognition and related fields.

The conference is sponsored by IAPR, Slovenian Pattern Recognition Society, IEEE Slovenia Section, Faculty of Computer and Information Science at the University of Ljubljana and Hermes SoftLab.

The scientific program of the conference will consist of plenary lectures by invited speakers, contributed papers presented in two parallel sessions and posters. The CAIP proceedings are published by Springer Verlag in the series Lecture Notes on Computer Science and will be distributed to the participants at the conference.

Scope of the Conference

- Image Analysis
- Computer Vision
- Pattern Recognition
- Medical Imaging
- Network Centric Vision
- Augmented Reality
- Image and Video Indexing
- Industrial Applications

Instructions to authors

Authors who wish to present a paper at the conference should send five copies of their paper to one of the two conference chairs marked CAIP'99. To enable double blind review there should be two title pages. The first with title, author's name, affiliation and address, telephone, fax and e-mail, abstract of 200 words and up to three keywords. The second title page should consist only of title, abstract and keywords. The papers excluding the title pages should not be longer than 10 pages. On a separate page the authors should answer the following three questions about their paper: (a) what is the original contribution?, (b) what is the most similar work?, (c) why is their work relevant to others?

A LaTeX template for the camera-ready version will be available on the conference home page. During the CAIP'99 review period the authors should not submitt any related paper with essentially the same content to any other conference.

Deadline for submission of papers: 15 January 1999

Registration

Information on registration will be available on the conference home page.

Venue

The conference will be held at the Faculty of Computer and Information Science at the University of Ljubljana. Ljubljana is the capital of Slovenia. The city which is a lively mixture of Mediterranean and northern influences offers all amenities within short distance. Alpine resorts, the Adriatic coast and several natural spas are close to Ljubljana.
The conference homepage is:
`http://razor.fri.uni-lj.si/CAIP99`

Second Call for Papers

Special Issue on

SPATIAL DATABASES

INFORMATICA
an International Journal of Computing and Informatics
http://orca.st.usm.edu/informatica

Edited by:

Maria A. Cobb
University of Southern Mississippi
Department of Computer Science & Statistics
Hattiesburg, MS 39406-5106
maria.cobb@usm.edu

Frederick E. Petry
Center for Intelligent and Knowledge-Based Systems
Tulane University
New Orleans, LA 70118
petry@eecs.tulane.edu

A special issue of Informatica on the topic of spatial databases is planned for publication as Vol. 23, No. 4. Max Egenhofer (University of Maine, NCGIA) has agreed to write a special introductory article for the issue. Others who have tentatively agreed to contribute to the issue include Robert Laurini and Alan Saalfeld. We extend invitations for papers in all areas related to spatial databases, including, but not limited to the following:

- spatial data and relationship models
- distributed spatial databases
- visualization
- querying
- uncertainty management
- GIS
- spatio-temporal reasoning

Dates:

Full papers are due on December 1, 1998
Author notification set for March 31, 1999.
Final revisions of accepted papers are due September 30, 1999 (in LaTeX format).

Authors interested in submitting a paper for the issue should contact one of the guest editors listed above for submission details.

Machine Learning List

The Machine Learning List is moderated. Contributions should be relevant to the scientific study of machine learning. Mail contributions to ml@ics.uci.edu. Mail requests to be added or deleted to ml-request@ics.uci.edu. Back issues may be FTP'd from ics.uci.edu in pub/ml-list/V<X>/<N> or N.Z where X and N are the volume and number of the issue; ID: anonymous PASSWORD: <your mail address> URL-http://www.ics.uci.edu/AI/ML/Machine-Learning.html

---

CC AI

The journal for the integrated study
of Artificial Intelligence, Cognitive Science
and Applied Epistemology.

CC-AI publishes articles and book reviews relating to the evolving principles and techniques of Artificial Intelligence as enriched by research in such fields as mathematics, linquistics, logic, epistemology, the cognitive sciences and biology.
CC-AI is also concerned with development in the areas of hard- and software and their applications within AI.

Editorial Board and Subscriptions

CC-AI, Blandijnberg 2, B-9000 Ghent, Belgium.
Tel.: (32) (9) 264.39.52,
Telex RUGENT 12.754
Telefax: (32) (9) 264.41.97
e-mail: Carine.Vanbelleghem@RUG.AC.BE

Call for Papers

Special Issue on Group Support Systems

Informatica - An International Journal of Computing and Informatics

A special issue of the Informatica Journal will focus on the different aspects of Group Support Systems. Original, unpublished contributions and invited articles will be considered for the issue.

As organizations are being directed to do more with less, personnel productivity issues are becoming more important. No single person has the knowledge, time, or experience to solve today's business problems, so organizations have adopted a team approach. One of the challenges to this team approach is to ensure the proper make-up of the teams. Getting the maximum mix of personnel usually means picking team members from across the organization. This can pose problems when the organization is national, or global in geographical span. Technology to support these teams has been in use for some time now, and research is beginning to emerge showing how successful this merging of technology and project teams has been

To be successful, research in this area should address issues relating to the technology. It should also focus on the effectiveness to the group, and the degree of success in solving problems. Topics of interest to this special issue will include, but not be limited to, the following:

- Critical success factors for implementing group support systems
- Factors affecting the diffusion of group support systems
- Reliable measures for predicting successful implementation
- Cross-cultural factors affecting use
- Management issues with applying group support systems
- Successful implementations
- Uses for group support systems in industry
- Perceived usefulness among project teams
- Ease of use issues
- Ease of implementation issues
- Required levels of technology support
- Barriers to implementation and adoption
- Any other interesting topic that is relevant to group support systems

Prospective authors should follow the regular guidelines of the Journal and submit their work electronically. You should e-mail your rtf or PDF file to one of the Guest Editors by the following due dates:
Full Manuscript Due May 15, 1999
Notification of Acceptance September 15, 1999
Final revisions of accepted papers December 1, 1999

Gary Klein or Morgan Shepherd, Guest Editors
College of Business and Administration
The University of Colorado at Colorado Springs
1420 Austin Bluffs Parkway
P.O. Box 7150
Colorado Springs, CO 80933-7150
Gklein@mail.uccs.edu or mshepher@mail.uccs.edu
Marcin Paprzycki
Department of Computer Science and Statistics
University of Southern Mississippi
Hattiesburg, MS 39406-5106, USA
phone: 601-266-6639 or: 601-266-4949
FAX: 601-266-6452

Call for Papers

Special Issue on Design Issues of Gigabit Networking

Informatica - An International Journal of Computing and Informatics

A special issue of the Informatica Journal will focus on the different aspects of the design of Gigabit networking. Original, unpublished contributions and invited articles will be considered for the issue. With the advent of the World Wide Web, the Internet has seen enormous growth from its roots as a network of modest proportions, mostly used by the research and academic community, to a large public data network. Several thousands of corporate users and several millions of dial-in residential users have gone online in the last few years, making the Internet a true public data network. This accelerated growth in subscription has led to a surge of data in the Internet backbone. In order to keep up with this demand in service and bandwidth, all Internet service providers have scaled their networks *many times*, in both size and bandwidth. The forecast for this continuing growth is even more astounding for the coming years. With fast emerging technologies, such as transfer of multimedia and electronic commerce, the need to scale the network beyond present capabilities is paramount. For this the Internet has to scale in several dimensions, including but not limited to bandwidth, routing, quality of service (QoS), and customer service provisioning. To be able to succeed, research has to investigate issues of the backbone network (SONET/SDH). It should also focus on the reliability of the network and the transparency of any self-healing to the user. Topics of interest to this special issue will include, but not limited to, the following:

- Design infrastructure of Gigabit networking (physical, data link and network issues)
- SONET/SDH architectures
- Protocol design for multimedia
- Fault-tolerant of backbone networks
- Routing and congestion protocols in Gigabit networking
- Switching issues in Gigabit networking
- QoS issues in using ATM in Gigabit networking
- Modeling and simulation
- Performance evaluation in Gigabit networking
- Management issues in Gigabit networking
- Any other interesting topic that is relevant to the backbone infrastructure design

Prospective authors should follow the regular guidelines of the Journal and submit their work electronically. You should e-mail your PDF or postscript file (preferably produced by dvips and viewable by ghostview) to one of the Guest Editors listed below:

Full Manuscript Due January 31, 1999
Notification of Acceptance July 31, 1999

Mohsen Guizani, Guest Editor
Electrical and Computer Engineering Architect
University of Missouri-Columbia/Kansas City
5605 Troost Avenue
Kansas City, MO 64110-2823.
E-mail: guizanim@umkc.edu
Kenneth Henriksen, Guest Editor
Chief Technology Integration Architect-Sprint
M/S: KSOPKB0803
9300 Metcalf Avenue
Oveland Park, KS 66212, USA.
E-mail: henriksen@sprintcorp.com
http://orca.st.usm.edu/informatica

Call for Papers

Special Issue on Advances in Simulation and Control

Informatica - An International Journal of Computing and Informatics

A special issue of Informatica on the topic of "Advances in Simulation and Control" is planned for publication as one of four issues in 2000 year. As computing power has been increased, most of the real world systems can be efficiently simulated and controlled in real time with the incorporation of advanced simulation and control techniques. The issue will be intended to serve as a medium for exchanging the latest research trends in the areas of "Simulation and Control."

Papers describing the state-of-the-art research on various simulation and control topics including, but not limited to, the following areas are solicited:

- Advances in Simulation and Control Methodology,
- Fuzzy and Neural Network Techniques in Simulation and Control,
- Real-time and Distributed Simulation and Control,
- Advanced Man-Machine Interfaces for Efficient Simulation and Control,
- Simulation and Control Applications in Complex Physical Systems such as Electricity Generating Power Plants and Power Systems.

Prospective authors should follow the regular guidelines of the Journal and submit their work electronically (by e-mail) to one of guest editors by the following due dates:

Important Schedules: Full papers are due May 7, 1999 in any format of PDF, PS, MS Word or WordPerfect, with author notification set for November 20, 1999. Final revisions of accepted papers are due February 26, 2000, only in "LaTex" format. Authors interested in submitting a paper for the issue should contact one of the guest editors listed below for submission details.

Robert M. Edwards, Guest Editor
Nuclear Engineering Department
231 Sackett Building
Pennsylvania State University
University Park, PA, 16802, USA.
Phone: +1 (814) 865-0037
FAX: +1 (814) 865-8499
Email: rmenuc@engr.psu.edu
Kwang Y. Lee, Guest Editor
Department of Electrical Engineering
The Pennsylvania State University
University Park, PA 16802, USA.
Phone: +1 (814) 865-2621
Fax: +1 (814) 865-7065
E-mail: kylece@engr.psu.edu
Se Woo Cheon, Guest Editor
Korea Atomic Energy Research Institute
Dukjin 150, Yuseong, Taejon 305-353, KOREA
Phone: +82-42-868-2261
Fax: +82-42-868-8357
E-mail: swcheon@nanum.kaeri.re.kr

Call for Papers

The Sixteenth International Conference on Machine Learning

June 27-30, 1999
Bled, Slovenia

The Sixteenth International Conference on Machine Learning (ICML-99) will be held at the Convention Center Festival Hall, Bled, Slovenia, from June 27 to June 30, 1999. ICML-99 will be co-located with the Ninth International Workshop on Inductive Logic Programming (ILP-99).

Submissions are expected that describe empirical, theoretical, and cognitive-modeling research in all areas of machine learning. Submissions that present algorithms for novel learning tasks, interdisciplinary research involving machine learning, or innovative applications of machine learning techniques to challenging, real-world problems are especially encouraged. For application papers, novelty and lessons for the machine learning community are of particular interest. All submitted papers will be reviewed by at least two members of the international programme committee.

Paper Submissions

Authors must submit four (4) hardcopies and an electronic version (PostScript) of each submission, as well as a copy of the title page via email.
The mailing address for hardcopies is:
Ivan Bratko & Saso Dzeroski / ICML-99
Jozef Stefan Institute
Jamova 39
SI-1000 Ljubljana
Slovenia
(phone +386 61 177 3528 for use on express mail forms)
Submissions must arrive by February 1, 1999, acceptance decisions will be mailed by March 27, 1999, and camera-ready copies will be due by April 19, 1999.
Electronic versions (PostScript, preferrably gzipped) should be emailed to `icml99@ijs.si` with a subject "postscript submission".

We depend on the electronic versions in PostScript to cut mailing costs and speed-up the reviewing process. Submissions not accompanied by an electronic version will only be processed exceptionally, when the authors cannot produce such an electronic version and have explained this on the title page.

A copy of the title page (plain ascii text) should also be emailed to `icml99@ijs.si` with a subject "title page" by January 28, 1999 (note the date).

The title page should accompany each hardcopy submission, in addition to being sent through electronic mail. The title page, both the electronic and hardcopy versions, should be formatted as follows:

Title:
Authors with addresses:
Abstract (250 word maximum):
Keywords:
Email address of contact author:
Phone number of contact author:
Electronic version: PostScript / None (please explain why not!)
Multiple submission statement (if applicable):

Please send your title page as plain ASCII. (Do NOT send PostScript, Latex, Microsoft Word, etc.)

Submissions (including the title page, figures, tables, and references) must not exceed 16 pages. Please start the body of your paper on page 2, but you need not repeat the title, abstract, etc. on that page. Staple each copy of the paper including the title page. Two-sided printing is encouraged.

Submissions must be formatted as follows: 12-point font (except for references, which can be formatted in a 10- or 11-point font), single-spaced (Baselineskip = 0.1875 inches or 0.4763 cm), a maximum per-page text width of 5.5 inches (14 cm), and a maximum per-page text height of 7.5 in (19 cm). For those using LaTeX, a style file and a sample source file are available through the on-line version of this call. In the interest of fairness, if a submission exceeds the length dictated by the above requirements, it will be rejected without review.

Simultaneous submissions to multiple conferences are allowed, as long as this is clearly indicated on the cover page. Accepted papers will be published in the ICML'99 Proceedings, provided they are withdrawn from the other conferences.

Workshop Proposals

ICML-99 Workshops will take place on June 30 at Hotel Park. Workshop proposals should be submitted by January 10, 1999 to the workshop chair Dunja Mladenic at `Dunja.Mladenic@ijs.si`, cc: `icml99@ijs.si`, subject "icml99 workshop proposal". Acceptance decisions will be mailed by February 1, 1999. Camera-ready proceedings from the workshop organizers are due June 1, 1999.

The proposals should not exceed three pages and should contain a title, organizer (with contact details), tentative program committee, workshop description and a draft call for papers for the workshop.

Tutorial Proposals

ICML-99 Tutorials will take place in the afternoon of June 26 at Hotel Park. Tutorial proposals should be submitted by January 10, 1999 to the tutorial chair Blaz Zupan at `Blaz.Zupan@ijs.si`, cc: `icml99@ijs.si`, subject "icml99 tutorial proposal". Acceptance decisions will be mailed by February 1, 1999. Camera-ready tutorial notes from the lecturers are due June 1, 1999.

The proposals should not exceed two pages and should contain a title, lecturer (with contact details), duration (hours), a description of the tutorial and a brief profile of the lecturer.

On-line version of this CFP:
`http://www-ai.ijs.si/SasoDzeroski/ICML99/cfp.html`
ICML-99 home page:
`http://www-ai.ijs.si/SasoDzeroski/ICML99/main.html`

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;
- stimulation and support to collaboration between research organisations and business, public, and other sectors;

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.

- 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.

	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)

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 postgraduates, 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 **Slove**nia (or S♡nia). 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

CONTENTS OF *Informatica* Volume 22 (1998) pp. 1–600

Papers

Editorials

Interview

Book and Conference Overviews

Calls for Papers

Professional Societies

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 LaTeX 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: .....

Title and Profession (optional): .....

.....

Home Address and Telephone (optional): .....

.....

Office Address and Telephone (optional): .....

.....

E-mail Address (optional): .....

Signature and Date: .....

Informatica WWW:

turing.ijs.si/Mezi/informatica.htm
http://orca.st.usm.edu/informatica/

Referees:

Witold Abramowicz, David Abramson, Kenneth Aizawa, Suad Alagić, Alan Aliu, Richard Amoroso, John Anderson, Hans-Jurgen Appelrath, Grzegorz Bartoszewicz, Catriel Beeri, Daniel Beech, Fevzi Belli, Istvan Berkeley, Azer Bestavros, Balaji Bharadwaj, Jacek Blazewicz, Laszlo Boeszoermenyi, Damjan Bojadžijev, Jeff Bone, Ivan Bratko, Jerzy Brzezinski, Marian Bubak, Leslie Burkholder, Frada Burstein, Wojciech Buszkowski, Netiva Caftori, Jason Ceddia, Ryszard Choras, Wojciech Cellary, Wojciech Chybowski, Andrzej Ciepielewski, Vic Ciesielski, David Cliff, Travis Craig, Noel Craske, Matthew Crocker, Tadeusz Czachorski, Milan Češka, Honghua Dai, Andrej Dobnikar, Sait Dogru, Georg Dorfner, Ludoslaw Drelichowski, Matija Drobnič, Maciej Drozdowski, Marek Druzdzel, Jozo Dujmović, Pavol Ďuriš, Hesham El-Rewini, Pierre Flener, Wojciech Fliegner, Vladimir A. Fomichov, Terrence Forgarty, Hans Fraaije, Hugo de Garis, Eugeniusz Gatnar, James Geller, Michael Georgiopolus, Jan Goliński, Janusz Gorski, Georg Gottlob, David Green, Herbert Groiss, Inman Harvey, Elke Hochmueller, Rod Howell, Tomáš Hruška, Alexey Ippa, Ryszard Jakubowski, Piotr Jedrzejowicz, Eric Johnson, Polina Jordanova, Djani Juričič, Sabhash Kak, 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, Alexander Linkevich, Raymond Lister, Doug Locke, Peter Lockeman, Matija Lokar, Jason Lowder, Andrzej Małachowski, Bernardo Magnini, Peter Marcer, 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, Elzbieta Niedzielska, Marian Niedq'zwiedziński, Jaroslav Nieplocha, Jerzy Nogieć, Stefano Nolfi, Franc Novak, Antoni Nowakowski, Adam Nowicki, Tadeusz Nowicki, Hubert Österle, Wojciech Olejniczak, Jerzy Olszewski, Cherry Owen, Mieczyslaw Owoc, Tadeusz Pankowski, Mitja Peruš, Warren Persons, Stephen Pike, Niki Pissinou, Ullin Place, Gustav Pomberger, James Pomykalski, Gary Preckshot, Dejan Rakovič, Cveta Razdevšek Pučko, Ke Qiu, Michael Quinn, Gerald Quirchmayer, Luc de Raedt, Ewaryst Rafajlowicz, Sita Ramakrishnan, Wolf Rauch, Peter Rechenberg, Felix Redmill, David Robertson, Marko Robnik, Ingrid Russel, A.S.M. Sajeev, Bo Sanden, Vivek Sarin, Iztok Savnik, Walter Schempp, Wolfgang Schreiner, Guenter Schmidt, Heinz Schmidt, Denis Sever, William Spears, Hartmut Stadtler, Olivero Stock, Janusz Stokłosa, Przemysław Stpiczyński, Andrej Stritar, Maciej Stroinski, Tomasz Szmuc, Zdzislaw Szyjewski, Jure Šilc, Metod Škarja, Jiři Šlechta, Zahir Tari, Jurij Tasič, Piotr Teczynski, Stephanie Teufel, Ken Tindell, A Min Tjoa, Wieslaw Traczyk, Roman Trobec, Marek Tudruj, Andrej Ule, Amjad Umar, Andrzej Urbanski, Marko Uršič, Tadeusz Usowicz, Elisabeth Valentine, Kanonkluk Vanapipat, Alexander P. Vazhenin, Zygmunt Vetulani, Olivier de Vel, John Weckert, Gerhard Widmer, Stefan Wrobel, Stanislaw Wrycza, Janusz Zalewski, Damir Zazula, Yanchun Zhang, Robert Zorc, Anton P. Železnikar

EDITORIAL BOARDS, PUBLISHING COUNCIL

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 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 list of referees. Each paper bears the name of the editor who appointed the referees. Each editor can propose new members for the Editorial Board or referees. Editors and referees inactive for a longer period can be automatically replaced. Changes in the Editorial Board 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. Informatica 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 Informatica is guaranteed in less than one year after the Executive Editors receive the corrected version of the article.

Volume 22 Number 4 December 1998 ISSN 0350-5596

Informatica

An International Journal of Computing and Informatics

Contents: