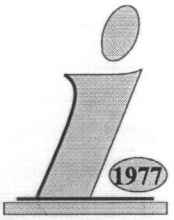
Volume 23 Number 2 May 1999 ISSN 0350-5596

Informatica

An International Journal of Computing and Informatics

Spatial Data Management

Informatica 23 (1999) Number 2, pp. 153–302



The Slovene Society Informatika, Ljubljana, Slovenia

Informatica

An International Journal of Computing and Informatics

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

Subscription Information Informatica (ISSN 0350-5596) is published four times a year in Spring, Summer, Autumn, and Winter (4 issues per year) by the Slovene Society Informatika, Vožarski pot 12, 1000 Ljubljana, Slovenia.
The subscription rate for 1999 (Volume 23) is
– DEM 100 (US$ 70) for institutions,
– DEM 50 (US$ 34) for individuals, and
– DEM 20 (US$ 14) 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 send checks or VISA card number or use the bank account number 900–27620–5159/4 Nova 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.

Introduction: Special Issue on Spatial Data Management

Guest Editors:

Frederick E. Petry
Department of Electrical Engineering & Computer Science
Tulane University
New Orleans, LA 70118
petry@eecs.tulane.edu
Maria A. Cobb
Department of Computer Science & Statistics
University of Southern Mississippi
Hattiesburg, MS 39406-5106
maria.cobb@usm.edu
Kevin B. Shaw
Digital Mapping, Charting & Geodesy Analysis Program
Mapping, Charting & Geodesy Branch
Naval Research Laboratory
Stennis Space Center, MS 39529
shaw@nrlssc.navy.mil

This special issue of Informatica focuses on several research topics in the area of spatial data management. Spatial databases have developed as extensions to ordinary databases in response to rapidly developing applications such as Geographic Information Systems (GIS), CAD systems, and many multimedia applications. These applications dictate the need for (1) additional data types, including point, line and polygon; (2) spatial operations such as intersection, distance, etc.; and (3) the ability to handle some combination of objects (vector data) and fields (raster data). The nature of spatial data requires multi-dimensional indexing to enhance performance, and much research has been devoted to this topic.

The first set of papers in this issue provides descriptions of extensions to the standard functionality in GIS databases. In particular, these first three papers discuss extensions for space-time visualization, sound as a spatio-temporal field and the inclusion of network facilities in a GIS. A realization of visual aspects of the space-time conceptual framework of Hagerstrand is discussed in the first paper by Hedley, Drew, Arfin and Lee. They demonstrate one of the first examples of an implementation of this approach and provide a real-world case study of visualizing worker exposure to hazardous materials. In the second paper (Laurini, Li, Servigne, Kang) a field-oriented approach to auditory data in a GIS is described. The special semantics of auditory information are presented and some techniques for indexing of such data are indicated. The third paper in this set adds additional levels of abstraction to extend the semantics of GIS networks. The authors, Claramunt and Mainguenaud, then show this leads to the development of a new interpretation of the database projection operator.

The second set of papers provides techniques for processing and modeling spatial data. The first paper by Havran provides a new approach for memory mapping of binary search trees that can improve spatial locality of data and thus spatial query performance. In the next paper by Chung and Wu, some improved spatial data structure representations including linear quadtrees are presented. These representations are shown to have better compression performance. Finally, the paper by Forlizzi and Nardelli describes the lattice completion of a poset to model spatial relationships. They prove some of the needed conditions for valid intersection and union relations among spatial objects with this representation.

Hagerstrand Revisited: Interactive Space-Time Visualizations of Complex Spatial Data

Nicholas R. Hedley*, Christina H. Drew**, Emily A. Arfin** and Angela Lee*
*Department of Geography and National Research Center for Statistics and the Environment,
University of Washington,
Seattle, WA 98195-3550, USA.
**Department of Geography, University of Washington,
Seattle, WA 98195-3550.
Phone: USA+ 206 616 3409
E-mail: nix@u.washington.edu

Keywords: visualization, spatial databases

Edited by: Frederick E. Petry, Maria A. Cobb and Kevin B. Shaw
Received: December 1, 1998 **Revised:** May 28, 1999 **Accepted:** April 15, 1999

Technological advances have rapidly changed the nature of spatial analytical tools in recent years. A tendency is for tool development to outpace theoretical and conceptual development. Occasionally, some conceptual frameworks must await the arrival of tools that can operationalize the elegance and sophistication they embody. This paper calls for reconsideration of a conceptual framework introduced by Hagerstrand (1970). It has taken until the late 1990s for tools appropriate for implementation of Hagerstrand's framework to develop, so that we may apply it in a meaningful and accessible manner that is grounded in pragmatic applications. This paper shows that the visual component of Hagerstrand's space-time conceptual framework may now be operationalized, using advanced spatial analytical visualization techniques. We demonstrate a robust visual representation, its value in a real-world case study, and discuss the potential for future applications of this technique. A key theme in this work is that the fusion of space-time conceptual frameworks and appropriate spatial analytical visualization techniques (such as GIS) can make significant progress in facilitating user access to spatial data bases (in terms of understanding the data, as well as physical access). This is necessary to foster more democratic processes in collaborative settings. The project described in this paper meets that challenge, and appears to be the first example of an implementation that explicitly attempts to bridge this gap.

1 Introduction

Spatial databases are powerful tools. When paired with rigorous spatial analysis, new understandings of previously unknown or misunderstood phenomena may be gained. Spatial data are widely used in research and commercial settings. This is facilitated by ongoing improvements in the performance and cost of computer hardware and software, and an increasing acceptance of information technology by society.

However, while tool development has progressed, the greatest advances are to be made where a fusion of technological and conceptual innovation occur. In the geographic domain, a clear conceptual innovation occurred in 1970, when Torsten Hagerstrand introduced his space-time geography concepts (Hagerstrand, 1970). This paper discusses the elegance and utility of Hagerstrand's framework, how it was appropriated, and also why its implementation was not as frequent as it might have been.

This work demonstrates the potential of space-time visualizations as a powerful tool for facilitating user access to spatial databases. Space-time representations are implemented in a manner that reflects the visual component of Hagerstrand's (1970) conceptual framework, using commercially-available GIS (geographic information system) software. This work suggests that the use of this approach to representing information has value in real-world settings. We suggest that this approach has particular value in the representation of interactions between mobile and stationary objects in multiple levels of abstraction of time and space. This work is one of the first projects to fuse Hagerstrand visual space-time concepts, advanced spatial analytical tools, and a contemporary real-world application. Finally, this work shows a methodological approach that may enhance spatial databases accessibility by users in both expert and non-expert domains.

2 Time Geography - A Neglected Framework

Torsten Hagerstrand (1970) is known in geography for research in spatial diffusion processes (Mark, 1997). His most lasting contribution to geography is known as 'time geography'. Associated with spatial diffusion processes, the space-time framework provides a conceptual framework of space and time as (limiting) constraints on spatial interaction and accessibility (Miller, forthcoming; Mark, 1997; Hagerstrand, 1970).

In 1970, Hagerstrand presented "space-time prisms", as a component of new time-geography conceptual frameworks . Lenntorp (1976), Parkes and Thrift (1980), Miller (1991), Chai and Wang (1997), Chrisman (1997) and Mark (1997) acknowledge its value as a means to conceptualize space and time. While the framework had great potential, actual implementations were infrequent (Miller, 1991). By 1998, they numbered less than ten. Of those ten, the majority aim to develop methodological tools to predict spatial diffusion based on alternative constraint models. Insufficient work has been done on the descriptive visualization of spatial data in the context of space-time representation.

3 Purpose and Function of the Space-Time Framework

Hagerstrand's (1970) approach joins space and time in a reference system for phenomena.: "...we need to understand better what it means for a location [or individual] to have not only space coordinates but also time coordinates." (Hagerstrand, 1970: p.9-10). A more recent articulation of space and time (based on time-geography), is: "Time and space act as containers of the material world; thus space and time are external or absolute." (Chrisman, 1997). Hagerstrand's framework has a highly visual conceptual design. Geographic space is collapsed into a two-dimensional plane, and the vertical (z) dimension represents time (increasing with height). It is in this setting that he proposed the representation of interacting life histories of individuals (Mark, 1997).

Hagerstrand's framework (1970) proposed the representation of 'paths' of mobile objects through space and time (see Figure 1 (i)). Focusing on spatial diffusion models, he extends this to 'space-time prisms', where a space-time volume bounds the maximum extent of all space-time paths of an individual based on space, time and accessibility constraints (Figure 1 (ii)).

The space-time 'prism' is acknowledged as a central idea in time geography (Mark, 1997). Space-time prisms have been applied in spatial diffusion, transportation, and planning settings (Miller, 1999; Kwan 1998; Kwan and Hong, 1998; Miller, 1991; Lenntorp, 1978).

Its use of space-time prisms is most beneficial in predicting individuals' behavior and potential for movement in space, in the face of constraining factors. The research presented here focuses on space-time trajectory visualization as a mechanism by which the space-time interaction of individuals in a real-world geographic space may be described.

The adoption of Hagerstrand's framework has been infrequent in the literature. Although much of the work done has great utility to measure, calculate, and predict individual movement based on constraining factors in an environment, examples are needed to demonstrate the value and utility of space-time representation for the communication of complex spatial data in applied settings. While the framework was acknowledged as both powerful and elegant (Miller, 1999), a challenging duality also surrounded the framework. Allen Pred (1977) articulated this concern, while addressing the time-geography concept as a whole: *"...the time-geography framework is at the same time disarmingly simple in composition and ambitious in design."* (Pred, 1977).

At the time of their introduction, space-time representations, while innovative and elegant, represented a major challenge to implement. The framework was highly visual and was commonly depicted as a perspective (or "2.5-D") sketch as shown by Figure 2, above. Perspective representations give the impression of three dimensions, by projecting three dimensions onto two. So, the space-time illustrations of Lenntorp (1978), Parkes and Thrift (1980), and Chrisman (1997) (see Figure 2) are 2.5-D representations. This is problematic.

While the concept developed by Hagerstrand is useful, depiction of space-time trajectories has typically been presented as a perspective sketch (Chrisman, 1997). While skilled artists may be able to give perspective (2.5-D) views, this approach makes some significant assumptions about the audience's ability to extract meaning from the diagram. These assumptions might include user experience with technology, information representation, or perhaps perceptual differences. In the version found in Chrisman (1997: 7, displayed in Figure 2 in this paper), where is the route taken by the woman on the bicycle with the child? Which objects move along which vectors? How do the movement vectors relate to possible routes in the road network shown. It is simply not clear or easy to determine this. An inherent limitation of 3D-to-2D projections is that information will invariably be lost. This difficulty must be resolved in order for the full utility of space-time representation to be seen.

Space-time implementations using Hagerstrand's (1970) space-time conceptual framework have been infrequent in the literature. This appears to be largely

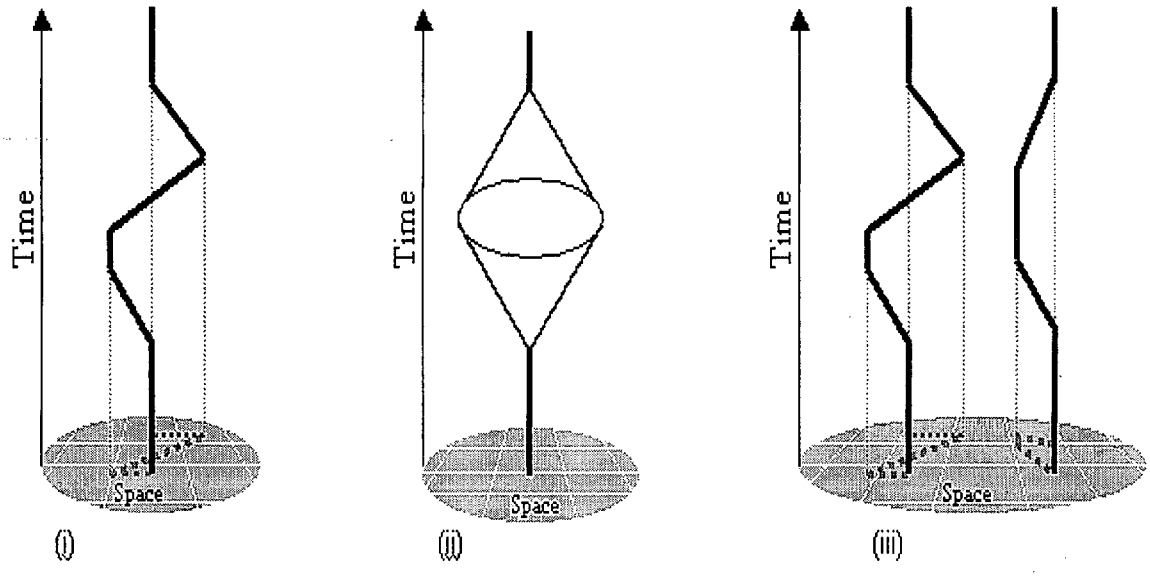


Figure 1: Principle space-time trajectory (i) and prism (ii) concepts, and application of trajectories to collocation (iii). Note: i and ii redrawn based on Miller (1991, p.290).

due to technical and technological difficulties in operationalizing the framework in a form that truly reflects the source concept (Mark, 1997; Pred, 1977). The examples that do exist in the literature (Miller, 1999; Miller, 1991; Lenntorp, 1976; Lenntorp, 1978; Kwan, 1998; Kwan and Hong, 1998) appear to focus on predictive, calculative modeling applications in the research domain (versus applied real-world settings).

Our response to this challenge is to produce manipulable dynamic 3D space-time visual representations. Three-dimensional representation is not an adequate response on its own. For to respond only with static 3-D would be to effectively produce the same end product but with a different generative procedure. To fully employ the power of Hagerstrand's conceptual framework, we employed 3-D representation tools in which the user can move around and visually query the objects in view. This is especially useful to examine objects that appear to float or are difficult to locate in space-time (such as the bicycle in Figure 2).

4 Case Study: A Recent Implementation based on Hagerstrand

Objectives. This research focuses on visualizing the space-time paths of workers and their exposure to radiological waste at Hanford Reservation, using the visual component of Hagerstrand's conceptual framework (Hagerstrand, 1970). The practical application of this is to allow a diverse stakeholder audience to understand the vulnerability of workers as they move about a site during spent nuclear fuel cleanup and relocation activities. The primary target user group of these visualizations are human resource managers (those who plan the roles, tasks and movements of workers in the cleanup process at Hanford).

Substantive challenge: Visualizing worker exposure to hazardous material The Hanford Nuclear Weapons Production Facility (known as the 'Hanford Site') is a former nuclear weapons manufacturing facility that occupies approximately 560 square miles in south-central Washington State, in the Northwest of the United States. From 1943-1989 Hanford produced weapons grade plutonium for nuclear weapons. In 1989, with the end of the Cold War, the mission of Hanford was dramatically changed from plutonium production to Waste Management (safe disposition of residual radioactive materials) and Environmental Restoration (clean-up of the contaminated areas on the site) (http://www.hanford.gov/top/welcome.html, 1998).

The cleanup effort at Hanford is a highly complex human and environmental resources management system. By definition the project necessitates the production and dissemination of complex interrelated spatial and temporal information to range of end-users.

This work focuses on the 100K area which is located close to the Columbia River (see center of, Figure 3). Preparations are underway to move spent nuclear fuel rods from the 100K to the 200 East site, to reduce risks of environmental contamination due to the encroaching water table close to the Columbia River. A key element of planning, strategy and evaluation for the 100K to 200 East project is to determine expected

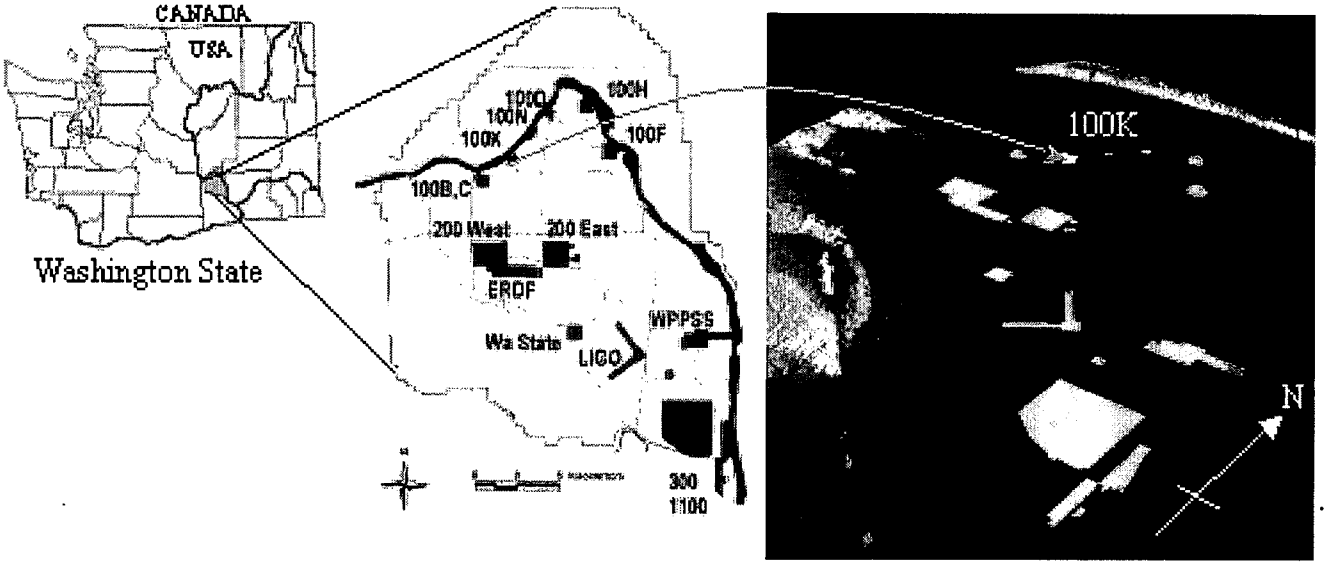


Figure 3: Map showing location within Washington State, map of the Hanford Reservation (center), and its representation in a 3D GIS (right).

radiological exposure for workers at all stages and locations of the spent nuclear fuel relocation process.

The location and character of radiological hazards at Hanford is information which must be accessible to a range of end users, or 'stakeholders'. Stakeholders include (but are not limited to) research scientists, decision and policy makers, and interested and affected public groups alike. There is a distinct need to move towards a situation in which information from data may be presented to audiences in a variety of settings. Such settings might include round-table 'openness' meetings of technical specialists, interested and affected parties, decision and policy makers. Alternatively, information may be disseminated within each of these primary stakeholder types.

A fundamental concern at Hanford is communication of information. As a result, information architecture that facilitates the efficient transmission of information within and between groups is a major interest. This is particularly critical at Hanford, as a very diverse range of stakeholder groups (including government, research and public interest groups) uses information. This work focuses on a particular case study at Hanford (worker paths) to demonstrate the value of operationalizing Hagerstrand's space-time representation. This has significant potential to foster semantic consistency in and across diverse information user groups.

Conceptual Basis. Employing Hagerstrand's (1970) space-time path concept, we can visualize the space-time trajectories of workers, groups and spatial entities (objects that have a spatially-located component, including people, buildings, Spent Nuclear Fuel handling rooms). More significantly, we may analyze the space-time intersections of people and radiological hazards.

The space-time path (or trajectory) approach is particularly significant for the Hanford case study. By being able to co-locate the space-time paths of workers and the spatial entities they interact with, we may quickly understand and communicate the character of their exposure at locations with known radiological doses. The cumulative radiation dose received by an individual is determined primarily by location relative to source, and duration of exposure. Using space-time trajectories, we may represent data and information in a way that makes their space-time relationship explicit. This allows us to express often difficult to access data in a form that requires less 'unpacking' (such as how data from a specific table relate to a management report of worker behavior). The data for worker exposure to radiological hazards have not to our knowledge been brought together in this manner previously. By identifying the trajectory segment that represents a given individual's co-location with a known hazardous source, we may quickly see the spatio-temporal intersection of the two (see Figure 4, below).

5 Data and Tools

Data were acquired from a variety of sources in order to develop the 3D GIS representation of the Hanford Reservation, the Department of Energy buildings, the radiological exposures associated with specific build-

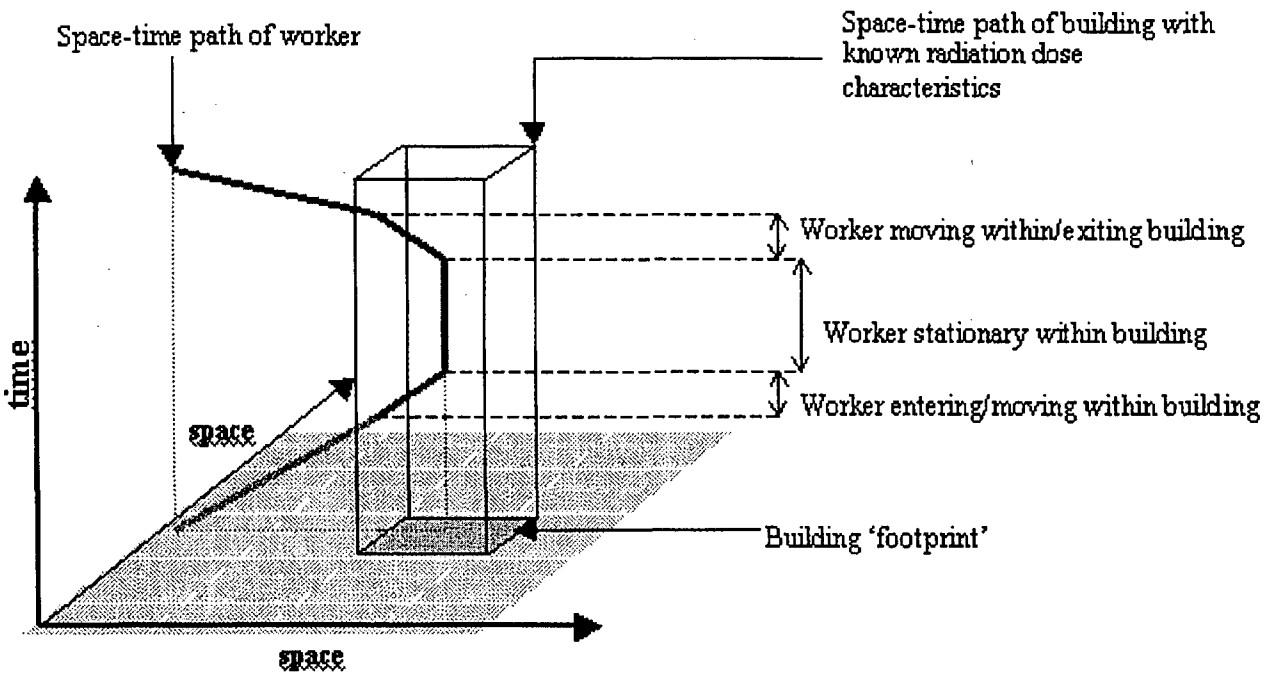


Figure 4: Applying Space-Time Trajectory Concept to Radiological Hazard Exposure.

ings, and finally the prescribed route of a generalized worker.

Bechtel Hanford Inc, the Hanford site contractor for Environmental Restoration (remediation/cleanup of contaminated areas), maintains the Hanford Geographic Information System (HGIS) and has made it available to the public through their internet site. Also, the Hanford Remedial Action Environmental Impact Statement, available on CD-ROM, represents the best publicly available characterizations of Hanford's environmental conditions. We also obtained the Radiation Exposure System, which contains historical radiation dosimetry data for over 150,000 people who have worked at the Hanford site. Several text-based reports were also used to develop the worker trajectory and the databases that corroborated it. They include the Spent Nuclear Fuels Operational Staffing Plan, The Spent Nuclear Fuels/Idaho National Environmental Engineering Laboratory Environmental Impact Statement, The Waste Inventory Data System, and the Columbia River Comprehensive Impact Assessment (see Data Sources at the end of this document). The tools employed for this research included a single 300Mhz Pentium class desktop computer equipped with 64MB RAM. The software used was Environmental Research Systems, Inc.'s ArcView GIS(tm) version 3.1, plus the software extension ArcView GIS(tm) 3D Analyst(tm).

6 Method

The visualization of a space-time trajectory for workers in the 100K area at Hanford was developed from a model day "scenario" for workers handling spent nuclear fuel in the 105KE building. This scenario was developed based on interviews with members of the Consortium for Risk Evaluation with Stakeholder Participation (CRESP) personnel based at Richland , Washington, and integrated with an existing set of GIS data using ArcView GIS(3.1.

Each worker 'behavior' (such as "walk from A to B along route X") was associated with line and/or polygon features from existing data in ArcView GIS(shapefiles. This was done in order to preserve the shape and orientation of transportation routes especially (so that the space-time trajectory did not end up being straight-line vectors between floating z-value points). A time index for a given behavior (such as moving an object from location A to location B) was added to this data.

Due primarily to the fact that the ArcView GIS(extension 3D Analyst(is relatively new and that

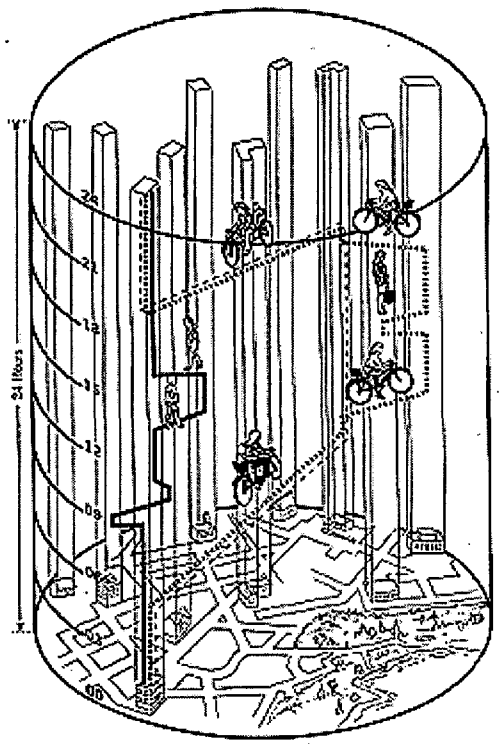


Figure 2: Illustration in Chrisman (1996) providing a conceptualization of space-time trajectories. Note: this diagram was redrawn from Parkes and Thrift (1980), which itself was based on Lenntorp (1978).Courtesy of John Wiley & Sons.

this research approach adopts a specialized conceptual framework, implementing the space-time trajectory visualization using this package required creativity to sidestep the package's limitations. A distinct challenge was the vertical sections of the space-time vector. One possibility in 3D Analyst(was to create a surface from data, bring it into a project view as a theme, then to drape a vector theme over the surface as a 3D shapefile. Making the surface transparent (invisible) would result in the illusion of the remaining vector hovering above the x, y geographic plane. This approach is problematic.

The main problem has to do with a worker using the same x, y route at different times (z-values) during the course of the period represented. While a worker route that never crosses an earlier segment can be represented using the surface-draping technique described earlier, it is very difficult to implement this for instances where the same routes are used at different times. Using the procedure just described, problems are quickly encountered when one surface is superimposed on another.

The solution to these implementation problems was to decompose the task duration information (from the CRESP interviews) into minute-by-minute elements. For each time step, a z-value was associated. This allowed the derivation of individual vector segments for each minute spent by workers in that location. The result was the ability to 'stack' vector segments on top of each other, according to the space-time already derived in the *.dbf file. The theme table for the resulting shapefile ended up having over 600 records, each corresponding to one minute of time in a worker's day. In this section, the visualization products produced in ArcView GIS(3D Analyst(are presented in a logical sequence that reflects the movement of workers through time and space according to the data gathered from CRESP personnel interviews.

7 Results of Implementation

The result of our work is the visualization of a generalized 'worker' trajectory based on prescriptive guidelines laid down by management at Hanford. The trajectory exists in a full-color 3D GIS environment that may be dynamically explored by the movement of the user using ArcView GIS(tm) 3D Analyst(tm). The reproductions presented here are illustrative, and readers are encouraged to contact the primary author for color imagery and demonstrations of the dynamic environment.

The appearance of the worker trajectory in the space-time visualizations shown must be explained. Reconstruction and representation of the physical landscape and the structures on it (such as buildings) at Hanford is straightforward from the sources already mentioned in the Data and Tools section. Data on the space-time behavior of specific worker roles is far harder to come by. There is no formal data produced to track worker behavior in Hanford project tasks. As a result, the worker trajectory seen in the case study shown had to be constructed from information given by Hanford personnel management, during interviews via email. This information was limited in that it gave no indication of rates of movement. The information only indicated locations and route segments within which a worker would be during a given stage of a shift. The appearance of the trajectory - that of vertically 'extruded' trajectory segments - results from temporal aggregation inherent in the information extracted from personnel manager interviews.

On the right side of Figure 3, we see an oblique-angle view of the Hanford Reservation. This image was produced to provide readers with a modest introduction to the appearance of objects in 3D Analyst(. This view demonstrates the ability to fairly realistically represent the 'lay of the land' at Hanford, using an elevation model over which objects are draped and encoded.

Figure 5 (above) presents the main section of the space-time trajectory. The time is the vertical (z) axis.

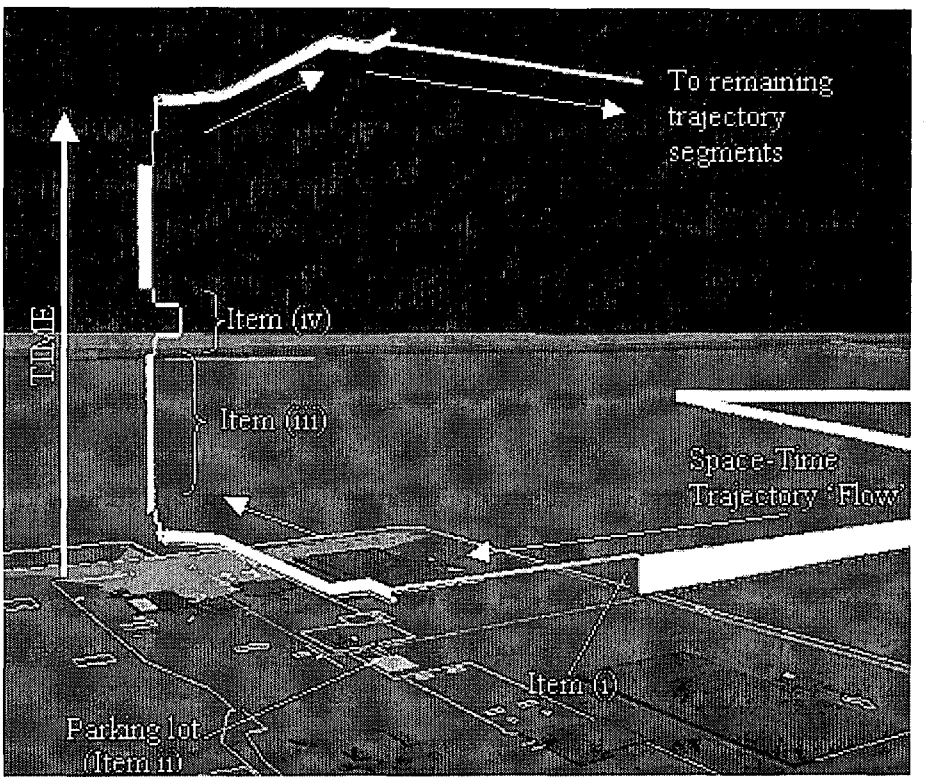


Figure 5: Worker Space-Time trajectory entering 100K area at Hanford.

The additional white arrows indicate the space-time 'flow' of the worker trajectory. There are some interesting features to note here:

- One can see the difference in time taken (vertical dimension of trajectory segment) to get from the Hanford entrance (off image, to the right) to the entrance of the 100K area (Figure 5, item i), compared with the time taken to get from the 100K entrance to the parking area (Figure 5, item ii). This is a good illustration of the logic structure operating in our implementation of Hagerstand's visual components using Hanford management infrastructure-based worker information. We were provided only a generalized account of worker movements around the 100K area. As a result of this, it was only possible to represent this in terms of time spent by a worker in a given route segment or building based on typical time taken to complete the entire task at/in that location. This explains the abrupt transitions between adjacent space-time trajectory segments.
- Along the left edge of the image, we see the main portions of a worker's day. From the parking lot (the right-hand green area), the workers follow the magenta surface route to the 105KE building, via offices and dressing area. The thick trajectory segment that joins with the bottom of the vertical space-time trajectory structure reflects this latter transition. In this vertical segment of the trajectory, we see the workers' first main shift of the day represented by the first segment of the trajectory (a 3-hour shift, item iii in Fig. E). The slight displacement to the right, half-way up the vertical stretch (item iv, Fig. E), indicates workers dressing down, going to lunch, and dressing up again ready for the afternoon shift (2 hours and 20 minutes). Note the stability of location in space during the main shifts of the day. Next, workers are debriefed and make their way to the parking lot and then to the entrance of 100K.

Figure 6 (above) completes the day's space-time trajectory for a worker, joining the 100K area with the Hanford entrance along the previously traveled route. Note the spatial (x, y, z) relationship between the entry space-time trajectory segment (lower) and the departure space-time trajectory segment (upper). This illustrates the problem of using the same path in space in two or more different time periods discussed earlier. Up until this point, we have shown the construction of the workers' space-time trajectory. The images in figures E and F begin to demonstrate the power of adopting the space-time representation in identifying the spatio-temporal relationship between workers and potential exposure to hazardous material.

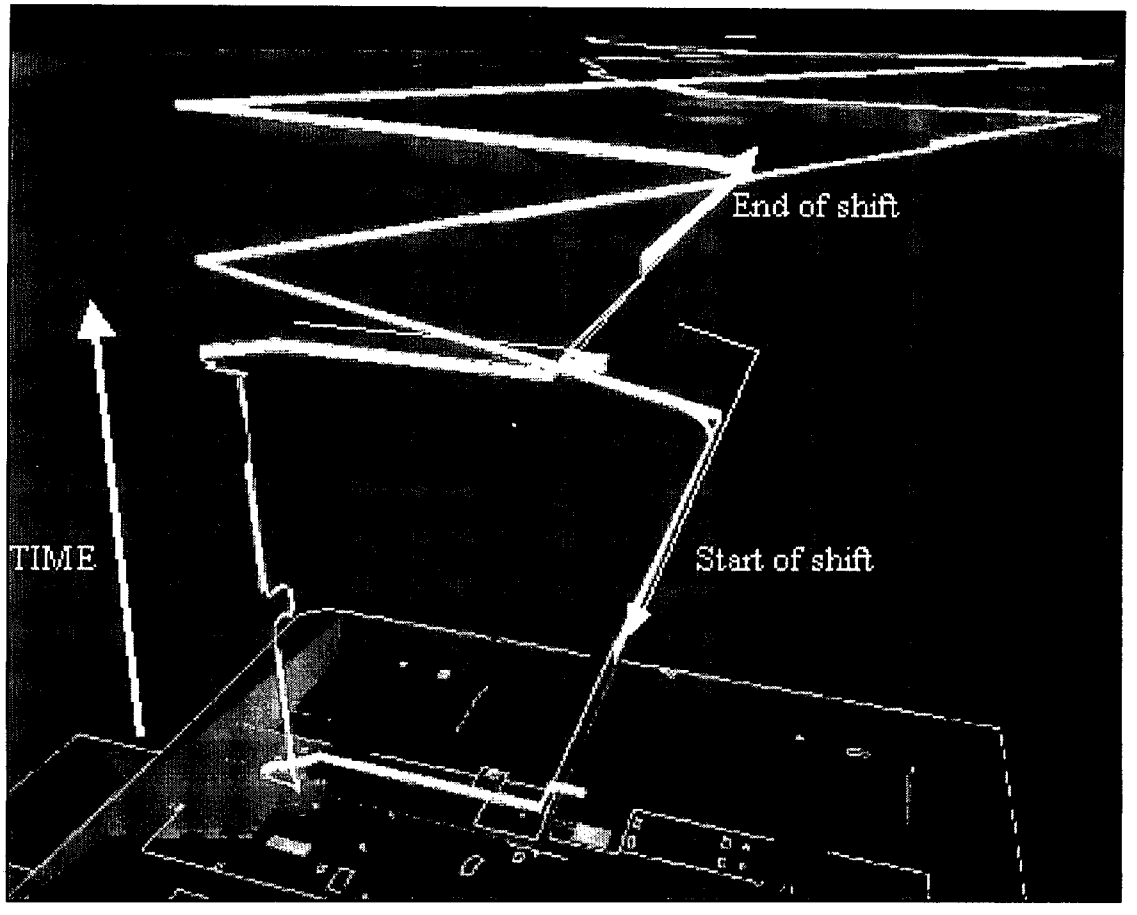


Figure 6: Complete Space-Time Trajectory for one worker).

The fifth image (Figure 7, above left) shows the addition of the space-time trajectory of the 105KE building, where workers spent most of their working day. Notice that the worker trajectory remains within the 105KE trajectory volume for a lengthy period. This represents the workers' vulnerability to hazard exposure for this duration.

The next image (Figure 8, above right) does not represent the trajectory of 105KE (though it is still there). The green volume represents the space-time trajectory of the spent nuclear fuel handling room within the 105KE building. However, notice that examination of the intersection of the worker trajectory and the handling room trajectory reveals that more of the worker time is spent in the handling room than is spent in the 105KE building alone. We know that worker vulnerability to hazard exposure is greater in this area than in the general 105KE building. This visualization approach is useful to express this difference. Combining radiological dose for this particular building and area with space-time information allows us to observe space-time intersections of workers with areas of different radiological exposure within the same building. This can be seen in Figure 9 (above). The space-time trajectory of workers passes through the 105KE building, but while inside, it also passes through the spent nuclear fuel handling room. This is a highly significant application for determining cumulative worker exposure, and understanding the space-time composition of that exposure. Finally, Figure 10 (above) visualization work provides a sense of the relative proportions of the space-time trajectory for workers at the 100K area with respect to the larger Hanford landscape.

8 Discussion

Visualization in 3D GIS The images shown in this paper do not do justice to the real environment in which this space-time visualization has been conducted. Using the mouse, a user may freely navigate through all environments shown in these images, viewing them from any angle. We suspect that the ability to manipulate rotation about three axes is more powerful for information exchange than previous static, perspective, non-3D space-time depictions. The ability to interactively switch themes (essentially objects) in the view on and off allows focused subsets of data to be viewed and minimizes occlusion difficulties. This is

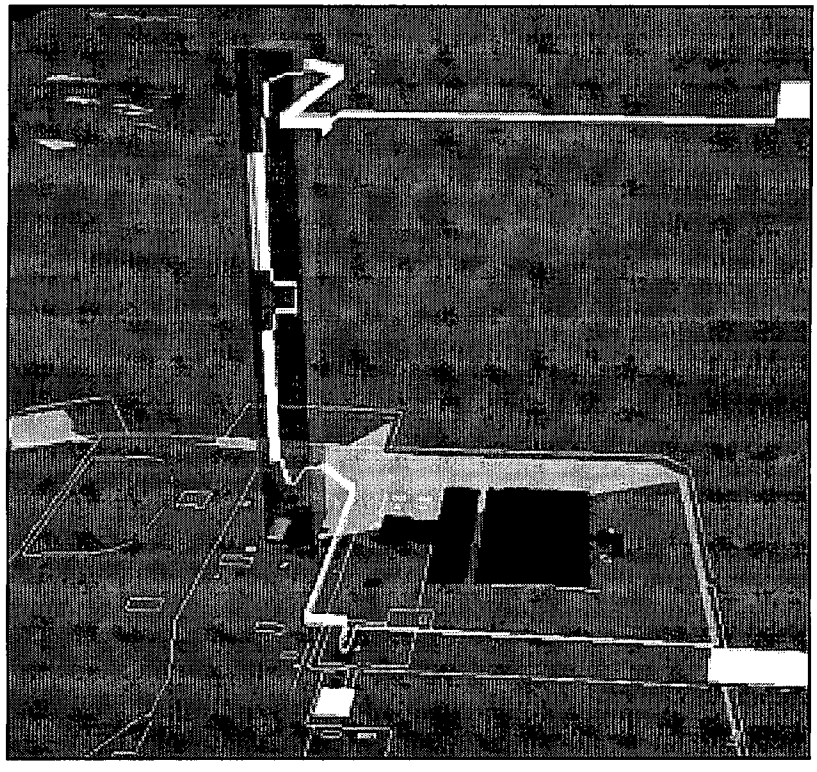


Figure 7: Addition of Space-Time trajectory of 105KE building and resulting intersection with worker Space-Time trajectory.

often encountered when information is presented using three dimensions, and clearly a problem for earlier 2.5-D representations.

Evaluating Utility: Initial Responses There is great value in presenting the spatio-temporal relationship between worker movement and hazardous material location and extent as we have done here. Once the basic 'rules' are explained (e.g., that the z-dimension is time), users may determine the intersection of objects. Early findings suggest that users become rapidly engaged by these representations. This is obviously something that should be empirically tested in a more thorough study of user interactions with this tool. Our working 3D GIS visualization was formally presented to research and management personnel from Hanford, primarily from the Consortium for Risk Evaluation with Stakeholder Participation (for more information, see http://www.cresp.org). Initial feedback was positive, and points to the value of such visualizations for technical specialists (e.g., scientists) and decision and policy makers. These responses indicate that in a more developed state, these visualizations would aid decision-making, policy analysis, and human resource and safety management at Hanford. The greatest unknown at this point seems to be the value to general public stakeholders. These unknowns include the ability to engage stakeholders with space-time data without alienating them as a result of the technology used, or the conceptual frameworks underpinning the representations.

Collaborative Applications Analyses using space-time geography are useful in the Hanford context for several reasons. They could provide a better way to build shared understanding among groups discussing occupational health vulnerabilities on site. Our approach provides a different way to introduce complex issues to those unfamiliar with the problems and circumstances at Hanford. This approach could be used for capturing and tracking cumulative exposures to radiation for individuals. The apparent potential for this approach to be extended to networks of workers, additional hazardous materials, and different occupational risk data, is great, as shown in the future directions discussion. Using 3D GIS allows powerful expression of the time dimension, it helps keep track of project status and progress, and it can combine individual trajectories to aggregated expressions of risk or vulnerability. Real-world settings that strive to attain ideal semantic consistency conditions in collaborative settings may find this conceptual framework an invaluable tool to illustrate and make accessible important environmental characterization and remediation information in public forums.

Data Considerations, Aggregation, and Misinterpretation Our visualization development was tailored to less-than-ideal data. While we were able

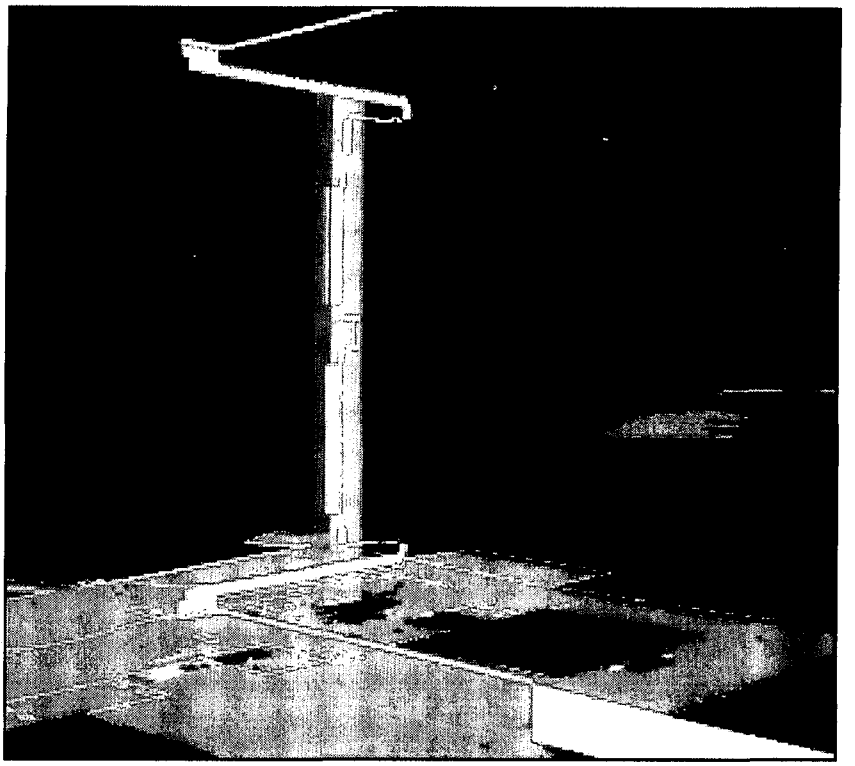


Figure 8: Spent Nuclear Fuel (SNF) Handling Room.

to represent the worker data we were provided, this involved significant temporal aggregation.. This raises questions about how 'better' (less aggregated) data would look, and impact users' spatial reasoning processes. It is suspected that above a certain temporal resolution, the resulting space-time trajectories may become challenging to interpret. With more detail, one might anticipate finer, smoother characteristics of the trajectory. While this would be beneficial in terms of precision, information such as rates of motion might cause difficulties in the interpretation of these visualizations. Different rates of motion might give an inaccurate impression of what is going, when viewed by the user. There is significant risk of misinterpretation of rate-of-motion slopes in the trajectory for route curvature.. The trajectory suspended in the time (z) dimension above the flat landscape might confuse users in mentally referencing different trajectory segments to the on-the-ground roads and pathways. Curved roads and pathways might compound this problem still further. Curved trajectories are likely to be more difficult for users to detect and reference to space-time metrics. There certainly appears to be a tradeoff to be made between revealing more information with linear trajectory segments instead of aggregated ones, versus misinterpretation of linear trajectory segments due to illusions caused by different rate-of-motion slopes and viewing angles.

Another opportunity for misinterpretation might arise from the use of conventional cues in unfamiliar ways. This is an observation that has been made in the cartographic literature (MacEachren, 1995). An example in the space-time visualizations discussed so far, might be the 'extruded' space time paths of building footprints being mistaken as volumetric buildings themselves. This issue could be addressed in the next stage of development, by developing a basic visual coding system for possible objects in a given 3D visualization.

Use of 3D GIS The implementation described in this work collapses the elevation dimension of the real-world landscape into a flat two-dimensional plane. A common use of 3D GIS is the representation of topography using some geometric model. This raises a challenging question for the space-time visualization presented here: Is it possible to represent both elevation and time simultaneously in the same z-dimension? We suspect that landscapes with large variations in elevation will complicate the process of efficiently representing time on the z-axis, while landscapes with minimal variations in elevation (such as the Bonneville Salt Flats in Utah, USA), would have negligible impact on the space-time trajectory. Bear in mind however, that 'large' and ' minimal' variations would be defined by the investigator and would depend upon the specific case. The general idea however, would be

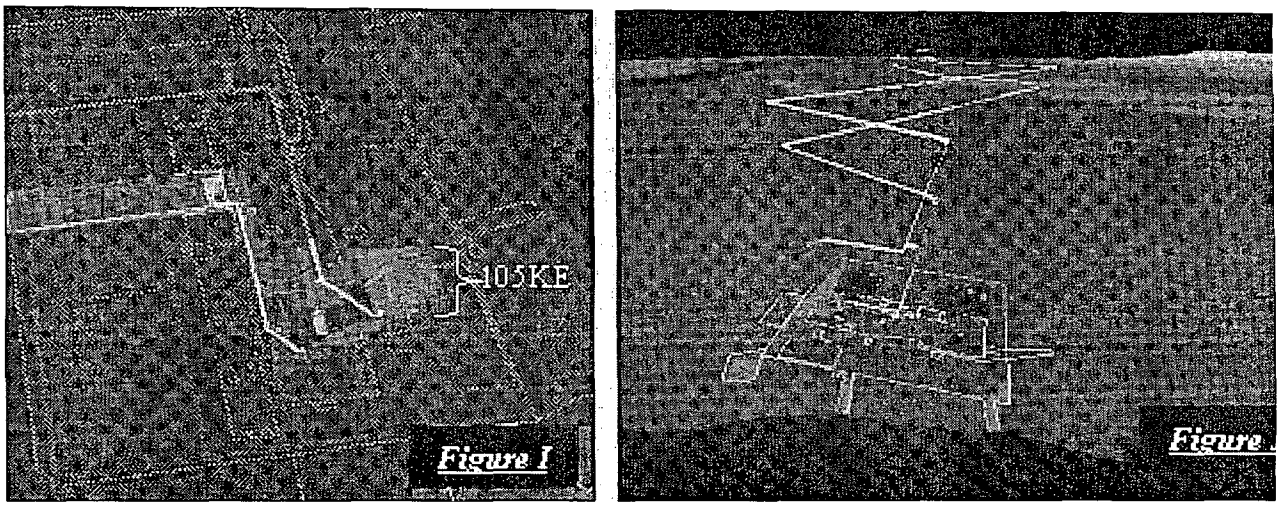


Figure 9: Space-time trajectory of the spent nuclear fuel.

that the start time would coincide with the maximum landscape elevation for that spatial location. All time dimensions would be added to that z-value.

Our case study at Hanford dealt with buildings. Fortunately for us, the buildings were single-level. What if they had multiple levels where a worker could move through? How would that situation be dealt with? This is particularly problematic to represent using the strategy we have proposed so far. The use of elevators or stairs (especially descending) would greatly complicate the meaning of the trajectory. At this time, there is no easy solution to this challenge. Perhaps instead, this is a situation whose characteristics are incompatible with the space-time trajectory strategy we use. This situation may help us identify the boundaries of an 'ideal' or appropriate operating resolution of information for workers. This might be a situation where an alternative representation strategy is used.

3D GIS and not VRML A natural question arises from this work: why 3D GIS and not something like VRML as a tool to represent the space-time trajectories of data at Hanford? VRML could comfortably deal with the geometric challenge posed by the same spatial path being used in multiple time periods. However, it was our intention from the beginning to maintain as much accessibility to the underpinning GIS data. VRML has distinct advantages over ArcView GIS(tm) 3D Analyst(tm) in terms of ability to manipulate 3D geometric models. Although 3D Analyst(tm) is equipped with a VRML conversion macro, being a new extension it is not fully reliable in transforming input data into VRML worlds yet. This was unacceptable. The next stage of development would most likely take place in a VRML world, with custom-built metrics and interfaces to ensure representative transformation of GIS data into manipulable geometric models.

9 Conclusions

This research has proposed space-time visualizations as a powerful tool for facilitating user access to complex spatial databases. This approach has potential in complex settings. We have provided an example of this in our Hanford case study, where it allows us to clearly represent the intersections of humans and hazards in space-time. This is a fundamental aspect of radiological exposure, and a valuable application of these techniques to a real-world. The research undertaken here shows a robust methodology with a sound conceptual basis, implemented using readily available software.

This research is one of few projects to fuse Hagerstrand's visual space-time components, advanced spatial analytical tools and a contemporary real-world application. It may be considered a stepping stone to a larger integrated undertaking to visualize the space-time trajectories of spatial entities (as suggested in Figure 12).

We have presented a methodological approach that may make spatial databases more accessible visually and conceptually to wider user audience, where fostering fairness and communication between groups is a fundamental requirement for democratic progress to occur. This visualization has already been enthusiastically received by task group members of the Consortium for Risk Evaluation with Stakeholder Participation. The logical next step is to perform formal subject testing of this representation approach and to compare its performance as an information tool with

existing representation approaches, before moving to the next stage of development.

In the next stage of development, several findings of this work will be applied. Firstly, 3D implementation of the visual components of Hagerstrand's space time framework appear to have great potential as a foundation for information tools used in complex human and resources management systems. Secondly, landscapes simplified to flat planes are likely to interfere least with the conceptual basis of space-time visualization where the z-dimension is time. Another consideration is that perhaps the space-time visualization strategy presented in this paper is unsuitable for multi-story structures through which trajectories pass. Careful consideration must be given before this specific situation is integrated into the existing approach. Fourthly, the next stage of development should attempt to integrate GIS and geometry-based visualization environments other than the standard ArcView GIS(tm) 3D Analyst(tm) environment. This environment inhibits user interaction and exploration in several ways, many of which are linked to the navigation through and manipulation of 3D space. The authors believe that development in this direction may produce tools and techniques that most efficiently maintain the conceptual underpinnings of the research strategy, while providing appropriate information tools to the real-world case study situation.

One final consideration, is that this approach to representing landscapes, individuals, hazards and structures in space-time requires fairly specific data. While it is possible to think of this as a constraint, we choose to consider it a tool that may facilitate the integration of formerly isolated data into an information system from which many will benefit.

10 Future Work

There are many potential applications for this approach that could be developed in the future. These include: moving beyond vulnerability assessment (worker exposure to hazards) to more detailed analyses; incorporating elements such as exposure, impacts, land use, documentation, uncertainty, institutional structures, and so forth. An example of improving the space-time visualization would be to add hazard values to the trajectory, which would allow us to display cumulative worker exposure to radiological hazards (see Figure 11, below).

At a larger scale, our approach to representing behavior, task and process at Hanford in space-time visualizations could be developed into a fully integrated tool (see Figure 12, next page). The tool would allow multi-scale visualization of multiple synchronous worker, group, and object (such as hazardous waste material) space-time trajectories. This may allow us to determine specific tasks that involve greater vulnerabilities than others, based on exposure/proximity to waste, duration of shifts in specific locations, and so forth.

Users of the tool would be able to define the desired information to be shown, using specific databases, variables and models. The tool would have several modes of operation (such as different scales at which to view and interact with trajectories and data), also allowing the user to focus on subsets of the Hanford site (in the same way this research has), simply by selecting the appropriate icon. This is conceptualized in the 'exploded' views projecting from the main body of Figure 12.

Essentially the tool would become a mechanism by which all parties involved at Hanford might gain access to relevant spatial databases. It would be possible to view the site at one of several levels of abstraction, defined by the user. Ultimately, we are aiming for technological transparency - where diverse stakeholder groups (as at Hanford) - view data, and understand it in the same way as any other user. This is an issue that requires extensive empirical testing in future work. The development of techniques and methodologies such as the one presented here are necessary if we wish to facilitate eventual semantic consistency in and across collaborative groups. The question remains however, whether we will able to determine this through empirical testing. Equally important is whether this particular technology (desktop 3D GIS) is capable of presenting information in ways that allow people to develop semantically consistent mental models of the world.

References

[1] Carlstein, T., Parkes, D.N., and Thrift, N.J., (Eds.). 1978. *Human Activity and Time Geography, Volume II.* London, Edward Arnold.

[2] Chai, Y. and E. Wang. (1997). The Basic Concept and Presentation of Time Geography, *Economic Geography*, 1997 (3), Vol. 17, p. 55-61.

[3] Chrisman, N.R. (1997). *Exploring Geographic Information Systems.* New York: John Wiley & Sons.

[4] Hagerstrand, T. (1970). What about people in regional science? *Papers of the Regional Science Association*, 24, 7-21.

[5] Kwan, M-P. (1998). Space-Time and Integral Measures of Individual Accessibility: A Comparative Analysis Using a Point-based Framework. *Geographical Analysis*, Vol. 30, No. 3 (July 1998), Ohio State University Press.

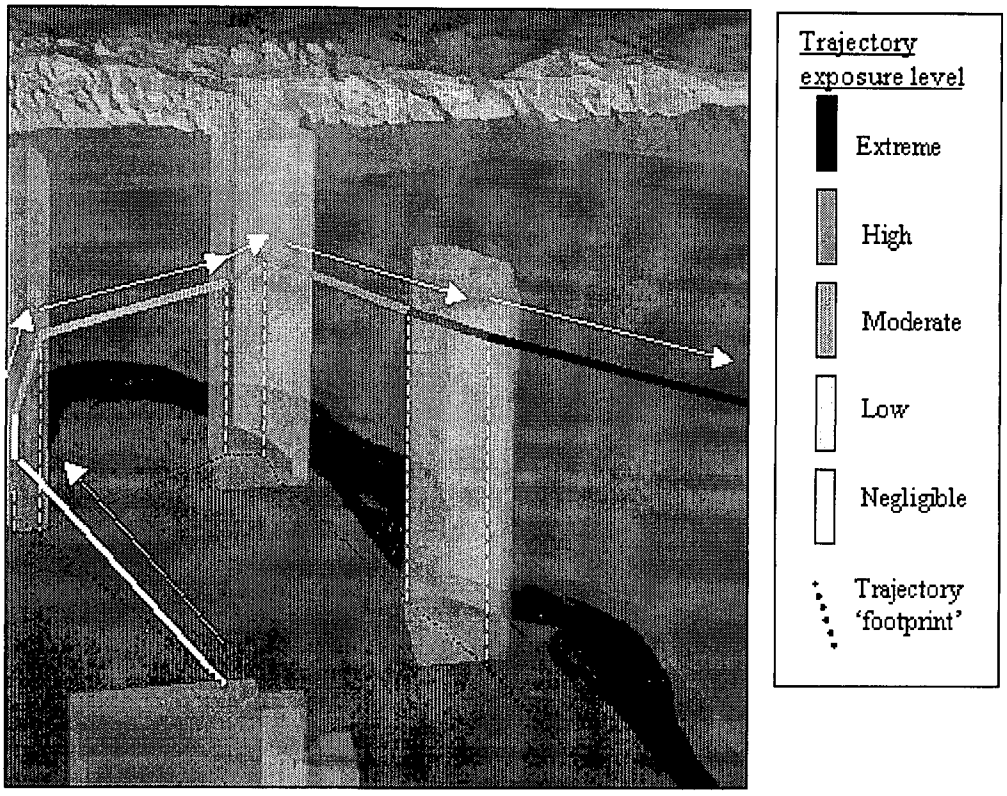


Figure 10: Space-Time Trajectory showing Cumulative Radiological Exposure.

[6] Kwan, M-P. and X-D. Hong. (1998). Network-Based Constraints-Oriented Choice Set Formation Using GIS. *Geographical Systems*, forthcoming (manuscript).

[7] Lenntorp, B. (1978). A time-geographic simulation model of individual activity programmes, in Carlstein, T., Parkes, D.N., and Thrift, N.J., (Eds.) *Human Activity and Time Geography, Volume II.* London, Edward Arnold, 162-180.

[8] Lenntorp, B. (1976). *Paths in Time-Space Environments: A Time Geographic Study of Movement Possibilities of Individuals.* Lund Studies in Geography B: Human Geography, Lund: Gleerup.

[9] MacEachren, A.M. (1995). *How Maps Work.* The Guilford Press: New York.

[10] Mark, D.M. (1997). Cognitive Perspectives on Spatial and Spatio-temporal Reasoning. In Craglia, M. and H. Couclelis, H., (eds.) *Geographic Information Research: Bridging the Atlantic.* London: Taylor and Francis.

[11] Miller, H.J. (1999). Measuring Space-Time Accessibility benefits within transportation Networks: Basic Theory and Computational Procedures. *Geographical Analysis* [Columbus] Vol. 31. No. 1. January 1999. p. 1-26.

[12] Miller, H.J. (1991). Modelling accessibility using space-time prism concepts within geographical information systems. *International Journal of Geographical Information Systems*, 5(3), 287-301.

[13] Parkes, D. and Thrift, N. (1980). *Times, spaces and places: A chronogeographic perspective.* New York: John Wiley & Sons.

[14] Pred, A. (1977). The Choreography of Existence: Comments on Hagerstrand's Time-Geography and its Usefulness, *Economic Geography*, Vol. 53 (2), 207-221.

[15] Webler, Thomas. 1995. "Right discourse in citizen participation: An evaluative yardstick" in Ortwin Renn, editor. *Fairness and competence in citizen participation.* Kluwer Academic Publishers: Dordrecht.

11 Data Sources

Columbia River Comprehensive Impact Assessment (CRCIA) is a study to assess the effects of Hanford-derived materials and contaminants on the Columbia River environment and to evaluate the potential risk

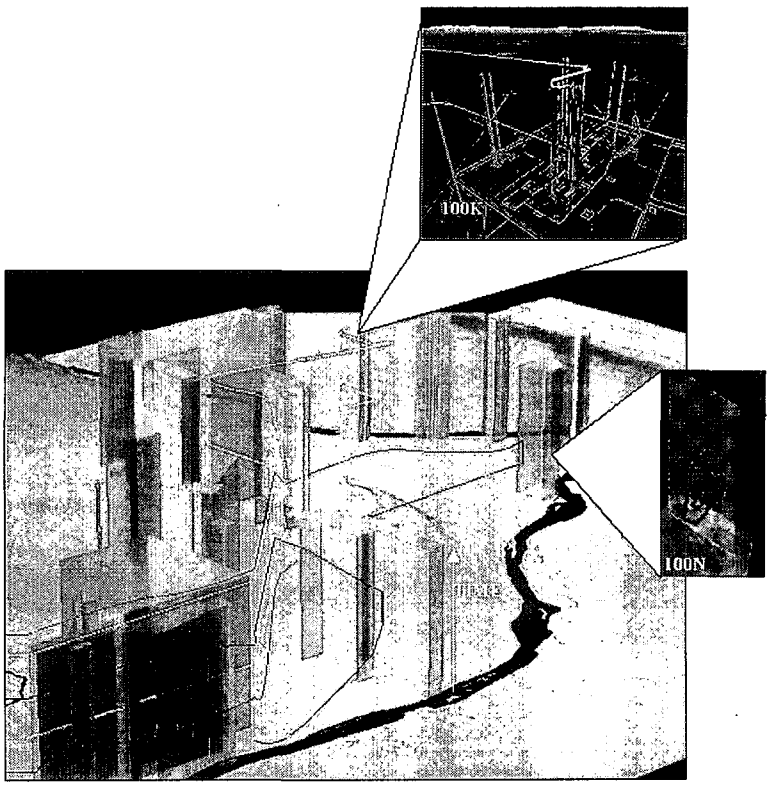


Figure 11: An Integrated Space-Time Tool, capable of handling multi-scale data, and user-defined data visualization.

to the environment and human health. The report divides the area into specific river segments looking at contaminants and receptors and includes various spatial files on the Hanford and the Columbia River.

Hanford Geographic Information System (HGIS) provides detailed maps of the Hanford Site and main features, including buildings, roads, topography, geology, wells, rivers, and so forth. It is currently available at the Bechtel Hanford web site.

Hanford Remedial Action Environmental Impact Statement and Comprehensive Land Use Plan (HRAEIS) is a report by Department of Energy to establish future land-use objectives for the Hanford Site. The scope of this HRAEIS includes RCRA and CERCLA-regulated waste sites in the Columbia River, Reactors on the River, Central Plateau, and all other geographic areas of the Hanford Site. The report includes spatial files for various waste and land use topics such as air emissions, areas, chemicals, elevations, rivers, roads, water, etc.

Radiological Exposure System (REX) maintains and reports individual Hanford worker, subcontractor and visitor radiological records since 1944. REX contains internal dosimetry and limited demographic information. A total of 150,000+ records currently exist on this database which is held by Battelle Pacific Northwest National Laboratory.

12 *Acknowledgements*

The authors would like to thank Dr. Timothy .L. Nyerges for his encouragement, and the anonymous reviewers for their suggestions.

This research was supported in part by the National Research Center for Statistics and the Environment (NRCSE), funded by the United States Environmental Protection Agency through agreement CR825173-01-0 to the University of Washington. This support does not constitute an endorsement by the Agency of the views expressed. This research was also supported in part by the Consortium for Risk Evaluation with Stakeholder Participation (CRESP) by the United States Department of Energy (DOE) Cooperative Agreement #DE-FC01-95EW55084. This support does not constitute an endorsement by DOE of the views expressed.

Modeling an Auditory Urban Database with a Field-Oriented Approach

Robert Laurini*, Ki-Joune Li**, Sylvie Servigne*, Myoung-Ah Kang*
*Laboratory for Information System Engineering
Claude Bernard University of Lyon / National Institute for Applied Sciences of Lyon
INSA - 404 - 69621 Villeurbanne Cedex - France
Email: {Robert.Laurini, Sylvie.Servigne, Myoung-Ah.Kang}@lisi.insa-lyon.fr

**Computer Science Department, National Pusan University
Kumjeong-Ku, Pusan - South Korea
Email: lik@spatios.cs.pusan.ac.kr

Keywords: Multimedia databases, auditory databases, geographic information systems, field-oriented approach, field indexing, continuous indexing

Edited by: Frederick E. Petry, Maria A. Cobb and Kevin B. Shaw
Received: January 4, 1999 **Revised:** May 25, 1999 **Accepted:** June 10, 1999

The goal of this paper is to give some elements in order to design a database dedicated to auditory information in cities. Auditory information is much more larger than traffic noise, because its includes all aspects of sounds which can be found in a city soundscape. Sounds levels can be modeled as a continuous spatio-temporal phenomenon, by means of the field-oriented approach which is very relevant for this kind of database. Indeed its allows the user to see really sounds as continuous phenomena when querying even if they come from samples captured in the city. In this paper, after having introduced the special semantics of auditory information, we will overall present the field-oriented approach and its characteristics. Finally, we will give some indications about the continuous indexing of such data in order to accelerate the queries.

1 Introduction

Sounds are very important in our daily life with a twofold attitude for any citizen. In one hand, when music, sounds are considered as enhancing the quality of life, but in the other hand, traffic noise deteriorates the quality of life. Due to those contradictory characteristics, the new concept of **soundscape** tries to combine both positive and negative aspects of the auditory environment. Presently, and more and more in the future, any urban planning activities try and must try to diminish noise levels everywhere in the cities and perhaps outside, for instance at the vicinity of airports.

Sounds are not only a physical phenomenon, but several psychological and physiological aspects must be taken into account when considering soundcapes. For instance any local government, and practically everywhere in the world, receives daily several complaints regarding noise, but with a very biased distribution such as along very noisy motorways, no people complaint. In contrast, in very quiet residential zones, people send regularly complaints regarding any kind of noise. So, if a city major wants to construct his/her noise reduction policy only on complaints, he/she will primarily act on very quiet precincts instead of carrying action plans along motorways. Bearing those considerations into account, it looks very interesting to develop a system for knowing objectively soundscape in order to act where and when necessary (KRYGIER 1994, SERVIGNE 1998). The French National Institute of Urban Engineering (INGU) has commissioned us to design an auditory information system in order:

- to better know soundscape, everywhere in a city; e.g. to have information everywhere in order to perform comparisons, to detect very noisy precincts, and to propose priorities to the city council;
- to propose a system for visualizing auditory information, perhaps with animated cartography;
- to estimate or simulate sounds impacts, especially for new urban developments.

In order to perform these tasks, measures were made along the city. Several sound recordist teams were sent throughout the city to measure not only sounds levels, but also to tape-record the acoustic signals, typically during one to two minutes. Conceptually, those measures will be considered as samples of a continuous phenomenon, taken randomly across space and time

as illustrated in Figure 1. But, in this paper, only sounds level values will be considered.

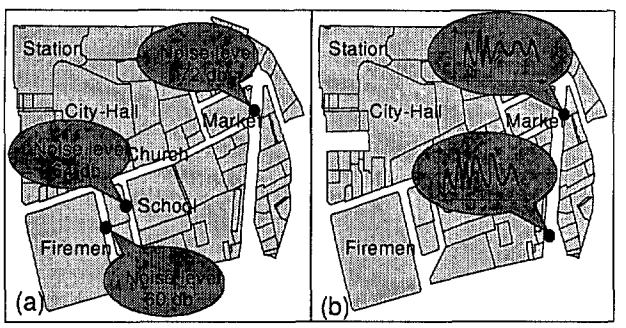


Figure 1: *Example of visualisation of auditory information. (a) Using bubbles to locate where the measures were made. (b) Location of the tape recorded auditory signals.*

One of the difficulties when designing such a computer system is that sounds constitute a continuous physical phenomenon. Usually information systems, and especially geographic information systems store entities representing land features. And the object-oriented model is an interesting tool for modeling geographic features. Mathematically speaking, sounds can be modeled by mathematical fields, which is an interesting way of modeling continuous information (COUCLELIS 1992, KEMP 1993).

More and more geographic phenomena are considered and modeled as spatio-temporal continuous fields. Look for instance at the modeling of temperature: it is easy to see that temperature cannot be modeled as an object, but as a field more precisely a scalar field, whereas winds are modeled as vector fields. One of the problems in field database is indexing. Whereas conventional indexing techniques were developed for discrete data, the application implies to design techniques for indexing continuous data, i.e., functions or more exactly spatio-temporal functions, and let us call it continuous indexing.

The goal of this paper will be to present an urban database especially devoted to store and query urban auditory information based on the field-oriented approach. The paper is organized as follows. After this short introduction, we will define the field-oriented approach, especially by comparing it to the entity approach. Then we will give the key-elements for designing a field-oriented database for sounds. And we will finish by some considerations regarding field indexing.

2 Entity Modeling versus Field Modeling

In our worlds, not all objects are clearly defined. As H. Couclelis (COUCLELIS 1996) said *" Is the world ultimately made up of discrete, indivisible elementary particles, or is it a continuum with different properties at different locations ? This question, already debated by the ancients Greeks, remains one of the major unanswered problems in the philosophy of physics"*. For us, a house, a car can be considered as elementary particles, or atomic objects; but a lake, a sea, a mountain can be divided in portions depending of the human activities. Later she continued *"somehow boundaries are intrinsic to the notion of atom, whereas in the case of extensive entities they are contingent. In other words, the notion of boundary a priori sits better with the atom view of things (and vector GIS) than the plenum view (and raster GIS), whereas the real geographic world forces us to consider both discrete and extensive entities"*.

2.1 What's in a field?

The term of **field** refers to the plenum, i.e. a spatial-temporal continuum which is common in domains such as electricity (electric field), magnetism (electromagnetic field), physics (gravity filed) and so on. Mathematically speaking, a field is defined by functions over space and time. See PARIENTE 1994, KEMP 1993-97, GORDILLO 1997.

When the field is scalar, then we have $f = F(x, y, z, t)$. In other words, a function is defined in any place at any time. To illustrate the problem, let us speak about heat and thermal fields. At any place, at any time, a temperature (scalar) exists and it is possible to find a function able to describe temperatures everywhere and every time.

In geoprocessing, the concept of fields can be elegantly used to model several features:

- the first one is terrain. Indeed elevation z can be defined as a function of x and y, such as $z = F(x, y)$;
- some other attributes such as acidity or permeability of soils, flows in hydrology, infiltration, population densities and so on.

In neighboring domains, such as meteorology, fields are very commonly used, for instance for winds, pressure, rain, humidity and so on.

Figure 2 illustrates the general problem. In Figure 2a, starting from some sampling points, we can imagine the concept of a field as a continuous phenomenon. Even so (2b), some sampled temperatures are not stored, by interpolating between previous information, an estimation can be given for any point

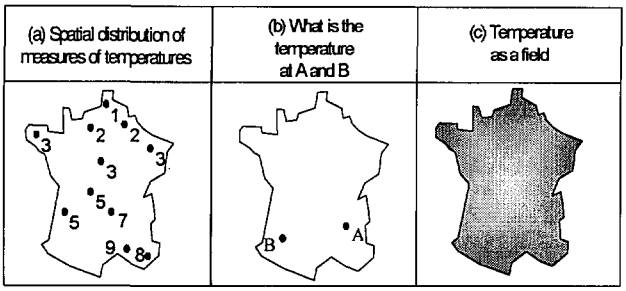


Figure 2: *Temperature as a field. (a) Spatial distribution of some temperature points. (b) What is the temperature of some villages, namely A and B? (c) Smooth representation of temperatures.*

located in the convex hull (See Preparata & Shamos 1986). Finally, Figure 2c portrays a continuous field as shaded values. More generally, temperatures can be considered as a scalar field (Figure 3a), whereas wind as a vector field as given in Figure 3b.

The field attributes can be issued from different domains. We can generally distinguish:

- nominal, which correspond to integer or real value on which any arithmetic operations can be done;
- ordinal, for which an order of magnitude can be defined, for instance the gravity of risks.

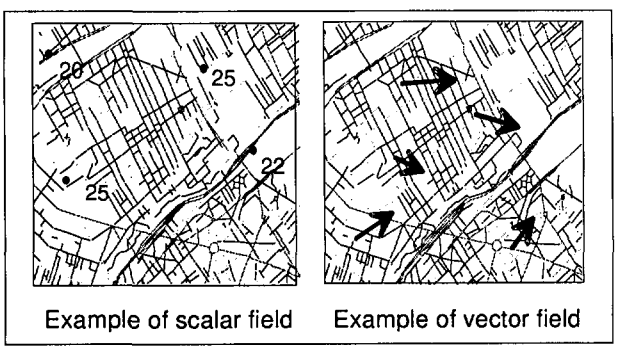


Figure 3: *Some examples of fields. (a) Scalar field. (b) Vector field.*

But in reality, a field is never totally a priori known by global acquisition. Generally, it can be measured in some points, or its average or integral can be measured in some zones. In this case, one has to find a function incorporating the previous values. In Figure 4 is given an example illustrating a case of temperatures (Figure 4a) for which a function, or a sort of smoothing must be performed in order to estimate values anywhere. So, along a cross-section, the shape of this function can be depicted (Figure 4b). Similarly, instead of temperature, a population field can be used. In this case, we will use the density instead of the average in order to generate the field.

2.2 Field Properties

Mathematically fields are defined by continuous functions. It is possible to distinguish several types of fields, at 2D or at 3D, scalar or vectorial. For example, a function $h = f(x,y)$ represents a scalar 2D field. For instance, for a digital terrain, elevation is a scalar 2D field which can be written as $z = f(x,y)$, and annual rain level can be written as $r = f(x,y)$ and daily rain as $r = f(x,y,t)$ where t corresponds to time. The wind is a vector 3D field with time, i.e. it needs three coordinates the represent the direction of the wind; it can be written as a vector such as $wx = f(x,y,z,t)$, $wy = f(x,y,z,t)$, $wz = f(x,y,z,t)$. But if we want to work with the wind velocity, we need to differentiate those functions according to time.

Dimension/ Type	Scalar	Vector
2D	$h = f(x,y)$	$hx = f(x,y)$ $hy = f(x,y)$ $hz = f(x,y)$
3D	$h = f(x,y,z)$	$hx = f(x,y,z)$ $hy = f(x,y,z)$ $hz = f(x,y,z)$

Table 1: *Different kinds of fields*

In the sequel, to alleviate the text, we will speak about the coordinates of a field. By coordinates, we mean be either x,y or x,y and z, or even x,y,z and t, depending of the number of variables in the field.

For the determination of a field, some elements are need:

- to have a set of points together with their value. Starting from those values, the function can be defined contingent to have a mathematical model. In the absence of such a model, some conventional estimation procedures can be launched.
- in some cases, not point values are known, but some zonal local densities or means: let us define them as statistical constraints.
- in some cases, some local characteristics lead to a very strong modification of the field; for instance a cliff in a digital terrain model; those are called morphological constraints.

Let us call seeds the sample points which are used to generate the field. And starting from those characteristics, the field can be estimated. An interesting system was made by Pariente for estimating fields with statistical and morphological constraints based on a

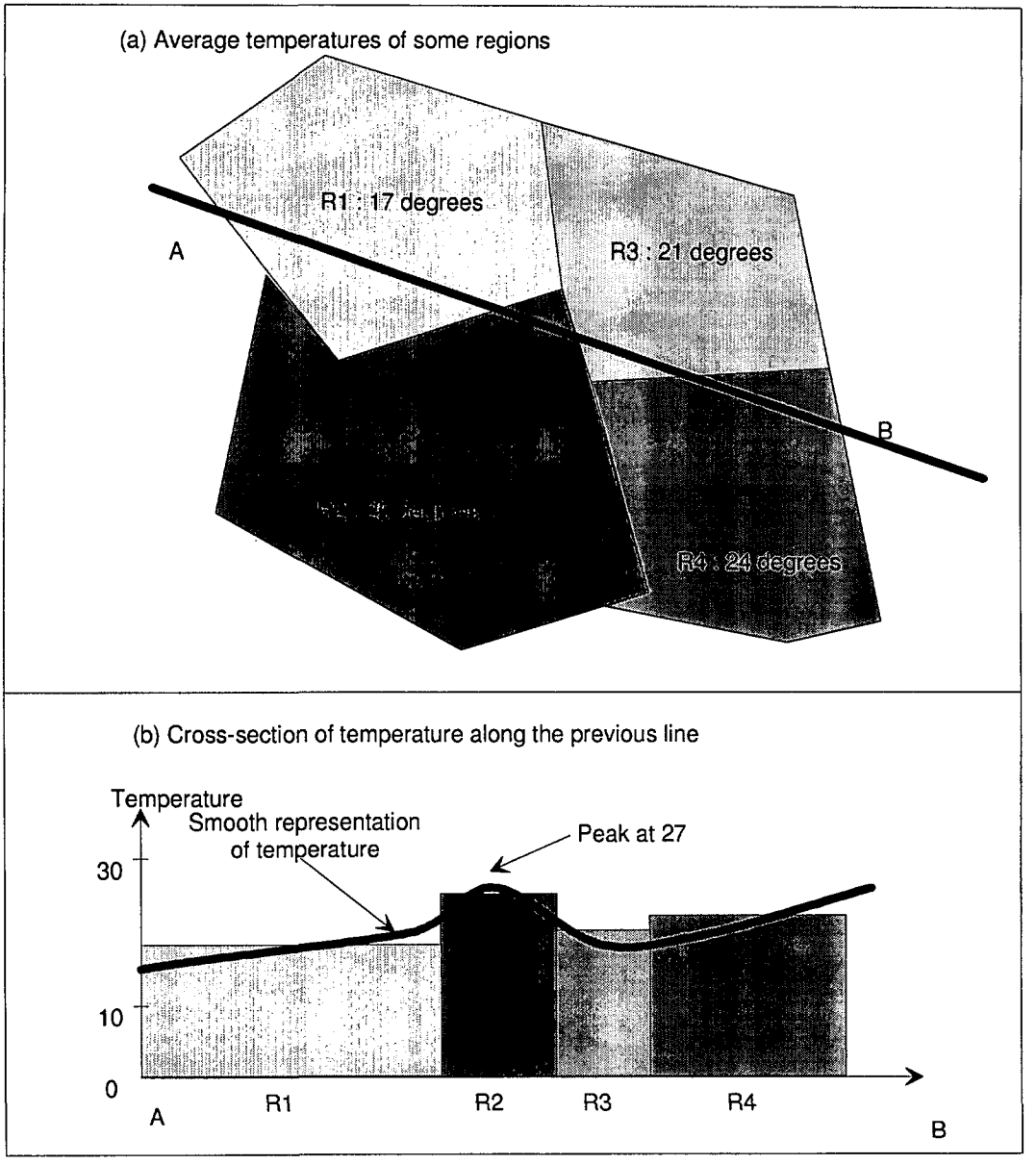


Figure 4: *From averaging to real values. (a) Some regions and their average temperatures. (b) Smooth temperature along a cross-section.*

neural network with a pyramid accelerator (PARIENTE & LAURINI 1993, PARIENTE 1994b, PARIENTE & SERVIGNE & LAURINI 1994).

2.3 Constraints

A Geographic Information System (LAURINI & THOMPSON 1992) should model reality as closely as possible. When attribute values have only been measured at a subset of all grid points, the values at the remaining grid points need to be estimated as accurately as possible, at the same time preserving salient features of the observed reality. The interpolation model must satisfy the user's needs, but must also respect morphological and mathematical constraints as well as information arising from a statistical study of the environment: all these constraints must be taken into account. The following section describes four main constraints. not all of which are taken into account by existing interpolation methods.

2.3.1 Morphological discontinuities

The system must be able to take into account the possible discontinuities in the studied phenomenon (modeled by means of a continuous field). A number of geographic objects can be considered as discontinuities, such as cliffs, roads, rivers, walls, geological fractures and political frontiers. These objects depend on the category of the phenomenon, and describe the relationships between geographic objects. If the boundary of a geographic entity corresponds to a discontinuity, it will be taken into account in the continuous field estimation process. The system must be able to take into account the possible discontinuities in the studied phenomenon (modeled by means of a continuous field). A number of geographic objects can be considered as discontinuities, such as cliffs, roads, rivers, walls, geological fractures and political frontiers. These objects depend on the category of the phenomenon, and describe the relationships between geographic objects. If the boundary of a geographic entity corresponds to a discontinuity, it will be taken into account in the continuous field estimation process.

2.3.2 Continuity and differentiability

The general trend of attribute values should be continuous, regular and differentiable (in order to avoid a crisp fracture in the values, unless it corresponds to a discontinuity discussed above). Values at sites that are close together in space are more likely to be similar than others located further away, and to depend on each other from a statistical point of view (spatial autocorrelation).

2.3.3 Preserving sample point attribute values

The estimation method must return the exact value of a sample point, and not an estimated one. Thus exact interpolation methods (returning exact values) are preferred to approximate ones (yielding estimates). Some mathematical methods do not meet this constraint, overall cause preference is given to differentiability rather than precision. It is important to preserve the original values of sample points because of possible problems of database integrity.

2.3.4 Attribute values for points or zones (statistical constraints)

Attribute values can be either the mean value of a zone (or the standard deviation), or can refer to the attribute value at one particular point. In the first case (attribute of a zone) it is useful to include statistical constraints in the estimation process, allowing constraints to be specified using the standard deviation, or regression parameters, or the mean value of a region. In this way, measurement errors can be reduced.

This (by no means exhaustive) list of constraints for a satisfactory interpolation method has been presented in order to meet precise requirements. In general, classical interpolation methods meet only a small subset of these constraints, which is why new methods of estimation are needed, together with a language including all specifications to improve the quality of estimates. See PARIENTE 1994a for details.

2.4 Field as an Abstract Data Type

As previously explained, fields, or more exactly field values are a new sort of attributes for object. So the key-idea is to consider fields as an abstract data type, in which we must mention:

- the nature of the field (temperature, altitude, hygrometry, sounds, etc.);
- the dimensions of the field where, e.g. a 2 or 3D, or 3D + time scalar field corresponds to spatial coordinates x, y and possibly z and t and one spatial variable (attribute value) w. If the field is a vector field, the dimension is greater, and this field can be described as x, y, z, t, w_1, w_2, ... , w_n where w_1, w_2, ... , w_n is the attribute vector of point x, y, z and t;
- the list of intervals of definition for each spatial dimension; for example, for a scalar field, the dimensions can be defined as [-3.4975, +128423.1] for x, and [5479.12, 10000] for y; so a sort a multidimensional bounding box will be defined;

- an interval of dates, corresponding to the validity of the different components of the field. Only components with a validity interval corresponding to that of the field will be taken into account (or with an infinite interval of validity);
- a set of sample points;
- a set of statistical constraints;
- a set of discontinuities.

```
Type FIELD
    Representation
        nature_of_field : string,
        dimension : integer,
        interval_for_each_dimension :
                list of (Interval),
        interval_of_validity : Interval,
        samples : set of (Sample_Point),
        statistical_constraints :
                set of (Statistical_Area),
        morphological_constraints :
                set of (Discontinuity)
    Methods
        Interpolation
```

`Interval` is an abstract data type corresponding to intervals of real numbers (such as [2.3, 78.23]; note that it is possible to specify an infinite interval by indicating an interval value R (set of real numbers) corresponding to $]-\infty, +\infty[$. The class `Sample_Point` can described such as:

```
Class Sample_Point inherit Point
    Representation
        Point_ID : Point,
        attribute_vector : list of (real),
        validity_sample : Interval
```

Each object of class `Sample_Point` possesses attributes, such as:

- the point coordinates (x, y and possibly z);
- the vector of attribute values (temperature, altitude, etc.); if it is a scalar field, this vector will contain only one element. In the case of a vector field, this vector will contain several attribute values according to the dimensions of the field;
- the period of validity of sampling.

The class `Statistical_Area`: modeling statistical constraints (here, only on the mean value of a zone, but it can be extended to standard deviation):

```
Class Statistical_Area
                    inherit Geographic_Area
    Representation
        Geographic_area_ID :
                Geographic_Area,
        mean_of_the_geographic_area :
                real,
        validity_stat : Interval
```

The attributes correspond to:

- the name of the geographic area concerned with the statistical constraint;
- the value of the mean to be reached;
- the period of validity of the statistical constraint.

The class `Discontinuity` allows the modelling of morphological constraints i.e. barriers.

```
Class Discontinuity inherit Line, Polyline
    Representation
        Geographic_line_ID :
                Geographc_Line
        nature_of_discontinuity : string
        validity_discontinuity : Interval
```

The attributes correspond to:

- the geographical line;
- the nature of the discontinuity (a wall, river, road, frontier, etc.);
- the period of validity of the discontinuity constraint

In addition, some functions and operators will allow us to handle the objects needed to represent the continuous field, as described above. The main operations can be classified as basic management (adding, removing or modifying objects, fields or attributes) and queries.

Let F be a continuous field of data. Now Field is regarded as a new abstract data type, providing the definitions of continuous scalar or vector fields. Declaration of scalar and vector fields with their components (sample points, statistical areas and discontinuities):

```
Field F ( name_of_field,
      nature_of_field,
      dimension,  /* of the field */
      (inter1, inter2),
       /* intervals of def for each dim */
      [begin_date, end\_date],
      /* interval of validity */
      (Sample_ID1,...),
       /* list of sample point */
      (Stat_area_ID1,...),
```

```
        /* list of constraints */
         (Discontinuity_ID1,...)
        /* discontinuities */
)
```

Other classes could be introduced in this model, like for the support of acquired data, such as the class of spatial models used (grid cell, polygons, triangular irregular networks, a regular grid of points, irregularly spaced points or contour lines) or the class of interpolation method used (see KEMP 1993) and these classes would be aggregated to the class `Field`. In this model (Figure 5), interpolating techniques are not mentioned, so classes like spatial data models and interpolation methods are not included in the specification of the object-oriented model. For another model of field data, refer to (GORDILLO & BALAGUER 1998).

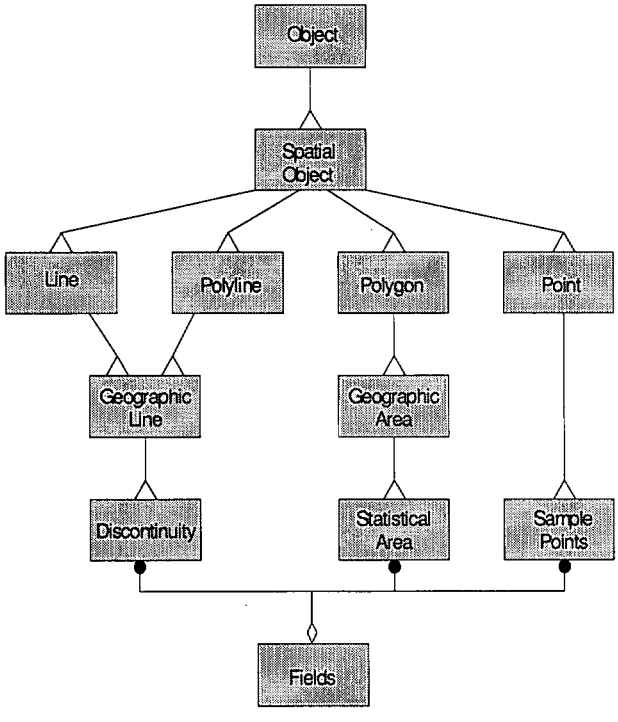


Figure 5: *Object-oriented model emphasizing the links between geographic objects and continuous fields*

2.5 Field Operators

For the manipulation of a field, among others we need to add or remove constraints. In addition, we need to compute the gradients, the derivatives, and the integrals.

2.5.1 Definitions of Field Operators

The first operation is to compute the value of the field at a given location starting from the coordinates. In other terms, we need an interpolation mechanism in order to estimate the value of the field. Once the value is determined, it is possible to estimate the gradients and the integral over a zone.

Finally, queries must return the estimated attribute value or the gradient of a point, or the integral or the density of a zone. For the attribute value of a point,

```
Estimation (F, #point )
```

returns a real value, the attribute value of the specified point, if the field is scalar. If it is vector field, it returns a vector. For the gradient of a point,

```
Gradient (F, #point )
```

returns a vector of three elements, which is the gradient of the specified point. For the integral of a zone,

```
Integral (F, #geographical_area )
```

returns the integral of the domain specified by the area. For the density of a zone,

```
Density (F, #geographical_area )
```

returns the density of the domain specified by the area.

When the field is temporal, all operators must include generalized dates (year, month, day, hour, minute, etc.) as parameters such as:

```
Estimation (F, #point, date )
Gradient (F, #point, date )
Integral (F, #geographical_area, date )
Density (F, #geographical_area, date )
```

In some cases, the integral and the density may be computed not only for a precise date, but for a period of time, so giving expressions as follows:

```
Integral (F, #geographical_area, period)
Density (F, #geographical_area, period)
```

The last functionalities refer to spatio-temporal continuity. in order to estimate those values, in addition to spatial interpolation some temporal interpolation procedures must be realized.

This list of query operators is by no means exhaustive; other operators can be defined in order to construct a complete for continuous fields.

2.5.2 Operators for Constraint Management

The second set of operators is linked to the management of constraints. For statistical constraints, we can add or remove samples and, means. For morphological constraints, we can add or remove some discontinuity. In some cases, we can modify some elements such as discontinuities. See PARIENTE (1994a) or LAURINI & PARIENTE (1996) for details.

2.6 Physical representation of fields

There are several ways to represent field data into a computer, especially depending on the structure of input sampling data. They can be represented by a generalization of well-known techniques for storing digital terrain models (see Laurini & Thompson 1992 for details). Among them, let us mention:

- when the input data are regularly distributed over space, for instance a grid, the raster format can be chosen. In this case, it is sometimes nicknamed as devil staircase or giant causeway;
- when the data are irregularly distributed, a Delaunay triangulation and a Voronoi tessellation can be constructed (Preparata & Shamos, 1986); this representation can be seen as an extension of TIN (Triangulated Irregular Networks);
- in some cases, isolines are interesting as a generalization of contour levels.

In general, grid and triangulation systems look very interesting for storing and querying, whereas isolines look more interesting for visualisation.

In the case of sounds, all these models can be used: in small places, or in the countryside, isolines (here called isophones) are the more relevant way. However, in cities due to the existence of barriers such as façades, isophones are not very convenient. Due to the spatial distribution of sampling points, a technique based on constrained triangulation can be used.

3 Design of a Field-oriented System for Sounds

For designing a relevant database systems for auditory information, we need to capture their semantics. Among the characteristics, we can mention:

- due to physiological aspects, sounds levels are measured logarithmically in decibels;
- sounds generally diffuse radially when there are no obstacles; but in cities due to the existence of lot of objects (car, trees, façades, and so on), we have some difficulty to accept this assumption. In other words, for making interpolation between samples, a very sophisticated model will be necessary;
- in this study, after discussion with acoustic experts, linear interpolation (Figure 6) of values in decibels will be implemented as a first step. When acoustic experts will give us the more sophisticated model, we will replace the linear interpolation method by this new one.

In addition to conventional queries, we can give three kinds of very specific soundscape queries:

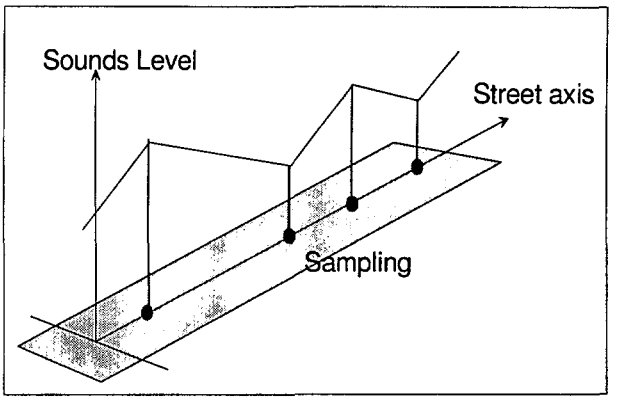


Figure 6: *Linear interpolation method for noise evaluation.*

- spatio-temporal queries based on field-oriented approach and interpolation methods. An example of this kind of query can be *"what is the mean sounds level in the 12th street at 4 o'clock am ?"*
- queries to identify similarity between sounds records based on signal similarity. In this cases, an example of sound is given in the framework of *"sounds by example"*. See FALOUTSOS 1996.
- queries based on key-words as criteria.

Only the first query will be addressed in this paper. In this section, after having given some fundamentals in acoustics, we will only explain some elements in the acquisition, the updating of auditory information and the necessary operators.

3.1 Some Elements on Acoustics

One of the ultimate goals of an urban sound information system is to detect noisy places where people are hindered so as to enhance their environment. Indeed people trouble is not necessary proportional to the noise level but depends also on physiological aspects, and that it is unavoidable to try to describe noise not also with quantitative aspects but with *qualitative* aspects. In this paper, only *quantitative* aspects will be considered.

Two types of quantitative criteria characterize a sound :

- the equivalent continuous noise (*Leq*) which need different measures during a day to obtain a representative average of noise environment. The *Leq* depends on:
  - delay of analysis period
  - level of instantaneous noise.
- the three statistical levels :

- L_{50} : level of average noise obtained during 50% of the analysis period,
- L_{100}, L_{99} : level of background noise,
- L_1 : level of peak noise.

3.2 Data acquisition

A lot of various information must be required to characterize a soundscape. Information can be *quantitative* (ex: sounds level) but also *qualitative* to be more realistic (ex: pleasant or not). The numerous information which is needed leads to difficulties, first to identify them as key variables, then to make data acquisition, and to finally model them.

Among others information required concerns :

- physical description of space: topography, buildings, road networks...,
- description of space occupation (sound impact): animals, buildings (ex: school), public places (ex: gardens, parks)...,
- urban acoustic data (sounds level of traffic, variation...),
- indicators of sound identity based on socio-economic and psychological information,
- sounds records.

Data acquisition is achieved by :

- collecting data in existing urban databases,
- sounds measurements and collecting in decibels (db) by sound recordists,
- organizations of surveys (city-dwellers, sites),
- recording sounds on various places (typically 1 to 2 mn).

3.3 Updating

Cities evolve and so sounds do. In the design of an information system, updating is a crucial issue. From a computer point of view, two aspects are emerging:

- the necessity to store and update spatio-temporal information, implying perhaps the use of versions in the database,
- the requirements of simulation of the future imply to work with different scenarios of evolution; here also the concepts of versions is a key-component.

Presently, the measures are daily stored with bulk updates. But in the future, one can imagine a real-time system with sensors distributed in selected places throughout the city. But this aspect is outside the goal of this paper.

3.4 Conclusions

To conclude this section about the modeling of sounds using the field-oriented approach, let us say that this approach is very interesting. However façades appear to act like barriers in the diffusion of sounds. So, the goal of this project is only to consider the storing and retrieving of auditory information in public spaces.

As an example of field-orientation, let us give very rapidly the declaration of a city in which we can find several zones, the city itself, downtown and the main square, and two punctual zones (town-hall and hospital). In addition, three fields are used, namely for modeling population, temperature and for auditory information, anywhere in the city.

```
City Elasty-City (
    town_area : Geographic_Area,
    downtown_area : Geographic_Area,
    town_hall_location : Point,
    hospital_location : Point,
    main_square : Geographic_Area,
    city_hall_temperature:
          TEMPERATURE
          (town_hall_location),
    hospital_temperature :
          TEMPERATURE
          (hospital_location),
     town_population :
           POPULATION(town_area),
     downtown_population :
          POPULATION(downtown_area),
    town_soundscape :
          SOUNDS(town_area),
     main_square_soundscape:
          SOUNDS(main_square),
....)
```

Now that we know more about the modeling and the querying of auditory information, let us give some elements regarding the indexing of sounds levels.

4 Field Indexing for Sounds Levels

As previously said, field databases consist of a large amount of data, like other types of geographical data, which requires to be stored on secondary memory. And, if the structuring is not adequately done, it may considerably degrade the performance of the system, especially due to the disk access time. It is, therefore very important to index field data for accelerating the access and processing time. The main goal of indexing is to find where requested data are stored on disk, without scanning the whole disk space. Each type of data needs its proper indexing method, and field data

also requires a specific indexing method. However, as far as we know in the literature, no proper indexing method has been proposed or developed. In this section, we will investigate the problems and possible approaches of field indexing for sounds.

4.1 Typical Field Queries for Sounds

In order to develop an indexing method for a system, we need first to clarify its purpose, which may be described by a set of operators, or a set of typical queries. While operators for alphanumeric or one-dimensional data are simple, those for spatial data are complex and diverse, and they are even more complex for field data. It is nearly impossible to investigate indexing method for all kinds of field data and we therefore restrict our focus on field operators only for sounds. And even sound field has several aspects, such as levels, directions or signal of sounds, in this section, we will deal only with the level of sounds as field data and leave other aspects as open issues.

In this section, we examine a list of typical field queries for sounds. It may be incomplete but long enough to show the problems and motivation of the field indexing for sounds data. Suppose that we have field data for noise levels within a certain region: they may be represented such as terrain models, for example contour lines, DEM (Digital Elevation Model) or TIN (Triangulated Irregular Network) (LAURINI & THOMPSON 1992). But we do not treat details on the physical representation of field in this section.

A field query consists of three variables, namely spatial variable S, temporal variable T, and level variable L concerning sounds level. Since a field can be defined by an mathematical equation, $f(L, S, T) = 0$, a field query can be represented by a function of one or two specified variables for calculating the other variables. Each variable can be a value or a set of values. First S is a point, a line or a region. Second, time T is also a value t, an interval $I=(t1, t2)$ or a recurrence $C = (I, P)$ where I is an interval within given period P. By recurrence, we mean, for instance, *Tuesdays afternoon, summer nights,* and so on. For example, a temporal condition for *"every day at 3:00 a.m."* can be described as (3:00, 24), which means 3:00 per every 24 hours. And L can be also a value or an interval. The response of the query depends on the type of S, T and L. Now we can classify the types of queries according to the types of query returned values as follows:

Q1 : $f^{-1}(S, T) = L$, (Estimation Query)
Find sounds level on a given area S for a given time T.
Q2 : $f^{-1}(L, T) = S$,
Find region with sounds level L for a given T.
Q3 : $f^{-1}(L, S) = T$,
Find time with sounds level L on a given region.
Q4 : $f^{-1}(L) = (S, T)$,
Find time and region (S, T) with sounds level L.
Q5 : $f^{-1}(Lmax) = (S, T)$ (Aggregation Query)
Find time and region (S, T) with maximum sounds level L.

Q1, the estimation query, is the most simple among the above query types, and the result depends on the types of S and T. For example, it is a surface, when S is a region and may be a value when S reduces to a point. The second type of query, Q2 is to find the region from conditions about time and sounds level. In contrast, Q3 is to find temporal set with spatial and sounds level conditions. While we inquire the region and corresponding time with a given sounds level by Q4, Q5 is to find the same thing but with maximum sounds level. And when S in Q5 is a region, one intends to find region and time that the average of the sounds level is maximum, as shown by Figure 7.

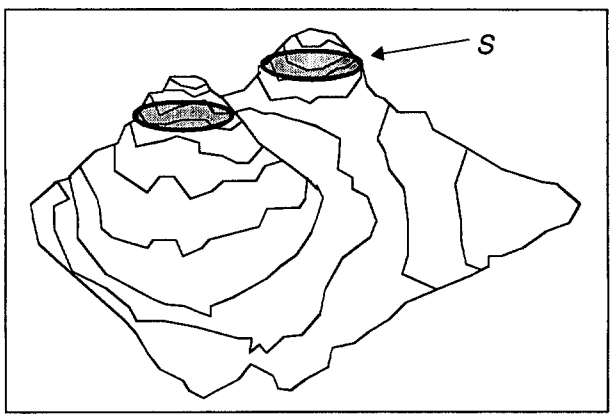


Figure 7: *Regions with maximum sounds level.*

Note that the queries listed above have only spatial aspect, if the value of temporal variable is infinite. For example, if T in Q1 is $[0, \infty)$, Q1 becomes only field query with only spatial aspect. Also if the value of spatial variable is infinite, the queries have only temporal aspect.

4.2 Field data indexing versus field indexing

The concept of field for modeling continuous phenomena is essentially an intensional way of modeling the real world based on seed points. Let us remind that an extensional view of a set is based on the declaration of all objects belonging to a set, whereas in an intensional view, the objects are described by properties. In our case, it is impossible to define the infinite number of objects (field points) belonging to the field. But those objects have in common to follow the Laplace equation of the field.

As a consequence, when the space is described with only with "all-fashioned" (i.e. extensional) spatial objects, conventional indexing are adequate, whereas since fields are described intensionnally, fresh indexing techniques must be used.

So, now the question is: do we have to index the field itself or the field data? What are the difference between those two indexing techniques ?

- **field data indexing** means that we want to index all the data supporting the field (field seeds); in this case, all conventional techniques used for indexing spatial and spatio-temporal data can be re-utilized. But if some peaks (for instance the peak at 27 degrees in Figure 4) and pits of the field are not present in the sample, they will never be retrieved by the user. In this case, field data indexing is equivalent to the indexing of only sampled data used in the field (seeds).

- in the other hand, **field indexing** means the indexing of the whole field, i.e. not only the seeds, but also all other field points, the number of which is infinite. In this case, since it is impossible to index an infinite number of points, new techniques must be introduced, for instance based on subfields. By subfield, we define a finite number of subsets of the field, each of them being described, for instance by a sort a multidimensional bounding box. So now, all peaks and pits will be either retrieved or estimated, and finally provided to the user. But the main difficulty remains the optimal splitting of the field into subfields.

Indeed let us examine some common spatial indexing techniques (GAEDE & GUENTHER 1998), temporal indexing techniques(SALZBERG & TSOTRAS 1997) or spatio-temporal indexing techniques. Their goal is to index discrete data never continuous data; in other words, they are valid to store field seeds but not the complete continuous field. Obviously some assumptions behind discrete indexing methods can be reused for continuous indexing.

Taking this idea into account, some spatial or spatio-temporal techniques can be used as a basis to field indexing, but with the difference that we are not indexing isolated objects, but continuous subfields and the corresponding seeds.

4.3 Indexing Structure

Before mentioning field indexing method for sounds, we need to explain the physical representation for field data of sounds. Since we treat only level of sounds, they may be represented as a surface like terrain. And terrain model such as DEM and TIN could be a good representation method for the field data for sounds. However, when we take into account the temporal aspect, its representation becomes very complex. Suppose that we have a TIN to represent a field, each vertex of triangle has a series of values according to time. However, we do not deal with detail method for field representation in this section, and rather generalize it to survey the indexing method. The only assumption that we make is that the field is divided into a number of auditory subfields, which are simplified by a tuple as follows,

$$F_s = (I_x, I_y, I_L, I_t) \quad (1)$$

where each I represents an interval of x, y, sounds level L and time t, respectively. In fact, this tuple means a minimum bounding box of a subfields in 4-D. Therefore, a field can be described by a set of minimum bounding boxes. This simplification implies that we are going to apply the *filtering and refinement strategy* to process queries. This strategy is very efficient in reducing the number of disk accesses by comparing only minimum bounding boxes during the filtering phase without examining the complex spatial objects stored on disk. Only the candidate objects selected during filtering phase are to be carefully examined during the refinement phase. The improvement of performance is considerable and it has become a common strategy in processing spatial operators.

And the simplification also means that it may be possible to apply one of traditional spatial indexing methods, such as R-tree (GUTTMAN 1984), Grid-File (NIEVERGELT et al. 1984), Quadtree (SAMET 1984), etc., to index field data for 4-dimension. But as mentioned by (THEODORIDIS et al. 1998), temporal dimension is not just like another dimension due to its peculiarity. For example, some properties such as the ordinality or the recurrence that we explained in the previous section, cannot be found in other dimensions. We will discuss on field indexing methods by separating them into two categories. First we are going to examine the field indexing methods without considering temporal aspect. And then we will try to investigate the indexing method taking temporal aspect into account.

4.3.1 Without temporal aspects

As we ignore the temporal aspect in a field database, the field reduces to a set of 3-dimensional auditory subfields $F_s = (I_x, I_y, I_L)$. Two approaches are possible to index field data without temporal aspect. The first is to generalize spatial access methods which are divided into two classed; PAMs (point access methods, for example, grid file) and SAMs (space access methods, for example, R^*-trees) (GAEDE & GUENTHER 1998). For indexing field data, 3-D SAM such as 3-D R^*-tees (BECKMANN et al. 1990) can be used because the subfields can be considered as 3-D rectangles.

However this approach has weak issues. The auditory subfields are not uniformly distributed along the axis of sounds level, but skewed around specific values as shown by Figure 8. It may lead to many overlaps between minimum bounding boxes of 3-D R^*-trees and may degrade the performance of indexing in comparison with the case of the uniform distribution of data (KIM et al. 1995). The second problem comes from the consideration of the dimension for sounds level as other dimension(x, y). In fact, the spatial operators or queries, such as containment, nearest neighbor, and spatial join differ from those for field data, described in the previous section.

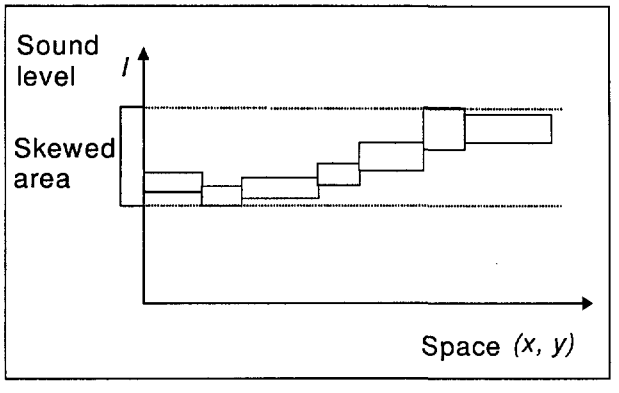


Figure 8: *Distribution of auditory subfields*

The second possible approach, called Hybrid Indexing, is to separate the dimension of sounds level from spatial dimensions. It means that we build two indexing structures, one for sounds level and one other one for spatial dimension. As far as spatial dimension is concerned, we simply use one of traditional spatial indexing method. And there are several possibilities for indexing sounds level, which are normally represented as an interval I_L=[L_{min}, L_{max}]. First, indexing methods for temporal data, such as AP-tree (GUNADHI 1993), IP-index (LIN et al. 1996) and interval B-tree (ANG 1995) may be employed. But since the temporal interval differs that of sounds level and they could give a bad performance. An example of such difference is that the intervals of sounds levels are skewed around some specific values as shown by Figure 8, while temporal intervals are relatively uniformly distributed. The second possibility is to transform each interval I_L=[L_{min}, L_{max}] to 2-dimension of (L_{min}, L_{max}). Each interval becomes a point in transformed space then any PAM can be used.

The transformation gives an interesting property. For example, Q2 can be simply processed by the indexing with transformation method as Figure 9 shows. When a query such as *"find the region where sounds level is between 50db and 60db"* is given, the subfields in area VI satisfy the query and those in area IV and V will be the candidates. The careful refinement on the candidates will give a correct answer.

Now we will examine how the 3-D SAM and hybrid method work to process the typical queries listed in the previous section. Suppose that we have a query of Q1 type, for example, *"find the sounds level for a given area W"*. With 3-D SAM and hybrid method, it is straightforward to find the result. By 3-D SAM, for example 3-D R^*-tree, we can easily find the minimum bounding boxes intersecting with area W by recursive manner. And the leaf nodes in R^*-tree point the candidate subfields which are eventually refined for the correct answer. It is also easy to process such query by hybrid method. It is sufficient to apply any PAM to find subfields intersecting with given region W.

For the second query type Q2, we need a more sophisticated processing than Q1. Suppose that a query *"find the region where sounds level is between 50db and 60db"* is given. By 3-D SAM, the query processing is also straightforward like Q1. We can find the candidate subfields by filtering minimum bounding rectangles whose values of sounds level intersect with [50db, 60db]. But the processing method of this query type by hybrid indexing differs from that of Q1. While the processing of Q1 uses the index for spatial dimension, we need to employ the index for sounds level dimension. For example, we can easily find the subfields satisfying the query condition, that is, falling on zones IV, V, and VI in Figure 8, by the transformation method. By other methods belonging to this catergory, such as AP-tree, interval B-tree, IP-tree or MAP 21 (NASCIMENTO & DUNHAM 1997), it is possible to find the subfields, though the performance may vary more or less.

Since the query types Q3 and Q4 are related with temporal aspect, we discuss on the corresponding methods in the end of the section. Instead, we see the indexing method for processing Q5 with 3-D SAM or hybrid method. Suppose that we have a query *"find the point where the sounds level is maximum"*. First, we can also employ 3-D SAM to process this query. With 3-D R^*-tree, we can easily find the subfield with maximum sounds level, which is in fact the minimum bounding box whose upper bound of sounds level is maximum. However when we want to find the region of given area (for example, $100m^2$) with maximum sounds level, the processing method becomes a little complicated. We select the minimum bounding box whose upper limit is maximum (bounding box A in Figure 10). And we remove the minimum bounding boxes whose upper limit is less than that of the selected bounding box. By repeating this procedure in recursive way, we can finally find the candidate auditory subfields that are to be carefully examined on refinement phase. This mechanism is in fact a typical example of filtering and refinement strategy based on R^*-tree and we can apply the similar process for other

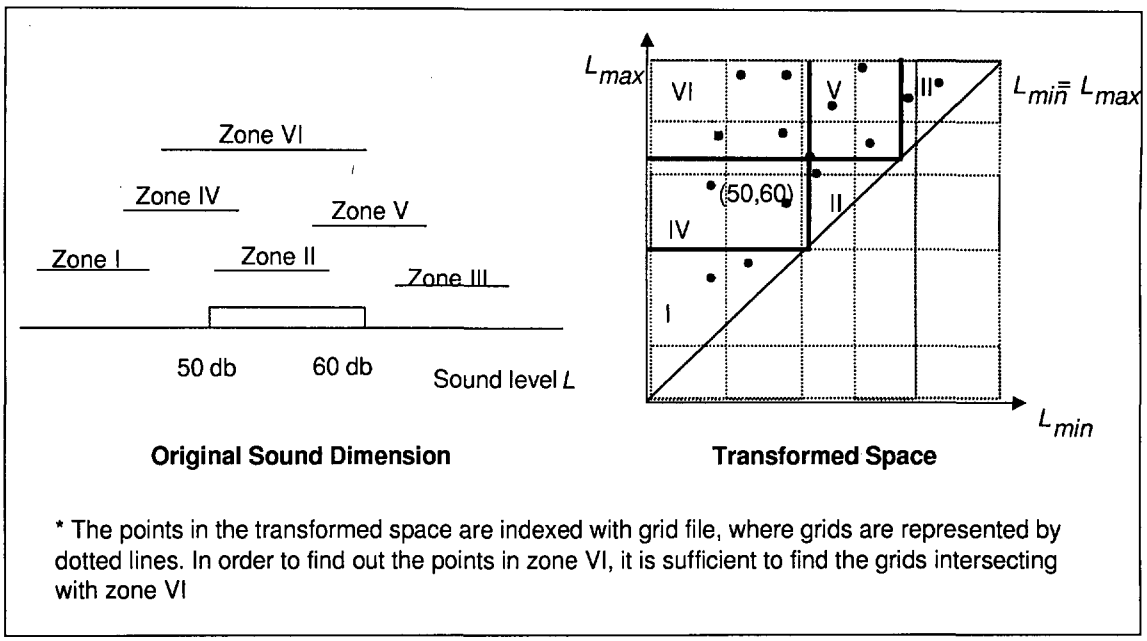


Figure 9: *Indexing by Transformation Method.*

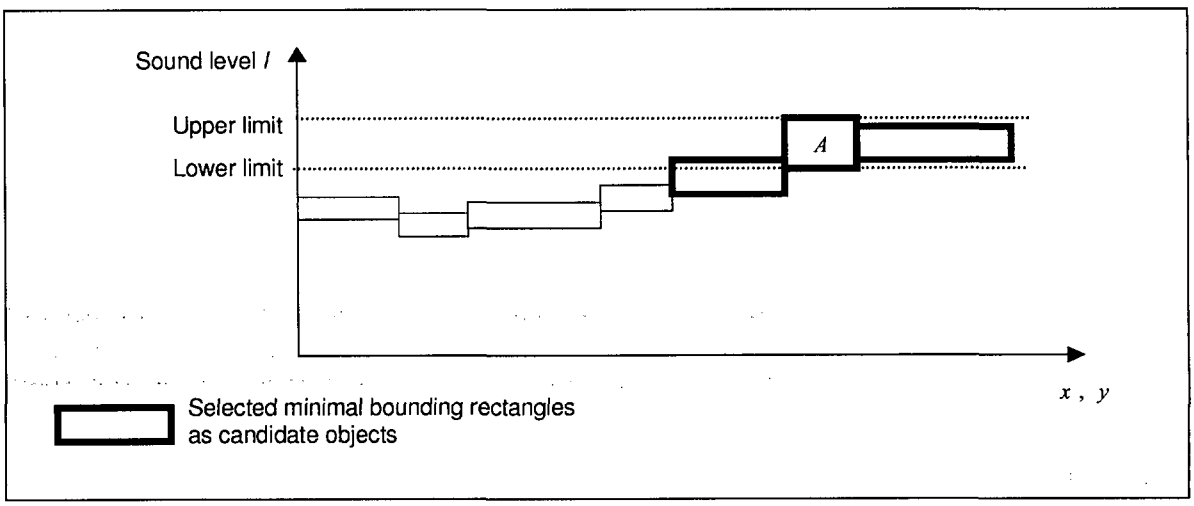


Figure 10: *Indexing for Q5 with 3-D R*-tree.*

3-D SAM.

As far as hybrid method is concerned, the processing method for this query depends on the type of index on sounds level. For example, we can find the subfield with maximum subfield by accessing to the rightmost leaf node on AP-tree or interval B-tree. And when transformation is employed, the uppermost point on transformed space corresponds to the subfield inquired.

4.3.2 Taking time into account

When we consider temporal aspect, the indexing method becomes much more sophisticated. The field indexing method with temporal aspect is related with spatio-temporal indexing, as field indexing without temporal aspect is with spatial indexing. It means that the indexing method on spatio-temporal data is an emergent research area and no remarkable researches have been done (THEODORIDIS et al. 1998). For the same reason, it is difficult to find any satisfactory work on field indexing with temporal aspect.

A number of spatio-temporal indexing methods has been proposed and they are distinguished into two categories with respect to the basic spatial index structure used as follows. The methods belonging to the first category are the extensions of R-tree (GUTTMAN 1984), which simplifies spatial objects by their minimum bounding boxes for spatial index: 3D R-trees (THEODORIDIS et al. 1996), MR-trees and RT-trees (XU & LU 1990), and HR-trees (NASCIMENTO & SILVA 1998).

- 3D R-trees treat time as another dimension and is used for the management of spatio-temporal multimedia objects in a authoring tool.
- MR-trees and HR-trees construct R-tree at each time-stamps and overlap them to obtain the advantage of utilization of common node and path.
- RT-trees incorporate the time information into the node of the index, in order words this method couples time intervals with the spatial ranges in each node of the tree in order to overcome the disadvantage of MR-trees when there is not many common path.

THEODORIDIS et al. (1998) introduce the specifications of spatio-temporal index structure, evaluates and compares the spatio-temporal access methods of this category with respect to the specifications.

The second category includes the methods which use quadtrees and the idea of overlapping B-tree in order to represent successive states of the objects according to the change of the time: Overlapping Linear Quadtrees (TZOURAMANIS et al. 1998). This method is used to store consecutive raster images according to transaction time by means of overlapping linear quadtrees in order to avoid storing identical subquadrants of successive instances of image data evolving over time.

	R-Tree	**Quadtree**
(a) Extension of temporal dimension	3D R-trees	Octree
(b) Overlap the spatial indexing structure	MR-trees, HR-trees	Overlapping Linear Quadtrees
(c) Modification of spatial index structure	RT-trees	

Table 2: *Comparing various indexing techniques*

If we classify the spatio-temporal access methods mentioned above according to the spatial access method used and to the technique for incorporating time information, the following table (Table 2) can be obtained.

In Table 2, the columns (R-Tree, Quadtrees) represent the used spatial indexing methods for spatio-temporal indexing method. And the rows ((a), (b), (c)) represent the tecniques for temporal information in spatio-temporal indexing method.

In general, the variants of R-tree, in special the case of (a) in the above table, are not efficient in cases where minimum bounding boxes include a large amount of dead space, i.e., the case of moving objects, for example the navigation of a car. By contrast, the methods employing overlapping technique are not suitable if a number of objects move from a temporal instant to another. The reason is that there is no many common paths between the index structures to be overlapped. The case (c) requires the good and sophisticated strategies which have not been well proposed and tested until now for node splitting considering the spatial and time characteristics in the same time.

However, it is practically impossible to directly employ spatio-temporal indexing methods to temporal field indexing due to the peculiarity of field and temporal data. And the spatio-temporal methods mentioned above did not pay a full attention on such characteristics. For example, the recurrence makes it difficult to represent and index field data. Suppose a query *"find sounds level on a region for every Sunday"*, which is a typical example of Q1. The simplest way to represent the recurrence is to extend the dimension to 4-D including temporal dimension and respectively locate all recurrences on the dimension. By this approach, we treat temporal data as others without any specific consideration, and it may simplify the method in a considerable way, but it is supposed to give a bad performance.

However, the temporal data possesses different properties from other kinds of data and a careful understanding and exploitation of them may facilitate

to develop a good temporal field indexing method. It remains as an open issue to develop temporal field indexing methods.

5 Conclusions

The goal of this paper was to give on overview of a new kind of database systems allowing to deal with spatio-temporal continuous data, and an application especially devoted to urban sounds. Due to the physical characteristics of sounds, and the psychological and physiological impacts over humans, it is very important to propose a new kind of database.

By considering sounds levels as mathematical fields, and sounds values as a new abstract data type, the key-elements of a new modeling technique were given in this paper, one of the difficulty being the indexing not only of sampling values, but also of other elements for speeding up queries.

Field-oriented models can thereof be considered as an interesting approach for all spatio-temporal continuous phenomena. Sounds are one of them if we consider level values. Here only a very simple linear interpolation mechanism was used. But in the future, when acoustic models will better mimic the diffusion of sounds especially in cities, more relevant information systems will be offered to urban decision-makers.

In addition, we have tried to present a solution of a new problem, i.e., continuous indexing method. Starting from ideas for discrete spatio-temporal indexing methods, we have extended those techniques in order to index continuous spatio-temporal data, namely subfields. Back to auditory database, one of the main difficulty is the visualisation of sounds, and an example is given Figure 11.

Another very challenging aspect is the modeling of tape-recorded auditory information. Here the problem will not be anymore the simple interpolation between sounds values, but also the interpolation of signals.

Presently, we are finalizing the specifications of a prototype. When the prototype will be in use, some other interesting research problems must be carried out, especially for the storing and retrieving of tape-recorded acoustic signals and their interpolation throughout the public space.

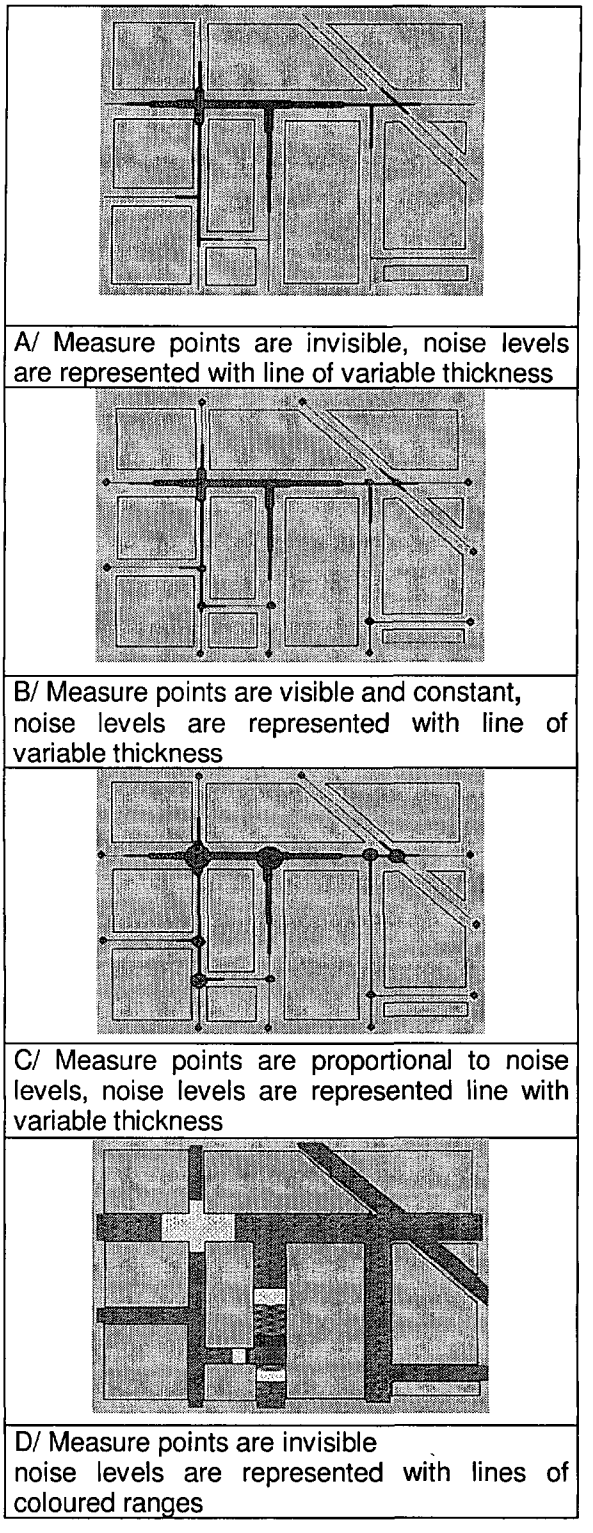


Figure 11: *Different representations of sounds levels in streets*

References

[1] ANG C., TAN K. (1995) The Interval B-tree. *Information Processing Letters, 53(2)*, pp.85-89.

[2] BECKMANN N., KREIGEL R., SCHNEIDER R., SEEGER B. (1990) The R*-tree: An efficient and robust access method for points and rectangles. *Proceedings of 1990 SIGMOD*, pp. 322-331.

[3] BURROUGH PA, FRANK AU (1996) Geographic Objects with Indeterminate Boundaries. Taylor and Francis. 345p.

[4] COUCLELIS, H. (1992) People Manipulate Objects (but Cultivate Fields): Beyond the Raster-vector Debate in GIS, in: Frank, A. U., Campari, I. and Formentini, U. (Eds), Theories and Methods of Spatio-temporal Reasoning in Geographic Space, *Lecture Notes in Computer Science 639*, pp. 65-77, Berlin: Springer.

[5] COUCLELIS, H. (1996) Towards an Operational Typology of Geography Entities with Ill-defined Boundaries, in Geographic Objects with Indeterminate Boundaries. Edited by Peter A. Burrough and Andrew U. Frank, 1996, Taylor and Francis, pp. 45-55.

[6] FALOUTSOS C. (1996) Searching Multimedia Databases by Contents. Kluwer Academic Press. 155p.

[7] GAEDE V., and GUENTHER O. (1998) Multidimensional Access Methods. *ACM Computing Surveys 1998*, Vol.30, No.2, pp.170-231

[8] GORDILLO S. (1997) Modelización de Campos Continuos en Sistemas de Información Geográfica. Master in Computing. National La Plata University, Buenos Aires, Argentina. 77 p.

[9] GORDILLO S., BALAGUER F. (1998) Refining an Object-oriented GIS design Model: Topologies and Field Data. *Proceedings of the* 6^{th} *International Symposium on Advances in Geographic Information Systems (ACM-GIS)*, pp.76-81, Washington, D.C., USA.

[10] GUNADHI H., SEGEV A. (1993) Efficient Indexing Methods for Temporal Relation, *IEEE Transaction on Knowledge and Data Engineering*, 5(3) pp.496-509.

[11] GUTTMAN A. (1984) R-trees - a Dynamic Index Structure for Spatial Searching, *Proceedings of the ACM SIGMOD Conference*, pp.47-57, Boston MA.

[12] KEMP, K. (1993) Environmental Modelling with GIS: A Strategy for Dealing with Spatial Continuity. Technical Report No. 93-3, National Center for Geographical Information and Analysis, University of California, Santa Barbara.

[13] KEMP K. (1997) Fields as a Framework for Integrating GIS and Environmental Process Models. Part 1: Representing Spatial Continuity; Part 2: Specifying field variables. *Transactions in GIS*, Vol 1, n 3, pp 219-234 and pp. 235-246.

[14] KIM M.S., SIN Y.S., CHO M.J. and LI K.J (1995) A Comparative Study on Spatial Access Methods. *Proceedings of the* 3^{rd} *ACM-GIS*, Baltimore, USA, 1995.

[15] KRYGIER J. (1994) Sound and Geographic Visualization. In Visualization in Modern Cartography. Edited by A. K. Mac Eachrew, D R Fraser-Taylor.

[16] LAURINI R, PARIENTE D. (1996) Towards a Field-oriented Language: First Specifications. In Geographic Objects with Indeterminate Boundaries. Edited by Burrough PA, Frank AU. Taylor and Francis. pp 225-235.

[17] LAURINI R., THOMPSON D. (1992) Fundamentals of Spatial Information Systems. Academic Press.

[18] LIN L., RISCH T., SKOLD M. (1996) Indexing Values of Time Sequences. *Proceedings of CIKM*, 1996.

[19] NASCIMENTO MA, DUNHAM MH (1997) Indexing Valid Time Databases Via B+-trees - The MAP21 Approach. Technical Report 97-CSE-08, Southern MethodistUniversity

[20] NASCIMENTO MA, SILVA JRO (1998) Towards Historical R-trees. *Proceedings of ACM Symposium on Applied Computing (ACM-SAC)*, 1998.

[21] NIEVERGELT J, HINTERBERGER H., SEVCIK KC (1984) The grid file: An adaptable Symmetric Multikey File Structure. *ACM TODS*, 9(1), pp. 38-71, March 1984.

[22] PARIENTE D. (1994) Estimation, Modélisation et Langage de Déclaration et de Manipulation de Champs Spatiaux Continus. PhD, Institut National des Sciences Appliquées de Lyon, 6 December 1994. 192p.

[23] PARIENTE D. (1994) Geographic Interpolation and Extrapolation by Means of Neural Networks. *Proceedings of the 4th European Conference on GIS, EGIS'94*, Paris, March 29-April 1, 1994. Utrecht: EGIS Foundation, 1994. p 684-693.

[24] PARIENTE, D. and LAURINI, R. (1993) Interpolation and Extrapolation of Statistical Spatio-Temporal Data Based on Neural Networks. Presented at the Workshop on New Tools for Spatial Analysis, DOSES/EUROSTAT, Lisbon, Portugal, November 18-20, 1993, pp 92-101.

[25] PARIENTE D., SERVIGNE S., LAURINI R.(1994) A Neural Method for Geographical Continuous Field Estimation. In: Neural Processing Letters, 1994, Vol. 1, No 2, p 28-31.

[26] PREPARATA F., SHAMOS M. (1986) Computational Geometry, an Introduction, Springer-Verlag.

[27] SALZBERG B. and TSOTRAS V.J. (1997) A Comparison of Access Methods for Temporal Data. to appear in ACM computing Surveys.

[28] SAMET H (1984) The quadtree and related hierarchical data structure. *ACM Computing Surveys*, Vol.16 No.2, pp.187-260.

[29] SERVIGNE S. (1998) Towards a Sonorous Urban Information System. In Proceedings of the COST-UCE C4 Rome Workshop: Information Systems and processes for Urban Civil Engineering Applications. Edited by U. Schiavoni. Luxembourg: Office for Official Publications of the European Communities 1998, EUR 18325 EN, pp. 191-202.

[30] THEODORIDIS Y., VAZIRGIANNIS M., SELLIS T. (1996) Spatio-Temporal Indexing for Large Multimedia Applications. *Proceedings of the* 3^{rd} *IEEE Conference on Multimedia Computing and Systems (ICMCS).*

[31] THEODORIDIS Y., SELLIS T., PAPADOPOULOS A., MANOLOPOULOS Y. (1998) Specifications for Efficient Indexing in Spatiotemporal Databases. *Proceedings of the 7th Conference on Statistical and Scientific Database Management Systems (SSDBM)*, pp.123-132, Capri, Italy.

[32] TZOURAMANIS T., VASSILAKOPOULOS M., MANOLOPOULOS Y. (1998) Overlapping Linear Quadtrees : a Spatio-temporal Access Method. *Proceedings of the 6th International Symposium on Advances in Geographic Information Systems (ACM-GIS)*, pp.1-7, Washington, D.C., USA..

[33] XU X., HAN J., LU W. (1990) RT-tree - an Improved R-tree Index Structure for Spatiotemporal Databases. *Proceedings of the 4th International Symposium on Spatial Data Handling (SDH)*, pp.1040-1049.

A Revisited Database Projection Operator for Network Facilities in a GIS

C. Claramunt
The Nottingham Trent University
Department of Computing
Burton Street, Nottingham NG1 4BU, UK
Email: clac@doc.ntu.ac.uk
AND
M. Mainguenaud
Institut National des Telecommunications
9 rue Charles Fourier, F91011, France
Email: Michel.Mainguenaud@int-evry.fr

Keywords: GIS networks, data models

Edited by: Frederick E. Petry, Maria A. Cobb and Kevin B. Shaw
Received: December 15, 1998 **Revised:** May 14, 1999 **Accepted:** June 10, 1999

Within GIS, networks have been mainly represented at the geometrical level. This paper demonstrates that the definition of additional levels of abstraction extend the semantics of GIS networks. Networks are analysed and modelled at the conceptual and logical levels, independently of the underlying geometrical representation. The management of this two-levels representation leads to the development of a new interpretation of the database projection operator. This new operator is oriented to a semantic representation of a network. Network component properties are identified at the node and link component levels. We define four semantic constants: Global, $Subset_N$, $Subset_L$ and $Subset_{N,L}$. The semantic of alphanumeric properties are identified and their propagation with the application of network operators is characterised in function of the semantic constants. The closure of network manipulations on the database schema supports the definition of a data model that integrates the result of a network operator. This data model is built as a subset of the data models involved in network operators.

1 INTRODUCTION

A lot of efforts are under progress to elaborate innovative solutions for the representation and exploration of complex database applications. In the context of geographical databases, several spatial data models have been identified through the integration of geometrical and topological principles (e.g., David 1993, Guting 1989, 1993 and 1995, Haas 1991, Kolovson 1993, Larue 1993, Scholl 1989, Van Oosterom 1993). Similarly, many proposals have been defined to formalise spatial query languages within databases (e.g., Brossier-Wansek 1995, Egenhofer 1990, Frank 1982, Orenstein 1988). The common and accepted model representation of discrete entities in space integrates the underlying structural constraints that organise the geographical representation. For instance, cartographic models use topological relationships to structure entities in space (Peuquet 1984). Similarly, graph structures are introduced into database models to model networks (Mainguenaud 1995). Network structures are particularly useful to represent physical or influence relationships in space. Physical networks include the distribution of electricity, gas, water or telecommunication resources. Influence networks describe economical or social patterns in space. Network can be also used to represent spatial navigation processes in space and time, i.e., a movement, generally a human one, between several locations in space. Such processes are represented throughout cognitive representations of space that integrate complementary levels of abstraction (i.e., large and local scales according to (Kuiper, 1978). In this case, network nodes represent symbolic and discrete location in space, edges displacements between these places (Claramunt 1995).

If several spatial database models and query languages have been proposed to represent the properties of networks in space (Claramunt 1996, Guting 1989, Haas 1991), the definition of a data manipulation language that operates, organises and presents a set of network queries within their geographical context is still an important research challenge to be addressed. Surprisingly, the definition of operators that address the presentation of query results has been hardly investigated within classic databases. Database query

languages are mainly based on a logic of predicates that restrict a query result to a set of tuples or objects. The projection operator restricts a query result to some of the relation attributes. To extend the semantics of the projection operator, aggregate functions have been proposed (e.g., sum, average, maximum, minimum, count). These functions can be integrated within the projection operator in order to extend the semantics delivered by the query result. In a GIS context, a query result combines network, spatial and alphanumerical properties. A first extension proposed to handle spatial data in query languages is the introduction of spatial predicates (e.g., in the where clause of an extended SQL-like query language). However, the semantics of these languages is not adapted to the complexity of geographical applications in which query operations are often oriented toward the spatial and logical manipulation of entities. The introduction of spatial operators, (e.g., in the select clause), improves the benefit of spatial operators as a new spatial semantics may be derived from their application (Guting 1989 and 1993, Haas 1991, Larue 1993, Orenstein 1988).

For alphanumeric attributes, current spatial query languages assume that the semantics of the operand components is still valid for the alphanumeric attributes delivered by a projection operator. This assumption is correct for queries based on the application of predicates in which the resulting semantics is not changed (i.e., no new attribute nor spatial representation are created). However the introduction of spatial operators within database queries change the resulting semantics as the spatial component of the query result may be derived (e.g., application of a spatial intersection operator). The propagation of alphanumeric properties within spatial operators have been studied in a previous work (Mainguenaud 1994). A classification defines the semantics of spatial entities at the topological level (i.e., an alphanumeric attribute is valid at either the interior, boundary or global spatial representation) and at a partial or global spatial representation (i.e., an alphanumeric attribute is valid, or not, for a subset of the spatial representation of this entity). The entity level defines the possible overlap, or not, in space for two instances of a same entity.

Accordingly, a data definition language must identify (1) an appropriate semantics of alphanumeric attributes and (2) rules for the propagation of these attributes within network operators. We can make a distinction between network operators that generate or not a new semantics (i.e., creation of new entities or not). That is a mandatory requirement to avoid inconsistencies or false interpretations in the analysis of network query results. This paper proposes an analysis of the semantics of GIS network operators and the definition of propagation rules that monitor the propagation of network component attributes. This analysis will support the redefinition of a projection operator suited for the presentation of query results that involve the manipulation of spatial data networks. From the database point of view, this application context represents a case study which may act as a reference and can be generalised with some minor adaptations to any network representations. From a GIS point of view, the context is of particular interest for applications oriented toward the representation of physical and semantic networks. The remainder of this paper is organised as follows. Section II presents the data model we use to explain the manipulations of networks. Section III discusses network operators and their semantics. Section IV develops the identification and application of a revisited projection operator. Section V draws the conclusion.

2 DATA MODEL SUPPORT

The definition of a network data model implies the integration of logical, geometrical and alphanumeric properties into a user-defined network representation. Therefore, modelling a network application requires the identification of the following elements:

- *logical connections* between network entities. The concept of graph is well adapted to model such connections.

- *geometrical properties* of the network and its component entities. In the context of our model, the geometrical level is defined by the concept of Abstract Data Type that allows an independent representation of the physical data model (Stemple 1986).

- alphanumerical properties of the network described within a so-called *data structural model.* We use a complex object data model for the description of alphanumerical properties with three constructors: [ ] for tuple, { } for set and < > for enumerated type (similar models with a matching semantics could be used without loss of generality).

- logical, geometrical and alphanumeric levels represented by an homogeneous *user-defined network* model.

We model a network with the concept of graph. A graph is a pair (V, E) where V is a set of vertices and E is a sub-set of V x V. An incident function maps an edge to a couple of vertices. This basic definition is completed by labelling functions. Two labelling functions allow the introduction of alphanumeric properties for vertices and edges. A node models a vertex with its labelling function. A link models an edge

G (V, E, ν, ε, Ψ)
V is a set of vertices, ν is a labelling function for vertices
E is a set of edges, ε is a labelling function for edges
Ψ is an incident function E -> V x V

Figure 1: Definition of a graph

with its labelling function. Let us consider a graph as defined in (Cruz 1987):

In order to simplify the notation, a network built on a graph G (V, E, ν, ε, Ψ) with labelled vertices and labelled edges is denoted N (N, L). For example a node is used to model an airport, a link to model a connection between two airports. Networks can be defined from one to many level of abstractions depending on the application semantic. The integration of networks defined at complementary scales can be realised by the application of logical operators that allow to compose and decompose network subsets (Claramunt 1996, Frank 1982, Mainguenaud 1996). For the purposes of this research, we restrict the domain of study to a set of networks defined at a same abstraction level (the same principles can be easily extended to multi-level representations).

3 Logical and geometrical levels

A network is structured through two complementary logical and geometrical levels which are defined as follows:

- **logical level**: A network is logically defined by a set of nodes and by a set of links. This level is independent of the underlying geographical representation. The logical level supports the application of network operators (e.g., path, paths)
- **geometrical level**: This level describes the geometrical properties of the network entities. It permits the application of metric operators (e.g., distance, area). Spatial constraints and operations are generally derived and defined from the geometrical level. They allow to manipulate the geometrical and topological properties of network entities (e.g., adjacency, inclusion).

This two-levels structure leads to the manipulation of spatial entities at two distinct levels (1) the logical level and the application of logical operators (e.g., logical connection) (2) the geometrical level and the application of spatial operators (e.g., spatial intersection). For instance, two distinct flight routes that intersect in space may be not logically connected. The distinction between these two structural levels are generally not represented and integrated within current database models and applications. However, a distinct representation of the geometrical and logical levels is more adapted to a finer representation of the semantics of spatial networks, particularly for applications which are oriented to the modelling of logical connections (e.g., maritime or aerial navigation).

A spatial network includes two orthogonal logical and geometrical levels. The logical level is mandatory by definition. The geometrical level is optional. We propose a definition of a network in function of the completeness of its geometrical level, with the definition provided in (Guptill 1995), that is, depending on the availability or not of the spatial representation instances. Let us define two Boolean functions to handle the existence or not of a spatial representation for a node and for a link, respectively:

IsSpatialNode: Node_type -> Boolean

IsSpatialLink: Link_type -> Boolean

Definition: A network is spatially complete whenever a spatial representation is available for all nodes and all links.

N (N, L) build on a graph G (V, E, ν, ε, Ψ) is spatially complete <=>

$\forall$ ni $\in$ N, $\forall$ $l_i \in$ L / IsSpatialNode (n_i) = True $\wedge$ IsSpatialLink (l_i) = True

Definition: A network is spatially incomplete whenever at least one spatial representation is not available for a node or a link and at least a spatial representation is available.

N (N, L) build on a graph G (V, E, ν, ε, Ψ) is spatially incomplete <=>

($\exists$ $n_i \in$ N/ IsSpatialNode (n_i) = False $\vee$ $\exists$ $l_i \in$ L / IsSpatialLink (l_i) = False) $\wedge$

($\exists$ $n_i \in$ N/ IsSpatialNode (n_i) = True $\vee$ $\exists$ $l_i \in$ L / IsSpatialLink (l_i) = True)

Definition: A network is a-spatial whenever no spatial representation is available for all nodes and all links.

N (N, L) build on a graph G (V, E, ν, ε, Ψ) is a-spatial <=>

$\forall$ $n_i \in$ N, $\forall$ $l_i \in$ L / IsSpatialNode (n_i) = False $\wedge$ IsSpatialLink (l_i) = False

From an application point of view, spatially complete networks allow to represent networks which are described by a complete geometry in space at both the node and link levels. On the other hand, spatially incomplete networks are oriented toward the representation of schematic or immaterial networks in space (i.e., spatial networks that describes influence or gravitational forces with no link geometry) or networks that are constituted by some incomplete data (i.e., some spatial representations are missing).

4 User defined level

Network applications require to handle the logical representation of a network (e.g., in order to define a query that delivers the flight connections or the paths between two airports), the alphanumeric properties (e.g., a query that delivers Air France flights) and the visualisation of a geometrical representation (e.g., a query that displays a map that presents the flight connections between two airports). The representation of a network must handle these complementary model levels and their semantic relationships in order to process and combine the semantics of these queries.

Let us consider an abstract data type modelling an object identifier, named Oid_type (e.g., a name, a number). Using the notations of complex objects, a graph and a network are defined with Abstract Data Types as follows:

Vertex_type = [Oid: Oid_type]

Vertices_type = { Vertex_type }

Node_type = [Oid: Oid_type, Representation: Spatial_Representation_type]

Nodes_type = { Node_type }

Edge_type = [Oid: Oid_type, Origin: Vertex_type, Destination: Vertex_type]

Edges_type = { Edge_type }

Link_type = [Oid: Oid_type, Origin: Node_type, Destination: Node_type,

Representation: Spatial_Representation_type]

Links_type = { Link_type }

Graph_type = [Vertices: Vertices_type, Edges: Edges_type]

Graphs_type = { Graph_type }

Network_type = [Nodes: Nodes_type, Links: Links_type]

Networks_type = { Network_type}

Node_type, Link_type and Network_type can be respectively considered as sub-types of Vertex_type, Edge_type and Graph_type. To simplify the presentation, the set of integrity constraints defined on such a schema is not presented.

Using the previous network definitions, we define an example database that describes an international flight network between some European airports: London, Brussels, Amsterdam and Paris (Figure 1). To explain by example the different logical and geometrical configurations of a network, let us consider that this database may be a-spatial (Figure 1A), spatially complete (Figure 1B) or spatially incomplete (Figure 1C). Using the notations of complex objects, the network and its components are defined with Abstract Data Types as follows (note that the logical representation of this network and its components is still valid if the spatial attributes Representation are not included in the network and component types):

Airport_type = [Oid: Oid_type, Representation: Spatial_Representation_type,

Category: string]

Flight_type = [Oid: Oid_type, Representation: Spatial_Representation_type,

Origin: Airport_type, Destination: Airport_type,

Duration: float, Price: float]

EU-Network_type = [Airports : {Airport_type},

Flights : {Flight_type}]

Airport_type represents the nodes of the netowrks. Flight_type the links of the network. Alphanumerical and spatial attributes for Airport_type (e.g., Category) and for Flight_type (e.g., Duration) correspond to the introduction of the functions n and e in the definition of a graph, respectively. They constitute a network modelled by the EU-Network_type. In particular, the a-spatial network may represent the point of view of a user which is only interested in the logical connections of the flights. The Paris - Brussels and Paris - Amsterdam links overlap in the geometrical level although they are logically distinct (Figure 1B/1A). Symbolic representations in Figure 1C such as Paris or the link Paris-Amsterdam have no particular spatial location. A link such as the London-Brussels one has a spatial location which is only relevant for its origin and destination (e.g., London and Brussels).

5 NETWORK OPERATORS

Several database query languages have been proposed for the manipulation of networks (e.g., Brossier-Wansek 1995, Cruz 1987, Guting 1989)). We analyse query operators that manipulate and derive the semantics of spatial networks. The definition of a minimal set of network operators is far away the scope of this paper. We propose a classification of network operators based on a distinction of the query results in order to identify different semantic levels in the re-analysis of the projection operator. That leads to a distinction between operators that manipulate networks and network components (i.e., nodes and/or links). We analyse these network operators from a *structural*, *topological*, *semantic* or *set-oriented* points of view:

- Structural operators correspond to the manipulation of graph components - network to node and link or node and link to network (i.e., decomposition or composition operations) -. They are applied on networks or network components.

- Topological operators manipulate graph properties (e.g., paths operator), they manipulate networks or a network and some of its components and deliver a network (or a set of networks).

- Semantic operators manipulate the semantics of alphanumerical attributes for networks or network

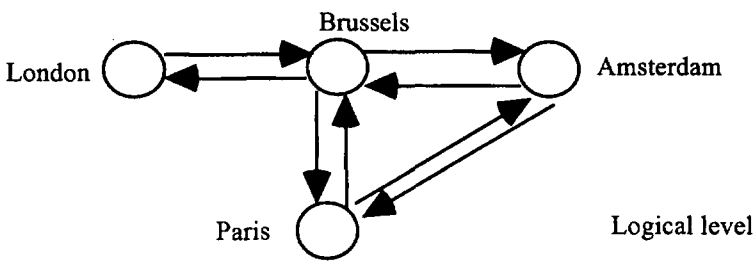


Figure 1A
Logical level of a graph

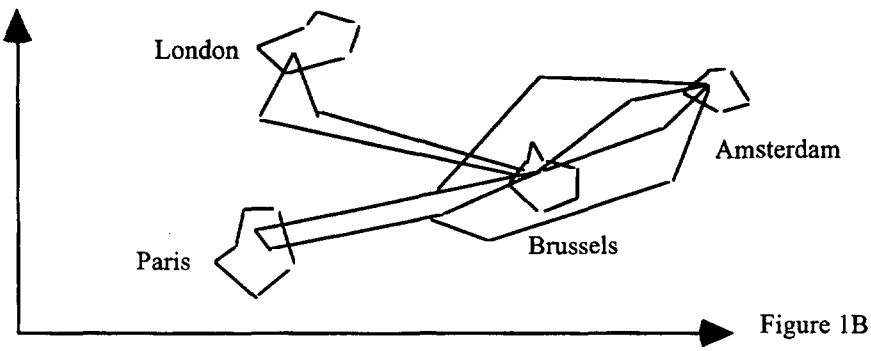


Figure 1B
Geometrical (spatially complete) level of a graph

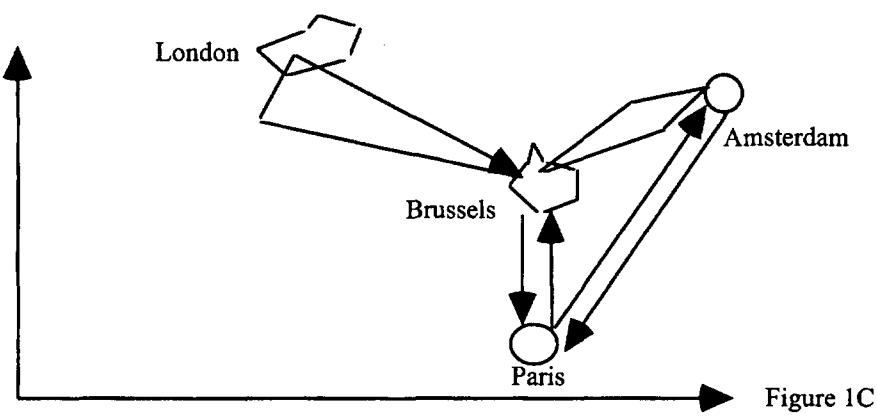


Figure 1C
Geometrical (spatially incomplete) level of a graph

Figure 2: Example database

components. They correspond to classic database operators.

- Set-oriented operators are conventional set theory operators (e.g., intersection). They are applied on networks or network components.

6 Network-oriented operators

The main objective of network-oriented operators is to provide one (or several) network(s) as a result. We hereafter detail the classification (structural, topological, semantic and set-oriented). The Build_In operator is an example of a structural network-oriented operator applied on a set of links and on a set of nodes to define a network, i.e., composition. The Build_In operator is the equivalent of the Insert operator in a conventional database. Nodes and Links may exist in the database without being regrouped in a network (similar to an association relationship in a Entity-Relationship model). The signature of the Build_In operator is as follows:

Build_In: Links_type x Nodes_type -> Network_type

The path operator is one of the most frequent topological network-oriented operator used in network applications. However, evaluating all possible paths in a network between an origin and a destination is not a relevant query in a GIS context (i.e., from a user point of view). A limitation must therefore be introduced (Mainguenaud 1996). Two levels of requirements can be defined:

- The first level operates at the link level (e.g., a query that delivers a path whose flight links are restricted to a specific company: Air France flights). Let us define an abstract data type Criteria_type to model such requirements.

- The second level operates at the path level (e.g., the global duration is less than 4 hours). Let us define an abstract data type Constraints_type to model such requirements.

A path is evaluated on an underlying network. Defining the most efficient algorithm to compute the transitive closure for the evaluation of a path operator is not relevant in our context. Let us define a generic path operator. In the context of a GIS query, several origins or destinations may be defined. A constrained definition is introduced on the path operator. The evaluation must be restricted to insure a realistic time of computation. The path evaluation can be required between a single origin and a single destination, between a single origin and a set of destinations or between a set of origins and a single destination. Conventional path operators, such as defined in Gral (Guting 1989) or Starburst (Haas 1991), have the following signature:

Path: Node_type x Node_type x Constraints_type x Criteria_type x Network_type ->
Graph_type

the result is a unique path (e.g., shortest path)

In order to provide a realistic solution for GIS applications, a relevant signature is:

Paths: Nodes_type x Nodes_type x Constraints_type x Criteria_type x Network_type ->
Graphs_type

the result is a set of possible paths

The result of the second signature is a set of graphs. Each graph from this set is a path without a cycle (an origin, a destination and the set of links between the origin and the destination). The application of a constraint such as "the path duration is less than four hours" may be verified but the real duration is not provided. However, we argue that a valuable decision must be taken depending on several selection criteria. A shortest path may not be the most convenient criteria as additional thematic parameters may be applied. Such a query is difficult to formulate from a computational point of view. Therefore, a human interaction with the query result is highly recommendable. In the case of our example database, a query delivering a set of paths between Paris and London is the most appropriate solution as a relevant choice implies a human decision process that combines additional criteria (e.g., company preferences, timetables). However, this objective implies the introduction of additional information that represent the network semantic in order to support such an interactive decision process. Then we introduce a labelling function in the definition of a network as follows:

N (N, L, α)

where α is a labelling function that complements the network semantic.

Using the function α, the limitation of the context in which a network is embedded (Figure 1) disappears. Our example database can now be extended. Alphanumerical and spatial attributes for EU_Network_type (e.g., Condition) correspond to the introduction of the function a in the definition of a network.

EU-Network_type = [Oid: Oid_type,
Airports : {Airport_type}, Flights : {Flight_type}
Condition: string, Cumulated_Length: float,
Companies: (string), Length_Air_Traffic_Corridor: float,
Air_Traffic_Noise: float,
Context : Spatial_Representation_type]

Accordingly the signature of the path operator is as follows:

Paths: Nodes_type x Nodes_type x Constraints_type x Criteria_type x Network_type ->
Networks_type

The Largest_Common_Subgraph operator is a second example of a topological network-oriented operator. This operator is applied on two networks, and returns a network. Its signature is as follows:

Largest_Common_Subgraph: Network_type x Network_type -> Network_type

Such operators are often NP-complete. In the case of a GIS query, a network mostly represents a path. A path (without cycle in our case) is a Directed Acyclic Graph (DAG). In a path a node has at most one predecessor and one successor. This topology allows the application of operators that are in theory NP-complete but with a realistic complexity in the context of our network model (even with a large graph).

The Networks_Selection operator is an example of a semantic network-oriented operator applied on a set of networks. This operator manipulates the semantics of a set of networks verifying some selection criteria. It is similar to the relational selection operator. Its signature is as follows:

Networks_Selection: Networks_type x Criteria_type -> Networks_type

The Networks_Intersection operator is an example of a set-oriented network-oriented operator applied on two sets of networks. The resulting networks belong to the two sets of networks from the left part of the signature (e.g., same Oid). Its signature is as follows:

Networks_Intersection: Networks_type x Networks_type -> Networks_type

Table 2 summarises relevant configurations for network-oriented operators with an operator example for each identified class (structural, topological, semantic and set-oriented). The left part of a signature is presented as a regular expression on the different types. This table represents the possible operator configurations. No other left part of the signature can be involved since the pair (Links_type, Network_type) implies the definition of Nodes_type to validate Links_type, and therefore corresponds to (Network_type)+. By definition, a structural operator can only be defined from its network components in order to compose a network (i.e., applies for nodes and links). Structural operators correspond to the manipulation of graph components from a level of abstraction to another one. In the context of our model, only nodes or links (i.e., composition: Build_In operator) can be involved in the left part of the signature to provide a network. This configuration delivers for example a network from a sequence of nodes and links. By definition, topological operators involve at least a network in the signature. No topological operator can be defined with a signature that does not involve at least one network. Semantic and set-oriented operators provide a set of networks and therefore can only be defined with (Networks_type)+ in their left part of the signature.

7 Node-oriented operators

The main objective of node-oriented operators is the extraction of nodes. They can be applied on a network or on one (or several) set(s) of nodes or links. The choice of relevant nodes is based on structural, semantic, or set-oriented purposes. The Nodes_Projection operator is an example of a structural node-oriented operator applied on a network, i.e., decomposition (e.g., a query that delivers the cities of a network). This operator transforms a network into its unique set of nodes. Its signature is as follows:

Nodes_Projection: Network_type -> Nodes_type

The Nodes_Projection operator is applied on a Network_type (i.e., a unique network). The Origins (resp. Destinations) operator is a second example of a structural node-oriented operator. It is applied on a set of links, i.e., decomposition. This operator manipulates the structural properties and returns a set of nodes verifying that resulting nodes are the origin (resp. destination) of a link (e.g., a query that delivers the airport which is the departure of a flight). Its signature is as follows:

Origins: Links_type -> Nodes_type

The Nodes_Selection operator is an example of a semantic node-oriented operator applied on a set of nodes. This operator manipulates the semantics of nodes and returns a set of nodes that verify some selection criteria from a set of nodes (e.g., a query that delivers a set of airports of a particular category). Its signature is as follows:

Nodes_Selection: Nodes_type x Criteria_type -> Nodes_type

The Nodes_Difference operator is an example of a set-oriented node-oriented operator. It is applied on two sets of nodes and delivers a set of nodes that verify that these nodes belong exclusively to the first set of nodes in the left part of the signature. Its signature is as follows:

Nodes_Difference: Nodes_type x Nodes_type -> Nodes_type

Table 3 summarises the relevant configurations for node-oriented operators with an operator example for each class (structural, semantic and set-oriented). This table represents the possible configurations. As a node can be derived from a network or a link, structural operators are defined from Network_type and Links_type. By definition no structural operator can be defined with Nodes_type in its signature. Topological operators are not relevant for node-oriented operators since a node is an atomic component of a network. Semantic and set-oriented operators provide a set of nodes and therefore can only be defined with Nodes_type in their left part of the signature.

Left part of the signature	structural	topological	semantic	set-oriented
(Nodes_type)+ (Links_type)+	Build_In			
(Nodes_type)+ Network_type		Paths		
(Network_type)+		Largest_Common_ Subgraph	Networks_ Selection	Networks_ Intersection

Table 1: Network-oriented operators

Left part of the signature	structural	semantic	set-oriented
Network_type	Nodes_Projection		
(Nodes_type)+		Nodes_Selection	Nodes_Difference
(Links_type)+	Origins		

Table 2: Node-oriented operators

8 Link-oriented operators

The main objective of link-oriented operators is the extraction of links from a network. They deliver a set of links. They can be applied on a network or on one (or several) set(s) of links or nodes. The choice of relevant links may be based on structural, semantic or set-oriented purposes. The Links_Projection operator is an example of a structural link-oriented operator applied on a network, i.e., decomposition. This operator transforms a network into a set of links (e.g., a query that delivers the links of a network). Its signature is as follows:

Links_Projection: Network_type -> Links_type

The Build_In_Link operator is a second example of a structural link-oriented operator applied on two sets of nodes, i.e., composition. This operator derives a set of links from two sets of nodes. The semantics of the Build_In_Link operator is similar to the Build_In operator but the level of interaction is the link instead of being the network. Its signature is as follows:

Build_In_Link: Nodes_type x Nodes_type -> Links_type.

The Links_Selection operator is an example of a semantic link-oriented operator applied on a set of links. This operator manipulates the semantics of a set of links verifying some selection criteria from a set of links (e.g., a query that delivers the flights that take less than two hours). Its signature is as follows:

Links_Selection: Links_type x Criteria_type -> Links_type

The Links_Intersection operator is an example of a set-oriented link-oriented operator applied on two sets of links. This operator delivers a set of links that belongs to the two sets of links of the left part of the signature. Its signature is as follows:

Links_Intersection: Links_type x Links_type -> Links_type

Table 4 summarises the relevant configurations for link-oriented operators with an operator example for each class (structural, semantic and set-oriented). As a link may be derived from a higher abstraction level, i.e., a network, or from a lower abstraction level, i.e., a set of nodes, structural operators are defined from Nodes_type or Network_type. By definition, no structural operator can be defined with Links_type in its

signature. Topological operators are not relevant for link-oriented operators since a link is an element of the Cartesian product of atomic components, i.e., nodes. Semantic and set-oriented operators are applied on sets of links and therefore can only be defined with Links_type in their left part of the signature.

9 Derived operators

The efficiency of a network query language can be improved by the definition of derived operators that combine the semantics of network, nodes and links operators. The signature of these operators may involve types such as Criteria_type, Constraints_type, Network(s)_type, Nodes_type or Links_type in the left part (i.e., operands) and types Nodes_type, Links_types or Network(s)_type in the right part (i.e., result). As an example, let us define the operator Starting_Places with the following signature:

Starting_Places: Network_type -> Nodes_type.

The Starting_Places operator returns the set of nodes which have no predecessor in a given network. This function does not increase the expressive power of the query language since it can be expressed by a composition of operators, on a network, Ne, using a functional notation by:

Nodes_Difference (Origins (Links_Projection (Ne)), Destinations (Links_Projection (Ne)))

Furthermore, it does not change the problematic of managing network operators as the right part of the signature is still based on the same concept: a node. However it provides a user-oriented and user-friendly operator in the context of network queries.

10 PROJECTION OPERATOR

11 Network semantic

A network is a composite entity (nodes and links) whose alphanumerical properties are constrained at the internal (within the network components) or at the external levels (e.g., by another network logically connected to one or many nodes of this network). These properties convey some analogies with the spatial domain whereas the alphanumeric properties of entities may be constrained either at the internal (e.g., interior, boundary) or external levels (e.g., merge of independent spatial maps that share some common entities).

Within network representations, the semantics of alphanumeric properties have to be analysed with respect to their node and link components. This leads to separate attributes defined (and restricted to) at the composite (i.e., network or link levels) and component levels (i.e., node, link, node and link levels). From the following three sorts, Network, Node and Link, we define the notion of derived attribute and inherited attribute. Network operators allow transformations between these three sorts: Network to Node or Link, Link to Node, and Link and Node to Network. The data model associated with the result of these operators is built from the original data model (i.e., functions a, n, e) and attributes that can be provided by the operand(s). An attribute is said to be derived whenever its definition is provided by the application of an aggregate function on operand(s) (i.e., creation of a new semantics). An attribute is said to be inherited whenever its definition is provided by the conventional projection operator or by a composition or decomposition operation (i.e., propagation of an existing semantics).

We introduce a set of semantic constants associated with network and network component attributes:

- A "Global" network attribute is an attribute whose semantics is restricted to the composite network as a whole, i.e., no inheritance of such an attribute is authorised at the node and link component levels from the network level. A "Global" network attribute is relevant for the representation of qualitative data (e.g., an attribute that represents the general condition of a network) as well as for a network attribute derived from the application of an aggregated function on nodes and links (e.g., a cumulated length attribute of a network). Similarly a "Global" link attribute is an attribute whose validity is relevant for the link as an element of the Cartesian product N x N, i.e., this attribute is not relevant for any node component (e.g., a duration attribute for a flight link). By default node attributes are classified as "Global".

- A "Subset$_{N,L}$" network attribute is an attribute whose semantic is also relevant at the node and link composite levels, i.e., inheritance of such an attribute is authorised at the node and link component levels from the network (e.g., an attribute that represents the list of companies that use a network).

- A "Subset$_N$" network attribute is an attribute whose semantic is also relevant at the node composite level, i.e., inheritance of such an attribute is authorised at the node component level from the network level (e.g., an air traffic noise attribute). A "Subset$_N$" link attribute is an attribute whose validity is relevant for a subset of the element of the Cartesian product N x N (i.e., based on origin and/or destination).

- A "Subset$_L$" network attribute is an attribute whose semantic is also relevant at the link composite level, i.e., inheritance of such an attribute

Left part of the signature	Structural	Semantic	Set-Oriented
Network_type	Links_Projection		
(Links_type)+		Links_Selection	Links_Intersection
Nodes_type (Nodes_type)+	Build_In_Link		

Table 3: Link-oriented operators

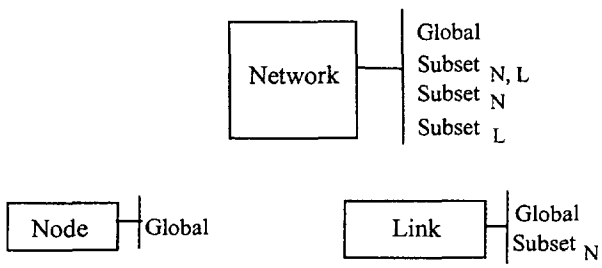


Figure 3: Semantic constants, static definitions

is authorised at the link component level from the network level (e.g., length of an air traffic corridor attribute).

A semantic constant "Subset$_L$" link attribute is not directly defined at the link level as it needs an origin and a destination to exist (i.e., equivalent to a global semantics). The only way to provide a "Subset$_L$" link attribute is an inheritance from the network level. No "Subset$_N$" semantic constant is defined for a node attribute from the static point of view as a node represents an atomic component for network operators. The only way to provide a "Subset$_N$" node attribute is an inheritance from the network or link levels. Figure 2 describes the static definition of semantic constants for network component attributes.

We introduce the semantic properties of the network components, from the static point of view, as follows:

Let N (N, L, α) represents a network with its labelling function

Let $A_{N,L}$ be the set of Network attributes provided by the labelling function α

Let A_L be the set of Link attributes

Let A_N be the set of Node attributes

Let a_i be an attribute of $A_{N,L}$ we qualify the characterisation of a_i as either a Global, Subset$_{N,L}$, Subset$_N$ or Subset$_L$ attribute depending on its semantics.

Let a_i be an attribute of A_L we qualify the characterisation of a_i as either a Global or Subset$_N$ attribute depending on its semantics.

Let a_i be an attribute of A_N, we qualify the characterisation of a_i as a Global attribute.

Let C be a function that provides the classification of an attribute

Sort_type = <Network, Node, Link>

C: string x Sort_type -> <Global, Subset$_{N,L}$, Subset$_L$, Subset$_N$>

12 Network operators

The left part of a signature of an operator is built with one to several sorts (Network, Link, Node). A basic right part of a query (i.e., a result) is defined as a sort. The data model associated with the result is built from the different data models involved in the left part of the signature. Semantic constants are associated with each attribute involved in the data model (i.e., in the left part of the signature). A transfer rule is applied from the left part to the right part of the signature in order to define the result data model (i.e., a set of data model attributes and their semantic constants).

The data model of the result is built with a subset of the respective data models involved in the left part and the derived attributes involved in the query. Conventional problems of database schema integration such as homonyms (i.e., two attributes with the same name but a different application semantics) are not relevant in the context of our model since the operators identified are closed on the same database schema. The unique source of conflict may be between two attributes with a same application semantic but different semantic constants. Therefore, we define an order for the semantic constants: from the lower level of abstraction to the higher: Global, Subset$_N$, Subset$_L$, Subset$_{N,L}$. Whenever a conflict arises, the priority is given to the higher level of abstraction (manipulations are closed on the database schema).

We classify network operators into two groups:

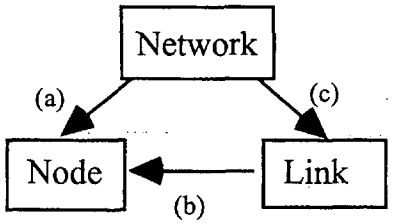


Figure 4: Decomposition

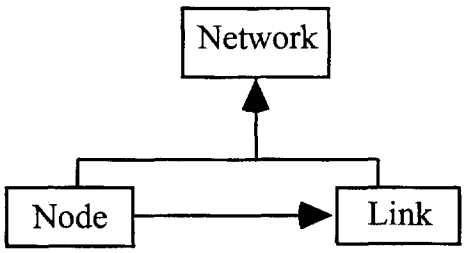


Figure 5: Composition

intra-component and heterogeneous operators. Intra-component operators involve the same sort in the left part and in the right parts of the signature. Intra-component operators regroup semantic and set-oriented operators. They do not provide a transfer of data models from the left part to the right part of the signature as the semantics of the data models is similar. The data model result is defined as the union as some attributes may appear in the data model of the sub-set of the operands, nevertheless the semantics is similar since the sorts are identical. Heterogeneous operators regroup structural and topological operators and therefore involve different sorts in the left part. The definition of transfer rules is relevant to define the data model of the result. Figure 3 and Figure 4 illustrate the different levels of abstraction involved in network operators. The decomposition reflects the different levels of abstraction from networks to nodes (Figure 3). The composition reflects the different levels of abstraction from node to link - to build a link - and node and link to network - to define a network - (Figure 4).

13 Heterogeneous operators

These operators provide a transfer since they manipulate network components rather than instances. Therefore transfer rules are defined from the conceptual point of view and are independent of the operator semantic (the approach is far different from spatial oriented operators as no creation of spatial representation is involved).

Node-oriented operators classified as structural deal with a decomposition and lead to a transfer from the network level or the link level. An attribute for a node is derived (arrow (a) Figure 3) from a network attribute classified as $Subset_{N,L}$ or $Subset_N$

$A_N = A_N \cup \{ a_i \in A_{N,L} / C(a_i, Network) = Subset_{N,L} \vee C(a_i, Network) = Subset_N\}$

An attribute for a node is derived (arrow (b) Figure 3) from a link attribute classified as $Subset_{N,L}$ or $Subset_N$

$A_N = A_N \cup \{ a_i \in A_L / C(a_i, Link) = Subset_{N,L} \vee C(a_i, Link) = Subset_N\}$

Link-oriented operators classified as structural deal with a decomposition and lead to a transfer from the network level. An attribute for a link is derived (arrow (c) Figure 3) from a network attribute classified as $Subset_{N,L}$, $Subset_N$ or $Subset_L$

$A_L = A_L \cup \{a_i \in A_{N,L} / C(a_i, Network) = Subset_{N,L} \vee$

$C(a_i, Network) = Subset\ N \vee C(a_i, Network) = Subset_L\}$

Network-oriented operators classified as structural deal with a composition and lead to a transfer from the left part to the right part of a signature. Functions ν, ε and α are involved. A signature defined with at least a Node(s)_type sort (resp. Link(s)_type) in the left part implies a transfer of attributes from the Node(s)_type (resp. Link(s)_type), i.e., the function ν (resp. ε), to the result. The extension of the data model - from the network on which the operator is applied or from the query definition process - is provided by aggregate constraints (i.e., function α). A network attribute is inherited from a node attribute classified as $Subset_N$. A network attribute is inherited from a link attribute classified as $Subset_{N,L}$, $Subset_N$ or $Subset_L$. A network attribute is derived from the query when it is required (e.g., calculated from an aggregate function) and is classified as Global.

$A_{N,L} = A_{N,L} \cup \{ a_i \in A_N / C(a_i, Node) = Subset_N\} \cup$

$\{a_i \in A_L / C(a_i, Link) = Subset_{N,L} \vee$

$C(a_i, Link) = Subset_N \vee C(a_i, Link) = Subset_L\}$

$\cup$

$\{ a_i / a_i$ is obtained by an aggregate function on N and/or L$\}$

Network-oriented operators classified as topological with a more complex signature than (Network_type)+ (e.g., path operator) can be considered as a special case of network-oriented operators providing a network from a network. The nodes or the links involved in the signature must already belong to the networks involved in the left part of the signature (e.g., the origin and the destination of a path operator). Table 5 summarises the transfer function for heterogeneous

operators which involve a decomposition (i.e., from a network to a link or from a link to a node).

The semantic constant associated with an attribute in the result data model may be different from its initial one. $Subset_{N,L}$ classification is concerned by this change as the link properties is not valid anymore when the result is a node sort. The new semantic constant is $Subset_N$ in this case. We extend the definition of derived attribute as follows: an attribute is said to be derived when its classification changes when its sort changes.

14 Dynamic of network operators

A sequence of network operators is realised at either a same (e.g., a sequence of Node_Difference operators applied on several node sets) or different abstraction level (e.g., a sequence of operations that successively decompose a network to links and these links to nodes - arrow (c) then (b) Figure 3). We qualify these sequences as horizontal and vertical, respectively.

The sequence of operators may be interpreted as the evaluation of a path in a graph G (V, E, ν, ε, Ψ) (Figure 5). Vertices are the different sorts involved in a signature. Edges model the transition from the left part of a signature to the right part. The node labelling function models the sorts involved in a signature. The edge labelling function models the classification of attributes involved in the transfer function.

V = {Network, Node, Link}

E = { (Network, Network), (Network, Link), (Network, Node),

(Link, Link), (Link, Node), (Node, Node) }

ν = sorts

ε: semantic constants concerned with the transfer function from the left part to the right part

15 APPLICATION

The semantic of derived results is provided by the successive propagation of semantic constants. Let us consider as an example, the Starting_Places operator. Its functional expression is:

Nodes_Difference (Origins (Links_Projection (Ne)), Destinations (Links_Projection (Ne)))

The successive propagation of attributes throughout the application of network operators (Table 5) within this Starting_Places composed operator is as follows:

Links_Projection: (Global, $Subset_N$, $Subset_L$, $Subset_{N,L}$) -> (-, $Subset_N$, $Subset_L$, $Subset_{N,L}$)

Origins: (-, $Subset_N$, $Subset_L$, $Subset_{N,L}$) -> (-, $Subset_N$, -, $Subset_N$)

Destinations (-, $Subset_N$, $Subset_L$, $Subset_{N,L}$) -> (-, $Subset_N$, -, $Subset_N$)

Nodes_Difference (-, $Subset_N$, -, -) -> (-, $Subset_N$, - , -)

Resulting node attributes are original network attributes which have $Subset_N$ or $Subset_{N,L}$ semantic constants ($Subset_{N,L}$ is transferred as a $Subset_N$ semantic constant by the Origin and Destination operators at the links level). Let us illustrate the semantic constants and the propagation rules for network operators with our example database. The application of the Starting_Places operator on a network requires the application of the Links_Projection operator. Table 6 presents the data model transfer.

Once the Links_Projection operator has been applied, the Origins and Destination operators are performed to transform a set of links into two sets of nodes. Table 7 presents the second data model transfer.

The Nodes_Difference operator is applied. The data model of a node is not changed as the Nodes_Difference operator belongs to the intra-component operators. Table 8 presents the final data model associated with the nodes. This data model is enriched with a subset of the Network_type (i.e., Companies, Air_Traffic_Noise and Context attributes).

16 CONCLUSION

The large diffusion of spatial database applications in scientific, planning and business domains implies the emergence of new requirements in terms of data representation and derivation. Particularly, current spatial data models and query language operations have to be extended in order to integrate a more complete semantic representation of complex domains. The integration and representation of graph structures within spatial data models is of particular interest for many application areas involved in the management of physical or immaterial networks.

This paper proposes the study of spatial network properties and their dynamic integration (i.e., using network operators) within the database projection operator. In order to represent the semantics of spatial network, we propose a classification that distinguishes the logical, spatially complete and spatially incomplete networks. The semantic of network components are studied at the network, link and node levels. The proposed model qualifies the semantics of alphanumeric attributes by a set of constants (i.e., Global, $Subset_N$ and $Subset_L$, $Subset_{N,L}$). These constants allows, or not, their propagation upward at a higher abstraction level (i.e., from nodes to link or network), downward at a lower abstraction level (i.e., from network to links / links to nodes / network to nodes) or at the same abstraction level. The propagation of these semantic properties are characterised and illustrated throughout the application of network operators that manip-

	Network -> Link	Link -> Node
Global	-	-
Subset $_N$	Subset $_N$	Subset $_N$
Subset $_L$	Subset $_L$	-
Subset $_{N,L}$	Subset $_{N,L}$	Subset $_N$

Table 4: Transfer data model for heterogeneous operators

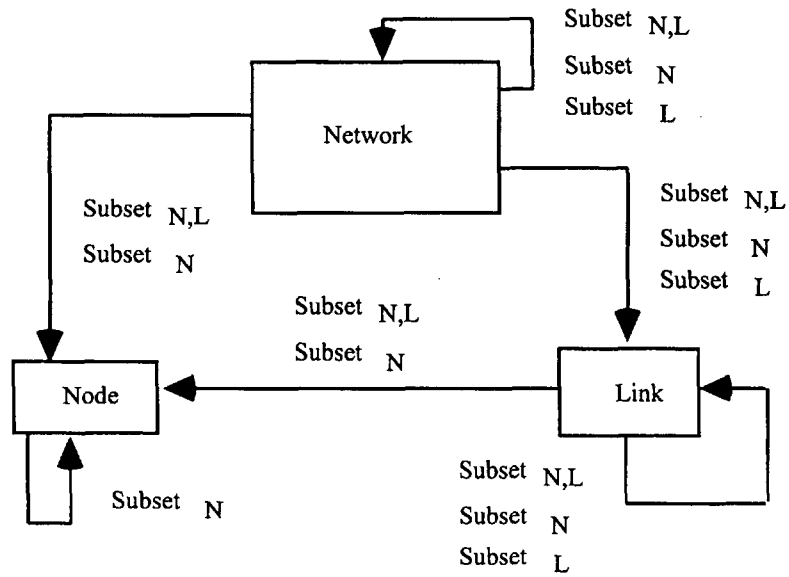


Figure 6: Dynamic of operators

ulate network components. Further work concerns the identification of an integrated semantic model that combines geometrical and network properties.

References

[1] Brossier-Wansek A. and Mainguenaud M. (1995) Manipulation of graphs with a visual query language: application to a Geographical Information System. In *Proceedings of the 3rd IFIP Conference on Visual Database Systems*, Spaccapietra, S. and Jain, R. Eds. Chapman and Hall, London, UK, pp. 227-246.

[2] Claramunt C. and Mainguenaud M. (1995) Dynamic and flexible vision of a spatial database. In *Proceedings of the DEXA95 International Workshop on Database and Expert System Applications*, London, UK, Revell, N. and Min Tjoa, A. Eds., Omnipress, pp. 483-492.

[3] Claramunt C. and Mainguenaud M. (1996) A spatial data model for navigation knowledge. In *Advances in GIS Research II*, Delft, The Netherlands, Kraak, M-J. and Molenaar, M. Eds., Taylor and Francis, pp. 345-357.

[4] Cruz I. F., Mendelzon A. O. and Wood P. T. (1987) A graphical query language supporting recursion. In *Proceedings of the ACM SIG Management Of Data -SIGMOD- Conference*, San-Francisco, USA, pp. pp. 323-330.

[5] David B., Raynal L., Schorter G. and Mansart V. (1993) GeO2: Why objects in a geographical DBMS. In *Advances in Spatial Databases*, Abel, D. J. and Ooi, B. C. Eds. Springer-Verlag, Singa-

Attributes	Semantic constant	Links_Projection
Oid	Global	-
Condition	Global	-
Cumulated_Length	Global	-
Companies	Subset $_{N,L}$	Subset $_{N,L}$
Length_Air_Traffic_Corridor	Subset $_{L}$	Subset $_{L}$
Air_Traffic_Noise	Subset $_{N}$	Subset $_{N}$
Context	Subset $_{N,L}$	Subset $_{N,L}$

Table 5: Data model transfer for the Links_projection operator

Attributes	Semantic constant	Origins/Destinations
Companies	Subset $_{N,L}$	Subset $_{N}$
Length_Air_Traffic_Corridor	Subset $_{L}$	-
Air_Traffic_Noise	Subset $_{N}$	Subset $_{N}$
Context	Subset $_{N,L}$	Subset $_{N}$

Table 6: Data model transfer for the Origins/Destination operators

pore, Lecture Notes in Computer Science, n. 692, pp. 264-276.

[6] Egenhofer M. (1990) Manipulating the graphical representation of query results in geographic information systems. In *Proceedings of the IEEE Workshop on Visual Languages*, Skokie, Illinois, pp. 119-124.

[7] Frank A. U. (1982) Mapquery - Database query language for retrieval of geometric data and its graphical representation, *ACM Computer Graphics*, 16 (3), pp. 199-207.

[8] Guptill S.C., Morrison J.L. (1995) El*ements of Spatial Data Quality*, Elsevier Science and the International Cartographic Association, Oxford, UK.

[9] Guting R. H. (1989) GRAL: An extensible relational database system for geometric applications. In *Proceedings of the 15th International Conference on Very Large Data Bases, VLDB*, Amsterdam, The Netherlands, pp. 33-44.

[10] Guting R. H. and Schneider M. (1993) Realms: A foundation for spatial data types in database systems. In *Advances in Spatial Databases*, Abel, D. and Ooi, B. C. eds, Springer-Verlag, Singapore, Lecture Notes in Computer Science, n. 692, pp. 14-35.

[11] Guting R. H., de Ridder T. and Schneider M. (1995) Implementation of the ROSE algebra: Efficient algorithms for Realm-based spatial data types. In *Advances in Spatial Databases*, Egenhofe, M. J. and Herring J. R. eds., Portland, USA, Springer-Verlag, pp. 216-239.

[12] Haas L. and Cody W. F. (1991) Exploiting extensible DBMS in integrated GIS. In *Proceedings of the 2nd International Symposium on Large Spatial Database*, Gunther O. and Schek H.-J. eds, Springer-Verlag, Zurich, Lecture Notes in Computer Science, n. 525, pp. 423-450.

[13] Kolovson P.C., Neimat M. and Potamianos S. (1993) Interoperability of spatial and attribute data managers: A case study. In *Advances in Spatial Databases*, Abel D. J. and Ooi B. C. eds. Springer-Verlag, Singapore, Lecture Notes in Computer Science, n. 692, pp. 239-263.

Attributes	Semantic constant
Oid	Global
Representation	Global
Category	Global
Companies	Subset $_N$
Air_Traffic_Noise	Subset $_N$
Context	Subset $_N$

Table 7: Final data model

[14] Kuiper B. (1978) Modelling spatial knowledge. *Cognitive Science*, 2, pp. 129-153.

[15] Larue T., Pastre D. and Viemont Y. (1993) Strong integration of spatial domains and operators in a relational database system. In *Advances in Spatial Databases*, Abel D. J. and Ooi B. C. eds., Springer-Verlag, Singapore, Lecture Notes in Computer Science, n. 692, pp. 53-71.

[16] Mainguenaud M. (1994) Consistency of geographical information system results. *Computers, Environment and Urban Systems*, Vol. 18,Pergamon Press, pp. 333-342, .

[17] Mainguenaud M. (1995) The modelling of the geographic information system network component. I*nternational Journal of Geographical Information Systems*, 9 (6), Taylor and Francis, pp. 575-593, .

[18] Mainguenaud M. (1996) Constraint-based queries in a geographical database for network facilities. *Computers, Environment and Urban Systems*, Pergamon Press, Vol. 20, pp. 139-151.

[19] Orenstein J. A. and Manola, F.A. (1988) PROBE spatial data modeling and query processing in an image database application. *IEEE Transactions on Software Engineering*, 14 (6), pp. 611-629.

[20] Peuquet, D. J. (1984) A conceptual framework and comparison of spatial data models. *Cartographica*, 21 (4), pp. 66-113.

[21] Scholl M. and Voisard A. (1989) Thematic map modelling. In *Proceedings of the 1st International Symposium on Large Spatial Databases*, Santa Barbara, USA, Springer-Verlag, Lectures Notes in Computer Science, n. 409, pp. 167-190.

[22] Stemple D., Sheard T. and Bunker R. (1986) Abstract data types in databases: Specification, Manipulation and Access. In *Proceedings of the 2nd Int. Conference on Data Engineering*, Los Angeles, pp. 590-597.

[23] Van Oosterom P. J. M. (1993) *Reactive Data Structures for Geographic Information Systems*, Oxford: Oxford University Press.

Analysis of Cache Sensitive Representation for Binary Space Partitioning Trees

Vlastimil Havran
Czech Technical University in Prague
Dept. of Computer Science and Engineering
Faculty of Electrical Engineering
Karlovo nám. 13, 12135 Prague
Czech Republic
Phone: +420 2 24357432, Fax: +420 2 298098
E-mail: havran@fel.cvut.cz

Keywords: binary tree, binary space partitioning, *kd*-tree, cache, spatial locality

Edited by: Frederick E. Petry, Maria A. Cobb and Kevin B. Shaw
Received: November 15, 1998 **Revised:** May 21, 1999 **Accepted:** June 10, 1999

Several variants of binary search trees were designed to solve various types of searching problems including geometrical ones as point location and range search queries. In complexity analysis we usually abstract from the real implementation and easily derive that the time complexity of traversal from the root of a balanced tree to any leaf is $O(\log m)$, where m is the number of leaves. In this paper we analyse a new method for memory mapping of a binary tree aiming to improve spatial locality of data represented by binary trees and thus performance of traversal algorithms applied on these data structures.

1 Motivation

The basic motivation for this research comes from binary search trees for computer graphics applications, where they are used to accelerate *ray–shooting*. To solve this problem *space subdivision* schemes are used, see [Watt92] for survey. One space subdivision is an orthogonal *binary space partitioning tree* (*BSP* tree or *BSPT* in the following text). It is often referred to as *kd*-tree in the context of computational geometry [Berg97]. It was initially developed to solve the hidden surface problem in computer graphics [Fuchs80].

A *BSPT* is a higher dimensional analogy to a binary search tree. The *BSPT* for a set S of objects in $\Re^n$ is a binary tree defined as follows. Each node v in *BSPT* represents a non–empty box (rectangular parallelepiped) R_v and set of objects S_v that intersects R_v. The box associated with the root node is the smallest box containing all the objects from S. Each interior node of *BSPT* is assigned cutting plane H_v that splits R_v into two boxes. Let H_v^+ be the positive halfspace and H_v^- the negative halfspace bounded by H_v. The boxes associated with the left and the right child of v are $R_v \cap H_v^+$ and $R_v \cap H_v^-$, respectively. The left subtree of v is a *BSPT* for a set of objects $S_v^- = \{s \cap H_v^- | s \in S_v\}$, the right subtree is defined similarly. The leaves of the *BSPT* are either occupied by the objects or vacant.

The *BSPT* is constructed hierarchically step by step until termination criteria given for leaf are reached. There are usually two termination criteria. First, maximum depth of *BSPT* is specified. Second, a node becomes a leaf if the number of objects associated with the node is smaller than a constant. The cutting plane H is for ease of computing search queries perpendicular to one of coordinate axes (*orthogonal cutting*). The example of *BSPT* in $\Re^2$ space is depicted in Fig. 1.

The most important operation carried out for any *BSPT* in any application is exhaustive traversal; for example the traversal in *depth–first–search* (DFS) order. It occurs when *BSPT* built for n-dimensional data is used for point location and range search queries [Samet90]. Several variants of *BSPT* are thus extensively used in GIS and other spatial database systems.

This decade I/O efficient algorithms and data structures for memory hierarchies have acquired noticeable research interest. The design of these data structures is driven by properties of external/internal memory hierarchy. Some achieved results are given for example in [Chiang95]. The example of design for sorting algorithm which takes into account the memory hierarchy is [Nyberg95].

In this paper we do not deal with external memory data structures, but with the internal memory hierarchy between the processor and the main memory including either on–chip cache or second–level cache. The main difference between this hierarchy and exter-

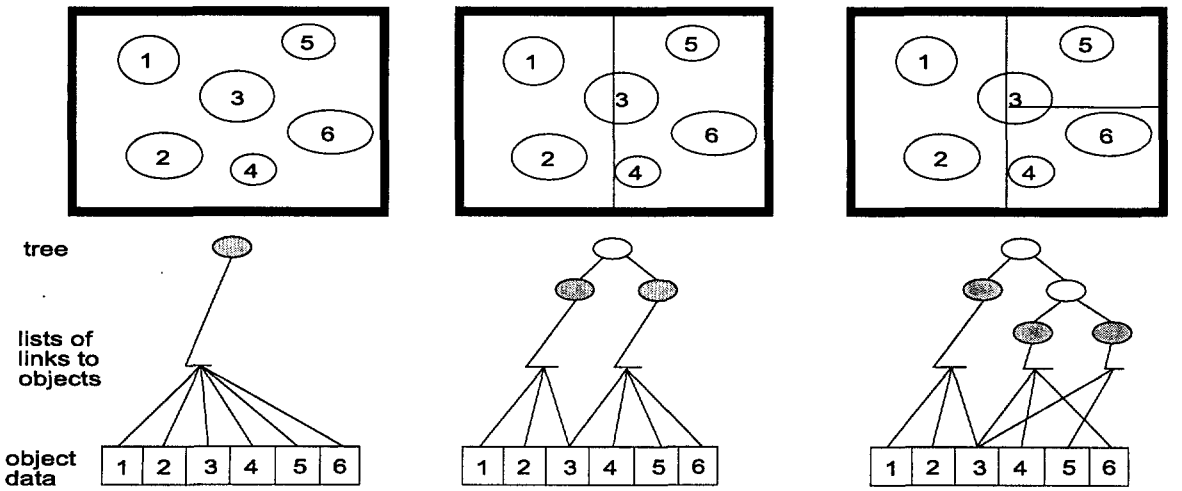


Figure 1: Binary space partitioning tree in $\Re^2$ space

nal memory one is the size of and the access time to one data block. Those for external memory hierarchy are much larger than the ones between the processor cache and the main memory.

Second, techniques for external memory data structures were developed mostly for one-dimensional search problems. For example, well known B–tree [Cormen90] cannot be used to decrease time complexity of the $BSPT$ traversal for $n > 1$, since the B–tree cannot represent n–dimensional data. In this paper we analyse novel methods to increase spatial locality of data in the cache and thus to decrease the execution time of any algorithm that uses $BSPT$. For the sake of simplicity we assume traversing $BSPT$ in DFS order from root to a leaf.

2 Preliminaries

In this section we recall several facts necessary to understand the concept of $BSPT$ nodes memory mapping. This includes memory allocation techniques and the structure of the memory hierarchy.

2.1 Memory Allocation

The key idea of this paper is mapping $BSPT$ nodes to addresses in the main memory. Allocation of dynamic variables is always provided by a *memory allocator*. Let us suppose the contiguous block of the unoccupied memory is assigned to the memory allocator at the beginning. This is used to assign the addresses within the block to the variables allocated so the variables do not overlap. We call this memory block a *memory pool*. Since the mapping is crucial for the main contribution of this paper, we discuss it in detail.

Common solution is to use a *general memory allocator*. Each $BSPT$ node is then represented as a specially allocated variable. Let S_I denote the size of memory to store information in a node. This is the position and the orientation of the splitting plane. Let S_P be the size of a pointer. Then the size required to represent one interior $BSPT$ node is $S_{IN} = S_I + 2.S_P$. Use of the general memory allocator requires to store two additional pointers with each allocated variable that are used later to free this variable from memory pool.

In this paper we also use another strategy to allocate the $BSPT$ node. We use a *special memory allocator* described for example in [Stroustrup97] to allocate variables of the same type and thus of the same *fixed size* S_V. We use $BSPT$ nodes of fixed size and dedicate them a special memory pool. During building up $BSPT$ the nodes are allocated from the memory pool as from an array in linear order.

2.2 Memory Hierarchy

The time complexity of a traversal algorithm using $BSPT$ is connected with the hardware used. Let us recall the organisation and the properties of the memory hierarchy. For analysis we suppose *Harvard* architecture with separated caches for instructions and data. Let T_{MM} denote *latency* of the main memory (time to read/write one data block).

The larger the memory and the smaller the access time, the higher the cost of the memory. The instruction/data latency of processors is smaller than T_{MM}. That is why a *cache* is placed between the memory and the processor. The cache is a memory of relatively small size with respect to the size of the main memory. The cache latency T_C is smaller than T_{MM}. This solution is economically advantageous; it uses *temporal* and *spatial locality* of data exposed by a typical program and the average access time can be significantly

reduced. Data between the cache and the main memory are transferred in blocks. The size of the block transferred is referred to as *cache line size* S_{CL}. Typical memory hierarchy is depicted in Fig. 2.

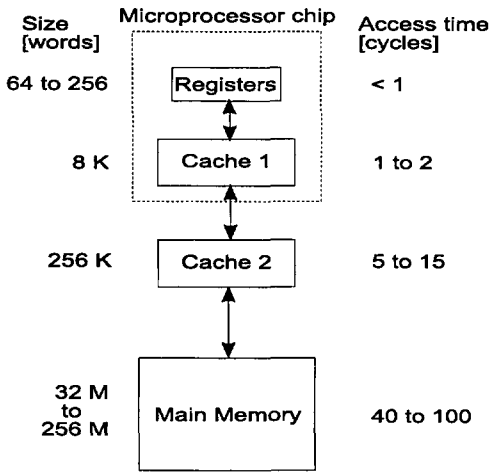


Figure 2: Typical memory hierarchy

In this paper we use for analysis only the one cache placed between the processor and the main memory. We denote the time consumed by operations in terms of cycles. Let T_W denote the average processing time on a $BSPT$ node to decide whether to follow its left or right descendant.

Typical values for today's superscalar processors and typical application are $T_{MM} = 55, T_C = 4, T_W = 5, S_{CL} = 128$ Bytes for MIPS R8000. These values are taken from [SGI96]. They are used further in the paper. Note that for a typical search algorithm on $BSPT$ holds $T_W \ll T_{MM}$.

3 *BSPT* Representations

As we already stated the $BSPT$ is actually represented by a binary tree. In general, a binary tree does not represent a valid instance of $BSPT$, since the splitting plane has to intersect the bounding box associated with the node. This is one of the reasons why the decomposition induced by $BSPT$ cannot be simply replaced by B-trees or some hashing scheme commonly used for one-dimensional search problems. The information stored in $BSPT$ node is the orientation and the position of splitting plane. The axis aligned bounding box is known explicitly for a root node only. The axis aligned bounding boxes associated with interior and leaf nodes are not stored explicitly in these nodes, but they can be derived by traversing down the tree.

Let us recall some terminology concerning binary trees. The *depth* of a node A in the tree is the number of nodes on the path from the root to the node A. The depth of root node is zero. We call a binary tree *complete* if all its leaves are positioned in the same depth d from the root node and thus the number of leaves is 2^d. An *incomplete* binary tree is the one that is not complete. Let h_C define *complete height* of a binary tree A as the maximum depth for which the binary tree constructed by the nodes of A is complete. The same definitions hold for $BSPT$.

The next four subsection gives details of $BSPT$ representations in the memory. This includes a usual method to represent $BSPT$ nodes using general memory allocator. We call this *random* representation. The less used method is *DFS order* representation. Finally, we describe two forms of a *subtree* representation that we designed to decrease further the average traversal time on $BSPT$ tree.

3.1 Random Representation

A common way to store the arbitrary $BSPT$ in the main memory is to represent each node as a special variable using general memory allocator. The representation is depicted in Fig. 3 (a).

This representation requires additional memory for pointers used by general memory allocator for each allocated variable, but it is the simplest technique to implement. The addresses of nodes in the main memory have no connection with their location in the $BSPT$. Assume that two additional pointers are needed to allocate the variable. Then the memory size M_S consumed by random representation to store N_{NO} nodes of $BSPT$ is:

$$M_S^{random} = N_{NO}.(4.S_P + S_I) \qquad (1)$$

3.2 Depth–First–Search (DFS) Representation

A DFS representation is implemented by using the allocator for fixed size variables described in subsection 2.2. In this representation the nodes are put subsequently in the memory pool in linear order, when $BSPT$ is built up in the DFS order, see Fig. 3 (b). The size of the memory consumed to represent N_{NO} nodes of $BSPT$ is:

$$M_S^{DFS} = N_{NO}.(2.S_P + S_I) \qquad (2)$$

Then $2.N_{NO}.S_P$ of memory taken by pointers to implement general memory allocator is saved in comparison with random representation.

3.3 Subtree Representation

The main goal of this paper is the analysis of $BSPT$ representation proposed originally in [Havran97] to reduce the time complexity of ray–shooting query per-

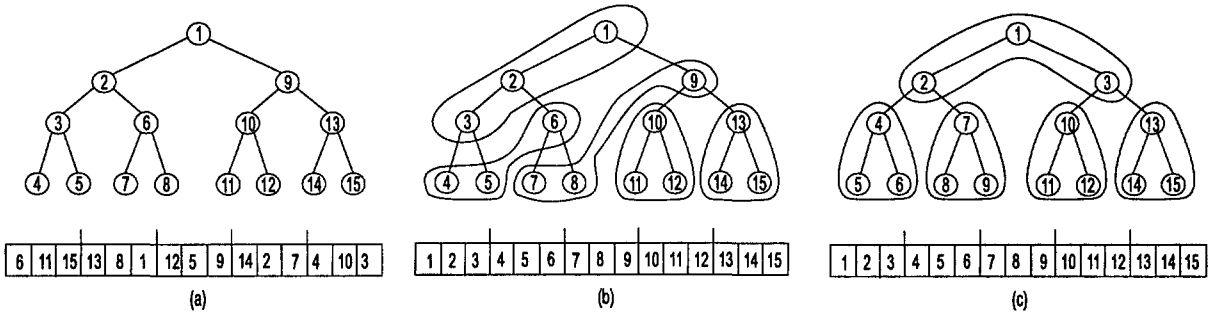


Figure 3: $BSPT$ representations (cache line size $S_{CL} = 3.size$(node of BSPT)) (a) Random (b) DFS (c) Subtree

formed on $BSPT$. Let us describe the representation in detail.

We also use allocator for fixed size variables, but the size of one allocated variable is equal to cache line size S_{SL}. The variable is subsequently occupied by the nodes organised into subtree. The whole $BSPT$ is then decomposed to subtrees, see Fig. 3 (c). Once the subtree is read to the cache, the access time to its nodes is equal to cache latency T_C. The subtree need not be complete. We distinguish between two subtree representations, see Fig. 4.

An *ordinary subtree* has all nodes of the same size, with two pointers to its descendants, regardless of whether the descendant lies in the subtree or not.

A *compact subtree* has no pointers among the nodes inside the subtree because their addressing is provided explicitly by a traversal program. The pointers are needed only to point between the subtrees. The leaves in an incomplete subtree have to be marked in a special variable stored in each subtree (one bit for each node).

The size of the memory described by both subtree representations is given in the next section.

4 Time Complexity and Cache Hit Ratio Analysis

In this section we analyse the time complexity of a DFS order traversal for all the $BSPT$ representations described in previous section. The theoretical analysis assumes that the $BSPT$ nodes data stored in the main memory are not loaded into the cache, i.e., cache hit ratio $C_{HR} = 0.0$. Further, we suppose that the $BSPT$ is complete and its height is h_l. An incomplete $BSPT$ requires to compute its average depth $\bar{h}_l$ and substitute it for h_l.

These simplifications enable us to express the average traversal time T_A on $BSPT$ in DFS order from its root to a leaf. We compute the T_A for an example of $BSPT$ of height $h_l = 23$. Further, we suppose random traversal and the probability that we turn left in a node is equal to $p_L = 0.5$.

If some data are already located in the cache ($C_{HR} > 0.0$), the analysis can be very difficult or even infeasible. The interested reader can follow e.g. [Arnold90]. Since the cache has asynchronous behaviour, we analysed the case by means of simulation.

4.1 Random Representation

We suppose $C_{HR} = 0.0$ during the whole traversal, i.e., the processing time of each $BSPT$ node is $T_{MM} + T_W$. As we know that the number of nodes along the traversal path from root to the leaf is $h_l + 1$, we can express the average traversal time T_A as follows:

$$T_A = (h_l + 1).(T_{MM} + T_W) \tag{3}$$

For values given above ($T_{MM} = 55$, $T_W = 5$, $h_l = 23$) we obtain $T_A = 1392.0$ cycles.

4.2 DFS Representation

The DFS representation increases the cache hit ratio by involuntary reading the descendant nodes for next traversal step(s) if traversal continues to the left descendant(s) of the current node. Assuming the size of the $BSPT$ node is $S_{IN} = S_I + 2.S_P$, we derive the average traversal time T_A as follows:

$$\begin{aligned} T_A &= (h_l + 1).[p_L.T_{MM}.\frac{S_{IN}}{S_{CL}} + T_W \\ &\quad + T_C.(1 - \frac{S_{IN}}{S_{CL}}) + (1 - p_L).T_{MM}] \end{aligned} \tag{4}$$

For $S_{IN} = 4 + 2.4 = 12$, $S_{CL} = 128$, and $p_L = 0.5$ we obtain $T_A = 859.1$ cycles.

4.3 Ordinary Subtree Representation

Assume that S_{CL} and S_{IN} are given. Let S_{ST} be the size of the memory needed for each subtree used to represent subtree type identification. We express the size of the memory taken by a complete ordinary subtree of the height h:

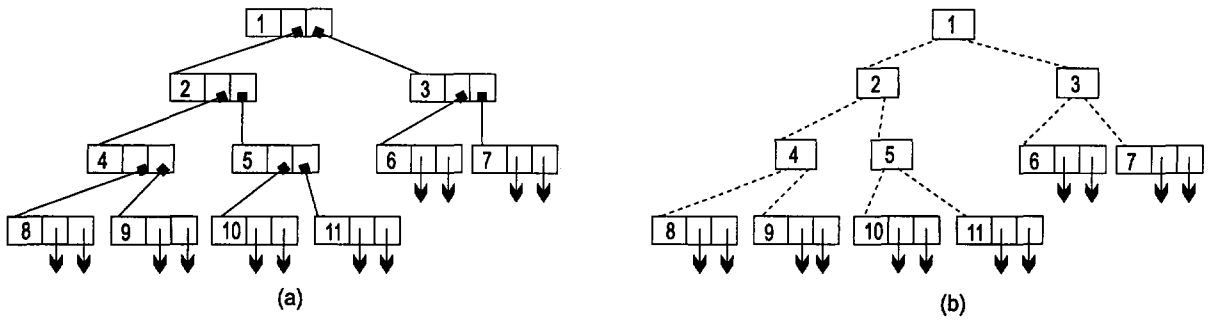


Figure 4: Subtree representation: (a) Ordinary (b) Compact

$$M(h) = (2^{h+1}-1).S_{IN} + S_{ST} \quad (5)$$
$$M(h) \leq S_{CL}$$

From Eq. 5 we derive the complete height of the ordinary subtree h_C:

$$h_C = \lfloor -1 + \log_2(\frac{S_{CL} - S_{ST}}{S_{IN}} + 1) \rfloor \quad (6)$$

The number of nodes in the incomplete ordinary subtree in the depth $d = h_C + 1$ is then:

$$N_{ODK} = \lfloor \frac{S_{CL} - (2^{h_C+1}-1).S_{IN} - S_{ST}}{S_{IN}} \rfloor \quad (7)$$

The average height of the subtree $h_A \geq h_C$ for $N_{ODK} > 0$ is computed as follows:

$$h_A = -1 + \log_2(2^{h_C+1} + N_{ODK}) \quad (8)$$

Finally, the average traversal time for the whole $BSPT$ of height h_l is:

$$T_A = (h_l + 1).(T_W + \frac{T_{MM} + T_C.h_A}{h_A + 1}) \quad (9)$$

The subtrees are placed in the main memory so they are aligned with the cache lines when read to the cache. Each subtree corresponds to one cache line. The size of the unused memory in the cache line is then:

$$M^{OSR}_{unused} = S_{CL} - (2^{h_C+1} - 1 + N_{ODK}).S_{IN} - S_{ST} \quad (10)$$

For $S_{IN} = 12, S_{ST} = 4$, we get $h_C = 2$, $N_{ODK} = 3$, $h_A = 2.46$, $M^{OSR}_{unused} = 4$, and the average traversal time $T_A = 555.9$ cycles.

4.4 Compact Subtree Representation

Let S_I be the size of the memory to represent the information in the $BSPT$ node, S_P the memory taken by one pointer. The size of the memory consumed by a complete subtree of the height h is expressed as follows:

$$M(h) = (2^{h+1}-1).S_I + 2^{h+1}.S_P + S_{ST} \quad (11)$$
$$M(h) \leq S_{CL}$$

The complete height h_C of subtree is from Eq. 11 derived similarly to Eq. 6 as follows:

$$h_C = -1 + \lfloor \frac{S_{CL} + S_I - S_{ST}}{S_I + S_P} \rfloor \quad (12)$$

In the same way as for the ordinary subtree representation we derive the number of nodes N_{ODK} located in the depth $d = h_C + 1$ in the subtree:

$$N_{ODK} = \lfloor \frac{S_{CL} - 2^{h_C+1}.(S_I + S_P) + S_I - S_{ST}}{S_I + S_P} \rfloor \quad (13)$$

The unused memory for one subtree in the cache line can be derived similarly as for ordinary subtree:

$$M^{CSR}_{unused} = S_{CL} - (2^{h_C+1} - 1 + N_{ODK}).S_N - 2.S_P.(N_{ODK} + 2^{h_C} - N_{ODK}/2) - S_{ST} \quad (14)$$

The average height of the subtree h_A and the average traversal time T_A are computed using Eq. 8 and Eq. 9. Given $S_P = 4$, $S_I = 4$, and $S_{ST} = 4$ we compute $h_C = 3$, $N_{ODK} = 0$, $h_A = 3.0$, $M^{CSR}_{unused} = 0$, and $T_A = 510.0$ cycles.

The h_C, N_{ODK}, h_A as the function of the cache line size for ordinary and compact subtree representations and T_A for all $BSPT$ representations are depicted in Fig 5.

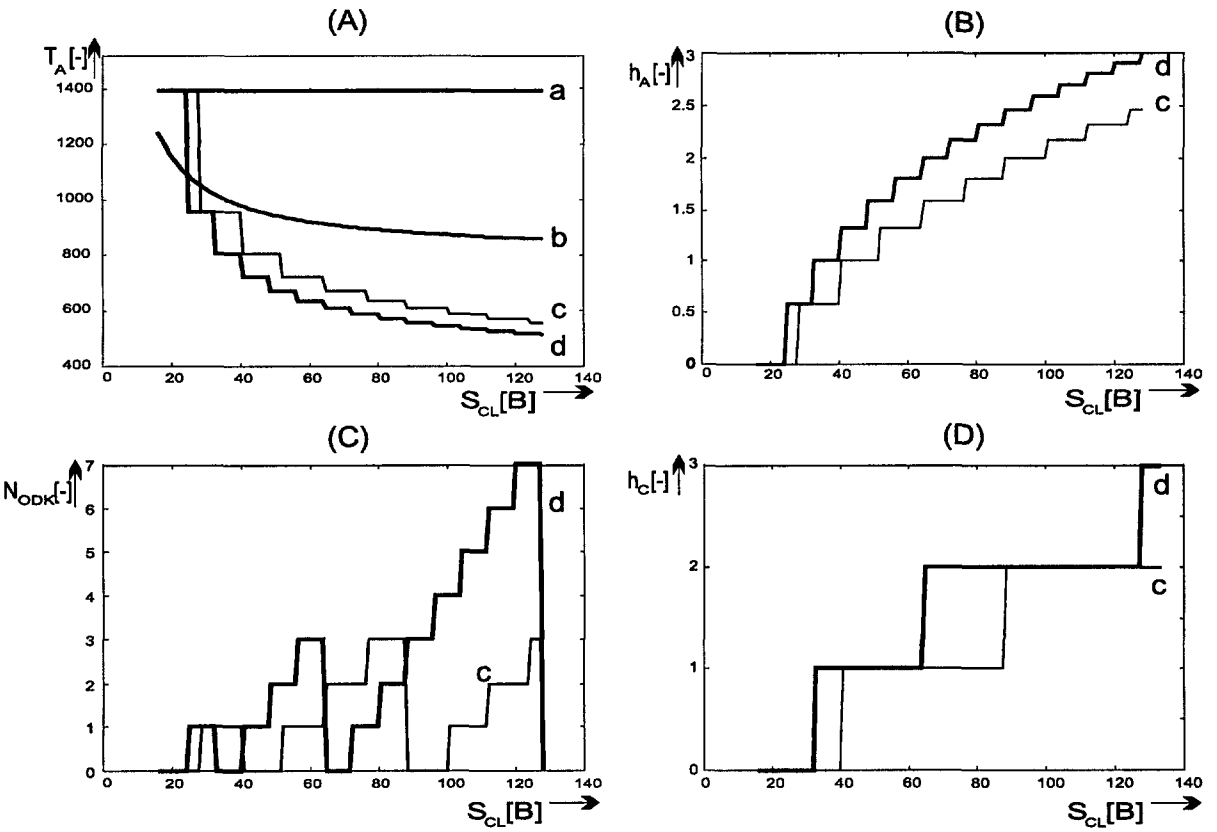


Figure 5: The analysis: (A) Average traversal time $T_A(S_{CL})$ for all *BSPT* representations, (B) $h_A(S_{CL})$, (C) $N_{ODK}(CL)$, (D) $h_C(CL)$ for subtree representations; Representation (a) Random (b) DFS, (c) Ordinary subtree, (d) Compact subtree

5 Simulation Results

We implemented a special program simulating the data transfer in a typical memory hierarchy for the DFS traversal on a complete *BSPT*. The simulation was carried out for the same memory hierarchy and *BSPT* properties as in previous section: $T_{MM} = 53$, $T_C = 4$, $T_W = 5$, $h_l = 23$, $S_P = 4$ Bytes, $S_I = 4$ Bytes, $S_{ST} = 4$ Bytes, four-way set associative cache with cache line size $S_{CL} = 2^7 = 128$ Bytes, the size of the cache was 2^{20} Bytes. The cache placement algorithm and its structure correspond to those found in current superscalar processors, e.g., MIPS R8000 or MIPS R10000 (see [SGI96]).

The theoretical, simulated times, and their ratio are summarised in Table 1. The parameter C_{HR} is the average cache hit ratio to access a *BSPT* node in the cache during traversal. The average cache hit ratio for the node as the function of its depth in *BSPT* is in Table 2.

Note that for $S_{CL} = 128$ the compact subtree is complete, so the cache hit ratio for all the nodes at the same depth in the *BSPT* is equal. This is the reason why C_{HR} for depth 12, 16, and 20 are quite different from neighbour values, since these *BSPT* nodes are often read from the main memory. The probability that these nodes are already loaded in the cache is smaller with the increasing depth.

The average traversal times obtained by simulation correlate well with those computed theoretically. It is obvious the times obtained by the simulation are smaller than these derived theoretically, since the theoretical analysis supposes in each step an initial value of $C_{HR} = 0.0$.

6 Conclusion

In a previous paper [Havran97] we showed experimentally that ordinary subtree representation can decrease traversal time for ray–shooting using *BSPT* by 40% in a ray tracing application. In this paper we have analysed the time complexity and cache hit ratio of different *BSPT* representations for DFS order traversal in detail. We have shown the time complexity of traversing of a *BSPT* is reduced by organising its inner representation that matches better the memory hierarchy. The subtree representation decreases the traversal time for DFS order by 62% and increases hit ratio from 35% to 90% for a given example of common

	Representation			
	Random	DFS	Ordinary subtree	Compact subtree
T_A (theoretical)	1392.0	859.1	555.9	510.0
T'_A (simulated)	987.1	629.4	445.6	379.3
$ratio = T_A/T'_A$	1.41	1.36	1.24	1.34
$C_{HR}[\%]$	35.8	69.8	83.5	90.3

Table 1: The average traversal time computed theoretically and obtained by the simulation

Depth	0	1	2	3	4	5	6	7	8	9	10	11
C_{HR} (Random)	100	100	100	100	97	91	62	52	39	25	21	18
C_{HR} (DFS)	100	100	100	100	100	93	79	84	58	56	63	51
C_{HR} (Ordinary subtree)	100	100	100	100	100	100	97	73	90	85	53	79
C_{HR} (Compact subtree)	100	100	100	100	100	100	100	100	69	100	100	100
Depth	12	13	14	15	16	17	18	19	20	21	22	23
C_{HR} (Random)	21	19	19	0	0	0	0	0	0	0	0	0
C_{HR} (DFS)	57	59	47	59	54	48	51	49	47	54	43	54
C_{HR} (Ordinary subtree)	80	64	66	79	66	70	72	74	61	75	74	62
C_{HR} (Compact subtree)	7	100	100	100	1	100	100	100	0	100	100	100

Table 2: The cache hit ratio $C_{HR}[\%]$ as the function of node depth in *BSPT*

memory hierarchy. Moreover, proposed representation decreases the memory required to store *BSPT* in the main memory by 57%.

7 Future Work

The presented technique is widely applicable to other hierarchical data structures as well. Future work should include research of variants of multi–dimensional binary trees and hierarchical data structures in general. Dynamization of these data structures with regard to cache sensitive representation is also interesting topic to be researched.

ACKNOWLEDGEMENTS

I would like to thank Jan Hlavička for delivering the subject of *Advanced Computer Architectures* and thus compelling me to write a report on this topic. Further, I wish to thank Pavel Tvrdík and all the anonymous reviewers for their remarks on the previous version of this paper.

This research was supported by Grant Agency of the Ministry of Education of the Czech Republic number 1252/1998 and by Internal Grant Agency of Czech Technical University in Prague number 309810103.

References

[Arnold90] Arnold, O.A. *Probability, statistics, and queuing theory with computer science applications*, Second edition, Academic Press, San Diego, 1990.

[Berg97] de Berg, M., van Kreveld, M., Overmars, M., Schwartzkopf, O. *Computational Geometry, Algorithms and Applications*, Springer Verlag, 1997.

[Chiang95] Chiang, Y.-J. *Dynamic and I/O-Efficient Algorithms for Computational Geometry and Graphs Problems: Theoretical and Experimental Results*, Ph.D. Thesis, Dep. of Comp. Sci., Brown University, 1995.

[Cormen90] Cormen, T.H., Leiserson, C.H., Rivest, R.L. *Introduction to Algorithms*, The MIT Press, Cambridge, Massachusetts, 1990.

[Fuchs80] Fuchs, H., Kedem, M.Z., Naylor, B. On Visible Surface Generation by A Priori Tree Structures, *Proceedings of SIGGRAPH'80*, Vol. 14, No. 3, July, pp. 124–133, 1980.

[Havran97] Havran, V., Cache Sensitive Representation for BSP trees, *Compugraphics '97*, International conference on Computer Graphics, Portugal, December 15-18, pp. 369–376, 1997.

[Nyberg95] Nyberg, C., Barclay, T., Cvetanovic, Z., Gray, J., Lomet D.B. *AlphaSort: A Cache-Sensitive Parallel External Sort*, VLDB Journal, Vol. 4, No. 4, pp.603-627, 1995.

[Samet90] Samet, H. *The Design and Analysis of Spatial Data Structures*, reprinted with corrections in 1994, Addison-Wesley, 1990.

[SGI96] Silicon Graphics *Power Challenge*, Technical Report, 1996.

[Stroustrup97] Stroustrup, B. *The C++ Programming Language, 3rd ed.*, Addison-Wesley, 1997.

[Watt92] Watt, A., Watt, M. *Advanced Animation and Rendering Techniques*, ACM-PRESS, Addison-Wesley, 1992.

Improved Representations for Spatial Data Structures and Their Manipulations

Kuo–Liang Chung
Department of Information Management and Institute of Information Engineering
National Taiwan University of Science and Technology
No. 43, Section 4, Keelung Road, Taipei, Taiwan 10672, R. O. C.
Phone: +886 2 27376771, Fax: +886 2 27376776
E-mail: klchung@cs.ntust.edu.tw
AND
Jung–Gen Wu
Department of Information and Computer Education
National Taiwan Normal University
No. 162, Section 1, Hoping E. Road, Taipei, Taiwan 10610, R.O.C.
Phone: +886 2 23622841 ext 19, Fax: +886 2 23512772
E-mail: jgwu@ice.ntnu.edu.tw

Keywords: Bintrees, Image Compression, Image Manipulations, Quadtrees, Spatial Data Structures

Edited by: Frederick E. Petry, Maria A. Cobb and Kevin B. Shaw
Received: November 16, 1998 **Revised:** May 19, 1999 **Accepted:** June 10, 1999

Based on some observations on quadtrees, this paper first presents two improved representations for the linear quadtree and DF–expression. Then, we present three improved representations for the S–tree, bincodes, and logical bincodes. Some experiments are carried out to evaluate the performance of the five proposed improved representations and the existing corresponding representations. Experimental results show that all the proposed improved representations have better compression ratios when compared to the existing ones. Especially, the improved representations for the linear quadtree, bincodes, and logical bincodes have some significant compression performance. Finally, a few image manipulations, such as area calculation, centroid calculation, and set operations, on the proposed improved representations are discussed.

1 INTRODUCTION

The quadtree [4] (QT) is a well–known binary image representation and can reduce the memory requirement through the use of aggregation of homogeneous blocks. It also can speedup many related image manipulations. Samet [9] has surveyed many applications in image processing, pattern recognition, computational geometry, computer graphics, spatial databases, geographic information systems (GIS), etc.

Instead of using a pointer–type data structure to represent a QT, Gargantini [1] presented a pointerless spatial data structure (SDS), called the linear quadtree (LQ), which uses a set of codes to encode the leaf nodes in the QT. The LQ improves the memory efficiency of the QT. Another approach called the DF–expression [6, 11] represents the QT as a string by traversing the QT in a preorder manner. Based on the bintree (BT) [5], Jonge, Scheuermann, and Schijf (1994) [2] presented the S–tree (ST) image representation. Bincodes (BCs) were proposed by Ouksel and Yaagoub [8] and were shown to have some space improvement over the LQ in empirical comparisons [10]. Some fast image algorithms on BCs for GIS applications were presented in [3]. Using BCs as the input, Wu and Chung [14] presented the logical bincodes (LBCs), which employ the logical expression, and can support some fast image manipulations.

Based on some observations on QT, this paper first presents two improved representations for LQ and DF–expressions. Following the similar observations on BTs, we present three improved representations for the ST, BCs, and LBCs. Some experimentations are carried out to evaluate the performance of the proposed five improved representations and the corresponding existing ones. Experimental results show that all the proposed improved representations have better compression ratios when compared to the existing ones. Especially, the improved representations for the LQ, BCs, and LBCs show a better compression performance. Finally, a few image manipulations, such as area calculation, centroid calculation, and set operations (union, intersection, and complement), on the proposed improved representations are discussed.

The remainder of this paper is organized as follows.

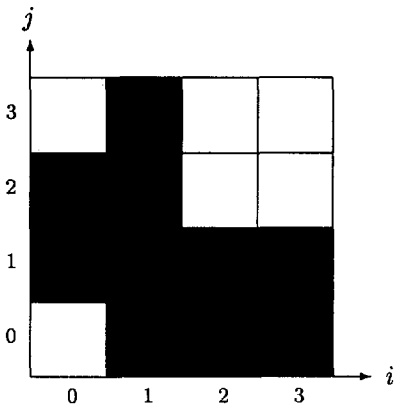


Fig. 1. An image example.

Section 2 reviews the existing five tree-based SDSs mentioned above in more detail. Some observations on QT and the proposed improved representations are presented in Section 3. In Section 4, some experiments on real images are carried out to demonstrate the compression performance. Some image manipulations on these proposed representations are discussed in Section 5. Some concluding remarks are given in Section 6.

2 TREE–BASED SPATIAL DATA STRUCTURES

In this section, we use a simple example to explain the five existing SDSs mentioned above.

For a QT, if the entire image is totally black or white, it is represented by a root node; otherwise, the root node is grey and the image is split into four equal–sized subimages, one for each quadrant, that are labeled *sw* (southwest), *se* (southeast), *nw* (northwest), and *ne* (northeast), respectively. This subdivision process is then repeated recursively for each of the four subimages until the subimage is totally black or white. A leaf node in a QT is called an external node; an internal node is called a grey node. Given a binary image with $2^2 \times 2^2$ pixels as shown in Fig. 1, where each black block is represented by a square box, the corresponding QT is shown in Fig. 2.

2.1 Linear Quadtree (LQ)

Without using pointers, an LQ [1] represents a QT by a set of codes, and each code is obtained by encoding a path from the root to that black node in the QT. Let the *sw* quadrant, the *se* quadrant, the *nw* quadrant, and the *ne* quadrant be encoded with 0, 1, 2, and 3, respectively. Then, the external node H_1 in Fig. 2 is encoded by 01; F is encoded by 02; H_2 is encoded

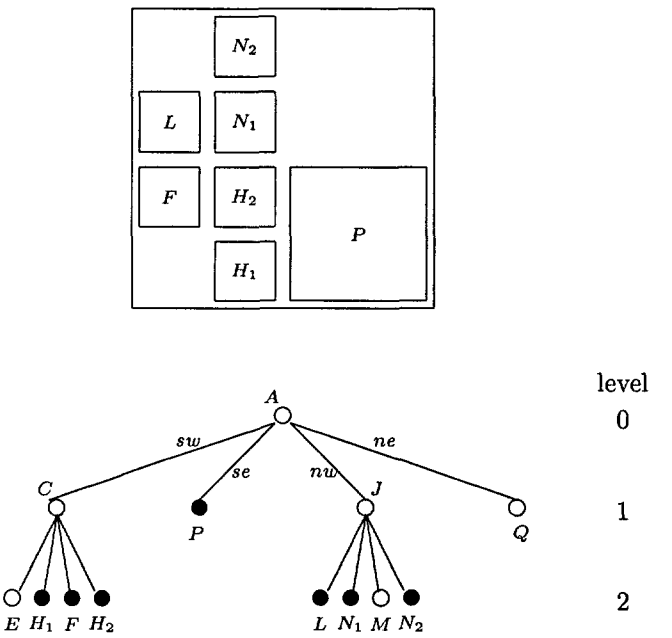


Fig. 2. The QT of Fig. 1.

by 03; P is encoded by $1X$, where X is a don't–care symbol, and so on. The LQ only encodes all the black nodes in the QT using a preorder traversal. Suppose the given image is of size $2^N \times 2^N$. For each black node in the QT, the length of the corresponding code in LQ is N. On the other hand, the code for a black node at level l will have $N-l$ consecutive X's in the rightmost part of that code. For each code, the first symbol in the LQ needs two bits, and each of the remaining symbols needs three bits since no symbol can start with an X symbol. Thus, it requires $2 + 3 \times (N-1)$ bits to represent any black node in the QT.

Using the above LQ encoding scheme by traversing the QT in a preorder way, the black nodes in Fig. 2 are encoded by the sequence $0102031X202123$.

2.2 DF–expression

Given a QT, the corresponding DF–expression [6] is based on a preorder traversal of the QT. During the traversal, if the encountered node is a black node, we append the symbol 'B' to the DF–expression; if the encountered node is a white node, we append the symbol 'W'; we append the symbol '(' when the encountered node is a grey node. Suppose the number of nodes in the given QT is q, both the DF–expression and the LQ can be obtained in $O(q)$ time.

The DF–expression of Fig. 2 is $((WBBBB(BBWBW$.

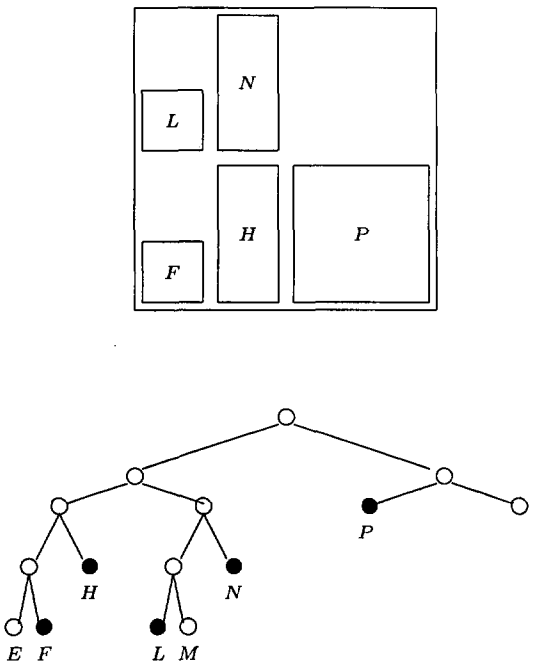


Fig. 3. The BT representation of Fig. 1.

linear–tree table	: 0 0 0 0 1 1 1 0 0 1 1 1 0 1 1
color table	: 0 1 1 1 0 1 1 0

Fig. 4. The ST of Fig. 3.

2.3 S–tree (ST)

The three image representations described in this and the following two subsections are based on the BT structure.

In this subsection, we first introduce the BT, then describe the ST [2]. The BT [9] is based on the recursive subdivision of the image into two equal–sized subimages. At each step, the splitting alternates between the horizontal $i-$, and vertical $j-$, axis. If the subarray does not consist entirely of 1's or entirely of 0's, it is further subdivided into two equal–sized subimages until blocks that consist entirely of 1's or entirely of 0's are obtained. The corresponding BT of Fig. 1 is shown in Fig. 3.

The ST consists of two array tables, namely, the linear–tree table and the color table. During a preorder traversal of the BT, at each time, we append a '0' when a grey node is encountered, and a '1' when a leaf node is encountered to the linear–tree table. Furthermore, we append a '1' ('0') when a black (white) leaf node is encountered to the color table. The ST for the BT of Fig. 3 is shown in Fig. 4.

2.4 Bincodes (BCs)

The BCs are obtained based on the BT structure and represent the BT as an ordered collection of black nodes in the BT. Given a $2^N \times 2^N$ binary image, each black node at level l and at location (x, y), which is located at left–bottom corner of the corresponding block, is encoded as $b(l, x, y) = \Sigma_{k=0}^{N-1}(x_k \times 2^{4k+3}) + \Sigma_{k=0}^{N-1}(y_k \times 2^{4k+1}) + \Sigma_{k=0}^{2N-1}(s_k \times 2^{2k})$, where $x_{N-1}x_{N-2}.....x_0$ and $y_{N-1}y_{N-2}.....y_0$ denote the binary representations of x and y, respectively, and $s = 2^{2N} - 2^{2N-l} = (s_{2N-1}s_{2N-2}.....s_0)_2$ with subscript 2 denoting base 2. In fact, the encoded BC for that black node is expressed as the sequence $(x_{N-1}s_{2N-1}y_{N-1}s_{2N-2}x_{N-2}s_{2N-3}...x_0s_1y_0s_0)_2$.

In Fig. 3, the block H at location $(1, 0)$ and at level 3 is encoded as $(01011100)_2 = 92$ since $i = (01)_2$, $j = (00)_2$, and $s = (1110)_2$. By traversing all the black nodes in the BT in a preorder fashion, the BCs of Fig. 3 are represented by the strictly increasing ordered sequence $\langle 87, 92, 117, 124, 208\rangle$.

2.5 Logical Bincodes (LBCs)

Observing the process in building the BT, we first divide the original image into two equal–sized subimages in the x–direction. All the pixels in the left (right) subimage have their x_{N-1} to be 0 (1); the other coordinate–variables x_{N-2},..., x_0, y_{N-1}, y_{N-2},..., and y_0 are viewed as don't–care symbols. Logically, we denote the left (right) subimage as $\bar{X}_{N-1}$ (X_{N-1}). Then, we divide the two resulting subimages in the y direction. The resulting four subimages, namely, the left–lower part, left–upper part, right–lower part, and right–upper part, are denoted by $\bar{X}_{N-1}\bar{Y}_{N-1}$, $\bar{X}_{N-1}Y_{N-1}$, $X_{N-1}\bar{Y}_{N-1}$, and $X_{N-1}Y_{N-1}$, respectively.

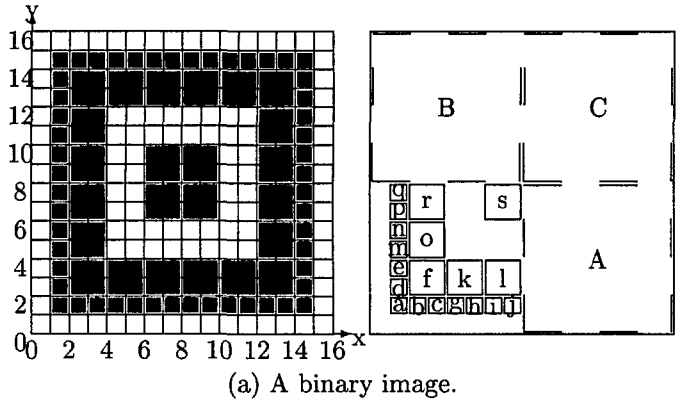
In [14], each pair of bits x_ks_{2k+1} or y_ks_{2k} in one BC is interpreted as a logical function of variable X_k or Y_k. $(01)_2$ is used to represent $\bar{X}_k$ or $\bar{Y}_k$, $(11)_2$ is used to represent X_k or Y_k, and $(00)_2$ is used to represent the don't–care symbol 'δ'. The BC of the block H in Fig. 3 is represented by $(01011100)_2$ and using this interpretation, the corresponding LBC is represented by $\bar{X}_1\bar{Y}_1X_0\delta$. Some block whose logical representation has more than one 'δ'. For example, the block P in Fig. 3 is represented logically by $X_1\bar{Y}_1\delta\delta$. Since all the 'δ's are at the end of the logical expression, only one 'δ' is needed to show the ending of one LBC, and this can save some storage space. Consequently, the LBCs of Fig. 3 can be represented by a logical expression $\bar{X}_1\bar{Y}_1\bar{X}_0Y_0 + \bar{X}_1\bar{Y}_1X_0\delta + \bar{X}_1Y_1\bar{X}_0\bar{Y}_0 + \bar{X}_1Y_1X_0\delta + X_1\bar{Y}_1\delta$.

3 IMPROVED REPRESENTATIONS

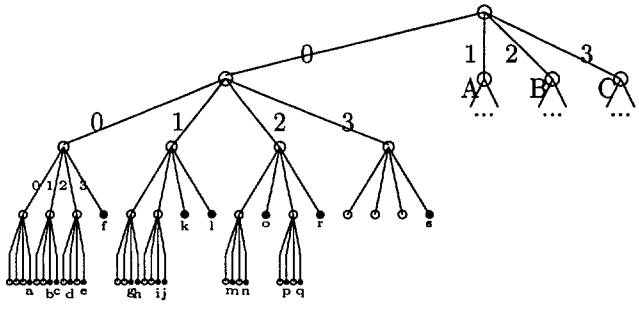
Let us consider the example shown in Fig. 5. The subtree from level 0 to level 3 is a complete subtree. Obviously, the codes used in the tree–based SDS mentioned in Section 2, e.g. LQ, only used to encode the information in the edges and nodes in the QT ranging from level 2 to level 4, but it is unnecessary to encode the information in the subtree from level 0 to level 1. Therefore, the topmost two levels can be discarded; this leads to an improved representation and has a storage–saving effect.

Similarly, the DF–expression can discard the topmost levels in the QT. However, the DF–expression can discard one more level than that of the improved LQ representation since the LQ representation encodes only the black nodes and we need to preserve the information one level above the bottom level of the complete subtree to maintain the order of the resulting LQ codes, which will be used to recover the original codes. The DF–expression stores both the white and black nodes, so one more level can be discarded without losing any information when compared to the improved LQ representation.

By the same arguments, for the ST, BCs and LBCs, we can use the above observation to reduce the memory requirement for representing them. In what follows, the QT as shown in Fig. 5(b) and the BT as shown in Fig. 6(b) are used to demonstrate the proposed five improved representations. In order to simplify the presentation, only the first quadrant of Fig. 5 is depicted in detail.



(a) A binary image.



(b) The QT.

Fig. 5. An example and its QT representation.

3.1 Improved LQ (ILQ)

For the first quadrant of Fig. 5(a), the codes of LQ are 0003, 0012, 0013, 0021, 0023, 003X, 0102, 0103, 0112, 0113, 012X, 013X, 0201, 0203, 021X, 0221, 0223, 023X, and 033X. Looking at these 19 codes, the leftmost significant digit, say the first digit, of each code is 0 since these codes all branch from the leftmost edge encoded by $(0)_4 = (00)_2$ from the root node of Fig. 5(b). Similarly, the second digit of the first six codes is 0; the second digit of each code from the 7th (13th) one to the 12th (18th) one is 1 (2); the third digit of the last code is 3. For this quadrant, if we discard the first digit, say 0, of these 19 codes, the resulting codes are 003, 012, 013, 021, 023, 03X, 102, 103, 112, 113, 12X, 13X, 201, 203, 21X, 221, 223, 23X, and 33X. This new representation is called the improved LQ (ILQ) for the first quadrant.

By the same arguments, for the LQ codes in the second quadrant, the third quadrant, and the fourth quadrant, the corresponding ILQ codes can be obtained by discarding the digits 1, 2, and 3, respectively.

To recover the LQ codes from the ILQ codes, we process the ILQ codes sequentially. For the first quadrant, the digit 0 is added to the left of each processed code unless it occurs that the leftmost digit of the processed code is 0 but that of its predecessor is 3. Afterwards, the digit 1 is added to the left of each processed code unless it occurs that the leftmost digit of the processed code is 0 but that of its predecessor is 3. Continuing this way, the LQ codes can be recovered from the LQ codes.

In general, given a QT with depth d, suppose there is a complete subtree rooted at the root node of the QT, which ranges from level 0 to level h. The ILQ only needs the information in the edges and nodes in the QT ranging from level $h-1$ to level d.

We now analyze the storage–saving performance of ILQ over LQ. Given a $2^N \times 2^N$ image, each LQ code needs $2+3(N-1)$ bits. We see that the complete subtree in the corresponding QT ranges from level 0 to level h. For each ILQ code, it needs $3N-3h+5 = (2+3(N-1)-3(h-2))$ bits, so we have the following result:

Result 1. Given a QT whose topmost h levels form a complete subtree, for each code in the corresponding ILQ representation, the storage–saving performance over the one in the LQ representation is equal to $\frac{3h-6}{3N-1}$ $(= \frac{3N-1-3N+3h-5}{2+3(N-1)})$.

In Fig. 5, it is known that $N = 4$ and $h = 3$,

so each ILQ code has 27.2% ($= \frac{3}{11}$) storage–saving performance over the corresponding LQ code.

Considering the first 'X' symbol from the left in each LQ code as a delimiter, Gargantini [1] encodes a black node in QT by using only one 'X' symbol in each LQ code. In Section 4, this version of LQ will be employed in experiments to demonstrate the storage–saving advantage of the proposed ILQ. For this simpler version of LQ, theoretically, we have the following result:

Result 2. Given a QT whose topmost h levels form a complete subtree, suppose a code in the corresponding LQ representation is at level l. For this code in the ILQ representation, the storage–saving performance over the one in the LQ representation is equal to $\frac{3h-6}{3l-1}$ ($= \frac{3l-1-3N+3h-5}{2+3(l-1)}$).

3.2 Improved DF–expression (IDF–expression)

The DF–expression of the image in Fig. 5 is represented by ((((*WWWB* (*WWBB* (*WBWBB* ((*WWBB* (*WWBBBB* ((*WBWBB* (*WBWBB* (*WWWB*.... Using the observation on QT mentioned above, the symbols '('s, which are used to represent the internal nodes in the top three levels, i.e. level 0, level 1, and level 2, in the original DF–expression can be discarded without losing any information. Thus, the improved DF–expression (IDF–expression) is represented by (*WWWB* (*WWBB* (*WBWBB* (*WWBB* (*WWBBBB* (*WBWBB* (*WBWBB WWWB*..., and totally 21 '('s are discarded. The IDF–expression discards one more level than the ILQ representation since the ILQ representation encodes only the black nodes, so we need to preserve the information one more level to maintain the preorder order and to be able to recover the original codes. Besides storing the internal nodes as '('s, the DF–expression also stores both the white and black nodes, thus one more level can be discarded than in the ILQ.

Suppose we use two bits 00 to represent a '(', two bits 10 to represent a 'W', and one bit 1 to represent a 'B'. Using the same condition of the complete subtree as described in Subsection 3.1, we have the following result since the IDF–expression can discard $\frac{4^h-1}{3}$ '('s in the DF–expression:

Result 3. Given a QT whose topmost h levels form a complete subtree, suppose it needs f bits to represent the corresponding DF–expression. Then, it needs $f - \frac{2\times 4^h-2}{3}$ bits to represent the IDF–expression. However, we need to store a number to indicate the number of levels been discarded, and assume we use four bits to represent this number since it is large enough for denoting the maximal levels allowable in any $2^{16} \times 2^{16}$ image. The storage–saving performance is $\frac{2\times 4^h-14}{3f}$ ($= \frac{\frac{2\times 4^h-2}{3}-4}{f}$).

The DF–expression of Fig. 5(b) needs 322 bits, but the corresponding IDF–expression needs 284 bits, the storage–saving performance of IDF–expression over DF–expression is about 11.8% ($= \frac{38}{322}$).

3.3 Improved S–tree (IST)

The ST representation of the image in Fig. 6 is represented by

linear–tree table:	00000001011011000110
	11100011011100000011
	01110001101110101 1...
color table:	00101010110101101011
	01011001...

Using the above observation on BT, the symbols 0's, which are used to represent the internal nodes in the top five levels, i.e. level 0 to level 4, in the linear–tree table of the original ST representation can be discarded without losing any information. Thus, the linear–tree table of the compact S–tree (IST) is represented by

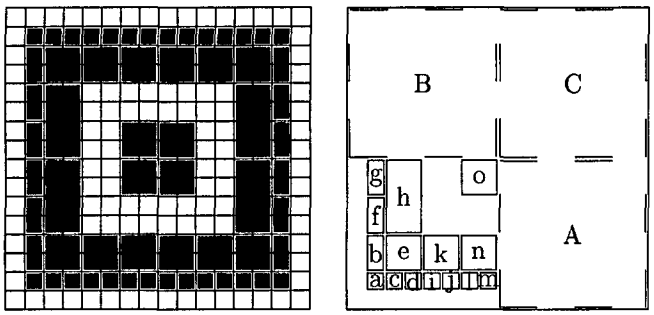
00101101100011011100110111000110111000
1101111011...

while the color table remains the same, and totally 31 0's are discarded. Basically, the storage–saving concept used in IST is very similar to the one in IDF–expression.

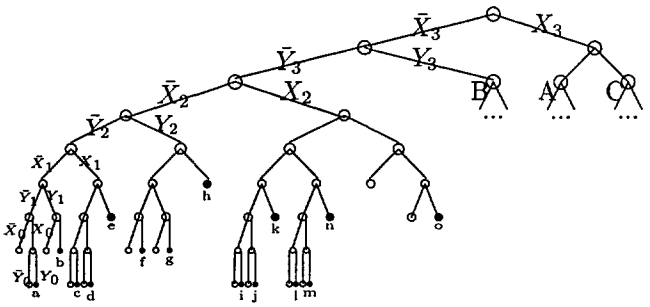
Only one bit is needed to represent 0 or 1 in the linear–tree table and the color table. Given a BT with depth d, suppose there is a complete subtree rooted at the root node in the BT, which ranges from level 0 to level h. We then have the following result since the IST can discard $2^h - 1$ 0's in the ST. Same as what described in Subsection 3.2, an extra 4–bit number is used to indicate the number of discarded levels in ST. We then have the following result:

Result 4. Given a BT whose topmost h levels form a complete subtree, suppose we need s bits to represent the corresponding ST. Then, we need $s - (2^h + 3)$ ($= s - 2^h + 1 - 4$) bits to represent the IST–expression. The storage–saving performance is $\frac{2^h+3}{s}$.

The ST of Fig. 6(b) needs 337 bits, but the corresponding IST needs 310 bits. Therefore, the storage–saving performance of IST over ST is about 8% ($= \frac{27}{337}$).



(a) The image subdivided to form a BT.



(b) The BT.

Fig. 6. The example of Fig. 5(a) and its BT representation.

3.4 Compact Bincodes (CBCs)

For the first quadrant in Fig. 6(a), the corresponding BCs are ⟨ 21855, 21884, 21975, 21983, 22000, 22364, 22396, 22464, 23895, 23903, 23920, 24023, 24031, 24048, 24560 ⟩ (= ⟨ 0101010101011111, 0101010101111100, 0101010111010111, 0101010111011111, 01010101 11110000, 0101011101011100, 0101011101111100, 0101011111000000, 0101110101010111, 01011101 01011111, 0101110101110000, 0101110111010111, 0101110111011111, 0101110111110000, 01011111 11110000 ⟩$_2$).

Looking at the binary representation of these 15 BCs, the leftmost four bits of them are all 0101's since they all branch from the left edge ($y_3 = 0$) and from the left son ($x_3 = 0$) of the root node of Fig. 6(b). By the same arguments, the next lower four bits in the first five codes are 0101; that of the codes from the sixth (ninth) code to the eighth (14th) code are 0111 (1101); that of the last code is 1111. For this quadrant, if we discard the leftmost six bits of each code among these 15 codes, the resulting codes are ⟨ 351, 280, 471, 479, 496, 860, 892, 960, 343, 351, 360, 471, 479, 496, 1008 ⟩ (= ⟨ 0101011111, 0101111100, 0111010111, 0111011111, 0111110000, 1101011100, 11011 11100, 1111000000, 0101010111, 0101011111, 0101110000, 0111010111, 0111011111, 01111 10000, 1111110000 ⟩$_2$). As a result, two subtrees are formed. The first subtree contains the first eight codes that discard their leftmost leading bits 010101; the second subtree contains the remaining codes that discard their leftmost leading bits 010111. The codes in each subtree are still listed in preorder. This new representation is called the compact BC (CBC) for the first quadrant.

By the same arguments, for the BCs in the second quadrant, third quadrant, and the fourth quadrant, the CBCs can be obtained by discarding the leftmost six bits 011101, 110101, 111101, in the first BCs, and discarding the leftmost six bits 011111, 110111 and 111111 in the last BCs, respectively.

To recover the BCs from CBCs, we process the CBCs codes sequentially in a preorder order. Each time, six bits 010101 are added to the left of the processed code unless it occurs that the predecessor is less than or equal to its preceding one. Afterwards, each time six bits 010111 are added to the left of the processed code unless it occurs that the processed code is less than or equal to its predecessor. Similarly, six bits 011101, 011111, 110101, 110111, 111101, and 111111 are added to the left of processed code sequentially started whenever the processed code is less than or equal to its predecessor.

Using the same condition of the complete subtree as described in Subsection 3.3, the CBCs only needs the information in the edges and nodes in the BT ranging from level $h-1$ to level d. We now analyze the storage-saving performance of CBC over BC. Given a $2^N \times 2^N$ image, each BC needs $4N$ bits. For each CBC, it needs $4N - 2h + 4$ $(= 4N - 2(h-2))$ bits, so we have the following result:

Result 5. Given a BT whose topmost h levels form a complete subtree, for each code in the corresponding CBCs, the storage-saving performance over the one in the BCs is equal to $\frac{h-2}{2N}$ $(=\frac{4N-(4N-2h+4)}{4N})$.

In Fig. 6, it is known that $N = 4$ and $h = 5$, so each CBC has 37.5% $(= \frac{3}{8})$ storage-saving performance over the corresponding BC.

3.5 Compact LBCs (CLBCs)

The LBC representation stores at most one don't-care symbol, $(00)_2$, at the end of each BC. The LBCs of the first quadrant of Fig. 6 are represented by ⟨ $\bar{X}_3\bar{Y}_3\bar{X}_2\bar{Y}_2\bar{X}_1\bar{Y}_1X_0Y_0$, $\bar{X}_3\bar{Y}_3\bar{X}_2\bar{Y}_2\bar{X}_1Y_1X_0\delta$, $\bar{X}_3\bar{Y}_3\bar{X}_2\bar{Y}_2X_1\bar{Y}_1\bar{X}_0Y_0$, $\bar{X}_3\bar{Y}_3\bar{X}_2\bar{Y}_2X_1\bar{Y}_1X_0Y_0$, $\bar{X}_3\bar{Y}_3\bar{X}_2\bar{Y}_2X_1Y_1\delta$, $\bar{X}_3\bar{Y}_3\bar{X}_2Y_2\bar{X}_1\bar{Y}_1X_0$, $\bar{X}_3\bar{Y}_3\bar{X}_2Y_2\bar{X}_1Y_1X_0$, $\bar{X}_3\bar{Y}_3\bar{X}_2Y_2X_1\delta$, $\bar{X}_3\bar{Y}_3X_2\bar{Y}_2\bar{X}_1\bar{Y}_1\bar{X}_0Y_0$, $\bar{X}_3\bar{Y}_3X_2\bar{Y}_2\bar{X}_1\bar{Y}_1X_0Y_0$, $\bar{X}_3\bar{Y}_3X_2\bar{Y}_2\bar{X}_1Y_1\delta$, $\bar{X}_3\bar{Y}_3X_2\bar{Y}_2X_1\bar{Y}_1\bar{X}_0Y_0$, $\bar{X}_3\bar{Y}_3X_2\bar{Y}_2X_1\bar{Y}_1X_0Y_0$, $\bar{X}_3\bar{Y}_3X_2\bar{Y}_2X_1Y_1\delta$, $\bar{X}_3\bar{Y}_3X_2Y_2X_1Y_1\delta$ ⟩. After discarding the leftmost three symbols $\bar{X}_3$, $\bar{Y}_3$, and $\bar{X}_2$ (X_2) in the first eight (last seven) LBCs, the resulting

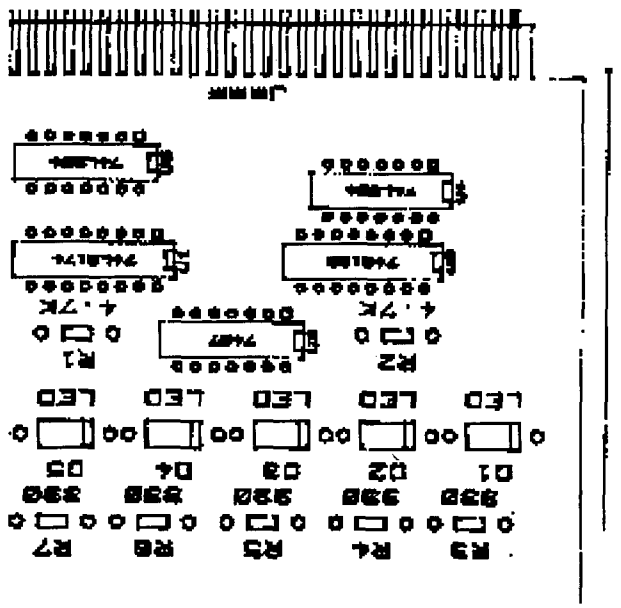


Fig. 7: A sample image.

compact LBCs (CLBCs) become $\langle\ \bar{Y}_2\bar{X}_1\bar{Y}_1X_0Y_0$, $\bar{Y}_2\bar{X}_1Y_1X_0\delta$, $\bar{Y}_2X_1\bar{Y}_1\bar{X}_0Y_0$, $\bar{Y}_2X_1\bar{Y}_1X_0Y_0$, $\bar{Y}_2X_1Y_1\delta$, $Y_2\bar{X}_1\bar{Y}_1X_0\delta$, $Y_2\bar{X}_1Y_1X_0\delta$, $Y_2X_1\delta$, $\bar{Y}_2\bar{X}_1\bar{Y}_1\bar{X}_0Y_0$, $\bar{Y}_2\bar{X}_1\bar{Y}_1X_0Y_0$, $\bar{Y}_2\bar{X}_1Y_1\delta$, $\bar{Y}_2X_1\bar{Y}_1\ \bar{X}_0Y_0$, $\bar{Y}_2X_1\bar{Y}_1X_0Y_0$, $\bar{Y}_2X_1Y_1\delta$, $Y_2X_1Y_1\delta\ \rangle$. By the similar arguments, for the other three quadrants, the corresponding LBCs can be compressed further. Under the same condition of the complete subtree as described in 3.4, we have the following result:

Result 6. Given a BT whose topmost h levels form a complete subtree, suppose one code in the LBCs is at level l. For this code in the CLBC representation, the storage–saving performance over the one in the LBC representation is equal to $\frac{h-2}{2l}$ $(= \frac{2(h-2)}{4l})$.

4 EXPERIMENTAL RESULTS

We take 16 real images to show the performance of the proposed five compact representations when compared to the existing ones. Among these sample images, Samples 1–5 are Chinese texts; Samples 6–10 are English texts; Samples 11–16 are figures and pictures. In our experiments, all images have 256 × 256 pixels, i.e. it needs 8K bytes of memory to save it. Fig. 7 illustrates Sample 16.

Tables 1, 2, 3, 4, and 5 illustrate the performance of ILQ, IDF–expression, IST, CBCs, and CLBCs over the existing LQ, DF–expression, ST, BCs, and LBCs, respectively. In each table, the column 'h' represents the number of the topmost levels that form a complete subtree. The column 'EXI' lists the number of bits needed to represent the original existing representation. The column 'COM' lists the number of bits needed to represent the proposed compact representation, and the column 'COM/EXI' lists the ratio of the number of bits required in the proposed compact representation over the original existing one. The storage–saving performance is listed in the last column 'SP' $(= 1 - \frac{\text{COM}}{\text{EXI}})$.

Tables 1, 4, and 5 show that the average storage–saving performance of ILQ, CBCs, and CLBCs over LQ, BCs, and LBCs is 15%, 21%, and 24%, respectively. However, Tables 2 and 3 show that the average storage–saving performance of IDF–expression and IST over DF–expression and ST is only 0.29% and 0.16%, respectively. The two proposed compact SDSs have a little compression improvement, but the storage–saving performance can be improved much more if the level of the related complete subtree becomes quite large.

Table 1: Performance of ILQ.

Sample #	#(codes)	h	EXI	COM	COM/EXI	SP
1	4006	3	86540	74526	86%	14%
2	2818	3	59777	51327	86%	14%
3	5633	3	121642	104747	86%	14%
4	3616	3	82535	71691	87%	13%
5	6326	3	144925	125951	87%	13%
6	8393	5	191380	115847	61%	39%
7	4702	3	107222	93120	87%	13%
8	6662	3	151828	131846	87%	13%
9	5625	3	127245	110374	87%	13%
10	7957	3	176072	152205	86%	14%
11	6844	3	155936	135408	87%	13%
12	1905	3	43392	37681	87%	13%
13	2337	3	53535	46528	87%	13%
14	1479	2	31242	31246	100%	0%
15	7234	3	164246	142548	87%	13%
16	6563	4	139717	100343	72%	28%

Table 2: Performance of IDF–expression.

Sample #	#(symbols)	h	EXI	COM	COM/EXI	SP
1	10940	3	17680	17640	99.77%	0.23%
2	7675	3	12407	12367	99.68%	0.32%
3	14672	3	23972	23932	99.83%	0.17%
4	13112	3	20005	19965	99.80%	0.20%
5	22512	3	34465	34425	99.88%	0.12%
6	30100	5	46017	45337	98.52%	1.48%
7	16756	3	25646	25606	99.84%	0.16%
8	23676	3	36256	36216	99.89%	0.11%
9	18536	3	28794	28754	99.86%	0.14%
10	21432	3	34746	34706	99.88%	0.12%
11	23700	3	36468	36428	99.89%	0.11%
12	6612	3	10169	10129	99.61%	0.39%
13	9380	3	14061	14021	99.72%	0.28%
14	4216	2	6748	6740	99.88%	0.12%
15	23292	3	36348	36308	99.89%	0.11%
16	16176	4	26782	26614	99.37%	0.63%

Table 3: Performance of IST.

Sample #	h	EXI	COM	COM/EXI	SP
1	5	17996	17982	99.92%	0.08%
2	5	12347	12333	99.89%	0.11%
3	5	23207	23193	99.94%	0.06%
4	5	27713	27699	99.95%	0.05%
5	6	40325	40295	99.93%	0.07%
6	9	46017	45337	98.52%	1.48%
7	5	29771	29757	99.95%	0.05%
8	5	42499	42285	99.97%	0.03%
9	5	30755	30741	99.95%	0.05%
10	5	38024	38010	99.96%	0.04%
11	5	42258	42244	99.97%	0.03%
12	5	11504	11490	99.88%	0.12%
13	5	15704	15690	99.91%	0.09%
14	3	7145	7143	99.97%	0.03%
15	6	38051	38021	99.92%	0.08%
16	7	26294	26232	99.76%	0.24%

Table 4: Performance of CBC.

Sample #	#(BCs)	h	EXI	COM	COM/EXI	SP
1	2902	5	92864	75456	81%	19%
2	2005	5	64160	52134	81%	19%
3	3948	5	126336	102652	81%	19%
4	2698	5	86336	70152	81%	19%
5	4755	6	152160	114124	75%	25%
6	6975	9	223200	125554	56%	44%
7	3852	5	123264	100156	81%	19%
8	5515	5	176480	143394	81%	19%
9	4294	5	137408	111648	81%	19%
10	6191	5	198112	160970	81%	19%
11	5649	5	180768	146878	81%	19%
12	1503	5	48096	39082	81%	19%
13	1795	5	57440	46674	81%	19%
14	1126	3	36032	33784	94%	6%
15	5311	6	169952	127468	75%	25%
16	4636	7	148352	101996	69%	31%

Table 5: Performance of CLBC.

Sample #	#(LBCs)	h	EXI	COM	COM/EXI	SP
1	2902	5	88886	67980	77%	23%
2	2005	5	60348	45680	76%	24%
3	3948	5	120580	91982	76%	24%
4	2698	5	85912	67636	79%	21%
5	4755	6	151778	110334	73%	27%
6	6975	9	222094	120506	61%	39%
7	3852	5	122648	97256	79%	21%
8	5515	5	175548	139244	79%	21%
9	4294	5	135988	106482	78%	22%
10	6191	5	192938	150166	78%	22%
11	5649	5	179784	142528	79%	21%
12	1503	5	47812	37750	79%	21%
13	1795	5	57296	45310	79%	21%
14	1126	3	33810	30234	89%	11%
15	5311	6	168506	121118	72%	28%
16	4636	7	139890	87694	63%	37%

5 IMAGE MANIPULATIONS

This section develops some image manipulations, such as computing geometric properties (area and centroid) and set operations, on the proposed compact SDSs. We only discuss the related manipulations on the IDF–expression and CBCs since in essence, the coding schemes of ILQ and CLBCs are some similar to that of CBCs; the coding scheme of IST is similar to that of the IDF–expression.

5.1 Computing Geometric Properties

In a QT/BT, the area of a black block represented by a black leaf node can be calculated from its level. For example, using a QT to represent an image with $2^n \times 2^n$ pixels, a black leaf node on level 1 has $2^{(n-1)} \times 2^{(n-1)}$ pixels; a black leaf node on level 2 has $2^{(n-2)} \times 2^{(n-2)}$ pixels. In the corresponding BT, a black leaf node on level 1 has $2^{(n-1)} \times 2^n$ pixels; a black leaf node on Level 2 has $2^{(n-1)} \times 2^{(n-1)}$ pixels, and so on.

To find the area of an image represented by an IDF–expression, we need to traverse the QT logically, where a node at level l has $4^{(n-l)}$ pixels. The algorithm for computing area on IDF–expression is listed below, where the IDF–expression discards the top–most c levels of the corresponding QT.

```
Area = 0; i = 0; l = 0;
Traverse(l); Traverse(l); Traverse(l); Traverse(l); /*
Four quadrants */
end
/* Traversing the QT logically to sum up each subarea
*/
Function Traverse(l)
If l <= c
  {
  l = l + 1;
  Traverse(l); Traverse(l);
  Traverse(l); Traverse(l); /* four quadrants */
  }
else
   {
    If IDF[i]='('
/* array IDF[] saves the IDF-expression */
  {
  l = l + 1;
  Traverse(l); Traverse(l);
  Traverse(l); Traverse(l); /* four quadrants */
  }
     else
      If IDF[i]='1'
       {
        Area = Area + 4^(n-l) ;
       }
   i = i + 1;
   }
```

The level of a BC can be found by counting the number of don't–care symbols, each don't–care symbol being a 2–bit pair $(00)_2$, in the BC. If there are b don't–care symbols, where the least significant $2b$ bits in the BC are all 0's, the corresponding block has $2^{\lfloor b/2 \rfloor} \times 2^{\lceil b/2 \rceil}$ pixels. The CBC discards some highest order bits of BC, while the number of don't–care symbols is not changed. The algorithm for computing the area on CBCs is listed below, where the CBCs are stored in the array CBC[].

```
/* Finding the area of an image represented by k
CBCs */
Area = 0;
For i = 1 to k
 b = the number of don't-care symbols in CBC[i];
 Area = Area + 2^⌊b/2⌋ × 2^⌈b/2⌉ ;
Next i
```

The centroid of all the concerning black pixels is calculated by averaging the x– and y–coordinates of the center of all the black blocks.

To find the centroid of an image represented by an IDF–expression, we traverse the QT logically to find the x– and y–coordinates of all the black leaf nodes. For an internal node at level l, its four sons have their bit–$(n-l-1)$ in the binary representation of x– and y–coordinates being $(0,0)$, $(1,0)$, $(0,1)$, and $(1,1)$, respectively. The algorithm for finding the centroid on the IDF–expression is shown below.

/* Finding the centroid (x, y) of an image represented by a IDF–expression which discards the top–most d levels of the corresponding QT */

$x = 0$; $y = 0$; $i = 0$; $l = 0$;
$a = 0$; $b = 0$; Traverse_and_find_xy(a, b, l);
$a = 2^{n-1}$; $b = 0$; Traverse_and_find_xy(a, b, l);
$a = 0$; $b = 2^{n-1}$; Traverse_and_find_xy(a, b, l);
$a = 2^{n-1}$; $b = 2^{n-1}$; Traverse_and_find_xy(a, b, l);
$x = x/k$;
$y = y/k$;
end

/* Traversing the QT logically and finding the x– and y–coordinates */
Function Traverse_and_find_xy(a, b, l)
If $l \leq d$
{
$l = l + 1$;
$a1 = a$; $b1 = b$;
Traverse_and_find_xy$(a1, b1, l)$;
$a1 = a \vee 2^{n-l-1}$; $b1 = b$;
Traverse_and_find_xy$(a1, b1, l)$;
$a1 = a$; $b1 = b \vee 2^{n-l-1}$;
Traverse_and_find_xy$(a1, b1, l)$;
$a1 = a \vee 2^{n-l-1}$; $b1 = b \vee 2^{n-l-1}$;
Traverse_and_find_xy$(a1, b1, l)$;
}
else
{
If DF[i]='('
{
$l = l + 1$;
$a1 = a$; $b1 = b$;
Traverse_and_find_xy$(a1, b1, l)$;
$a1 = a \vee 2^{n-l-1}$; $b1 = b$;
Traverse_and_find_xy$(a1, b1, l)$;
$a1 = a$; $b1 = b \vee 2^{n-l-1}$;
Traverse_and_find_xy$(a1, b1, l)$;
$a1 = a \vee 2^{n-l-1}$; $b1 = b \vee 2^{n-l-1}$;
Traverse_and_find_xy$(a1, b1, l)$;
}
else
If DF[i]='1'
{
$x = x + a + 2^{(n-l-1)}$;
$y = y + b + 2^{(n-l-1)}$;
}
$i = i + 1$;
}

For a black block represented by a BC, $(b_{4n-1}b_{4n-2}....b_2b_1b_0)_2$, we can extract the x– and y–coordinates of its left–bottom corner and level l. The x–coordinate and y–coordinate of the BC are $(b_{4n-1}b_{4n-5}...b_7b_3)_2$ $(b_{4n-3}b_{4n-7}...b_5b_1)_2$, respectively. The centroid of the corresponding block is $(x + 2^{\lfloor l/2 \rfloor}, y + 2^{\lceil l/2 \rceil})$. The CBC discards some highest order bits of BC. We need to recover these highest bits in the x– and y– coordinates to find the centroid. If only the root of the BT is discarded, then two sub–BTs are generated. We need to add a 0 to the highest bit of the x–coordinate for the CBCs in the left sub–BT, and add a 1 to that of the right sub–BT. If the topmost two levels of the BT are discarded, then we need to add one bit to the leftmost bit of the x–coordinate and one bit to that of the y–coordinate. The added bits for the four split sub–BTs are (0,0), (0,1), (1,0), and (1,1), respectively. If the topmost h levels of the BT are discarded, then there are 2^h generated sub–BTs. We need to add $2^{\lceil h/2 \rceil}$ bits $(x_{n-1}x_{n-2}...x_{n-\lceil h/2 \rceil})$ to the leftmost bits of the x–coordinate and $2^{\lfloor h/2 \rfloor}$ bits $(y_{n-1}y_{n-2}...y_{n-\lfloor h/2 \rfloor})$ to the y–coordinate in the CBC. The sequence of $x_{n-1}y_{n-1}x_{n-2}y_{n-2}...x_{n-\lceil h/2 \rceil}y_{n-\lfloor h/2 \rfloor}$'s to be added to the CBCs in the 2^h sub–BTs are 0...0, 0...01, ..., and 1...1. The algorithm for finding the centroid on CBCs is shown below.

/* Finding the centroid (x, y) of an image represented by k CBCs */
$x = 0$; $y = 0$;
For $i = 1$ to k
l = the number of don't–cares of $CBC[i]$;
a = the x–coordinate of $CBC[i]$;
b = the y–coordinate of $CBC[i]$;
$x = x + a + 2^{\lfloor l/2 \rfloor}$; $y = y + b + 2^{\lceil l/2 \rceil}$;
Next i
$x = x/k$; $y = y/k$;

5.2 Set Operations

We only sketch the set operations on BTs since the detailed algorithms for set operations on split sub–BTs mentioned above can be obtained by calling the existing related algorithms.

If we discard the topmost h levels of a BT, then 2^h sub–BTs are generated. Each sub–BT is a BT which represents $1/2^h$ of the original image. To find the intersection and union of the two images, we can perform the related operations on the corresponding subimages. If the compact SDSs of the images, say IST, CBC, and CLBC, discard the same number of levels, set operations can be performed on the corresponding sub–BTs. However, if the improved SDSs of an image discards more levels than the other one, we need to add some levels back in order to make the two improved SDSs have the same number of sub–BTs. The related process for finding the intersection of two images represented by one improved SDS is listed below.

/* Finding the intersection of two images represented by BTs, one discards the topmost d_1 levels and the

other discards the topmost d_2 levels. */
If $d_1 \neq d_2$
{
Recover $|d_1 - d_2|$ levels to the improved SDS with $d = max(d_1, d_2)$;
}
For $i = 1$ to $2^{min(d_1,d_2)}$
Find the intersection of the corresponding i–th sub–BTs;
Next i

The intersection of two images represented by sub–BTs can be found using the algorithms described in [3, 14]. The number of arithmetic operations in the intersection of the corresponding sub–BTs is proportional to the total number of black leaf nodes in the two sub–BTs, so the time complexity for the intersection of two images is proportional to the total number of the black nodes in the two given BTs.

The union of two images can be found in a similar way. We first find the union of corresponding subimages. Then, we check whether there are two neighboring subimages, that are both whole black, if so, they can form a larger black block. The related procedure is listed below.

If $d_1 \neq d_2$
{
Recover $|d_1 - d_2|$ levels to the improved BT with less d;
}
For $i = 1$ to $2^{min(d_1,d_2)}$
Find the union of the corresponding sub–BTs;
Next i
Check each of the resulting $2^{min(d_1,d_2)}$ sub–BTs to find the whole black ones and combine them if possible.

To find the complement of an image, we just find the complement of all the subimages separately. The algorithm for finding the complement of a subimage represented by BCs is presented in [3]. Note that for the DF–expression, we need only change each symbol '0' to a '1', and change each for symbol '1' to a '0'. The algorithm is listed below.

For $i = 1$ to 2^d /* d denotes the number of levels been discarded */
Find the complement of the sub–BTs;
Next i

6 CONCLUSIONS

We have presented five improved SDSs to represent binary images. Experimental results show that the proposed methods have better compression performance when compared to the existing five well–known SDSs. In addition, some image manipulations on the proposed improved SDSs have been developed.

It is an interesting research issue to apply our method to compress similar images [7] and to extend our method to color/gray images [13]. In addition, it is also an interesting research issue to analyze the savings of our new representation by using the analytical model [12].

ACKNOWLEDGEMENT

The authors would like to thank the two refrees and Prof. M. A. Cobb for their valuable comments.

References

[1] I. Gargantini, An effective way to represent quadtrees, *Comm. ACM*, **25** (12), 905-910 (1982).

[2] W. D. Jonge, P. Scheuermann, and A. Schijf, S^+–trees: an efficient structure for the representation of large pictures, *CVGIP: Image Understanding*, **59**, 265–280 (1994).

[3] C. Y. Huang and K. L. Chung, Fast operations on binary images using interpolation–based bintrees, *Pattern Recognition*, **28** (3) 1995, pp. 409–420.

[4] G. M. Hunter and K. Steiglitz, Operations on images using quad trees, *IEEE Trans. on Pattern Analysis and Machine Intelligence*, **1** (2), 145–153 (1979).

[5] K. Knowlton, Progressive transmission of gray-scale and binary pictures by simple, efficient, and lossless encoding schemes, *Proc. IEEE*, **68** (1980) 885–896.

[6] E. Kawaguchi and T. Endo, On a method of binary–picture representation and its application to data compression, *IEEE Trans. on Pattern Analysis and Machine Intelligence*, **2** (1), 27–35 (1980).

[7] T. W. Lin, Compressed quadtree representation for storing similar images, *Image and Vision Computing*, **15** (11), 833–843 (1997).

[8] M. A. Ouksel and A. Yaagoub, The interpolation–based bintree and encoding of binary images, *CVGIP: Graphical Models and Image Processing*, **54** (1), 75–81 (1992).

[9] H. Samet, *The Design and Analysis of Spatial Data Structures*, Addison Wesley, New York, 1990.

[10] C. A. Shaffer, R. Juvvadi and L. S. Health, Generalized comparison of quadtree and bintree storage requirements, *Image and Vision Computing*, **11** (7), 402–412 (1993).

[11] M. Tamminen, Encoding pixel trees, *Computer Vision, Graphics, and Image Processing*, **28** (1), 44–57 (1984).

[12] M. Vassilakopoulos and Y. Manolopoulos, Analytical comparison of two spatial data structures, *Information Systems*, **19** (7), 569–582 (1994).

[13] J. R. Woodwark, Compressed quadtrees, *The Computer Journal*, **27** (3), 225–229 (1984).

[14] J. G. Wu and K. L. Chung, A New Binary Image Representation: Logicodes, *J. of Visual Communication and Image Representation*, **8** (3), 291–298 (1997).

Characterization Results for the Poset Based Representation of Topological Relations – I: Introduction and Models

Luca Forlizzi[1] and Enrico Nardelli[1,2]
[1] Dipartimento di Matematica Pura ed Applicata, Univ. of L'Aquila,
Via Vetoio, Coppito, I-67010 L'Aquila, Italia.
E-mail: {forlizzi,nardelli}@univaq.it

[2] Istituto di Analisi dei Sistemi ed Informatica,
Consiglio Nazionale delle Ricerche, Viale Manzoni 30, I-00185 Roma, Italia.

Keywords: Multimedia databases, auditory databases, geographic information systems, field-oriented approach, field indexing, continuous indexing

Edited by: Frederick E. Petry, Maria A. Cobb and Kevin B. Shaw
Received: December 24, 1998 **Revised:** May 25, 1999 **Accepted:** June 10, 1999

Formal methods based on the mathematical theory of partially ordered sets (i.e., posets) have been used for the description of topological relations among spatial objects since many years.
In particular, the use of the lattice completion (or normal completion) of a poset modeling a set of spatial objects has been shown by Kainz, Egenhofer and Greasley to be a fundamental technique to build meaningful representations for topological relations.
In this paper and in the companion one [9] we discuss the expressive power of the lattice completion as a formal model for a set of spatial objects, by proving sufficient and necessary conditions for its use to give a correct representation of intersection and union relations among spatial objects.
We also show how to use lattice completion when working on a subset (i.e., a view) of the set of spatial objects so that the computation only considers objects relevant to the view itself.

1 Introduction

Spatial databases for applications in which is only important to represent knowledge about the relative positions of spatial objects while precise data about their absolute position can be discarded, e.g. a railway network for travelers, can be suitably modeled by the so-called *topological data model.* In such a model two spatial objects are considered the same if there exists an isotopy that transform one of them into the other.

In this work[1] we therefore deal only with topological spatial information associated to a collection of spatial objects, ignoring both non-topological spatial information and non-spatial information of the objects. For this reason from now on we denote a collection of spatial objects with the more abstract term of *class of sets* and use a set-theoretical terminology.

A class of sets together with a set-containment relation among them models many common situations in spatial databases. For example it may represent a containment relation between geographical objects of the plane or a hierarchical relation between administrative units.

[1]Research partially supported by the European Union TMR project "Chorochronos".

With this modeling approach many natural operations in the modeled reality correspond to operations of set-union and set-intersection among elements of the representation.

For example, the overlapping among the spatial regions where rain is heavy and the spatial regions where slope is high is a natural operation for building a map of potentially dangerous zones from a geological point of view. It directly corresponds to set-intersection in the representation by means of a class of sets with a set-containment relation.

Concerning operations on the reality of interest that can be mapped to set-union in the representation, examples are collecting all the countries belonging to the same state, which corresponds to identifying the state itself.

The topological data model was introduced in 1979 by Corbett, and called PLA data model, in the context of modeling data for the Census Bureau of the United States [5]. A database theory formalization was given by Paredaens and co-workers [26, 24, 23] in terms of the concepts of a finite number of points, continuous curves between these points, and areas formed by these curves in the real plane. Their formalization has a planarity constraint, i.e. curves may only intersect in

their endpoints.

Another formalization of the PLA-model using combinatorial topology was given by Worboys [33]. He used the topological notions of 0-, 1-, 2-simplex and simplicial 2-complex in the real plane. A 0-simplex is a point, a 1-simplex is a segment line, and a 2-simplex is a triangle. A simplicial 2-complex is a collection of 0-, 1-, and 2-simplices with the constraint that the intersection of any two simplices is either empty or is a face of both simplices. Also in this formalization a planarity constraint is assumed.

Using the combinatorial topology formalization, Kainz *et al.* [22] discussed how to use partially ordered sets (i.e. *posets*) as a representation structure for spatial data in the topological data model. They showed how the use of a poset operator called *lattice completion* (or *normal completion*) allows to model the important topological relations of containment, adjacency, and connectivity in terms of the set relations of intersection, union and containment and of the closure of these. A more extended treatment of these aspects is given in Sect. 2. Discussions on the use of posets and lattices to represent spatial relations and their connections with topological models are also provided by Saalfeld [28] and Kainz [19, 20, 21].

Lattice completion transforms a given poset in a *lattice* by adding to the poset new objects and relations so that, informally speaking, every set of objects has unique representatives for their intersection and union, while maintaining order relations.

As an example of the use of lattice representation to compute topological relations, consider the instance of the topological data model in Fig. 1(left) and its (transitively reduced and with top and bottom elements omitted for clarity) lattice representation in Fig. 1(right). You can now detect, as an example, that: (1) regions D and E intersect in the plane since they have as intersection in the lattice representation A (a region); (2) regions A and B are adjacent in the plane since they have as intersection in the lattice γ (a line); (3) lines β and ϵ are not connected by a common point since their intersection in the lattice representation is empty.

The importance of a poset-based representation and lattice completion is twofold [1, 2, 12, 16, 17, 31, 32]: on one side they give a formal basis for representing and reasoning about spatial relations, on the other side very efficient data structures can be defined to represent and manipulate them. Also, a poset based representation can be used as a formalization device for spatial query languages [8].

Using the combinatorial topology formalization, Kainz *et al.* [22] showed that the lattice completion of a poset modeling a set of spatial objects is a fundamental technique to build meaningful representation for topological relations.

In fact they proved that the new elements introduced by the normal completion process can (and have to) be interpreted as being the intersection of spatial objects. This is fundamental, from a mathematical point of view, since it means that the lattice resulting from the normal completion is the closure of the given set of spatial objects with respect to the intersection operation.

This result, however, leaves it open the question of the closure of the set of spatial objects with respect to the other fundamental operator to manipulate spatial entities, namely the union operator.

In this paper we precisely clarify the limitations for the use of a lattice as a formal model for a set of spatial objects, by proving sufficient and necessary conditions.

In the companion paper [9] we show that a technique already known in lattice theory, namely the construction of the maximal antichain lattice of a given poset, can be used to define another completion operator that builds the closure of the given set of spatial objects with respect to the union operation.

We also show that this new completion operator commutes with the normal completion and that the lattice obtained from the application of both completion operators is minimal and unique up to isomorphism.

Finally we show how to apply these operators when working on a subset (i.e., a *view*) of the set of spatial objects so that the computation only considers objects relevant to the view itself.

Our results give further theoretical motivations to the use of lattices built on simplicial complexes as a model for spatial objects and topological relations, since this kind of lattices are, by construction, closed with respect to both the union and the intersection operations.

The structure of the two papers is the following. In Sect. 2 we review the poset-based representation for the topological data model, examine problems arising from a naive extension to the most general case, and shortly describe how we tackle them. This section is a synthesis of all the results of the two papers. Section 3 introduces definitions related to posets and lattices and some basic facts about them. More advanced results are recalled in Appendix A. In Sect. 4 we introduce formally the definition of closure of a class S of objects with a set-containment relation with respect to a certain set operator, of representation and of universal partition. Advanced technical details are contained in Appendix B. The companion paper [9] is dedicated to the study of the representation of the closure of a class with respect to set-intersection and set-union operators. It also contains conclusions and final remarks.

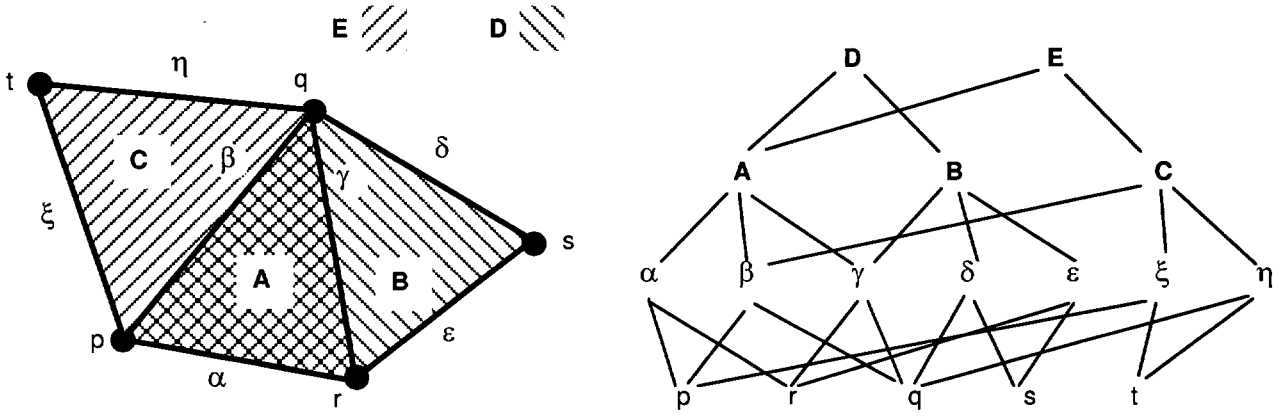


Figure 1: (left) A class S of spatial objects in the topological data model. (right) A lattice representation for this class.

2 A Poset Representation for the Topological Data Model

In this section we discuss in more details the use of posets to represent a collection of spatial objects in the topological data model approach and highlight its shortcomings.

Note that since we focus on set-union and set-intersection operations on spatial regions, it is not necessary to explicitly consider the lowest two levels of the poset representation of the topological data model, see Fig. 1 (right), i.e. the levels modeling curves and points.

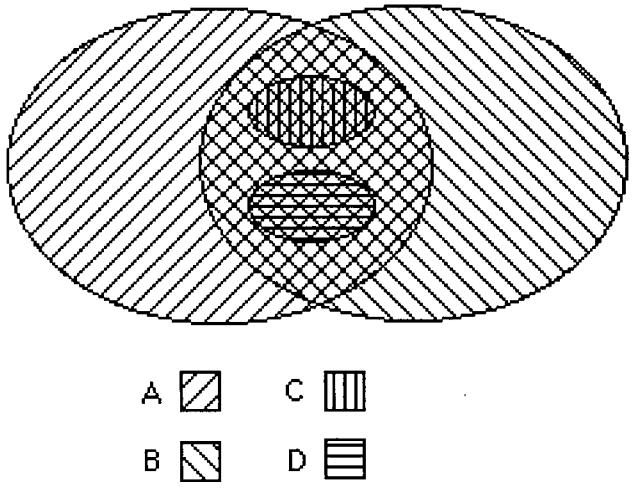


Figure 2: A class of spatial objects in the topological data model

A poset representation is not always rich enough to represent the closure of set-intersection and set-union on the sets of the class. Kainz *et al.* proposed to transform the poset representing a class of spatial sets in the topological data model in a lattice. This transformation can be obtained by means of a well-known operator of poset theory, namely the lattice completion (or normal completion) [4, 6]. Kainz *et al.* suggested to represent the result of set-intersection between spatial sets by means of the new elements added by the lattice completion operator . In such a way it is possible to compute topological properties via set-intersection and set-union on the lattice representation.

Consider for example the class of sets S containing the four sets A, B, C, and D shown in Fig. 2. Each set is represented by means of a different filling pattern. Zones filled with more than one pattern belong to more than one set. We can represent the class S with the poset P shown in Fig. 3(left). Note that elements of P have the same labels of the sets they represent.

2.1 Representing Set-Intersection Closure

Now suppose we want a representation of the closure $S^{\cap}$ of the class S with respect to the set-intersection operator (i.e. the class obtained intersecting each possible pair of sets taken from S). The only element of the closure that is different from sets in S and from the empty set is $A \cap B$. The set $A \cap B$ is contained in the sets A and B and it contains the sets C and D. The normal completion of poset P is the lattice L, shown in Fig. 3(right). We always represent posets and lattices by a drawing showing their transitive reduction. The lattice L is composed by the elements of P plus a top (T) and a bottom ($\emptyset$) element, and a new element labeled X, which is smaller than the elements (representing the sets) A and B and greater than those (representing the sets) C and D. Therefore, since the

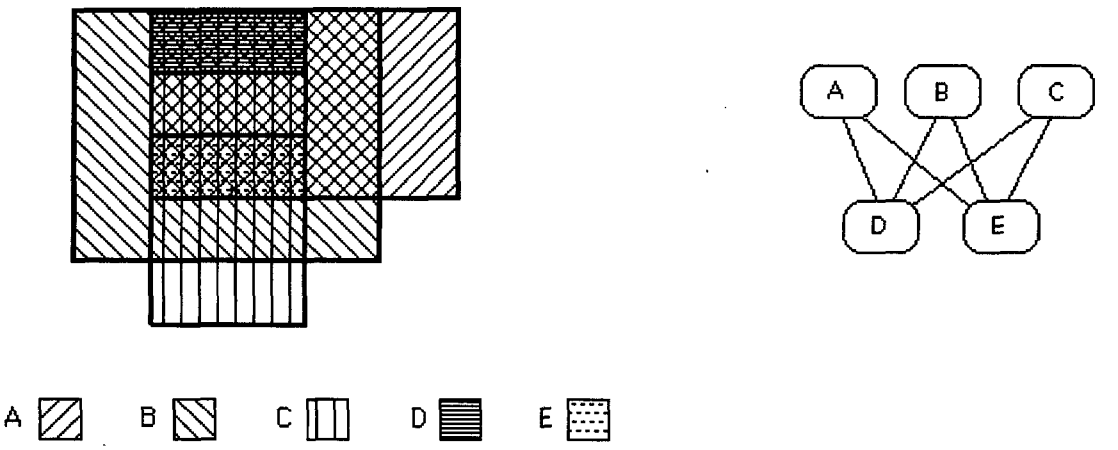


Figure 4: (left) A class S of spatial objects in the topological data model. (right) A poset representation P for this class.

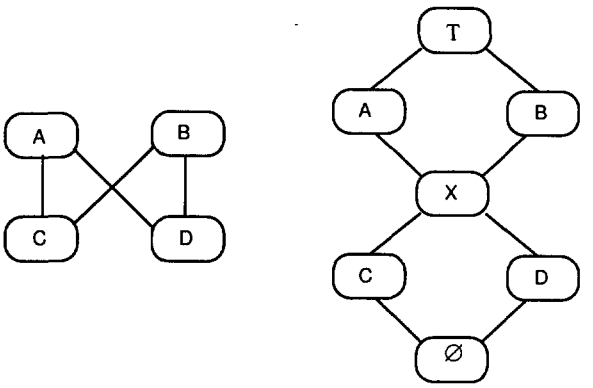


Figure 3: (left) A poset representation of the class S of Fig. 2. (right) The normal completion of the poset to the left.

relation of set $A \cap B$ with respect to other sets of S is analogous to that of the element X with respect to other elements of L, the lattice L can represent the class of sets $S^{\cap}$, provided that the element X represents the set $A \cap B$. This was the proposal of Kainz *et al.* [22].

In the general case, however, using the normal completion operator to represent the set-intersection operator, may lead to wrong results [10, 11]. Consider the example shown in Fig. 4(left) containing the five sets A, B, C, D, and E. Its poset representation is shown in Fig. 4(right), where elements of P have the same labels of the sets they represent.

Suppose we want a representation of $S^{\cap}$. The only elements of the closure that are different from sets in S and from the empty set are the sets $A \cap B$, $B \cap C$, $A \cap C$, and $A \cap B \cap C$. The normal completion of poset P is the lattice L, shown in Fig. 5(left). The lattice L is composed by the elements of P plus a top (T) and a bottom ($\emptyset$) element, and one new element labeled X. But lattice L cannot represent $S^{\cap}$, since a correct representation of $S^{\cap}$, shown in Fig. 5(right), has three new elements, labeled F, G and H. Element F represents $A \cap B$, element G represent $B \cap C$, while element H represent two coincident sets of $S^{\cap}$, namely $A \cap B \cap C$ and $A \cap C$.

This example shows that to represent set-intersection operator by means of a poset operator we have to provide more information to our representation. A way to do this is to include in the class S a partition of the whole domain on which S is defined, that we call *universal partition* of S. Note that a basic reference layer analogous to our universal partition, is commonly used in the modeling of geometrical entities [15, 7]. Such a basic reference layer is indeed the starting point for many efficient data structures based on a space-partitioning criteria, e.g. quadtree [29], grid-file [25], k-d tree [3], cell-tree [14].

Also note that in [7] it is shown that the mathematical definition of a partition and the spatial one disagree. In fact, since geometrical objects are closed sets in the spatial view, their boundaries can be common to more than one object. This is not possible in the mathematical view of a partition. But this does not affect our work, since if we consider the extended poset model of a class of spatial object, where also the two lowest layers are present (see Fig. 1(right)), we note that such a model represents also the spatial view of geometrical object, where common intersections exist.

In a poset representing a class that has a universal partition there is a representative element for each of the element of the partition of the domain of S. In Fig. 6 we show a class S of sets containing five sets A, B, C, D, and E which have exactly the same containment relations as the regions in Fig. 4. But the

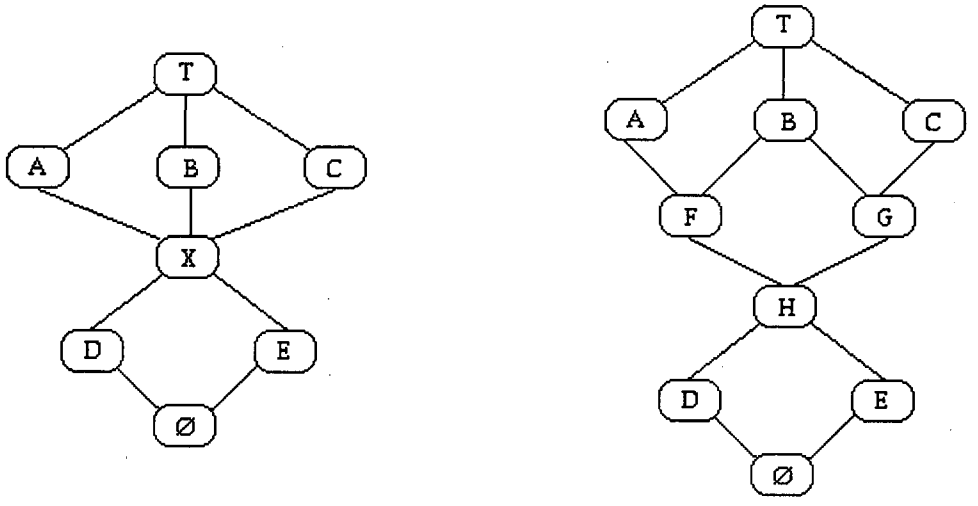


Figure 5: (left) The lattice completion of poset P representing class S of Fig. 4. (right) A poset representation of the set-intersection closure for class S.

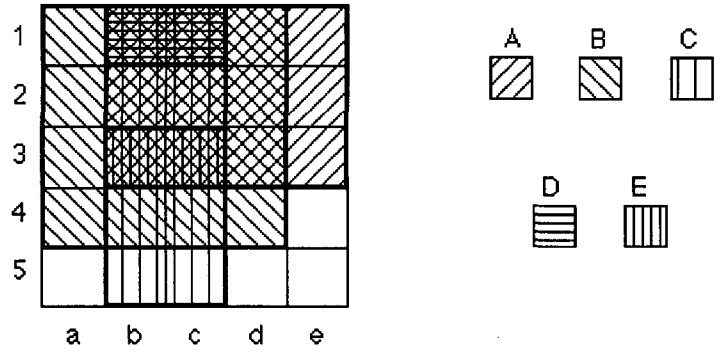


Figure 6: A class S of spatial objects with a universal partition

class also contains a universal partition, whose elements coincide with the unit squares of the grid, denoted $1a, 1b, \ldots, 5e$. Sets A, B, C, D, and E are distinguished by means of different filling patterns.

A poset representation P for this class of sets is shown in Fig. 7. We want to construct a representation of $S^{\cap}$, namely a representation which contains also elements that represent sets $A \cap B$, $B \cap C$ and $A \cap B \cap C$. Comparing P with the poset in Fig. 4(right) we can see that the universal partition provides information on the class S that were missing in the poset in Fig. 4(right).

For example elements $1d, 2d$, and $3d$ represent elementary areas contained in both sets A and B but not in set C. This fact means that $A \cap B$ and $A \cap B \cap C$ are different sets. Figure 8 shows the normal completion $M(P)$ of poset P (in Fig. 8 also, the top and the bottom of the lattice completion have been omitted for clarity). Inspecting Fig. 8 (and recalling Fig. 5(right)) we can see that $M(P)$ is a correct representation of class $S^{\cap}$, since elements labeled X, Y and Z represent respectively sets $A \cap B$, $B \cap C$ and $A \cap B \cap C$.

In the companion paper [9] we formally prove that the existence of a partition of the domain on which a class of sets is defined is a sufficient but not – in general – necessary condition for the normal completion of the representation of the class to be a correct representation of the closure of the class with respect to the set-intersection operator. In the same section we also give necessary conditions for a correct representation of the closure of the class with respect to the set-intersection operator by means of the normal completion. Finally, we show that for classes of sets satisfying a reasonable assumption (informally, that the union of the smallest elements in the class is equal to the union of the greatest ones), the existence of a partition of the domain on which a class of sets is defined is a necessary and sufficient condition to this aim.

2.2 Representing Set-Union Closure

We also study the problem of representing the closure of a class of sets with respect to the set-union operator. To represent the closure with respect to the union op-

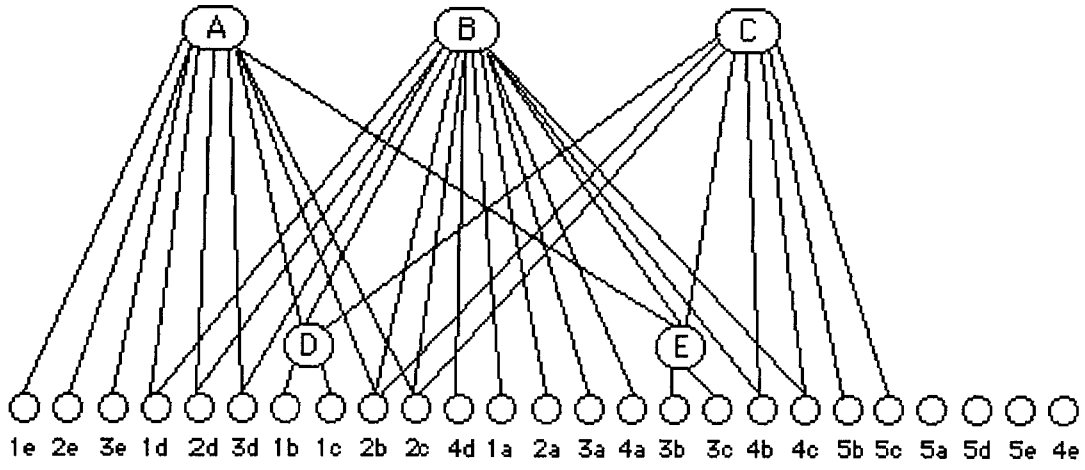


Figure 7: A poset representation of the class S of spatial objects of Fig. 6

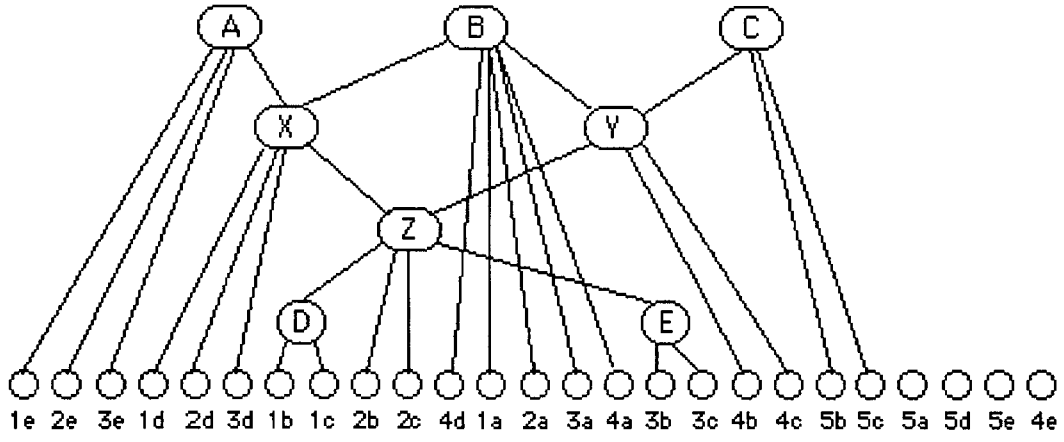


Figure 8: The normal completion of the poset in Fig. 7

erator of a class of sets S, we need a new poset operator since lattice completion does not provide a correct answer. The following example illustrates the problem. Let us consider the class of sets S in Fig. 9. The poset representation P for S is shown in Fig. 10(left) where elements of P have the same labels of the sets they represent.

Suppose now we want to introduce in P an element, denoted $C \cup D$, to represent the union of sets C and D. The union of C and D is the smaller set containing both C and D. Hence the representative of $C \cup D$ should be the least upper bound in P of C and D. We could provide such a least upper bound extending the poset in Fig. 10(left) to a lattice by means of the normal completion. In this way we obtained the lattice in Fig. 10(center). The newly created element X would be the desired least upper bound of C and D. However X would also be greater than E in the poset. Hence X cannot represent the union of C and D since such a union does not contain E.

So to represent $C \cup D$ we need to provide a least upper bound to representatives of C and D by means of a different poset-operator. More exactly we need a poset-operator that applied to the poset in Fig. 10(left) provides a least upper bound for C and D that is greater than only the representatives of sets contained in $C \cup D$. The correct representation is the lattice shown in Fig. 10(right), where element X is the representative of $C \cup D$.

For this purpose we introduce a poset operator, called by us *U-completion*, that builds the quotient lattice modulo a certain congruence relation of the antichain lattice of P. We give necessary and sufficient conditions for the U-completion to represent $S^{\cup}$, the closure of a class of spatial sets S with respect to the set-union operator.

We address also the issue of the simultaneous application of the lattice completion and U-completion opera-

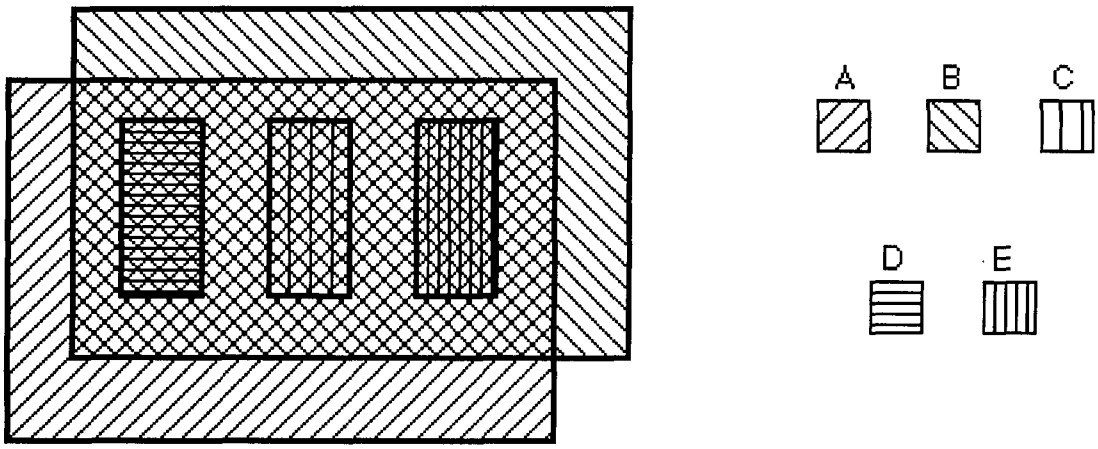


Figure 9: A class S of spatial objects in the topological data model.

tors. We prove that U-completion and lattice completion can be applied in whatever order, always producing the same outcome. Namely, we show that for any class S, $(S^{\cup})^{\cap} = S^{\cap\cup}$ and $(S^{\cap})^{\cup} = S^{\cap\cup}$ and that the lattice obtained by the application of both completion operations is minimal and unique up to isomorphism. Hence, starting from a representation of class S, we can apply in in whatever order the normal completion and the U-completion obtaining a representation of $S^{\cap\cup}$. We finally show that the existence of a universal partition is a necessary and sufficient condition for the simultaneous application of the U-completion and the lattice completion to represent $S^{\cap\cup}$.

2.3 Working on a View

Note that one is often interested in applying the intersection and union operator only to a subclass T of a given class S of spatial sets, for example when a view to operate on a subclass T of S has been defined in the spatial database. In such a case it is not convenient to build the normal completion or the U-completion of the whole representation of the class, since it is likely that it contains much more elements than the ones we are interested to. Consider that, in general, the completion of a given poset P has in the worst case a size which is exponential with respect to the size of P. The naive solution of building only completions of a poset representation of the subclass T has the disadvantage of producing poset representations for closures of T of that are disjoint from the representation of S. We give in the companion paper [9] necessary and sufficient conditions so that the poset representation of S and those of closures of subclasses can be joined together into a correct poset representation of the whole reality of interest.

As an example, let us consider the class S of sets in Fig. 11, where S contains also the elements of a universal partition. Namely, the elements of the universal partition are the small unit squares and are denoted using a matricial notation $1a, 1b, \ldots, 4e, 4f$. The remaining sets are each distinguished by a filling pattern. Areas filled with more than one pattern are contained in more than one set.

Suppose now we are interested in building a representation of the closure with respect to both set-union and set-intersection operators for the view consisting only of elements $\{A, B, C, D\}$.

In Fig. 12(left) we can see a poset representation of S, where representatives of elements of the universal partition not contained in any other set have been omitted for clarity. We apply normal completion and U-completion to the subposet representing the view $\{A, B, C, D\}$ and we merge the obtained lattice with the original poset obtaining the desired result, shown in Fig. 12(right), where X represents $A \cap B$, Y is the representative of $C \cup D$, and the top and bottom elements have been omitted for clarity.

3 Poset Basics

In this section we recall definitions regarding posets and lattices and some basic facts about them. It can be skipped by a reader with some knowledge of the field. More advanced results are recalled in Appendix A.

Definition 3.1 *A partial order relation $\Re$ is a binary relation which is reflexive ($a\Re a$), antisymmetric ($a\Re b$ and $b\Re a$ implies $a = b$) and transitive ($a\Re b$ and $b\Re c$ implies $a\Re c$).*

Definition 3.2 *A partially ordered set, i.e. a poset $\langle P, \leq \rangle$ is an algebraic structure formed by a set P and a partial order relation (normally indicated with the symbol $\leq$), among the elements of P.*

Posets are well known mathematical structures. This fact allows us to lean on a well founded theory,

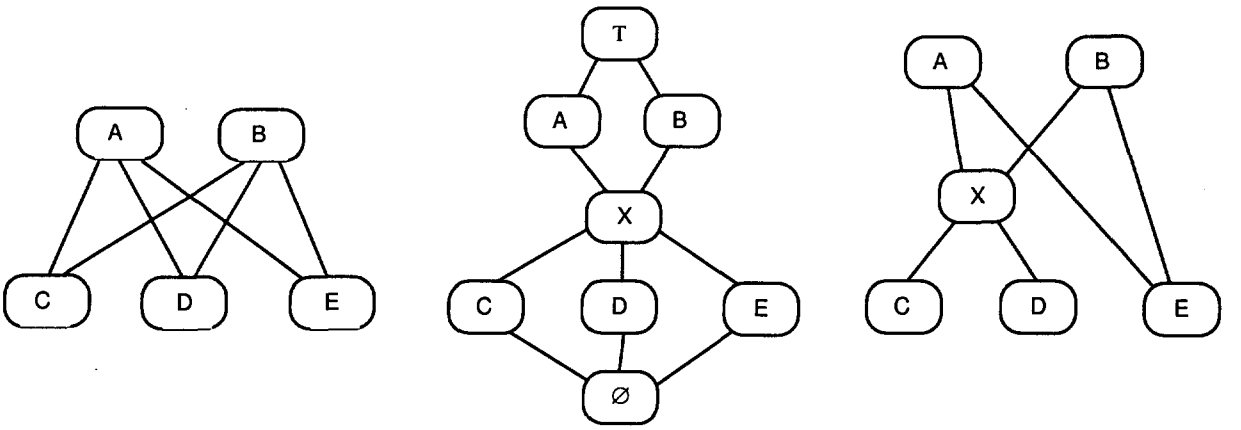


Figure 10: (left) A poset representation P of the class S of sets of Fig. 9. (center) The lattice completion of poset P. (right) The correct lattice representation of class S plus the set $C \cup D$.

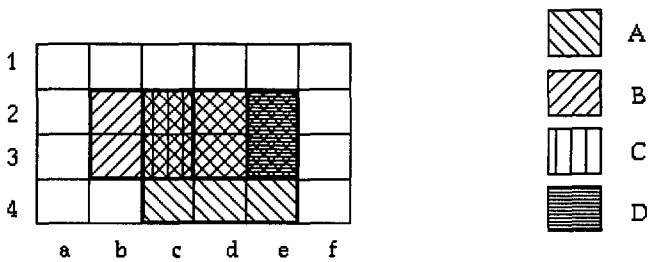


Figure 11: A Class S of sets containing a universal partition.

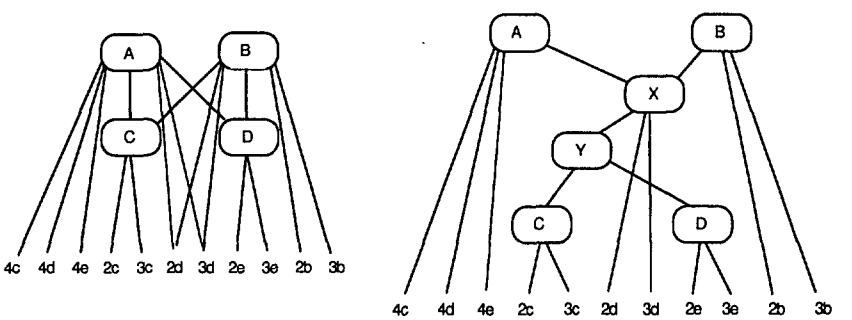


Figure 12: (left) A poset representation of the class S of sets of Fig. 11. (right) A representation of the closure with respect to both set-union and set-intersection operators of the class S.

covered by an extensive amount of literature. We signal for the interested reader Birkhoff's and Gratzer's classic books [4, 13].

Finite posets can be graphically represented by a Hasse diagram. In a Hasse diagram, each poset's element is represented by a dot; also, given $A, B \in P$ such that $A \leq B$, and for no $C \in P$ we have $A < C < B$, then the dot that represents B is drawn in a position higher than those of the dot that represents the element A, and the two dots are connected by a line. Note that the Hasse diagram of the poset $\langle P, \leq \rangle$ is a representation of the transitive reduction of the relation $\leq$.

Definition 3.3 *Let $\langle P, \leq \rangle$ be a poset and let $Q \subseteq P$. Then:*

1. *$a \in Q$ is a* maximal *element of Q if $\forall y \in Q$, $a \leq y$ implies $a = y$;*
2. *$a \in Q$ is the* greatest *(or* maximum*) element of Q if $\forall y \in Q$, $y \leq a$;*
3. *$a \in Q$ is a* minimal *element of Q if $\forall y \in Q$, $y \leq a$ implies $a = y$;*
4. *$a \in Q$ is the* least *(or* minimum*) element of Q if $\forall y \in Q$, $a \leq y$.*

The following definition introduces some important families of sets associates to a poset.

Definition 3.4 *Given a poset $\langle P, \leq \rangle$ and $Q \subseteq P$, we define:*

1. *$Q^* = \{x \in P \mid \forall y \in Q, y \leq x\}$ (*upper-star ideal*);*
2. *$Q_* = \{x \in P \mid \forall y \in Q, x \leq y\}$ (*lower-star ideal*);*
3. *$\uparrow Q = \{x \in P \mid \exists y \in Q, y \leq x\}$ (*up Q*);*
4. *$\downarrow Q = \{x \in P \mid \exists y \in Q, x \leq y\}$ (*down Q*);*
5. *$Q^o = \{x \in Q \mid x$ is a maximal element of $Q\}$ (set of the maximal elements of Q or* maxima *of Q);*
6. *$Q_o = \{x \in Q \mid x$ is a minimal element of $Q\}$ (set of the minimal elements of Q or* minima *of Q);*

Also for each $p \in P$ we define:

1. *$\uparrow p = \{x \in P \mid p \leq x\}$;*
2. *$\downarrow p = \{x \in P \mid x \leq p\}$;*

Note that for each $p \in P$, trivially $\uparrow p = \uparrow \{p\} = \{p\}^*$ and $\downarrow p = \downarrow \{p\} = \{p\}_*$.

Definition 3.5 *Given a poset $\langle P, \leq \rangle$ and $Q \subseteq P$, if there exist $u \in P$ such that u is the least element of Q^*, we call u* least upper bound (lub) *of Q and write $u = lub_P(Q)$. Analogously, if there exist $l \in P$ such that l is the greatest element of Q_*, we call l* greatest lower bound (glb) *of Q and write $l = glb_P(Q)$.*

Note that by definition the *glb* and the *lub* are unique, when they exist. In the rest of the paper, given a poset $\langle P, \leq \rangle$ and $Q \subseteq P$, we write $lub(Q)$ instead of $lub_P(Q)$ and $glb(Q)$ instead of $glb_P(Q)$, every time no ambiguity is possible. Also, given $x, y \in P$ we simply write $lub(x, y)$ and $glb(x, y)$ to denote respectively $lub(\{x, y\})$ and $glb(\{x, y\})$.

A *lattice* is a poset such that for every pair of elements a least upper bound and a greatest lower bound exist.

Definition 3.6 *Given a poset $\langle P, \leq \rangle$ if $lub(x, y)$ and $glb(x, y)$ exist $\forall x, y \in P$ then P is called a lattice.*

The next Definition deals with mappings from a poset to another poset.

Definition 3.7 *Let $\langle P, \leq \rangle$ and $\langle Q, \leq' \rangle$ be posets. A map $F : P \mapsto Q$ is said to be*

1. order preserving *if $x \leq y$ in P implies $F(x) \leq' F(y)$ in Q;*
2. *an* order embedding *if $x \leq y$ in P if and only if $F(x) \leq' F(y)$ in Q;*
3. *an* order isomorphism *if it is an order embedding surjective mapping.*

Given a generic poset, we are often interested in extending it to a lattice which preserves the ordering relation. We call such a lattice the *completion* of the given poset.

Definition 3.8 *Given a poset $\langle P, \leq \rangle$, and a lattice $\langle L, \leq' \rangle$ we say that L is a* completion *of P if there exists an order embedding from P to L.*

Among the completions of a given poset, there is a particularly interesting one, called Normal Completion or MacNeille Completion [6].

Definition 3.9 *Given a poset $\langle P, \leq \rangle$, we define its MacNeille completion as the poset $\langle M(P), \subseteq \rangle$ where $M(P) = \{Q \mid Q \subseteq P$ and $Q = (Q^*)_*\}$.*

It is well known that $\langle M(P), \subseteq \rangle$ is a completion of $\langle P, \leq \rangle$ and that it has the additional property of preserving all least upper bounds and greatest lower bounds that exist in $\langle P, \leq \rangle$ [6]. The following definition provides a canonical order embedding between the posets $\langle P, \leq \rangle$ and $\langle M(P), \subseteq \rangle$.

Definition 3.10 *Let $\langle P, \leq \rangle$ be a poset and let $\langle M(P), \subseteq \rangle$ be its MacNeille completion. The canonical order embedding $\varphi : P \mapsto M(P)$ between a posets and its MacNeille completion is defined as $\varphi(x) = \downarrow x$.*

The following definition introduces a useful property of the MacNeille completion.

Definition 3.11 *Let P be a poset and let $Q \subseteq P$. Then Q is called* join-dense *in P if for every element $s \in P$ there is a subset A of Q such that $s = lub(A)$ Analogously, Q is called* meet-dense *in P if for every element $s \in P$ there is a subset A of Q such that $s = glb(A)$. To provide a compact notation, for all $x \in P$, we define $(\uparrow x)_Q = \{y \in Q \mid x \leq y\}$ and $(\downarrow x)_Q = \{y \in Q \mid y \leq x\}$.*

4 Representations and Closures

In this section we define formally what we mean by closure of a class of sets with respect to a certain set operator, and what we mean by representation of a class of sets with a set-containment relation by means of a poset. We also introduce in this section the concept of universal partition of a class S with a set-containment relation. It will be used later as a tool to operate efficiently on sets belonging to S and on sets belonging to closures of S.

We consider only finite classes, i.e. classes containing a finite number of sets. For technical reasons it is useful to work with classes of sets with a set-containment relation that contain a greatest set (namely a set that contains every other set of the class) and a least set (namely a set that is contained in every other set of the class). This is not a restriction since if a finite class of sets has not a greatest or a least set, we can always expand it adding respectively the set-union of all the sets of the class or the empty set, and then work with the expanded class. From now on, when we speak of a class of sets with a set-containment relation, we always refer to the expanded class. All results proved in this section are almost straightforward. Their proofs can be found for completeness in Appendix B (p.236).

We first define the closure of a class of sets with respect to set-union and set-intersection operators.

Definition 4.1 *Let S be a class of sets with a set-containment relation. We define $S^{\cap}$, the* closure of S with respect to the set-intersection operator, *by the following rules:*

1. *if $s \in S$ then $s \in S^{\cap}$;*
2. *$\forall s_1, s_2 \in S^{\cap}$, $s_1 \cap s_2 \in S^{\cap}$.*

Definition 4.2 *Let S be a class of sets with a set-containment relation. We define $S^{\cup}$, the* closure of S with respect to the set-union operator, *by the following rules:*

1. *if $s \in S$ then $s \in S^{\cup}$;*
2. *$\forall s_1, s_2 \in S^{\cup}$, $s_1 \cup s_2 \in S^{\cup}$.*

Definition 4.3 *Let S be a class of sets with a set-containment relation. We define $S^{\cap\cup}$, the* closure of S with respect to both set-union and set-intersection operators, *by the following rules:*

1. *if $s \in S$ then $s \in S^{\cap\cup}$;*
2. *$\forall s_1, s_2 \in S^{\cap\cup}$, $s_1 \cap s_2 \in S^{\cap\cup}$, $s_1 \cup s_2 \in S^{\cap\cup}$.*

Obviously, we have $S^{\cap} \subseteq S^{\cap\cup}$ and $S^{\cup} \subseteq S^{\cap\cup}$. The following theorem shows the relation existing among the closure operations introduced by the above definitions.

Theorem 4.1 *We have $(S^{\cup})^{\cap} = S^{\cap\cup}$ and $(S^{\cap})^{\cup} = S^{\cap\cup}$.*

A key role is played by elements in the lowest layer.

Definition 4.4 *We define* base *of S (and we denote it as B_S) the set of minimal elements of $S \setminus \{\emptyset\}$.*

Note that since the base is defined without considering the empty set, then when this set is part of the considered class, either directly or because of its expansion, we usually assume definitions and properties do not refer to it. To avoid an heavy presentation, we usually do no explicit how definition and properties have to be extended to the empty set, unless we feel this extension is hard to figure out for the reader on the basis of the current context.

We introduce a mapping to relate each set of a class with sets of the base containing it. The powerset of set I is denoted by 2^I.

Definition 4.5 *Let S be a class of sets with a set-containment relation. We define the mapping $S_{\text{Base}} : S \mapsto 2^{B_S}$ as*

$$S_{\text{Base}}(s) = \{r \in B_S | r \subseteq s\} \ .$$

When the base of a class is a partition of the whole domain of S a lot of interesting properties turn out to be true. The following definition formalizes the concept of partition of the whole domain of S.

Definition 4.6 *We say that base B_S of a class of sets S is a* universal partition *of S if $\forall r_1, r_2 \in B_S$, we have $r_1 \cap r_2 = \emptyset$, and $\forall s \in S$ there exist $r_1, r_2 \ldots r_n \in B_S$ such that $s = \bigcup_i r_i$.*

The following lemma shows that for each set s of a class of sets with a set-containment relation, there exists a unique collection of sets of the universal partition whose set-union is equal to s, and that this collection is exactly $S_{\text{Base}}(s)$.

Lemma 1 *If S is a class of sets with a set-containment relation and a universal partition U_S, there exists a unique set $\{r_1, r_2 \ldots r_n\} \in 2^{U_S}$ such that $s = \bigcup_i r_i$. Also $\forall s \in S, s = \bigcup_{r \in S_{\text{Base}}(s)} r$.*

From the previous lemma the following corollary follows, which clarifies the links between S and its universal partition by showing that each set of S can be generated by a collection of sets of the universal partition.

Corollary 2 *The mapping $S_{\text{Base}}(.)$ is an order embedding from the poset $\langle S, \subseteq \rangle$ to the poset $\langle 2^{U_S}, \subseteq \rangle$.*

The universal partition U_S of a class S of sets with a set-containment relation is also a universal partition of $S^{\cap}$, $S^{\cup}$ and $S^{\cap\cup}$. We show in the following corollary this important property with respect to $S^{\cap\cup}$, leaving the analogous proofs with respect to $S^{\cap}$ and $S^{\cup}$ to the reader.

Corollary 3 *Let S be a class of sets with a set-containment relation and a universal partition U_S. Then U_S is a universal partition of $S^{\cap\cup}$.*

Thanks to the corollary above, we can apply Definition 4.6 and Corollary 2 also to $S^{\cap\cup}$.

The next two corollaries show important consequences of the existence of the universal partition that we will discuss in the companion paper [9].

Corollary 4 *Let S be a class of sets with a set containment relation and a universal partition U_S. Then $S^{\cup} = S^{\cap\cup}$.*

Corollary 5 *Let S be a class of sets with a set containment relation and a universal partition U_S. The mapping $S_{\text{Base}}(.)$ is an order isomorphism from the poset $\langle S^{\cap\cup}, \subseteq \rangle$ to the poset $\langle 2^{U_S}, \subseteq \rangle$.*

Now we define formally what a *representation* is by means of a poset of a class of sets with a set-containment relation.

Definition 4.7 *Let S be a class of sets with a set-containment relation and let $\langle P, \leq \rangle$ be a poset. We say that P is a representation of S if there exists an isomorphism between S and P.*

From now on, every time we deal with a representation P of a class S of sets with a set-containment relation, we refer to the isomorphism between S and P as $Rep : S \mapsto P$. Of course there exists $Rep^{-1} : P \mapsto S$. Note that since the classes of sets with a set-containment relation we consider have a greatest and a least set, their representations have a greatest and a least element. For any $s \in S$ we say that $Rep(s)$ is the *representative* of s in P.

Among the elements of P, representatives of set contained in B_S play a special role. Hence it is useful to introduce a notation for them.

Definition 4.8 *Let S be a class of sets with a set-containment relation and let P be a representation of S. We define* base *of P the set*

$$B_P = \{x \in P \mid x = Rep(r) \text{ and } r \in B_S\}.$$

If B_S is a universal partition of S, we say that B_P is a universal partition *of P and we denote it as U_P.*

We often need to relate each representative of a set s with the representatives of the sets of the base contained in s. Therefore we introduce the mapping $P_{\text{Base}}(\cdot)$ from elements in P to subsets of B_P.

Definition 4.9 *Let S be a class of sets with a set-containment relation and let P be a representation of S. For each $p \in P$, we define the mapping $P_{\text{Base}} : P \mapsto 2^{B_P}$ as*

$$P_{\text{Base}}(p) = \{x \in P \mid x = Rep(r), r \in S_{\text{Base}}(Rep^{-1}(p))\} .$$

In a class of sets with a set-containment relation and a universal partition, a set is 'composed' by sets of the universal partition by means of the set-union operator. In the representation of the class an analogous 'composition' is obtained by means of the $lub(.)$ operator that assign to each subset of a poset its least upper bound, as the following theorem shows.

Theorem 4.2 *Let S be a class of sets with a set-containment relation and a universal partition. If P is a representation of S then for each $s \in S$ we have:*

$$\begin{aligned} Rep(s) &= lub(\{y \mid y = Rep(r) \text{ and } r \in S_{\text{Base}}(s)\}) \\ &= lub(P_{\text{Base}}(Rep(s))) . \end{aligned}$$

If one thinks of $P_{\text{Base}}(.)$ as a mapping between the posets $\langle P, \leq \rangle$ and $\langle 2^{B_P}, \subseteq \rangle$, the previous theorem translates into the following corollary:

Corollary 6 *The mapping $P_{\text{Base}}(.)$ from the poset $\langle P, \leq \rangle$ to the poset $\langle 2^{B_P}, \subseteq \rangle$ is order preserving. Moreover if B_S is a universal partition, $P_{\text{Base}}(.)$ is an order embedding.*

References

[1] H. Aït-Kaci, R. Boyer, P. Lincoln and R. Nasr, "Efficient Implementation of Lattice Operations", ACM TOPLAS, 11(1):115-146, Jan 89.

[2] E. Apolloni, F. Arcieri, S. Ercoli, E. Nardelli, M. Talamo, "Un modello di riferimento per l'interazione con sistemi per la gestione di dati geografici", (in italian), Convegno Nazionale Sistemi Evoluti per Basi di Dati, Gizzeria Lido, Giugno 1993.

[3] J.L.Bentley, "Multidimensional binary search trees used for associate searching", Communications of ACM, 18, 509-517, 1975.

[4] G. Birkhoff, "Lattice Theory", American Mathematical Society Colloquium Publications Vol. 25, (Providence, RI: American Mathematical Society), 1967.

[5] J.P.Corbett, "Topological Principles in Cartography", Technical Paper 48, US Bureau of Census, Washington DC, 1979.

[6] B.A.Davey, H.A.Priestley, "Introduction to Lattices and Order", Cambridge University Press, 1991.

[7] M.Erwig, M.Schneider, "Partition and Conquer", in Spatial Information Theory: A Theoretical Basis for GIS, Vol. 1329 of LNCS, Springer Verlag 1997.

[8] L.Forlizzi, B.Kuijpers, E.Nardelli, "Region-based query language for spatial databases in the topological data model", manuscript.

[9] L.Forlizzi, E.Nardelli, "Characterization Results for the Poset Based Representation of Topological Relations – II: Intersection and Union", to appear on Informatica.

[10] L.Forlizzi, E.Nardelli, "On the use of posets as a formal model for spatial data", Technical Report 1/98, Dip. di Matematica, Univ. di L'Aquila, Feb 1998.

[11] L.Forlizzi, E.Nardelli, "Some Results on the Modelling of Spatial Data", 25th Annual Conference on Current Trends in Theory and Practice of Informatics (SOFSEM'98), Vol. 1521 of LNCS, Springer Verlag 1998.

[12] D.D.Ganguly, C.K.Mohan, S.Ranka, "A Space-and-Time Efficient Coding Algorithm for Lattice Computations", IEEE TKDE, 6(5):819-829, Oct. 1994.

[13] G.Gratzer, "General Lattice theory", (New York, NY: Academic press),1978.

[14] O.Günther, J.Bilmes, "Tree-based access methods for spatial databases: implementation and performance evaluation", IEEE TKDE, 3(3):342-356, 1991.

[15] R.H. Güting, M. Schneider, "Realm-based spatial data types: the ROSE algebra", VLDB Journal, 4(2), 213-289, April 1995.

[16] M.Habib, L.Nourine, "Bit-Vector Encoding for Partially Ordered Sets", Int. Workshop on Orders, Algorithms and Applications (ORDAL'94), Lyon, France, Jul.94, LNCS 831, V.Bouchitté and M.Morvan (Eds.).

[17] S.C.Hirtle, "Representational Structures for Cognitive Space: Trees, Ordered Trees and Semi-Lattices", in Spatial Information Theory: A Theoretical Basis for GIS, Vol. 988 of Lecture Notes in Computer Science, Springer Verlag 1995.

[18] C.Jard, G.V.Jourdan, J.X.Rampon, "Some online computation of the ideal lattice of posets", IRISA Research Report n.773, 1993.

[19] W.Kainz, "Application of Lattice Theory in Geography", 3rd Int. Symp. on Spatial Data Handling, 135-142, 1988.

[20] W.Kainz, "Order, Topology and Metric in GIS", ASPRS/ACSM Annual Convention, vol. 4, 154-160, 1989.

[21] W.Kainz, "Spatial Relationships - Topology Versus Order", 4th Int. Symp. on Spatial Data Handling, 814-819, 1990.

[22] W.Kainz, M.Egenhofer, I.Greasley, "Modelling spatial relations and operations with partially ordered sets", Int. J. of GIS, vol. 7, no. 3, 215-229., 1993.

[23] B.Kuijpers, J.Paredaens, J.Van den Bussche, "Lossless Representation of Topological Spatial Data", 4th Int. Symp. on Large Spatial Databases (SSD'95), LNCS 951, 1-13, 1995.

[24] B.Kuijpers, J.Paredaens, J.Vandeurzen, "Semantics in Spatial Databases", LNCS 1358, 1998.

[25] J.Nievergelt, H.Hinterberger, K.C.Sevcik, "The grid file: an adaptable symmetric multikey file structure", ACM TODS, 9(1):38-71, March 1984.

[26] J.Paredaens, "Spatial Databases, the Final Frontier", ICDT'95, LNCS 893, 14-32, 1995.

[27] L.M.Perry, "Extending (Finite) Partially Ordered Sets to Lattices: An Incremental Approach", Master's Thesis, Univ. of Maine, Dep. of Surv. Eng., Orono, ME, 1990.

[28] A.Saalfeld, "Lattices Structures in Geography", 7th Int. Symp. on Computer-Assisted Cartography, AUTOCARTO 7, 482-489, 1985.

[29] H. Samet, "The design and analysis of spatial data structures", Addison-Wesley Reading, MA, 1990.

[30] G. Steiner "An algorithm to generate the ideals of a partial order", Operation Research Letters, vol.5 n.6, 1986.

[31] M. Talamo, P. Vocca, "A Data Structure for Lattice Representation", TCS, 175(2):373-392, 97.

[32] M.Talamo, P.Vocca, "An Optimal Time×Space Data Structure for Lattices Representation", to be published on SIAM Journal on Computing.

[33] M. F. Worboys, "A generic model for planar geographical objects", Int. J. Geographical Information Systems, vol 6, n.5, 353-372, 1992.

A Properties of Posets

In this section we recall some more advanced results on posets that are needed in our proofs. It can be skipped by the reader that has a working knowledge of the field.

Upper-star and lower-star ideals, introduced at page 231, have interesting properties, stated in the following lemma. We refer to [6, 27] for proofs of these properties.

Lemma 7 *Given a poset $\langle P, \leq \rangle$ and $Q_1, Q_2 \in P$, we have:*

1. $Q_1 \subseteq (Q_1^*)_*$ *and* $Q_1 \subseteq (Q_{1*})^*$;
2. *if* $Q_1 \subseteq Q_2$, *then* $Q_2^* \subseteq Q_1^*$, $Q_{2*} \subseteq Q_{1*}$ *and* $(Q_1^*)_* \subseteq (Q_2^*)_*$;
3. $Q_{1*} = ((Q_{1*})^*)_*$ *and* $Q_1^* = ((Q_1^*)_*)^*$;
4. $p \in \{p\}_*$ *and* $p \in \{p\}^*$, $\forall p \in P$;
5. $(\{p\}^*)_* = \{p\}_*$, $\forall p \in P$;
6. *if* $p \leq q$, *then* $\{p\}_* \subseteq \{q\}_*$ *and* $\{q\}^* \subseteq \{p\}^*$, $\forall p, q \in P$;
7. $\{p\}_* = \{q\}_*$ *if and only if* $p = q$, $\forall p, q \in P$.

The next Lemma shows important properties of join-dense and meet-dense subsets, introduced at page 232. For proofs of this and other properties see [6].

Lemma 8 *Let P be a poset and let $Q \subseteq P$. If Q is join-dense in P then for each $x \in P$ we have $x = lub_P((\downarrow x)_Q)$. If Q is meet-dense in P then for each $x \in P$ we have $x = glb_P((\uparrow x)_Q)$.*

The following lemma shows that a poset (via the canonical order embedding) is both join-dense and meet-dense in its MacNeille completion. For proofs of this and other properties of the MacNeille completion see [6].

Lemma 9 *Let $\langle P, \leq \rangle$ be a poset and let $\langle M(P), \subseteq \rangle$ be its MacNeille completion. The set $\varphi(P)$ is both join-dense and meet-dense in $M(P)$.*

The definition of the MacNeille completion may also be used to define an algorithm for its construction. Such an algorithm has an exponential worst-case time complexity with respect to the number of elements of the poset. Recently, Perry [27] has proposed an incremental algorithm that has a polynomial complexity with respect to the number of elements of the produced lattice.

In general, in a poset there can be elements that are not related to each other.

Definition 1 *Let $\langle P, \leq \rangle$ be a poset and let $Q \subseteq P$. We say that Q is an* antichain *if $\forall x, y \in Q$, $x \leq y$ implies $x = y$.*

Note that for each $R \subseteq P$, the set R^o of the maximal elements of R and the set R_o of the minimal elements of R are antichains. It is possible to define an order relation among the antichains of a poset. The resulting poset is indeed a lattice, as it is shown by the next Lemma. For its proof see [4, 18].

Lemma 10 *Let $\langle P, \leq \rangle$ be a poset and let $A(P)$ be the set of all the antichains of P. The poset $\langle A(P), \tilde{\preceq} \rangle$, where $\tilde{\preceq}$ is the order relation defined, $\forall A_1, A_2 \in A(P)$, as $A_1 \tilde{\preceq} A_2$ if and only if $\downarrow A_1 \subseteq \downarrow A_2$, is a lattice, called lattice of antichains of $\langle P, \leq \rangle$. We write $A_1 \tilde{\prec} A_2$ when $A_1 \tilde{\preceq} A_2$ and $A_1 \neq A_2$. We have also $lub(A_1, A_2) = (\downarrow A_1 \cup \downarrow A_2)^o$ and $glb(A_1, A_2) = (\downarrow A_1 \cap \downarrow A_2)^o$.*

Several algorithms are known to construct the lattice of antichains of a given poset. In particular we signal to the interested reader [18, 30].

Now we define the concept of *congruence* and that of *quotient lattice.*

Definition 2 *Let $\langle L, \leq \rangle$ be a lattice and let $\cong$ be an equivalence relation defined on it. We say that $\cong$ is a* congruence *if $\forall a, b, c, d \in L$, $a \cong b$ and $c \cong d$ imply $lub(a, c) \cong lub(b, d)$ and $glb(a, c) \cong glb(b, d)$.*

Given a lattice $\langle L, \leq \rangle$ and an equivalence relation $\cong$ defined on L, we can try to define a partial order relation on $L_{\cong}$, the quotient set of L modulo $\cong$, saying that given $a, b \in L$, $[a] \leq' [b]$ if $a \leq b$. It turns out that the relation $\leq'$ is well defined (namely it is independent from the choice of the representatives of the equivalence classes) if the relation $\cong$ is a congruence (see [6]). The poset $\langle L_{\cong}, \leq' \rangle$ is indeed a lattice, as it is shown by the next Lemma. For its proof see [6].

Lemma 11 *Let $\langle L, \leq \rangle$ be a lattice and let $\cong$ be a congruence defined on it. Let $L_{\cong}$ be the quotient set of L modulo $\cong$ and let $\leq'$ be the partial order relation on $L_{\cong}$ defined as $[a] \leq' [b]$ if and only if $a \leq b$, $\forall [a], [b] \in L_{\cong}$. The poset $\langle L_{\cong}, \leq' \rangle$ is a lattice, called the* quotient lattice *of L modulo $\cong$.*

We close this section by recalling in the next lemma a basic result of elementary algebra which is used in the paper. Its proof can be found in elementary algebra textbooks.

Lemma 12 *Let A and B be sets and let $f : A \mapsto B$ be a mapping. Consider the equivalence relation $\cong$ on the set A defined as $x \cong y$, $\forall x, y \in A$, if and only if $f(x) = f(y)$. Let $A_{\cong}$ be the quotient set of A modulo $\cong$. Then the mapping $g : A_{\cong} \mapsto B$ defined as $g([x]) = f(x)$ is well defined (namely it is independent from the choice of the representatives of the equivalence classes) and is injective. Moreover, $g(.)$ is bijective if and only if $f(.)$ is surjective.*

B Proofs of Section 4

Theorem 4.1. *We have $(S^{\cup})^{\cap} = S^{\cap\cup}$ and $(S^{\cap})^{\cup} = S^{\cap\cup}$.*

Proof. We only show the latter equality because the proof of the former is analogous. We first prove that for each $s \in (S^{\cap})^{\cup}$ we have $s \in S^{\cap\cup}$. Observe that for each $s \in S^{\cap}$ we have either $s \in S$ or s is generated applying recursively rule 2 given in Definition 4.1 (p.232). In either cases, $s \in S^{\cap\cup}$ applying rules 1 or 2 given in Definition 4.3 (p.232), hence $S^{\cap} \subseteq S^{\cap\cup}$. Consider now $s \in (S^{\cap})^{\cup}$: s is generated by the application of rules 1 or 2 given in Definition 4.2 (p.232), starting from sets in $S^{\cap}$. Then, since $S^{\cap} \subseteq S^{\cap\cup}$, applying rules 1 or 2 given in Definition 4.3 (p.232) to the same sets in $S^{\cap}$ we have $s \in S^{\cap\cup}$.

We now prove that for each $s \in S^{\cap\cup}$ we have $s \in (S^{\cap})^{\cup}$. Let us consider $s \in S^{\cap\cup}$. If $s \in S$ then $s \in (S^{\cap})^{\cup}$ by definition. Otherwise, s is generated by the application of rule 2 given in Definition 4.3 (p.232). To show the thesis we proceed by induction on the number of applications of rule 2. If rule 2 is applied 0 times, then $s \in S$, hence the thesis is true. Assume s is generated by N applications of rule 2 to distinct couples of distinct elements. Then we have $s = s_1 \cup s_2$ or $s = s_1 \cap s_2$ with $s_1, s_2 \in S^{\cap\cup}$ and s_1 and s_2 are generated respectively by N_1 and N_2 applications of rule 2, with $N_1 < N$ and $N_2 < N$. By inductive hypothesis $s_1, s_2 \in (S^{\cap})^{\cup}$, hence we have $s_1 = \bigcup_i (\bigcap_j s_{ij})$ and $s_2 = \bigcup_h (\bigcap_k t_{hk})$, with $s_{ij} \in S\ \forall i, j$ and $t_{hk} \in S\ \forall h, k$. If $s = s_1 \cup s_2$ then we have

$$s = (\bigcup_i (\bigcap_j s_{ij})) \cup (\bigcup_h (\bigcap_k t_{hk})),$$

hence the thesis is trivially true. If $s = s_1 \cap s_2$ then we have

$$s = (\bigcup_i (\bigcap_j s_{ij})) \cap (\bigcup_h (\bigcap_k t_{hk})).$$

Applying the distributive property we have

$$s = \bigcup_{i,h} (\bigcap_j s_{ij}) \cap (\bigcap_k t_{hk}),$$

hence the thesis is proved. ■

Lemma 1. *If S is a class of sets with a set-containment relation and a universal partition U_S, there exists a unique set $\{r_1, r_2 \ldots r_n\} \in 2^{U_S}$ such that $s = \bigcup_i r_i$. Also $\forall s \in S$, $s = \bigcup_{r \in S_{\text{Base}}(s)} r$.*

Proof. Assume there exist two collections $\{r_1, r_2 \ldots r_n\}$ and $\{t_1, t_2 \ldots t_m\}$ of sets of the universal partition such that $s = \bigcup_i r_i = \bigcup_j t_j$ but such that they differ in at least one element. Without loss of generality assume that there exists $r_i \in \{r_1, r_2 \ldots r_n\}$ such that $\forall j$, $r_i \neq t_j$. By definition of elements in the universal partition then $\forall j$, $r_i \cap t_j = \emptyset$ and it cannot be $r_i \subseteq s = \bigcup_j t_j$, hence we have a contradiction. To show the second part of the proposition, let us consider an $s \in S$. By definition for each $r \in S_{\text{Base}}(s)$ we have $r \subseteq s$, hence $\bigcup_{r \in S_{\text{Base}}(s)} r \subseteq s$. Since a universal partition exists, for the first part of this proposition, there is a unique set $\{r_1, r_2 \ldots r_n\} \in 2^{U_S}$ such that $s = \bigcup_i r_i$. For each $r_i \in \{r_1, r_2 \ldots r_n\}$ we have $r_i \subseteq s$, then $r_i \in S_{\text{Base}}(s)$ and $s = \bigcup_i r_i = \bigcup_{r \in S_{\text{Base}}(s)} r$. ■

Corollary 2. *The mapping $S_{\text{Base}}(.)$ is an order embedding from the poset $\langle S, \subseteq \rangle$ to the poset $\langle 2^{U_S}, \subseteq \rangle$.*

Proof. We know from Lemma 1 that $\forall s \in S$, $s = \bigcup_{r \in S_{\text{Base}}(s)} r$. Then given $s_1, s_2 \in S$ we have $s_1 \subseteq s_2$ iff $\bigcup_{r \in S_{\text{Base}}(s_1)} r \subseteq \bigcup_{r \in S_{\text{Base}}(s_2)} r$. Since the elements of U_S are a partition, $\bigcup_{r \in S_{\text{Base}}(s_1)} r \subseteq \bigcup_{r \in S_{\text{Base}}(s_2)} r$ iff $S_{\text{Base}}(s_1) \subseteq S_{\text{Base}}(s_2)$. ■

The following proposition is needed for the proof of the subsequent Corollary.

Proposition 13 *If S is a class of sets with a set-containment relation and a universal partition U_S, then $\forall s \in S^{\cap\cup}$ there exists a set $\{r_1, r_2 \ldots r_n\} \in 2^{U_S}$ such that $s = \bigcup_i r_i$.*

Proof. Let us consider $s \in S^{\cap\cup}$. If $s \in S$, the thesis is true by hypothesis. Otherwise, s is generated by the application of rule 2 given in Definition 4.3 (p.232). To show the thesis we proceed by induction on the number of applications of rule 2. If rule 2 is applied 0 times, then $s \in S$, hence the thesis is true. Assume s is generated by N applications of rule 2 to distinct couples of distinct elements. Then we have $s = s_1 \cup s_2$ or $s = s_1 \cap s_2$ with $s_1, s_2 \in S^{\cap\cup}$ and s_1 and s_2 are generated respectively by N_1 and N_2 applications of rule 2, with $N_1 < N$ and $N_2 < N$. By inductive hypothesis we have $s_1 = \bigcup_i r_{1i}$ and $s_2 = \bigcup_h r_{2h}$, with $r_{1i} \in U_S$, $\forall i$ and $r_{2h} \in U_S$, $\forall h$. If $s = s_1 \cup s_2$ then we have $s = (\bigcup_i r_{1i}) \cup (\bigcup_h r_{2h})$, hence the thesis is trivially true. If $s = s_1 \cap s_2$ then we have $s = (\bigcup_i r_{1i}) \cap (\bigcup_h r_{2h})$. Applying the distributive property we have $s = \bigcup_{i,h} (r_{1i} \cap r_{2h})$. Since for every $r_a, r_b \in U_S$ we have $r_a \cap r_b = \emptyset$ if $r_a \neq r_b$, the former equality gives the thesis. ■

Corollary 3. *Let S be a class of sets with a set-containment relation and a universal partition U_S. Then U_S is a universal partition of $S^{\cap\cup}$.*

Proof. It follows trivially from Proposition 13. ■

Corollary 4. *Let S be a class of sets with a set containment relation and a universal partition U_S. Then $S^{\cup} = S^{\cap\cup}$.*

Proof. Obviously we have $S^{\cup} \subseteq S^{\cap\cup}$. We have to show that $S^{\cap\cup} \subseteq S^{\cup}$. Since (for corollary 3) U_S is a universal partition of $S^{\cap\cup}$, $\forall s \in S^{\cap\cup}$ there exists a set $\{r_1, r_2 \ldots r_n\} \in 2^{U_S}$ such that $s = \bigcup_i r_i$. Since $U_S \subseteq S$, $\forall r_i \in \{r_1, r_2 \ldots r_n\}$ we have $r_i \in S$, so $s = \bigcup_i r_i \in S^{\cup}$. ■

Corollary 5. *Let S be a class of sets with a set containment relation and a universal partition U_S. The mapping $S_{\text{Base}}(.)$ is an order isomorphism from the poset $\langle S^{\cap\cup}, \subseteq\rangle$ to the poset $\langle 2^{U_S}, \subseteq\rangle$.*

Proof. We know from Corollary 3 that U_S is a universal partition of $S^{\cap\cup}$, hence, by Corollary 2, mapping $S_{\text{Base}}(.)$ is an order embedding from poset $\langle S^{\cap\cup}, \subseteq\rangle$ to poset $\langle 2^{U_S}, \subseteq\rangle$. We only have to show that $S_{\text{Base}}(.)$ is a surjective mapping. But this is trivial, since for each $\{r_1, r_2 \ldots r_n\} \subseteq U_S$ we have $\{r_1, r_2 \ldots r_n\} \subseteq S^{\cap\cup}$ and $s = \bigcup_i r_i \in S^{\cap\cup}$ and, by Lemma 1 (p.232), $S_{\text{Base}}(s) = \{r_1, r_2 \ldots r_n\}$. ■

Theorem 4.2. *Let S be a class of sets with a set-containment relation and a universal partition. If P is a representation of S then for each $s \in S$ we have:*

$$\begin{aligned} Rep(s) &= lub(\{y \mid y = Rep(r) \text{ and } r \in S_{\text{Base}}(s)\}) \\ &= lub(P_{\text{Base}}(Rep(s))) \ . \end{aligned}$$

Proof. Since S has a universal partition, we know, from Lemma 1, that $\forall s \in S$, $s = \bigcup_{r \in S_{\text{Base}}(s)} r$. For every $r \in S_{\text{Base}}(s)$ it is $r \subseteq s$, so $Rep(r) \leq Rep(s)$ since P is a representation. Consider now an $x \in P$ such that for every $r \in S_{\text{Base}}(s)$, $Rep(r) \leq x$. We have $r \subseteq Rep^{-1}(x)$ for every $r \in S_{\text{Base}}(s)$, so $s = (\bigcup_{r \in S_{\text{Base}}(s)} r) \subseteq Rep^{-1}(x)$. But since P is a representation we have $Rep(s) \leq x$, hence the first equality in the thesis is proved. To show the second one we simply observe that Definition 4.9 implies $P_{\text{Base}}(Rep(s)) = \{y \mid y = Rep(r) \text{ and } r \in S_{\text{Base}}(s)\}$ for any $s \in S$ ■

The following Proposition is used in the proof of the subsequent Corollary and in the companion paper [9] to manage easily universal partitions on posets.

Proposition 14 *Let S be a class of sets with a set-containment relation and let P be a representation of S. For each $p \in P$ we have $P_{\text{Base}}(p) = B_P \cap {\downarrow} p$.*

Proof. By definition we have $P_{\text{Base}}(p) \subseteq B_P$. Also, $\forall x \in P_{\text{Base}}(p)$ there exists $r \in S_{\text{Base}}(Rep^{-1}(p))$ such that $x = Rep(r)$. Since P is a representation of S, $Rep^{-1}(x) \subseteq Rep^{-1}(p)$ implies $x \leq p$. Conversely, let us consider $x \in B_P \cap {\downarrow} p$. Since $x \leq p$ we have $Rep^{-1}(x) \subseteq Rep^{-1}(p)$. Also, $x \in B_P$ implies $Rep^{-1}(x) \in B_S$, hence $Rep^{-1}(x) \in S_{\text{Base}}(Rep^{-1}(p))$ so we have $x \in P_{\text{Base}}(p)$. ■

Corollary 6. *The mapping $P_{\text{Base}}(.)$ from the poset $\langle P, \leq\rangle$ to the poset $\langle 2^{B_P}, \subseteq\rangle$ is order preserving. Moreover if B_P is a universal partition, $P_{\text{Base}}(.)$ is an order embedding.*

Proof. The fact that $P_{\text{Base}}(.)$ is order preserving follows from Proposition 14 and from the well known fact that $p_1 \leq p_2$ implies ${\downarrow} p_1 \subseteq {\downarrow} p_2$, which in turn implies $B_P \cap {\downarrow} p_1 \subseteq B_P \cap {\downarrow} p_2$. To show the second part of the thesis (when B_P is a universal partition) let us consider $Q_1, Q_2 \in B_P$ such that $Q_1 = P_{\text{Base}}(p_1)$, $Q_2 = P_{\text{Base}}(p_2)$ and $Q_1 \subseteq Q_2$, with $p_1, p_2 \in P$. For each $q \in Q_1$ we have $q \leq p_1$ and since $q \in Q_2$ we also have $q \leq p_2$. But for Theorem 4.2 we have $p_1 = lub(Q_1)$ hence $p_1 \leq p_2$ ■

ASYNCHRONOUS MICROPROCESSORS

Jurij Šilc
Computer Systems Department, Jožef Stefan Institute, Ljubljana, Slovenia
email: jurij.silc@ijs.si
AND
Borut Robič
Faculty of Computer and Information Science, University of Ljubljana, Slovenia
email: borut.robic@fri.uni-lj.si

Keywords: asynchronous logic, asynchronous processor, self-timed processor

Edited by: Rudi Murn
Received: April 6, 1999 **Revised:** May 19, 1999 **Accepted:** May 25, 1999

The asynchronous processors attack clock-related timing problems by asynchronous (or self-timed) design techniques. Asynchronous processors remove the internal clock. Instead of a single central clock that keeps the chip's functional units in step, all parts of an asynchronous processor (e.g., the arithmetic units, the branch units, etc.) work at their own pace, negotiating with each other whenever data needs to be passed between them. In this paper, several projects are presented, two of these – the Superscalar Asynchronous Low-Power Processor (SCALP) and AMULET – are presented in more detail.

1 Introduction

Conventional synchronous architectures are based on global clocking whereby global synchronization signals control the rate at which different elements operate. For example, all functional units operate in lockstep under the control of a central clock [16].

With progress of time and improvement of technology, clocks get faster, the chips have higher circuit density and the wires get finer. As a result, it becomes increasingly difficult to ensure that all parts of the processor are ticking along in step with each other. Even though the electrical clock pulses are travelling at a substantial fraction of the speed of light, the delays in getting from one side of a small piece of silicon to the other can be enough to throw the chip's operation out of synchronization. Even if the clock were injected optically to avoid the wire delays, the signals issued as a result of the clock would still have to propagate along wires in time for the next clock pulse, and a similar problem would remain. For example, the 1997 National Technology Roadmap for Semiconductors [23] forecasts that CMOS technology will reach a point where the switching delay for a single gate will be close to 10 ps while a single chip area will be nearly 7.5 cm^2. It will take 30 clock cycles for the electric signal to cross such a chip. Moreover, the interchip clock skew already represents a major problem.

The clock-related timing problems have been recently attacked by *asynchronous* (or *self-timed*) design techniques. These asynchronous processors do away with the idea of having a single central clock keeping the chip's functional units in step. Instead, each module of the processor – for example, the arithmetic units, the branch units, etc. – all work at their own pace, negotiating with each other whenever data needs to be passed between them. The communication protocol synchronizes the modules involved in the communication and allows data to be shared between them.

Without a global clock, asynchronous systems enjoy [19]:

- Data-dependent cycle time rather than worst case cycle time: The conventionally clocked chip has to be slowed down so that the most sluggish function does not get left behind. To deal with this problem one can either use some extra circuitry to try to speed up these slow special cases, or alternatively just accept it and slow everything down to take account of the lowest common denominator. Either way the result is that resources are wasted or the chip's speed is determined by an instruction that may hardly ever be executed. In the asynchronous approach the chip only becomes more sluggish when a tricky operation is encountered.
- Potential for low power consumption: The conventional processors are becoming increasingly power consuming. For example, DEC's Alpha and the IBM/Motorola PowerPC 620 emit around 20 W to 30 W in normal operation. If we were to continue to use 5 V supplies, we could expect by the end of 1999 a 0.1 micron processor dissipating 2 kW. Reducing the supply to 3 V (or 2 V) would only reduce the power dissipation to 660 W

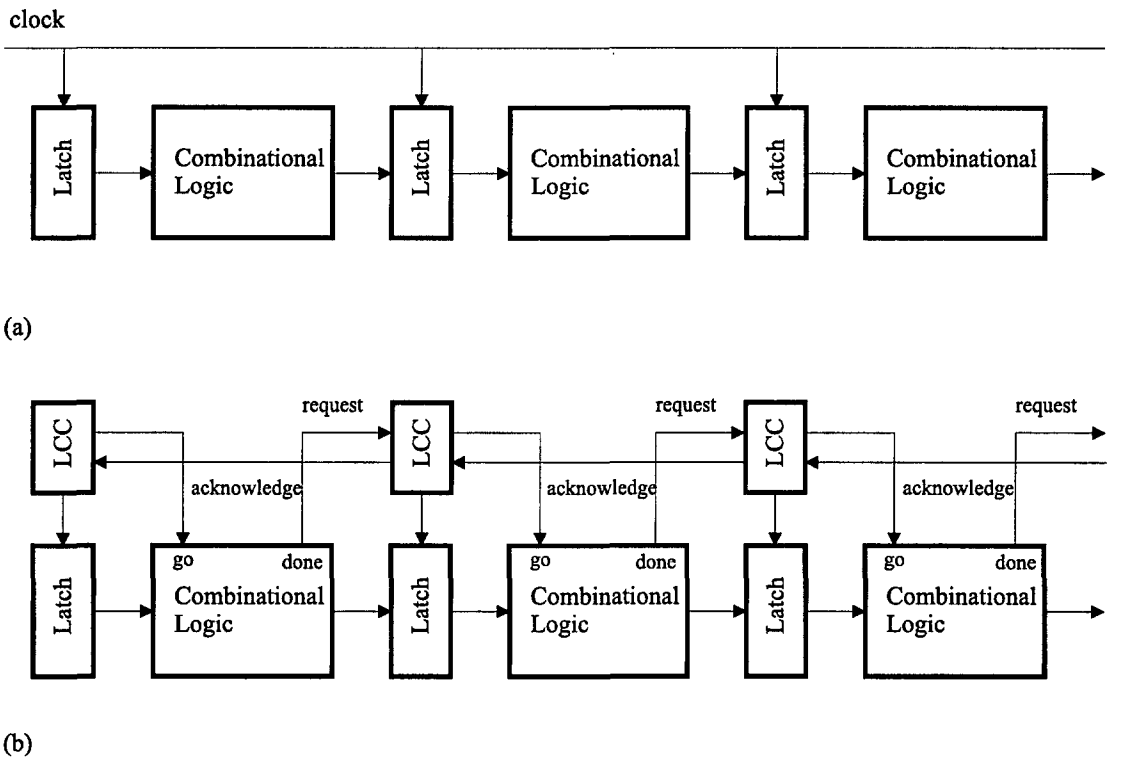


Figure 1: A simple pipeline (a) synchronous (b) asynchronous

(or 330 W). One of the reasons is that many of the logic gates switch their states simply because they are being driven by the clock, and not because they are doing any useful work. Removing the clock in asynchronous processors also removes the unnecessary power consumption as CMOS gates only dissipate energy when they are switching.

- Ease of modular composition, i.e., circuits can be assembled as plug-and-play.
- Optimization of frequent operations while rare operations can spend more time.
- No need for clock alignment at the interfaces.
- Timing fault-tolerance.

There are also several shortcomings to the asynchronous approach:

- clock-based computers are easier to build than asynchronous;
- it is easier to verify a synchronous design due to its deterministic operation (by comparison, verifying an asynchronous design, with each part working at its own pace, is difficult).

2 Asynchronous Logic

Virtually all digital design today is based on a synchronous approach whereby each subsystem is a clocked finite state machine that changes its states on the edges of a regular global clock. Such a system behaves in a discrete and deterministic way, provided the delays are managed so that the flip-flop setup and hold times are met under all conditions.

As a contrast, in asynchronous design, there is no clock to govern the timing of state changes. Subsystems exchange information at mutually negotiated times with no external timing regulation. An asynchronous pipeline, such as *micropipelines* [18], manages the flow of data according to the state of the next and previous pipeline stages. In a synchronous pipeline, if a stage is late in completing operation of the combinatorial circuit, the entire pipeline delayed by an amount of time equal to the clock period. For an asynchronous pipeline, however, if a stage is late by an duration ω, the entire pipeline is delayed just by ω. Also in contrast to synchronous pipelines, an asynchronous pipeline controller only judges the state of adjoining stages. Therefore decentralized control is possible [19].

Figure 1a shows the structure of a synchronous pipeline with latches and combinational logic blocks. All latches are controlled by a single global clock signal and operate simultaneously.

An asynchronous implementation of the pipeline is shown in Fig. 1b. The latches and the combinational logic block are the same as in the synchronous pipeline. The timing, however, is controlled differently. Each latch has an associated *latch control circuit* (LCC)

which opens and closes the latch in response to *request* signals from the previous stage and *acknowledge* signals from the following stage. There are a few key features which describe most current approaches:

- *Delay-insensitive* vs *speed-independent* design: Delay-insensitive designs make no assumptions about delays within the system. That is, any gate or interconnection may take an arbitrary time to propagate a signal. Speed-independent systems are tolerable to variations in gate speeds but assume instantaneous transmissions along wires.
- *Dual-rail encoding* vs *data bundling* communication protocol: In dual rail encoded data, each Boolean is implemented as two wires. This allows the value and the timing information to be communicated for each data bit. Bundled data, on the other hand, has one wire for each data bit and a separate wire to indicate the timing.
- *Level* vs *transition* encoding: Level-sensitive circuits typically represent a logic one by a high voltage and a logic zero by a low voltage. In transition signaling, only changes in the level of signals are taken into account.

Delay-insensitive circuits with dual-rail communication and encoding with transition signaling proved to be ideal for automatic transformation into a silicon layout, as the delays introduced by the layout compiler cannot affect the functionality. The most popular form in recent years has been dual-rail encoding with level-sensitive signaling. Delay insensitivity is achieved at the cost of more power dissipation than with transition signaling. The advantage of this approach over transition signaling is that the logic processing elements can be much simpler. A well-known form of delay-insensitive circuit with bundled data communication and encoding with transition signaling is the *micropipelined* approach, which was proposed in [18] and adopted in the AMULET project (see below).

3 Microprocessors

A number of asynchronous microprocessors have been proposed or built recently. The processors described can be divided broadly into two categories:

- Those that were built using a conservative timing model, suitable for formal synthesis or verification, but with a simple architecture. Among these are CAP, TITAC, ST-RISC, ARISC, and ASPRO-216.
- Those that were built with a less cautious timing model using an informal design approach, but with a more ambitious architecture. These include the AMULET processors, NSR, Fred, CPP, Hades, ECSTAC, STRiP, and SCALP.

Table 3 summarizes these characteristics.

Let us describe the architecture and the asynchronous design of asynchronous superscalar processors SCALP and AMULET and, only briefly, some other projects [7, 22].

3.1 Superscalar Asynchronous Low-Power Processor

The first asynchronous superscalar processor was designed in 1996 at the University of Manchester [7]. The processor was named SCALP, for Superscalar Asynchronous Low-Power Processor. SCALP's main architectural innovation is its lack of a global register file and its result forwarding network. Most SCALP instructions do not specify the source of their operands and destination of their results by means of register numbers. Instead, the idea of *explicit forwarding* was introduced whereby each instruction specifies the destination of its result. That destination is the input to another functional unit consuming the value. Instructions do not specify the source of their operands at all; they implicitly use the values provided for them by the preceding instructions.

Figure 2 shows the organization of the SCALP processor. SCALP does have a register file; it constitutes one of the functional units. It is accessed only by read and write instructions which transfer data to and from other functional units by means of the explicit forwarding mechanism. Several instructions are fetched from memory at a time. Each instruction has a *functional unit identifier*, which is a small number of easily decoded bits that indicate which functional unit will execute the instruction. The instructions are statically allocated to functional units. If there is more than one functional unit capable of executing a particular instruction, one must be chosen by the compiler. This simplifies the instruction issuer and is essential to the explicit forwarding mechanism. The instruction issuer is responsible for distributing the instructions to the various functional units on the basis of the functional unit identifier. Each functional unit has a number of input queues: one for instructions and one for each of its possible operands. An instruction begins execution once it and all of its necessary operands have arrived at the functional unit. The functional unit sends the result, along with the destination address, to the result-routing network. This places the result into the appropriate input queue of another functional unit.

There are some similarities between the SCALP approach and dataflow computing. In particular, it is possible to describe SCALP programs by means of dataflow graphs. Nevertheless, the flow of control in SCALP is determined by a conventional control-flow mechanism, not a dataflow mechanism.

Processor	Design Style	ISA	Organization
CAP	4-phase, dual-rail delay-insensitive	own 16-bit RISC-like	fetch-execute pipeline
FAM	4-phase, dual-rail delay-insensitive	own RISC-like	pipelined
STRiP	variable clock synchronous	MIPS-X	pipelined forwarding
ST-RISC	dual-rail delay-insensitive	own	fetch-execute pipeline
NSR	2-phase bundled data	own 16-bit RISC-like	pipelined no forwarding decoupled branch & load/store
CFPP	2-phase bundled data	SPARC	pipelined multiple execution stages single issue result pipeline forwarding using counter-flow
AMULET1	2-phase bundled data	ARM	pipelined no forwarding
TITAC-1	2-phase, dual-rail quasi delay-insensitive	own 8-bit	nonpipelined
Fred	2-phase bundled data	based on MC 88100	pipelined multiple functional units single issue no forwarding decoupled branch & load/store
Hades	unspecified	own	pipelined multiple functional units multiple issue forwarding
ECSTAC	fundamental mode	own variable length	pipelined no forwarding
AMULET2e	4-phase bundled data	ARM	pipelined forwarding
SCALP	4-phase bundled data	own	pipelined multiple functional units multiple issue explicit forwarding
TITAC-2	2-phase, dual-rail scalable delay-insensitive	own 32-bit	pipelined multiple functional units
AMULET3i	4-phase bundled data	ARM	pipelined branch prediction out-of-order completion unrestricted forwarding
ARISC	4-phase bundled data	MIPS-II, MIPS16	pipelined multiple functional units
ASPRO-216	4-phase, multi-rail quasi delay-insensitive	own 16-bit	pipelined out-of-order completion

Table 1: Recent asynchronous microprocessors

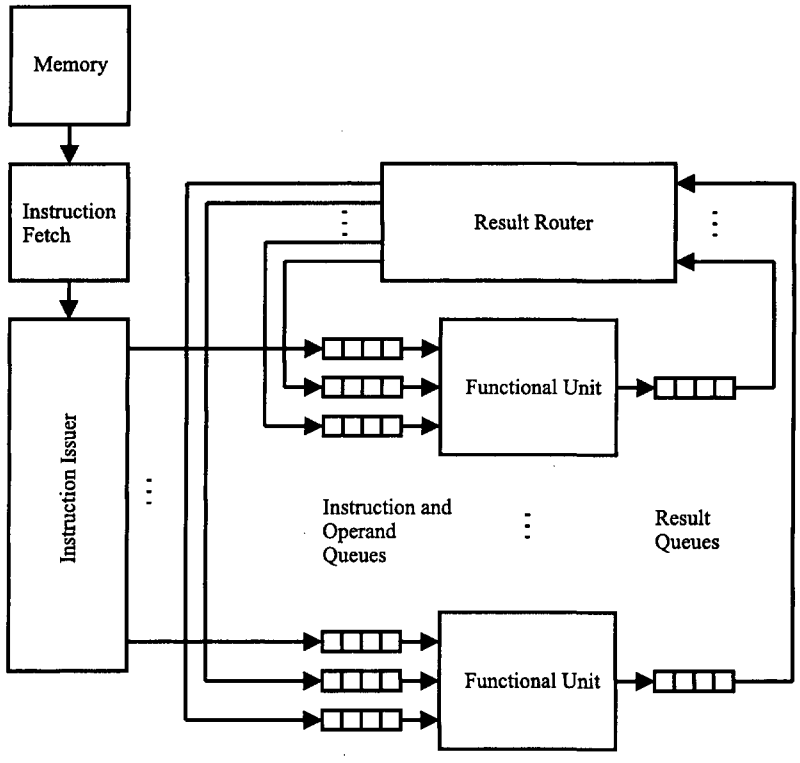


Figure 2: SCALP overall organization

3.2 AMULET

At the University of Manchester, several asynchronous processors called AMULET1, AMULET2, and AMULET3 were implemented [8, 9, 10, 20].

AMULET1 was the first asynchronous implementation of a commercially important instruction set architecture (ARM's instruction set architecture version 3 used in the ARM6 processor) [8], and appeared in early 1993 (see Fig. 3). It was designed using a 2-phase bundled data design style, with a 5-stage pipeline and no result forwarding. An interlocking was used to stall instructions at the register read stage until their operands had been written by previous instructions. After being fetched, a branch instruction had to pass through ten pipeline or FIFO stages before the target address was sent to memory. This resulted in large numbers of prefetched instructions that were discarded and a significant number of bubbles. As a result, the pipeline throughput was low. AMULET1 permitted out-of-order completion of load instructions relative to other instructions. AMULET1 was about 70 % the speed of a 20 MHz ARM6 processor, but with faster simple operations (e.g., with three times faster multiplication).

In October 1996, the AMULET2 processor was designed [9], based on the ARM instruction set architecture version 4. The processor used a 4-phase bundled data design style because this was believed to have benefits in terms of speed, size, and power relative to AMULET1. The processor had a slightly shorter pipeline than AMULET1 and employed both forwarding and branch prediction. It also incorporated limited forwarding by employing a last-result register at the output of the ALU, and forwarding mechanisms to use the result of a load instruction in a following instruction. A more sophisticated register-interlocking mechanism was used to remove write-after-write hazards. The AMULET2e chip consisted of 454 000 transistors including a 4 kbyte fully associative cache. The synchronous equivalent ARM810 used almost twice as many transistors, but also an 8 kbyte cache. With its 40 MIPS, the AMULET2 was 3.2 times faster than AMULET1, and with half the performance of a 75 MHz ARM810.

The next in the AMULET line, AMULET3, is expected to be a commercial product in 1999 [10]. It is expected that the key feature of this microprocessor will be a reorder buffer capable of solving the problems of result forwarding and exception handling in an asynchronous pipeline. This will allow a high degree of flexibility in operation, such as out-of-order completion, whilst avoiding read-after-write hazards (by stalling until the relevant value appears) and write-after-write hazards (averted by the reorder buffer).

3.3 Caltech Asynchronous Processor

The Caltech Asynchronous Processor (CAP) [11] was built in the late 1980s at California Institute of Tech-

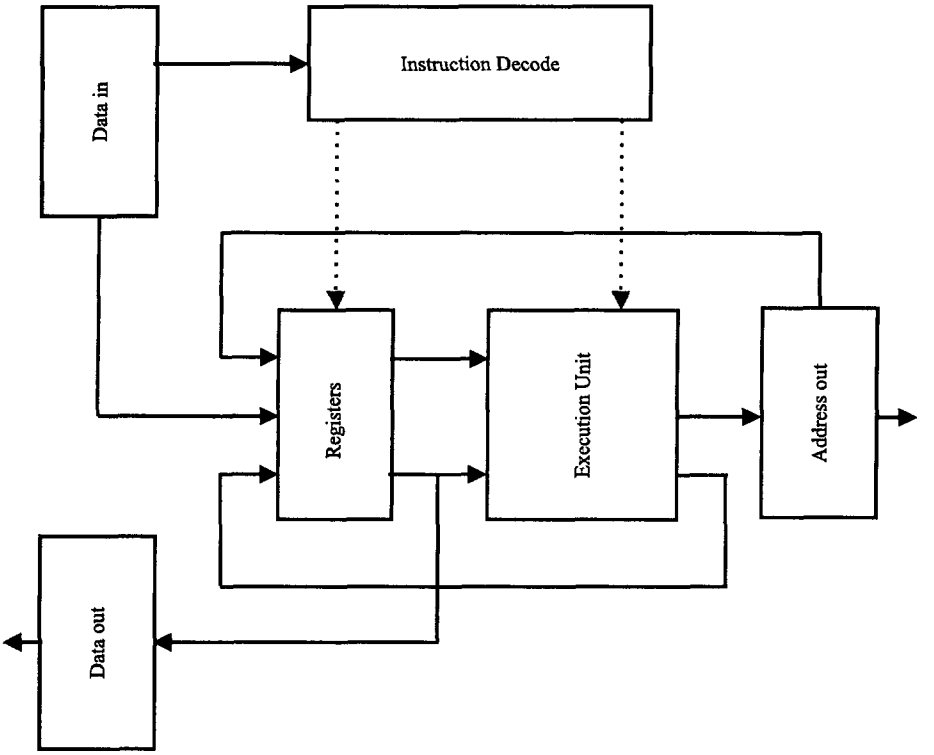


Figure 3: The AMULET1 organization

nology. The circuit design was delay-insensitive with dual-rail encoded communication. The processor featured a RISC-like load/store instruction set with 16 registers. A number of concurrent processes were responsible for instruction fetch, operand read, ALU operate, etc. The processor was implemented in a 1.6 micron CMOS process, and operated at 18 MIPS at room temperate and 5 V. The circuit continued to function at very low supply voltages, with optimum energy per operation at around 2 V. It was also tested in liquid nitrogen at 77 K when its performance reached 30 MIPS. More recently, GaAs version of CAP has been implemented [21].

3.4 Fully Asynchronous Microprocessor

The 32-bit Fully Asynchronous Microprocessor (FAM) [3] developed in the early 1990s at the Korean Institute of Science and Technology and the Tokyo Institute of Technology, was a dual-rail asynchronous processor with a RISC-like load/store instruction set. It had a 4-stage pipeline but register read, ALU, and register write occurred in a single stage eliminating the need for any forwarding. FAM, like CAP, is experimental.

3.5 Self-Timed RISC Processor

A Self-Timed RISC Processor (STRiP) [5] was built at Stanford University. Its instruction set was that of the MIPS-X processor. STRiP is included here even though it has a global clock signal and could be considered synchronous. It is unusual in that the speed of the global clock is dynamically variable in response to the instructions being executed, giving much of the advantage of an asynchronous system. The performance of STRiP was typically twice that of an equivalent synchronous processor. By maintaining global synchrony STRiP was able to implement forwarding in the same simple way as synchronous processors. For a 2 micron technology, the designers reported a 63 MHz effective clock frequency and a 62.5 MIPS performance rating.

3.6 ST-RISC

ST-RISC [4] was an architecture from the Israel Institute of Technology. It was delay-insensitive with a dual-rail datapath. ST-RISC had a 2-stage pipeline (fetch and execute) and a 3-address-register-based instruction set.

3.7 Nonsynchronous RISC

The Nonsynchronous RISC (NSR) processor [1] was built using FPGA technology at the University of Utah in 1993. It was implemented using a 2-phase bundled data protocol. NSR was a pipelined processor with pipeline stages separated by FIFO queues. The idea of the FIFO queues is that they decouple the pipeline stages so that an instruction that spends a long time in

one stage need not hold up any following instructions. The disadvantage of this approach is that the latency of the queues themselves is significant and, because of the dependencies within a processor pipeline, the increase in overall latency is detrimental. NSR used a locking mechanism to stall instructions that need operands produced by previous instructions. There was no forwarding mechanism. NSR had a 16-bit datapath and 16-bit instructions. The instructions included three 4-bit register specifiers and a 4-bit opcode. Some aspects of its instruction set were specialized for the asynchronous implementation: branch, load, and store instructions made the FIFOs that interconnected the functional units visible to the programmer. Conditional branch instructions were decoupled from the ALU that executed comparison instructions by a Boolean FIFO. Computed branch instructions also used a FIFO to store computed branch addresses. Load instructions had two parts. One instruction specifies a load address. A subsequent instruction used the load result by reading from a special register `r1`. There could be an arbitrary separation between the two instructions, and it was possible to have several load operations outstanding at one time. Store instructions worked similarly by writing the store data to `r1`.

3.8 Counterflow Pipeline Processor

The Counterflow Pipeline Processor (CFPP) [17], developed in 1994 at Sun Microsystems, was based on extensions of the techniques proposed in [18]. The CFPP executed SPARC instructions. Its novel contribution was the result forwarding in an asynchronous pipeline. CFPP had two pipelines. In one pipeline, instructions flowed upwards; in the other results flowed downwards. As instructions flowed upwards they watched out for results of previous instructions that they had to use as operands flowing downwards. If they spotted any such operands, they captured them; otherwise, they eventually received a value that flowed down from the register file which was at the top of the pipelines. When an instruction obtained all of its operands it continued to flow upwards until it found a pipeline stage where it could compute the result. Once computed, the result was injected into the result pipeline for use by any following dependent instructions and was also carried forward in the instruction pipeline to be written into the register file. The counterflow pipeline neatly solved the problem of result forwarding. This particular CFPP was not fabricated.

3.9 TITAC

TITAC-1 [13] was a simple 8-bit asynchronous processor built at the Tokyo Institute of Technology. It was based on the quasi delay-insensitive (QDI) timing model (where additional assumptions are introduced into the delay-insensitive model) and so had to use dual-rail encoding communication. This resulted in about twice as many gates in the datapath compared to an equivalent synchronous datapath. Architecturally, TITAC-1 was very straightforward with no pipelining and a simple accumulator-based instruction set.

In 1997, a 32-bit asynchronous microprocessor TITAC-2 [19] was built whose instruction set architecture was based on the MIPS R2000. It uses a scalable delay-insensitive (SDI) model, which unlike the QDI model, assumes that the relative delay ratio between any two components is bounded. SDI circuits can run faster than equivalent QDI ones. The measured performance of TITAC-2 was 52.3 MIPS using the Dhrystone benchmark.

3.10 Fred

Fred [15] is a development of NSR and also built at the University of Utah. Like NSR, Fred is implemented using 2-phase data bundling. It is modeled using VHDL. Fred extends the NSR to have a 32-bit datapath and 32-bit instructions, based on the Motorola 88100 instruction set. Fred has multiple functional units. Instructions from the functional units can complete out of order. However, the instruction issuer can only issue one instruction at a time, and the register file is only able to provide operands for one instruction at a time. This allows for a relatively straightforward instruction issue and a precise exception mechanism, but limits the attainable level of parallelism. There is no forwarding mechanism; instructions are stalled at the instruction issuer until their operands have been written to the register file. There is no out-of-order issue. Like the NSR, Fred uses programmer-visible FIFO queues to implement decoupled load/store and branch instructions. This arrangement has the possibility of deadlock if the program tries to read from an empty queue or write to a full one. Fred chooses to detect this condition at the instruction issuer and generate an exception.

3.11 Hades

Hades [6] is a proposed superscalar asynchronous processor from the University of Hertfordshire. It is in many ways similar to a conventional synchronous superscalar processor; it has a global register file, forwarding, and a complex (though in-order) instruction issue. Its forwarding mechanism uses a scoreboard to keep track of which result is available and from where.

3.12 ECSTAC

ECSTAC [12] is an asynchronous microprocessor designed at the University of Adelaide. ECSTAC is im-

plemented using fundamental mode control circuits. It is deeply pipelined with a complex variable-length instruction format. It has 8-bit registers and ALU. The variable-length instructions and the mismatch between the address size and the datapath width made the design more complex and slower. There is no forwarding mechanism within the datapath, and a register interlocking scheme is used to stall instructions until their operands are available.

3.13 ARISC

A joint project between the Technical University of Denmark and LSI Logic Denmark resulted in ARISC [2], which is an asyncronous re-implementation of the LSI Logic's TinyRISC TR4101 embedded microprocessor. Four-phase bundled data protocol is used throughout the entire design in combination with a normally opaque latch controller. All logic is implemented using static logic standard cells. In 0.35 micron CMOS technology ARISC performance is 74 MIPS (with power efficiency 635 MIPS/W), while the performance of the 83 MHz TR4101 is 48 MIPS (539 MIPS/W).

3.14 ASPRO-216

ASPRO [14] is a CMOS standard-cell QDI microprocessor which is being developed at the ENST Bretagne and CNET Grenoble. It is based on a simple RISC architecture with 16 general purpose registers. ASPRO-216 is a scalar processor, which extensively uses an overlapping pipelined execution scheme involving asynchronous units. The processor issues instructions in order and may complete their execution out of order. In 0.25 micron technology the expected peak performance is 200 MIPS with 0.5 W power dissipation.

4 Conclusions

Power consumption in VLSI chips may be a significant issue for two reasons. First, to optimize battery life, portable equipment such as lap-top computers and mobile telephones demand low power consumption. Second, high performance processors now dissipate enough power to make chip cooling a problem in an office environment. Thus, it has been predicted that asynchronous techniques will find their way into certain niches, in particular, embedded applications where the work required is extremely burst-intensive or where power-saving requirements make the approach attractive. Clocked chips with some asynchronous parts may also be expected. The asynchronous processor paradigm has the potential to solve the clocking problems in large processor chips. As a result, several universities and microprocessor manufacturers are actively investigating new asynchronous processor architectures.

References

[1] E. Brunvand: The NSR processor. *Proc. 26th Annual Hawaii International Conference on System Sciences*, (1993), pp. 428–435.

[2] K.T. Christensen et al.: The design an asynchronous TinyRISC TR4101 microprocessor core. *Proc. 4th International Symposium on Advanced Research in Asynchronous Circuits and Systems*, (1998).

[3] K.-R. Cho, K. Okura, K. Asada: Design of a 32-bit fully asynchronous microprocessor (FAM). *Proc. 35th Midwest Symposium on Circuits and Systems*, (1992).

[4] I. David, R. Ginosar, M. Yoeli: Self-timed architecture of a reduced instruction set computer. *Proc. IFIP Working Conference on Asynchronous Design Methodologies*, (1993).

[5] M.E. Dean: Strip: A self-timed RISC processor. Technical Report CSL-TR-92-543, Stanford University, 1992.

[6] C.J. Elston et al.: Hades – Towards the design of an asynchronous superscalar processor. *Proc. 2nd Working Conference on Asynchronous Design Methodologies*, (1995), pp. 200–209.

[7] P.B. Endecot: SCALP: A superscalar asynchronous low-power processor. Ph.D. Thesis, University of Manchester, 1996.

[8] S.B. Furber et al.: A micropipelined ARM. *Proc. FIP TC10/WG 10.5 International Conference on Very Large Scale Integration*, (September, 1993), pp. 5.4.1–5.4.10.

[9] S.B. Furber et al.: AMULET2e. *Proc. EMSYS'96 - OMI 6th Annual Conference*, (September, 1996).

[10] S.B. Furber, J.D. Garside, D.A. Gilbert: AMULET3: A high-performance self-timed ARM microprocessor. *Proc. IEEE International Conference on Computer Design*, (October 1998).

[11] A.J. Martin et al.: The design of an asynchronous microprocessor. *Proc. Decennial Caltech Conference on VLSI*, (1989), pp. 351–373.

[12] S.V. Morton, S.S. Appleton, M.J. Liebelt: ECSTAC: A fast asynchronous microprocessor. *Proc. 2nd Working Conference on Asynchronous Design Methodologies*, (1995), pp. 180–189.

[13] T. Nanya et al.: TITAC: Design of a quasy-delay-insensitive microprocessor. *IEEE Design and Test of Comp.*, **11**(2):50–63, 1994.

[14] M. Renaudin, P. Vivet, F. Robin: ASPO-216: A standard-cell Q.D.I. 16-bit RISC asynchronous microprocessor. *Proc. 4th International Symposium on Advanced Research in Asynchronous Circuits and Systems*, (1998).

[15] W.F. Richardson, E. Brunvand: Fred: An architecture for a self-timed decoupled computer. Technical Report UUCS-95-008, University of Utah, 1995.

[16] J. Šilc, B. Robič, T. Ungerer: *Processor Architecture - From Dataflow to Superscalar and Beyond.* Springer-Verlag, Berlin, 1999.

[17] R.F. Sproull, I.E. Sutherland, C.E. Molnar: Counterflow pipeline processor architecture. *IEEE Design and Test of Comp.*, **11**(3), 1994.

[18] I.E. Sutherland: Micropipelines. *Comm. ACM* **32**(6):720–738, 1989.

[19] A. Takamura et al.: TITAC-2: An asyncronous 32-bit microprocessor based on scalable-delay-insensitive model. *Proc. IEEE International Conference on Computer Design*, (October 1997), pp. 288–294.

[20] G. Theodoropoulos, J.V. Wood: Building parallel distributed models for asynchronous computer architecture. *Proc. World Transputer Congress*, (September 1994), pp. 285–301.

[21] J.A. Tierno et al.: A 100-MIPS GaAs asynchronous microprocessor. *IEEE Design and Test of Comp.*, **11**(2):43–49, 1994.

[22] T. Werner, V. Akella: Asynchronous processor survey. *Computer*, **30**(11):67–71, 1997.

[23] The National Technology Roadmap for Semiconductors. SEMATECH Inc., Austin, TX, 1997.

Discrete-Event Simulation Software : A Comparison Of Users' Surveys

Vlatka Hlupic
Department of Information Systems and Computing
Brunel University
Uxbridge, Middlesex UB8 3PH, England

Keywords: Discrete-Event Simulation Software, Users' Requirements, Surveys

Edited by: S.J.E. Taylor
Received: May 22, 1998 **Revised:** February 13, 1999 **Accepted:** May 4, 1999

Simulation is being widely used as modelling and analysis tool in various application areas such as manufacturing, traffic, defence or health. Such popularity of simulation has resulted in a large number of simulation software tools available on the market. This paper presents and compares the results of surveys on the use of discrete-event simulation software conducted in 1992 and in 1997. Findings of the survey indicate which types of simulation software are primarily being used, users' opinion about positive and negative software features and possible ways of improving simulation software.

1 Introduction

Many companies are facing pressures to improve efficiency and reduce operating costs due to increasing competition. Simulation is a valuable tool that enables organisations to investigate possible strategies for performance improvement. It is widely used in various application areas, and as a result of this, there is a plethora of simulation packages available on the software market.

This paper presents the results of surveys on the use of discrete-event simulation software, which were conducted in 1992 and in 1997. These surveys have been carried out mainly to discover how the users are satisfied with the simulation software they use and how this software can be further improved. Although the same questionnaire was distributed to survey participants in both surveys, and similar number of questionnaires were completed and returned, it was not possible to have the same survey sample for both surveys. Survey conducted in 1992 included regular simulation users in educational institutions and industry around Europe, whilst participants in 1997 survey included members of the Simulation Study Group of the Operational Research Society of Great Britain both from academic and industrial institutions. Findings of the surveys indicate which types of simulation software are primarily being used common positive and negative features of simulation packages, and users' recommendations for further improvement of simulation software.

The paper is structured as follows. Following a brief review of previous simulation software surveys and their main findings, a survey conducted in this research is described. Results obtained from both surveys are presented and compared. Conclusions outline the main findings of this research.

2 User Surveys On Simulation Software

Several publications related to the users' surveys are found in simulation literature. A dated survey carried out by Kleine (1970; 1971), has examined users' views of eleven discrete simulation languages. The results of this survey showed that it was difficult to interpret the results mainly because a limited number of respondents were proficient in more than one language. In addition, the expertise of some respondents was difficult to specify.

Christy and Watson (1983) have used a survey of non-academic users to explore issues such as the functional area that use simulation, the method of selecting simulation software, the popularity of various software tool for simulation applications etc. This analysis has revealed that, that of the total applications of simulation, 59Concerning the simulation software, the results showed that generally there is a reluctance to implement and learn new programming languages for simulation applications.

Kirkpatrick and Bell (1989) have used a survey approach to investigate the issues related to visual interactive simulation in industry. These issues included the types of problems being addressed, reasons for using visual interactive modelling, and the ways in which this type of modelling affects problem solving. The results have revealed that although the some of the

participants are aware of the significant set-up costs and need for learning new software and new methodology, most of them have agreed that visual interactive modelling provides enhanced interaction with decision maker, more useful and easier-to-understand models, and better decisions.

Van Breedam (1990) have conducted a survey in order to evaluate several simulation software tools. They have distributed a questionnaire to experienced users of simulation, who were asked to rate a sample of simulation packages on the various criteria. On the basis of received answers, they have classified the software evaluated into clusters according to the main software features.

MacKulak and Savory (1994) undertook a survey in order to ascertain important features of industrial simulation environments. The results of the survey revealed that some of the most important software features specified by survey participants were consistent and friendly user interface, database storage capability for input data and a troubleshooting section in a documentation.

A comprehensive research on simulation software is published in Simulation News Europe 1998). This research relates to comparisons of simulation software and simulation techniques, where about 65 simulation tools were compared using various examples. Despite being so comprehensive, this research does not include a survey where simulation software users would specify their preferences and opinions about such software.

An analysis of the above presented surveys reveals that although a majority of the survey studies investigate various issues related to simulation software, none of them examine users' opinions about possible ways to further improve simulation and to reduce some of the inherent problems associated with developing computer simulation models.

3 Users' Surveys in 1992 and 1997

3.1 Purpose of the Survey

The main purpose of the survey was to investigate the users' requirements of discrete-event simulation software and users' opinion about possible ways of improving current simulation software tools and better satisfying their needs.

The questionnaire distributed to the participants in both surveys consisted of nine questions dealing with the type of discrete-event simulation software used (1), the specification of particular packages used (such as WITNESS, SIMFACTORY II.5, SIMAN/CINEMA, ProModelPC, XCELL+, INSTRATA or other) (2), the purpose of use of simulation (3), general opinion about each software used (4) and the application areas of simulation (5). Other questions include estimation of how successful the simulation studies carried out were from the point of view of the software used (6). In particular, users had to appraise whether substantial approximations had to be made due to limitations of software, or whether all desirable features of the systems under consideration could be modelled. The participants were also asked to list the main weaknesses and limitations of software used (7), as well as the most important positive features included in software used (8). Finally, they were requested to specify the most important features that should be included in the existing simulation packages that were to their knowledge not yet provided (9). Majority of the questions regarding the opinion about the software and possible ways of its improvement (questions 4,6,7,8 and 9) were open-ended instead of providing several alternatives to select from. It was believed that this approach would avoid any suggestions to the participants and give better and unprejudiced response.

3.2 Survey Samples

The 1992 survey sample included participants from both educational institutions and industrial companies around Europe, for whom it was known or believed to be regular users of simulation. A response rate was about 35% out of 120 distributed questionnaires. The ratio of responses from universities and responses from industry is about 70% to 30%, although an approximately equal number of the questionnaires were distributed to each group of users.

The 1997 survey sample includes members of the Simulation Study Group (special interest group) of the Operational Research Society of Great Britain, both from industry and academic institutions. It was believed that survey participants were actively involved in simulation (the results of the survey have confirmed that assumption) and /or had a substantial interest in simulation. A number of academics from universities across Great Britain have participated in the survey as well as participants from the industry working for various manufacturing, service, consulting and research companies. A response rate was about 25% out of 220 distributed questionnaires. The ratio of responses from universities and responses from industry is about 66% to 33% (similar to 1992 survey), although again an approximately equal number of the questionnaires was distributed to each group of users.

Both survey samples were not selected by any formal statistical method. The participants, for whom it was known or believed to be regular users of simulation, were selected deliberately. On the other hand, in both surveys not only the response rate was significantly higher from the users from universities, in average each response from academic participants provided more information then the response from the users in industry.

All these facts might raise the question of statistical significance of obtained results. However, it is believed that intentional selection of survey participants in both surveys experienced in simulation enhances the importance and representativeness of results.

4 Findings Of Surveys

In 1992 survey, a majority of the users used only simulators (56.5%), 38.4% used both simulators and simulation languages, and 5.1% used only simulation languages (simulators are usually data driven packages that require little or no programming, whilst simulation languages require bespoke programming for model development). An analysis of the number of simulation software tools used reveals that a majority (64. 1%) of the users used only one software tool, but the other 35.9% of users use more than one software, up to six different software packages. Regarding the purpose of simulation, 38.5% of participants used simulation only for modelling real systems, 7.7% used simulation only for education, whilst the majority of 53.8% used simulation both for modelling real systems.

In 1997 survey, 74% of the users used simulators, and at the same time 48% used simulation languages as well. Analysis of the specification of simulation software tools used reveals that about one third of the users use only one software tool, another third use two software tools, and the rest use between three and seven different software tools. With regard to the purpose of simulation, 81.5% of participants use simulation for modelling real systems, 66.6% use simulation for education (51.8% of the users use simulation both for modelling real systems and education), 7.4 % of participants use simulation for research and 3.7% use simulation for consulting work. Table 1 summarises the results related to the number of simulation packages used for both surveys.

Tables 1 to 5 present the results of 1992 and 1997 surveys. Items in rows were grouped according to percentages of users that specified certain software features. Each percentage is calculated as a proportion of 100%, and the rules of arithmetic cannot be applied to all percentages in a certain column.

Common elements from the responses regarding the general opinion about the software used are summarised in table 2, together with the percentage of users that have specified a certain software feature in each survey.

With regard to the types of systems being modelled, 35.9% of users in 1992 survey used simulation to model only manufacturing systems, 41% were involved in modelling both manufacturing and other types of systems, whilst 23.1% modelled only other types of systems such as chemical processing, health or traffic. In 1997 survey, manufacturing is also dominant application area of simulation (66.6%). Other application areas include, for example, health, communication systems, service industry, oil terminals and traffic modelling.

When being asked about the success of modelling, 30.8% of participants in 1992 survey declared that they have been able to model desirable features of the systems being modelled, 38.4% have managed to model most of the features, whilst 30.8% had problems. In 1997 survey, 51.9% of participants declared that they have been able to model desirable features of the systems being modelled, whilst 48.1% had problems in modelling due to the software limitations and inflexibility.

Table 3 summarises the responses obtained in both surveys regarding the main limitations and weaknesses of the software used including the percentages of the certain responses, whilst table 4 summarises the responses related to the most important positive features of software used.

Finally, a summary of the features that users in both surveys would like to be incorporated in the simulation software that could improve the software they use is presented in Table 5.

5 An Analysis Of Surveys' Results

The above presented results of the survey reveal some interesting trends in the use of simulation software. In both surveys, a majority of participants used simulators rather than simulation languages. Over third participants in 1992 survey and almost half participants in 1997 survey used both simulators and simulation languages. The reason for a high percentage of participants using both simulators and simulation languages in both surveys might be the fact that about 36% of users in 1992 survey and two thirds of participants in 1997 survey use more than one simulation package (up to seven different packages). These results indicate the trend of using several simulation packages rather than only one package, which probably cannot be suitable for all applications.

In both surveys, many survey participants are combining education, research and real life projects, and these are usually academics that have probably obtained most of these software tools with an educational discount. On the other hand, users employed in industry tend to use more flexible simulation and general purpose languages, have less tools at disposal and they usually have to pay a full price of the package.

With regard to the purpose of simulation, it is interesting that in 1992 survey 38.5% survey participants used simulation only for modelling real systems, whilst in 1997 survey over 80% of participants used simulation for modelling real systems. This could indicate that many of academic participants in the surveys are

Number of Simulation Packages Used	Percentage of Users 1992 Survey	Percentage of Users 1997 Survey
1	64.1	37
2	11.1	29.6
3	5.0	11.1
4	13.7	7.4
5	2.1	7.4
6	4.0	3.7
7	0	3.7

Table 1: The number of simulation packages used by surveys participants

increasingly involved in research and working on real life projects (as they represent a majority of survey participants in both surveys).

A general opinion about the software used shows that the main objection by the users in both surveys is that software is too limited for complex problems (1992 survey) and is lacking extensive modelling facilities and flexibility (1997 survey) (although at the same time a slightly smaller percentage of users in 1997 survey indicated the opposite, i.e. that packages do provide extensive facilities and flexibility). A significant percentage of survey participants indicated that software has good graphics (1992 survey), is in general easy to use (both surveys), but difficult to learn (1997 survey).

An analysis of the application areas of simulation reveals that manufacturing is dominant in both surveys. The main software limitations for survey participants in both surveys are limited software flexibility. Participants in 1992 survey also indicated that software is too slow, whilst participants in 1997 survey complained that software is difficult to learn and expensive. Other notable problems include validation difficulties (1992 survey), a lack of facility for output analysis (both surveys), inadequate guidance in experimentation, lack of software compatibility, problems with data input and the age of software packages (1997 survey). A complaint about the lack of flexibility is probably caused by a significant majority of both surveys' participants using simulators, which are believed to be less flexible than simulation or general purpose languages.

When positive software features are considered it is apparent that the main positive features are similar in both surveys, with slightly different order of importance. These features include the ease of model development, animation facilities and simulation modelling speed.

Desirable features that users would like included in software vary considerably within both groups of surveys participants, showing that software preferences are to a large extent matter of an individual taste and expectations. Some of the common features specified by participants in both surveys are better experimental support, facility for output analysis, better links with other packages (software compatibility) and further developments to make packages easier to learn and use. There are also various other desirable features specified by a small percentage of participants in both surveys, which is mainly a result of an open-ended style of questions that enabled users to express personal views and preferences. Some similarity in responses obtained in both surveys could indicate that many problems related to simulation software in 1992 still exist. .

6 Conclusions

The above presented results of simulation software users' surveys conducted in 1992 and in 1997 reveal to what extent the users are (were) satisfied with software and how they would like this software to be further improved. A general analysis of all results obtained shows that simulation software used by participants in these surveys is predominantly easy to use, with good visual facilities, but too limited for complex and non-standard problems, too expensive and incapable of providing adequate guidance in experimentation. A substantially dominant application area of simulation is manufacturing, although it is apparent that simulation can be used successfully in other domains such as business process modelling (Hlupic 1998). There are a variety of features that users have requested that dominantly refer to more assistance in experimental design, easier to learn and use packages and improved software compatibility. Although both surveys reveal that a major limitation specified is inadequate flexibility, only about 10% of participant in 1992 survey and 7.4% of participants in 1997 survey indicated this feature as desirable.

The results from both surveys indicate that there are significant similarities in users' responses, which could lead us to believe that many problems and limitations of simulation software indicated several years ago have not been eliminated. It is also apparent that no single package could possibly incorporate all desirable features, being at same time very easy to learn and use, inexpensive, with excellent graphical facil-

Software Features 1992 Survey	Perc of Users 1992 Survey	Software Features 1997 Survey	Perc of Users 1997 Survey
Too limited for complex problems	35.8	Lack of modelling facilities/flexibility	37
Good graphics	30.7	Extensive modelling facilities/flexibility	33.3
Slow	25.6	Easy to use Difficult to learn	18.5
Easy to use Generally very good	17.9	Expensive Good for teaching	14.8
Easy to learn Biased to manufacturing problems	15.3	Inexpensive and good value Easy to learn Dated Good for developing models of real systems	11.1
Poor statistical support User friendly	17.7	Good graphical interface Models are easy to develop Difficult to link to other software Lack of language interface Powerful tool	7.4
Difficult to validate models Inadequate experimentation facilities Difficult to use for non standard problems	15.1	Lack of good user interface Average modelling facilities Slow to run Good speed Poor logic facilities Simple Good automatic statistics collection Lack of hierarchical/modular approach Ease of animation Easy to use Good value for the price Presentable Easy to create reusable code Inadequate graphic front end It takes long to develop models	3.7
Quick Lack of good support for fluid processing	12.5		

Table 2: A summary of users' general opinion about software in both surveys

Software Limitations 1992 Survey	Perc of Users 1992 Survey	Software Limitations 1997 Survey	Perc of Users 1997 Survey
Restricted flexibility	33.3	Limited standard features/flexibility	22.2
Slow	17.9	Difficult to learn	14.8
		Expensive	
Validation difficulties	12.8	Inadequate guidance in experimentation	11.1
Lack of facility for output analysis	10.2	Lack of software compatibility	7.4
		Lack of output analysis facilities	
		Data input	
		Dated	
Difficult to use	7.7	High cost of support and training	3.7
Difficult to learn		Use of dongle	
Lack of facility for experimental design		Lack of on-line help	
Poor statistics		Lack of complex languages within the package	
Manufacturing bias and terminology problem		Inadequate processing power	
		Inadequate graphical accuracy	
		Lack of real-time accuracy	
		Poor logic	
		Too much distracting emphasis on animation	
		No access to system events	
		Poor facilities for developing own user interface	
		Slow	
		Difficulty of validating models	
		Lack of portability	
		Requires too much expertise	
		Output presentation	
		Inadequate user interface	
		No possibility for stand alone executable models	
		The lack of ability to build a modular type of simulation	
		Crude results package	
		Need to have more work entries in a model then correspond to reality	
		the use of dummy work-centres	
		Lack of the integration of scheduling and simulation packages	
		Initialisation	
Lack of database linkages	5.1		
Limits to the size of models			
Expensive			
Inadequate graphics	2.5		
Lack of a support for fluid processing			
Lack of support for object oriented concepts			
Big models are not understandable			

Table 3: A summary of users' opinion about the main limitations of software used in both surveys

Main Positive Features of Software	Perc of Users 1992 Survey 1992 Survey	Main Positive Features of Software	Perc of Users 1997 Survey 1997 Survey
Graphics (animation)	35.8	Ease of modelling	48.1
Ease of use	23	Good animation/visual facilities	44.4
Ease of learning	10.2	Modelling speed	22.2
Automatic report generation			
Interactivity	7.7	Flexibility/linking to	18.5
User support		external code	
User interface			
Speed of modelling			
Flexibility	5.1	Graphical interface	7.4
Documentation			
Good statistical			
analysis			
Interface with other software	2.5	Input and output analysis features	3.7
Support for UNIX platforms		features	
Incorporated cost analysis		Linking to other packages	
Easy check of 'what-if		Low price	
questions		Ease of statistics collection	
Cheap		Interactivity	
Being menu driven		Variable animation speed	
		Modularization of models	
		The results summary	
		VR functionality	
		Number crunching	
		Unlimited functionality via C	
		coding	
		Easy to create reusable code	
		Interactivity	
		Portability	

Table 4: A summary of users' opinion about the most important positive features of software used in both surveys

ity, extensive flexibility and standard features, and intelligent features for experimental design and output analysis. However, the desirable features specified by both surveys participants could be a useful indicator to software developers how to further improve simulation software by providing more flexibility achieved by less modelling efforts.

References

[1] Christy D.P. and Watson H.J. The Application of Simulation: A Survey of Industry Practice", *Interfaces*, Vol 13, No 15, October 1983, 47-52, 1983.

[2] Hlupic V. :Simulation Modelling and Business Process re-engineering: Bridging the Gap, *Proceedings of European Simulation Multiconference*, ESM'98, Manchester, June 1998. (Ed.by Zobel R. and Moeller D.), 641-645, SCS,1998.

[3] Kleine H.: A survey of users' views of discrete simulation languages", *Simulation*, May 1970, 225-229, 1970.

[4] Kleine H. :A survey of users' views of discrete simulation languages (the second survey), *Simulation*, August 1971, .89-94, 1971.

[5] Kirkpatrick P., Bell P.C.: Visual Interactive Modelling in Industry: Results form a Survey of Visual Interactive Model Builders, *Interfaces*, Vol 19, No 5, September-October 1989,.71-79, 1989.

[6] Mackulak G.T. and Savory P.A.: Ascertaining Important Features For Industrial Simulation Environments, *Simulation*, 63 (October 1994), 211-221, 1994.

[7] *Simulation News Europe*, No 23, July 1998, 36-49, 1998.

[8] Van Breedam A. :Segmenting the Simulation Software Market, *OR Insight*, Vol 3, No 2, April-June 1990, 9-13, 1990.

Desired Software Features	Perc of Users 1992 Survey	Desired Software Features	Perc of Users 1997 Survey
Better software compatibility Link to databases Link to spreadsheet Link to CAD software Link to statistical packages Link to MRP scheduling software	17.9	Better experimentation support	18.5
Facility for output analysis	12.8	Further developments making packages easier to learn and use Better links with other packages	11.1
More flexibility Help in experimental design	10.2	Better analysis of results and data displays Extensive standard features Internal system for creating user logic More but easier flexibility Output design and analysis Iconic programming/graphical model building Better presentation of the model on the screen and in the printout	7.4

Table 5: A summary of users' opinion about the features that should be included in simulation software in both surveys (part 1)

Desired Software Features	Perc of Users 1992 Survey	Desired Software Features	Perc of Users 1997 Survey
Better and more intelligent on line-help Better experimentation facilities Support of standard programming concepts	7.7	An intelligentinterface that would advise in number of replications, warm up period, batch size etc. Virtual reality Complete accuracy with the physical world Real-time animation Access to system events Good facilities for developing own user interface (to create sub-simulators) Facilities for batch running and collection of statistics Better Graphical User Interface Better statistical facilities Extra blocks to support various application areas Stand alone executable models Model pre-analyser Optimisation facilities Time series analysis Tutorial distribution fitting Library of reusable models Completeness check Multimedia features Interactive handling of parameters during execution of experiments Confidence intervals Hypothesis testing Graphical display of simulation output A several purpose library of facilities to extract ready-built components of simulation Ability to do IF/THEN/ELSE logic A facility to print out by one command, all the parameter values, object specifications and routings/logic within a model A cross referencing capability, that is providing ready answers to questions such as where are all references to a given attribute to be found	3.7

Table 6: A summary of users' opinion about the features that should be included in simulation software in both surveys (part 2)

Desired Software Features	Perc of Users 1992 Survey	Desired Software Features	Perc of Users 1997 Survey
Elimination of memory limitations	5.1		
Better documentation			
Easy model editing			
Dedicated systems for more specific applications			
Higher execution speed			
Ability to create run-time applications	2.5		
Automatic save			
More prompt to save			
Hierarchical model building			
Low cost of software			
Easy design of on-line reports			
Availability on standard hardware and software systems			
Improved editing facilities			
Removal of unnecessary constrains			
Enhancement of fluid processing facilities			
Automatic generation of entity cycle diagrams			

Table 7: A summary of users' opinion about the features that should be included in simulation software in both surveys (part 3)

More Efficient Functionality Decomposition in LOTOS[1]

Monika Kapus-Kolar
Jožef Stefan Institute, P.O.B. 3000, SI-1001 Ljubljana, Slovenia
Phone: +386 61 1773 531, Fax: +386 61 1262 102
E-mail: monika.kapus-kolar@ijs.si

Keywords: distributed service implementation, automated protocol synthesis, LOTOS

Edited by: Rudi Murn
Received: June 30, 1998 **Revised:** January 16, 1999 **Accepted:** May 18, 1999

An improved functionality-decomposition transformation for Basic LOTOS specifications is proposed which, given a specification of the required external behaviour (the expected service) of a system and a partitioning of the specified service actions among the system components, derives the behaviour of individual components implementing the service. There may be an arbitrary finite number of components, pairwise communicating by executing common actions and/or by exchanging messages over queues with infinite capacity.

1 Introduction

The top-down design strategy starts by identifying the required external behaviour of a system (its service), followed by gradual refinement of its internal structure. The crucial top-down design transformation is functionality decomposition, used to decompose a given process into a number of concurrent interacting subprocesses. It is widely used in many design cases, both for decomposing hardware and software, e.g. in design of circuits, computers, computer networks, distributed systems and telecommunication networks - for distributed implementation of various types of servers, controllers, testers etc. The implementation of a service of a particular layer in the standard reference model for open systems interconnection requires for example derivation of a suitable behaviour of the protocol entities of the layer.

By decomposition, an abstract system specification is refined into a less abstract specification better reflecting the inherent system structure, i.e. its functional components and their spatial distribution. Thereby a specification becomes more tractable for formal reasoning and implementation. The most elegant way is to start with a verified specification of the system service and then apply to it only verified correctness-preserving transformations, i.e. transformations refining the system structure without affecting its service, to avoid the need for posterior verification of the obtained detailed system specification.

If a system is specified in a formal language (Turner, 1993), correctness-preserving transformations can be expressed as well-defined syntactic manipulations that can usually be completely automated. It must be admitted that design by automated transformations is rather uncommon in the current engineering practice, but the reason is definitely not designers' aversion to such style of work. Rather it is the lack of transformations that are well formalized, proven to be correct, efficiently implemented, easy to use and understand and, above all, that correspond to practically useful design steps. In the future, high-quality tools for transformational design seem to be the best way to make formal languages and methods accessible to non-experts.

For discovering, understanding and employing correctness-preserving specification transformations, it is convenient if a formal language offers operators for composing specifications of system components directly representing real-life composition of the components. Such are for example languages based on process algebras, among which we concentrate on Basic LOTOS, the core sublanguage of LOTOS (Bolognesi and Brinksma, 1987), a standard specification language originally intended for specification of Open Systems Interconnection standards, but now widely employed for specification of all kinds of systems (both software and hardware) where external behaviour and/or inter-component interaction are of primary importance - see the Applications section in (WELL).

The problem addressed in our paper is system decomposition into a pre-defined number of components pairwise communicating by executing common actions and/or by exchanging messages over unbounded reliable first-in-first-out (FIFO) channels. It is assumed that all actions in the service specification are already pre-allocated to individual components, so that the task is only to derive specification of the inter-component communication, i.e. the protocol. The transformation is expected to be compositional, i.e. to

[1] A preliminary version of the paper appeared in the IEEE CS proceedings of the EUROMICRO'97 conference.

reflect the structure of the given service specification in the derived component specifications.

Saleh in his very exhaustive survey (Saleh, 1996) identifies 37 protocol synthesis methods, many of them further diversified into several variants. Among the 37 methods, 10 are based on LOTOS or similar models. For Basic LOTOS and multi-component servers with asynchronous internal channels, probably the most representative reference is (Kant et al., 1996), whose synthesis method has evolved from a long chain of similar methods, starting with (Bochmann and Gotzhein, 1986). For servers with synchronous channels, such a reference synthesizing earlier approaches is (Brinksma and Langerak, 1995), though its method is only intended for two-party servers. Thus in our paper we take (Kant et al., 1996) as the starting point, and (Brinksma and Langerak, 1995) as an additional source of ideas. We generalize (Kant et al., 1996) to servers with both synchronous and asynchronous channels, suggest how to correct the errors identified in (Kant et al., 1996) (one also in (Brinksma and Langerak, 1995)), and provide a hint for reducing the number of the necessary protocol interactions. Our paper is a corrected and enhanced version of (Kapus-Kolar, 1997), whose main deficiency is that the adopted specification language is semantically not exactly equivalent to Basic LOTOS, as it is here.

The paper is organized as follows. Section 2 introduces the adopted specification language. In Section 3 we explain the basic principles of protocol derivation. In Section 4 we propose distributed implementation for each individual type of service behaviour or subbehaviour. Section 5 includes discussion and conclusions.

The paper should be quite self-sufficient, though to keep it reasonably short, the presentations in the remaining sections are rather concise. Thus a reader unfamiliar with LOTOS-based protocol synthesis is advised to first study the more tutorial-like papers (Kant et al., 1996; Brinksma and Langerak, 1995). The two papers should also be referred to for discussion on earlier methods. Additional pointers to comparative studies can be found in (Saleh, 1996).

Before proceeding to details, let us once more try to motivate the reader by emphasizing that the algorithm described below is not intended solely for embedding in some CAD tool. The following sections provide many hints on how to systematically design correct and efficient distributed service implementations, that might be useful for the design of any multi-component system.

2 Specification Language

The language defined in Table 1 has been conceived with an aim to include in an abstract and concise way all constructs defined in Basic LOTOS (Bolognesi and Brinksma, 1987), in the exclusive setting of the protocol derivation problem formally defined in Section 3.1.

The following typographical convention has been adopted: If $\mathcal{X}$ is some universe of elements where "x" denotes its name and stands for any letter, then variables x, x', ..., x_1, ... range over elements of $\mathcal{X}$ and variables X, X', ..., X_1, ... over subsets of $\mathcal{X}$, if not stated otherwise. In particular, a b stands for a behaviour (for a process exhibiting it), a c for a system component, an a for a non-δ action, a g for an interaction gate or an action on it, a u for a user-defined action name, an s for **i** or a u (for a service-action name), an m for a protocol message, an r for a gate renaming, a p for the name of an explicitly specified process, an n for a process-parameter name, a v for a process-parameter value, and an e for a service specification subexpression. Let $b' \leq b$ and $b' < b$ respectively denote that b' is a subbehaviour or a proper subbehaviour of b.

We concentrate on the semantics of the language, informally overviewed below, but its syntax is intentionally kept simple, to simplify the presentation of protocol derivation. Throughout the paper, the usual self-understood forms of "syntactic sugar" are used where convenient, e.g. switching between the infix and the prefix notation, parentheses, omission of implicitly implied parts of the specification, etc. Several illustrative service and protocol specifications can be found in Sections 3 and 4.

stop denotes inaction of the specified process, e.g. of the system as a whole, of an individual system component, or of some other partial system behaviour.

Actions with reserved names δ (in the original LOTOS syntax specified by **exit**) and **i** respectively denote successful termination and an internal action of the specified process. In a service specification they are furnished with a superscript c indicating the component controlling their execution, but the superscript doesn't belong to the action name - the selection of c influences the protocol derivation algorithm, but is irrelevant for the service itself.

u^c denotes a service primitive, i.e. a type u interaction between a system user and the system component c.

If two components c and c' communicate synchronously (the condition encoded as $\neg FIFO(c, c')$), they can exchange a protocol message m in an interaction $\mathbf{sync}_{c,c'}!m$, where the order of c and c' is irrelevant. If a c and a c' communicate asynchronously (i.e. $FIFO(c, c')$), c can send an m to c' by a $\mathbf{send}_{c,c'}!m$, while c' receives the message by a $\mathbf{rec}_{c,c'}!m$. When a protocol action appears in a local behaviour specification, explicit specification of the location qualifier identifying the particular system component is unnecessary. The parameter m is supposed to be always specified as a constant, so that it can be considered a part of the action name. **sync**, **send** and **rec** actions

Name of the construct	Type(s)	Syntax
Inaction	b	**stop**
Successful termination	b	δ^c
Internal action	a,s^c	$\mathbf{i}^c$
Service primitive	a,g,s^c	u^c
Protocol synchronization	a,g	$sync_{c,c'}!m$ where $\neg FIFO(c,c')$
Protocol message transmission	a,g	$send_{c,c'}!m$ where $FIFO(c,c')$
Protocol message reception	a,g	$rec_{c,c'}!m$ where $FIFO(c,c')$
Sequential composition	b	$b_1 >> b_2$
Action prefix	b	$a;b_2$
Choice	b	$b_1[]b_2$
Parallel composition	b	$b_1\|G\|b_2$
Disabling	b	$b_1[> b_2$
Hiding	b	**hide** G **in** b_1
Gate renaming	r	$g \rightarrow g'$
Renaming	b	**ren** R **in** b_1
Process definition		$p(n) := b_1$
Process instantiation	b	$p(v)$

Table 1: The specification language abstract syntax

are only allowed in the derived protocol specifications.

"$b_1 >> b_2$" denotes a process behaving after successful termination of b_1 as b_2, where δ of b_1 is interpreted in "$b_1 >> b_2$" as **i**. "$a;b_2$" is the special case of the sequential composition where b_1 is an individual action, so that no **i** is needed for transfer of control to b_2.

"$b_1[]b_2$" denotes a process ready to behave as b_1 or as b_2.

"$b_1|G|b_2$" denotes parallel composition of processes b_1 and b_2. Actions listed in G and δ are only executable as common actions of the two processes, while other actions the processes execute independently. By connecting processes by "|||" or "||" one shortly specifies their minimal or maximal synchronization, respectively. Specification of a parallel composition of an empty list of processes is an empty string identifier ε, while parallel composition of a singleton set of processes equals the only member of the set.

"$b_1[> b_2$" denotes a process with behaviour b_1 potentially disrupted by behaviour b_2. While b_1 is still active, the process might terminate by executing δ in b_1.

"**hide** G **in** b_1" denotes a process behaving as b_1 with its actions listed in G renamed into **i**, i.e. the gates made internal to the process (hidden from its environment). Hiding of an action doesn't influence its location, i.e. hiding of a u^c results in $\mathbf{i}^c$.

"**ren** R **in** b_1" denotes a process behaving as b_1 with its visible gates renamed as specified in R.

$p(n) := b_1$ defines a process p with behaviour b_1, and a $p(v)$ defines an instantiation of the process. Parametrization of processes constituting a service specification is not allowed, while in the derived protocol specification, the local mappings of the processes might need input parameters. Process parametrization is not allowed in Basic LOTOS, but we don't consider that a problem, for parameters can usually be avoided, if so desired, by switching to a less concise specification style interpreting the parameter value as a part of the process name. Anyway, in the full LOTOS, parametrization is legal.

A *specification* is a list of process definitions. The first process on the list is supposed to denote the behaviour that the specification is defining, i.e. the *main* process. As such, it must never be instantiated within the specification. Specifically, let *Server* denote the main process in the specification of the service that is being implemented.

In the original LOTOS syntax, explicit processes are defined on formal gates, that are associated with actual gates upon process instantiation. In our simplified language, the gate instantiation can be expressed as renaming of the gates on which a process is originally defined applied to the particular process instance (see for example the instantiation of process "Proc" in Table 22).

The relation used throughout the paper for judging equivalence of behaviours is observational equivalence $\approx$ (Bolognesi and Brinksma, 1987), i.e. we are interested only into the external behaviour of processes, that is into the actions that they make available for synchronization with their environment (all actions except **i** and actions transformed into **i** by hiding).

The protocol derivation mapping defined below in some cases introduces an ε specifying no actions. ε is equivalent to δ (as execution of no actions is a success by definition), except for an additional absorption rule (introduced for the purposes of protocol synthesis) stating the $(\varepsilon >> b)$ is equivalent to b.

$Server := Loop$
$Loop := ((Idle[> (event^1; ((report^2; \delta^1)\|\|\|(report^3; \delta^1)))) >> Loop)$
$Idle := ((test^1; Idle)[](play^1; Idle))$
$\forall\{c, c'\} : FIFO(c, c')$
$Server_1 := ((Idle_1[> ((event^1; ((send_2!1; \delta)\|\|\|(send_3!1; \delta))) >> ((rec_2!2; \delta)\|\|\|(rec_3!3; \delta)))) >> Server_1)$
$Idle_1 := ((test^1; Idle_1)[](play^1; Idle_1))$
$Server_2 := (rec_1!1; report^2; send_1!2; Server_2)$
$Server_3 := (rec_1!1; report^3; send_1!3; Server_3)$

Table 2: An example service and its derived protocol

In many of the protocol examples given below, elimination of ε is not the only simplification modulo observational equivalence that the specifications have undergone. In addition, process parametrization has been omitted where redundant.

3 Principles of Protocol Derivation

3.1 Problem definition

The protocol derivation problem is defined as follows. Given

- the above described system of components and channels for protocol interactions, with all FIFO channels initially empty,
- a specification of the required system service (non-blocking, with no non-executable or otherwise irrelevant parts), and
- a suitable (defined by the restrictions in Section 4) partitioning of the specified actions among the system components,

derive such behaviour of individual components that, when the **sync**, **send** and **rec** actions are hidden, the overall system behaviour is observationally equivalent to the specified service and the server never stops with any of its FIFO channels non-empty.

An illustrative example of a service and its protocol is given in Table 2. The protocol has been derived as suggested below and subsequently simplified. There is a server consisting of asynchronously communicating components 1 to 3, supporting users 1 to 3, respectively. Whenever user 1 signals a particular event, the server reports it to users 2 and 3, and subsequently becomes ready for a new signal from user 1. Protocol messages of type 1, issued by component 1, serve for reporting the signalled event to components 2 and 3, while messages 2 and 3 confirm to component 1 that the event has been reported to users 2 and 3, respectively. While there is no event to report, the server idles, i.e. allows user 1 to play and execute tests, both locally at component 1.

3.2 Mapping T

We are looking for a mapping $\mathbf{T}_c(e, z)$ which would take any service specification subexpression e and translate it into its counterpart at any individual component c, within a given context z. The mapping of individual subexpression types is given in Section 4. A $\mathbf{T}_c(e, z)$ implements the service actions within e allocated to c and the necessary protocol interactions between c and the rest of the system components.

During service execution, each instantiation of an explicit process p gives raise to a new instance of any b within the process body. Since the aim of mapping a specification e of such a b is a proper implementation of each particular instance of b, z for mapping the e must be the identifier of the particular instance of p. The particular instance of b is then unambiguously identified by "$z.Z(e)$", where $Z(e)$ identifies e within the specification of p's body. Protocol optimization by re-use of instance identifiers shall not be systematically considered, though often possible.

3.3 The basic principles of protocol construction

Like (Kant et al., 1996) and previous similar algorithms, our algorithm is based on a small set of intuitive rules that can be easily expressed in an informal way:

1) Only the server components responsible for actions in the service specification should participate in the execution of the service.
2) An individual service action must always be implemented at the component to which it has been pre-assigned.
3) Every protocol message should unambiguously identify the behaviour that it helps to implement.
4) Service choices should always be resolved exclusively by service actions. Protocol actions only communicate the decisions between server components.
5) To simplify protocol derivation, we partition the service actions in such a way that
   - conflicts between distributed implementations of concurrent service parts are a priori avoided, and
   - all service choices can be resolved locally at individual components.

6) With the previous heuristics, *the sole purpose of protocol actions is to report on local terminations of individual service parts.*

3.4 Service behaviour attributes, particularly termination types

Mapping **T** is guided by various pre-calculated attributes of individual service subbehaviours, actually (speaking in terms of the specification syntax) of the service specification subexpressions by which they are represented. Likewise, the attributes are also defined for each of the explicitly specified processes constituting the service. Computation and selection of attributes is the key activity in the protocol derivation process. After we find a set of attributes that is both consistent and known to lead to an efficient protocol, the protocol derivation itself (i.e. application of mapping **T**) is trivial. (That doesn't mean that the mapping has also been trivial to conceive and prove!)

For any attribute $X(\ldots)$ whose value is a set of elements x, let $X_x(\ldots)$ indicate $x \in X(\ldots)$, and vice versa.

The basic attributes of a behaviour b are its starting components (SC), its ending components (EC) and its participating components (PC), respectively listing the system components executing a starting, an ending, or any service action within b. E.g., for a behaviour

$$b = (((((\mathrm{a}^1;\delta^3)|||(\mathrm{b}^2;\delta^3)) >> (\mathrm{c}^3;stop))[]\delta^1)||| \\ ((\mathrm{d}^1;\delta^1)[> (\mathrm{a}^1;\delta^4)))$$

we have $SC(b) = \{1,2\}$, $EC(b) = \{1,4\}$, $PC(b) = \{1,2,3,4\}$.

Boolean attribute $IT(b)$ indicates whether b might immediately terminate (the above b can not, while its part "$(\ldots[]\delta^1)$" in isolation could). Boolean attribute $ST(b)$ indicates whether b synchronizes its termination with its environment, i.e. its δ is not "consumed" by a sequential composition operator (as is for example δ^3 in the above b). Boolean attribute $DT(b)$ indicates whether a termination of b might be decision-making (as are in the above b both occurrences of δ^1).

All the above attributes are *generic properties of behaviours*. Solving a system of recursive equations for such an attribute, one should choose a solution minimizing all the process attributes, to respect the natural semantics of the attribute.

There are two more attributes, TC and TC^+, but they are not generic properties of behaviours. They are selected by a designer, to implement his/hers specific protocol derivation strategy. The exact role of TC and TC^+ is to dictate the *termination type* of individual service parts within the distributed service implementation, as follows:

An individual service action must of course be implemented at the component to which it is pre-allocated. At any other component, we propose that it is *selectively* mapped into a **stop** or an ε (semantically equivalent to δ). (Kant et al., 1996; Brinksma and Langerak, 1995) strictly map to ε, and consequently the quality of the derived protocol suffers.

Depending on the mapping of the ending actions of a particular b, in the case that the server is running a terminating alternative of b the behaviour $\mathbf{T}_c(b,z)$ of a component c concludes either by **stop** (not preceded by δ) or by δ. In the latter case, c is a terminating component of b, i.e. $TC_c(b)$. It is important that $\mathbf{T}_c(b,z)$ concludes for all terminating alternatives of b in the same manner, i.e. always by **stop** or always by δ.

$TC_c(b)$ is always a consequence of $TC_c^+(b)$ indicating that the surrounding context of b requires c to conclude its implementation of b by δ. Besides, $EC_c(b)$ implies $TC_c(b)$, for the ending components of a b are responsible for detecting its termination.

If $TC_c(b)$, $\mathbf{T}_c(b,z)$ terminates on its own and so allows c to *sequentially* proceed to the subsequent activities. That is the usual behaviour-termination type in LOTOS-based protocol derivation. If $\neg TC_c(b)$, $\mathbf{T}_c(b,z)$ concludes by inactivity, that is *disrupted* by a subsequent activity of c specified outside $\mathbf{T}_c(b,z)$. The *opportunity for protocol optimization* lies in the fact that setting $TC_c(b)$ to $false$ renders the protocol messages serving solely for termination of $\mathbf{T}_c(b,z)$ unnecessary. For better understanding, observe that for a service behaviour "$b_1 >> b_2$", δ in b_1 is invisible for the service users and hence its implementation is irrelevant, as long as control is properly transferred to the implementation of b_2.

As TC and TC^+ are attributes that are rather selected than computed, we don't provide strict rules for them - just the restrictions that must be respected are stated in Section 4. Within those limits, TC and TC^+ should be selected according to the relevant optimization criteria. An always present criterion is minimization of the (worst-case or average) inter-component communication. Another criterion, often conflicting with the former one, might be declaring for some service parts b and components c that $TC_c(b)$ is desirable, for c should terminate $\mathbf{T}_c(b,z)$ as soon as possible, e.g. to release some resources. A third criterion might be to make a pair of server components exchange a protocol message at a particular point of service execution, e.g. to convey some data. Moreover, scheduling of protocol exchanges on FIFO channels usually requires prevention of channel buffers overcrowding.

When trying to find a solution better than the old ($TC^+(b) = TC(b) = C$), one must be aware that in some cases a different solution might also result in a *less* efficient protocol, depending on subtle properties of the service structure. Thus the optimization should be performed by thorough analysis of the entire service specification. Nevertheless, one should never simply retreat to ($TC^+(b) = TC(b) = C$), for that might re-

$\mathbf{Send}_c(C,m)$	$:= \|\|\|(\{(send_{c'}!m;\delta)\|((c' \in (C \setminus \{c\})) \wedge FIFO(c,c'))\} \cup$ $\{(sync_{c'}!m;\delta)\|((c' \in (C \setminus \{c\})) \wedge \neg FIFO(c,c'))\})$
$\mathbf{Rec}_c(C,m)$	$:= \|\|\|(\{(rec_{c'}!m;\delta)\|((c' \in (C \setminus \{c\})) \wedge FIFO(c,c'))\} \cup$ $\{(sync_{c'}!m;\delta)\|((c' \in (C \setminus \{c\})) \wedge \neg FIFO(c,c'))\})$
$\mathbf{Exch}_c(C,C',m)$	$:=$ (if $(c \in C)$ then $\mathbf{Send}_c(C',m)$ else ε endif $\|\|\|$ if $(c \in C')$ then $\mathbf{Rec}_c((C \setminus \{c'\|((c \in C) \wedge (c' \in C') \wedge \neg FIFO(c,c'))\}),m)$ else ε endif)
$\mathbf{Proj}_c(G)$	$:= \{u^c\|(u^c \in G)\}$
$\mathbf{Proj}_c(R)$	$:= \{(u^c \to u'^c)\|((u^c \to u'^c) \in R)\}$
$\mathbf{Select}(C,z)$	:= a (within context z) deterministically selected element of C
$\mathbf{Term}_c(b,z)$	$:= \mathbf{Term'}_c(b,TC^+(b),z)$
$\mathbf{Term'}_c(b,C,z)$	$:=$ if $(EC(b)=\phi)$ then $\mathbf{T}_c(b,z)$ else if $EC_c(b)$ then $(\mathbf{T}_c(b,z) >> \mathbf{Send}_c((C \setminus TC(b)),z.Z(b)))$ else if $((c \in C) \wedge \neg TC_c(b))$ then if $(\|EC(b)\|=1)$ then $(\mathbf{T}_c(b,z)[> \mathbf{Rec}_c(EC(b),z.Z(b)))$ else $((\mathbf{T}_c(b,z)[> \delta)\|\|\mathbf{Rec}_c(EC(b),z.Z(b))$ endif else $\mathbf{T}_c(b,z)$ endif endif endif
$\mathbf{Alt}_c(b,b',z)$	$:= \mathbf{Alt'}_c(b,(PC(b') \cap TC(b')),z)$
$\mathbf{Alt'}_c(b,C,z)$	$:=$ if $(EC(b)=\phi)$ then $\mathbf{T}_c(b,z)$ else if $(c = \mathbf{Select}((TC^+(b) \cap PC(b)),Z(b)))$ then $(\mathbf{Term}_c(b,z) >> \mathbf{Send}_c((C \setminus (PC(b) \cup TC^+(b))),z.Z(b)))$ else if $((c \in C) \wedge \neg PC_c(b) \wedge \neg TC_c^+(b))$ then $\mathbf{Rec}_c(\mathbf{Select}((TC^+(b) \cap PC(b)),Z(b)),z.Z(b))$ else $\mathbf{Term}_c(b,z)$ endif endif endif

Table 3: Functions used in mapping **T**

sult in *protocol errors*, as explained in Section 4, if not to mention that for some service types, that straightforward solution is extremely inefficient (as demonstrated by the example in Table 7, discussed in Section 4.1).

To reduce the computational complexity, the rules for attribute evaluation in Section 4 are not exact, in the sense that they sometimes assume that the protocol derivation algorithm must pay attention to a service scenario that a more careful examination of the service specification would identify as non-executable. Particularly the rules neglect the positive impact of synchronization between parallel behaviours. Consequently, the decomposition transformation might generate some redundant protocol interactions, or the transformation might be unjustly declared inapplicable to a particular action partitioning. In other words, more exact attribute evaluation rules would result in a more generally applicable transformation generating more efficient protocols. But as the proposed rules are strictly pessimistic, they are nevertheless sufficient for protocol correctness (with the harmless exception that the generated component specifications might comprise some parts that are non-executable within the context of the system), provided that the mapping **T** itself is correct.

Of course, computation of attributes might also fail, implying that it is impossible to satisfy all the given restrictions simultaneously, or result in a protocol not satisfying the adopted optimization criteria. In that case, one should try to find a better service action partitioning, or artfully insert in the service specification some dummy internal actions (as long as the service remains observationally equivalent to the original).

3.5 Some auxiliary specification-generating functions, particularly implementation of terminations

Table 3 defines a set of auxiliary specification-generating functions for the mapping **T**. The first two functions implement sending or receiving at a c of a message m between c and any c' in $(C \setminus \{c\})$, independently for each c'. $\mathbf{Exch}_c(C,C',m)$ implements the actions of c necessary to perform the task of each member of C' receiving m from each member of C other than itself. Since message exchanges in **sync** actions are bi-directional, that allows some optimizations.

$\mathbf{Proj}_c(G)$ and $\mathbf{Proj}_c(R)$ respectively project onto the gates of c a set of gates and a set of renamings. $\mathbf{Select}(C,z)$ deterministically selects within context z a component in C.

If the surrounding context of b requires a c to conclude its implementation of the terminating alternatives of b (if any) by δ, formally $TC_c^+(b)$, that might be implemented by the mapping $\mathbf{T}_c(b,z)$ itself, i.e. c included into $TC(b)$. Alternatively, it might be better to have $\neg TC_c(b)$ and to make c exit its implementation of b upon reception of special messages (sent by

$b = (((a^1;\delta^1)[](b^1;c^2;d^1;\delta^1))\|\|\|((a^1;\delta^1)[](b^1;c^2;d^1;\delta^1)))$, $\neg FIFO(1,2)$
if $TC_2(b)$ then $\mathbf{T}erm'_1(b,\{1,2\},z) := (((a^1;sync!z.1;\delta)[](b^1;sync!z.2;sync!z.3;d^1;\delta))\|\|\|$ $((a^1;sync!z.4;\delta)[](b^1;sync!z.5;sync!z.6;d^1;\delta)))$
$\mathbf{T}erm'_2(b,\{1,2\},z) := (((sync!z.1;\delta)[](sync!z.2;c^2;sync!z.3;\delta))\|\|\|$ $((sync!z.4;\delta)[](sync!z.5;c^2;sync!z.6;\delta)))$
else $\mathbf{T}erm'_1(b,\{1,2\},z) := ((((a^1;\delta)[](b^1;sync!z.2;sync!z.3;d^1;\delta))\|\|\|$ $((a^1;\delta)[](b^1;sync!z.5;sync!z.6;d^1;\delta))) >> (sync!z.7;\delta))$
$\mathbf{T}erm'_2(b,\{1,2\},z) := (((sync!z.2;c^2;sync!z.3;stop)\|\|\|(sync!z.5;c^2;sync!z.6;stop))$ $[> (sync!z.7;\delta))$ endif

Table 4: A service behaviour b and some of its possible implementations wrt. its termination type

$b = (a^1;((b^3;\delta^3)\|\|\|(c^4;\delta^4)))$, $\forall\{c,c'\} : FIFO(c,c')$, $TC^+(b) = \{1,3,4\}$, $TC(b) = \{3,4\}$
$\mathbf{A}lt'_1(b,\{1,2,3\},z) := (((a^1;((send_3!z.2;\delta)\|\|\|(send_4!z.2;\delta)) >> stop)[> \delta)$ $\|\|\|((rec_3!z.1;\delta)\|\|\|(rec_4!z.1;\delta)))$
$\mathbf{A}lt'_2(b,\{1,2,3\},z) := (rec_3!z.1;\delta)$
$\mathbf{A}lt'_3(b,\{1,2,3\},z) := (rec_1!z.2;b^3;send_1!z.1;send_2!z.1;\delta)$
$\mathbf{A}lt'_4(b,\{1,2,3\},z) := (rec_1!z.2;c^4;send_1!z.1;\delta)$

Table 5: Illustration of function $\mathbf{A}lt'$

the ending components of b) together indicating global termination of b. Function **Term**, an extension of mapping **T**, implements the additional protocol interactions. Function $\mathbf{T}erm'$ is a generalization of **Term** that signals the global termination to a component set C.

For illustration of function **Term**, consider the service behaviour b in Table 4 with requirement $TC_2^+(b)$. If we set $TC_2(b)$ to *true*, component 2 implements all the "a" actions as ε. Consequently, whenever component 1 selects an "a" alternative, it must send a special indication to component 2, for component 2 does not participate in such an alternative and needs a message for its detection and subsequent proper local termination. In the worst case, execution of b requires two such messages. If we set $TC_2(b)$ to *false* instead, component 2 simply executes its part of b and then stops. Later its inactivity is disrupted by a message (a single one!) implemented by **Term** applied to b. Obviously, $\neg TC_2(b_1)$ is the more efficient solution.

Function **Alt** is an extension of function **Term** in the sense that *one* of the components knowing that an alternative b of a b' has been activated (and terminated) indicates the fact to the yet non-informed participants of b' that need the information. Function $\mathbf{A}lt'$ is a generalization of **Alt** that sends the information to a component set C, where necessary. Note that both $\mathbf{A}lt'$ and $\mathbf{T}erm'$ can serve for adding δ to implementation of a b at a c, but there is a slight difference: $\mathbf{T}erm'$ implements δ upon c receiving messages from *all* the ending components of b and is thus also suitable if $(\mathbf{T}_c(b,z) \neq stop)$ or if c must detect *global* termination of b. Otherwise c may enable δ already after receiving a *single* message from a *selected* participant of b (function $\mathbf{A}lt'$), for the reception may disrupt $\mathbf{T}_c(b,z)$, supposed to be equivalent to **stop**, at any time.

The effects of function $\mathbf{A}lt'$ are illustrated in Table 5. Because of $(TC_1^+(b) \wedge \neg TC_1(b))$, component 1 detects global termination of b by receiving messages from the ending components 3 and 4, introduced by the **Term** part of $\mathbf{A}lt'$. On the other hand we have $(\neg TC_2^+(b) \wedge \neg PC_2(b))$, thus $\mathbf{A}lt'$ makes component 3 (selected among {1,3,4}) indicate activation of b to component 2. 1 and 3 in the second argument of $\mathbf{A}lt'$ are irrelevant, since both components participate in b, and component 3 even terminates on its own.

3.6 Assumptions (obligations) for mappings T, $\mathbf{T}erm'$ and $\mathbf{A}lt'$

This section is only intended for readers deeply interested in technical details, while others are advised to refer to it only if having difficulties with the understanding of Section 4.

Mapping **T** is compositional, i.e. a mapping of a behaviour b is defined in terms of the mappings of its subbehaviours b'. Thus when designing $\mathbf{T}_c(b,z)$, we always make some assumptions for each individual $\mathbf{T}_c(b',z)$. One level higher, analogous statements appear as proof obligations for $\mathbf{T}_c(b,z)$. As mapping $\mathbf{T}erm'$ or $\mathbf{A}lt'$ shall often be used instead of **T**, the obligations (suitably adapted) also apply to them.

Let $Func(b,Arg)$ denote behaviour of the considered distributed system where every component c behaves like $Func_c(b,Arg)$, where $Func$ is **T**, $\mathbf{T}erm'$ or $\mathbf{A}lt'$ and Arg are the remaining arguments of the mapping. Let $Func(b,Arg)^*$ denote $Func(b,Arg)$ with protocol interactions hidden. Below we list the correctness criteria adopted for the system be-

haviour $Func(b, Arg)$ and the individual component behaviours $Func_c(b, Arg)$ within its context, assuming that all the FIFO channels are initially empty. Most of the rules are just common sense or can be understood in the light of the protocol derivation guidelines stated so far, while the others support implementation of various individual behaviour composition operators, explained in Section 4. The obligations are indeed numerous, but the correctness proof (Kapus-Kolar, 1998) for the protocol derivation transformation has identified them as necessary, as will the reader if trying to thoroughly understand the mappings in Section 4.

1) If $\neg PC_c(b)$, then
$$(((Func = \mathbf{T}) \wedge \neg TC_c(b)) \vee ((Func(b, Arg) = \mathbf{T}erm'(b, C, \ldots)) \wedge (c \notin (TC(b) \cup C))) \vee ((Func(b, Arg) = \mathbf{A}lt'(b, C, \ldots)) \wedge (c \notin (TC^+(b) \cup (C \setminus PC(b)))))$$
implies $Func_c(b, Arg) = stop$ and $TC_c(b)$ implies $Func_c(b, Arg) = \varepsilon$.
2) Any visible action offered by $Func_c(b, Arg)$ is either a service action pre-allocated to c within b, a protocol interaction associated with termination of a $b' \leq b$, or δ.
3) In b, every $b' \leq b$ is assigned its unique identifier, that is present in every protocol message associated with termination of b'.
4) If $(((Func = \mathbf{T}) \wedge TC_c(b)) \vee ((Func(b, Arg) = \mathbf{T}erm'(b, C, \ldots)) \wedge (c \in (TC(b) \cup C))) \vee ((Func(b, Arg) = \mathbf{A}lt'(b, C, \ldots)) \wedge (c \in (TC^+(b) \cup (C \setminus PC(b))))))$,
then $Func_c(b, Arg)$ terminates in all the terminating alternatives of b, otherwise never terminates.
5) Within $Func(b, Arg)$, for any terminating alternative of b, any $Func_c(b, Arg)$ not terminating for the alternative enters inaction before the last of the ending components c' terminates $Func_{c'}(b, Arg)$.
6) $Func(b, Arg)$ inherits from b its starting actions, with the exception that a starting δ in b is considered equivalent to a starting $\mathbf{i}$ in $Func(b, Arg)^*$, provided that there is a c guarding both the δ and the $\mathbf{i}$.
7) Within $Func(b, Arg)$, for any terminating alternative of b, any $c \in SC(b)$ is activated before the last of the ending components c' terminates $Func_{c'}(b, Arg)$.
8) Pretending that δ of $Func(b, Arg)$ is already executable when enabled by all c having it in $Func_c(b, Arg)$, $Func(b, Arg)^* \approx b$, with the exception that enabling of δ is allowed to be an internal decision of $Func(b, Arg)^*$ if $\neg ST(b)$. If b is the entire service, exact observational equivalence is required.
9) For each c, $Func_c(b, Arg)$ contains only actions that are executable within $Func(b, Arg)$.
10) No $Func_c(b, Arg)$ ever requires that the incoming protocol messages on FIFO channels are received in an order other than that of their transmissions within $Func(b, Arg)$.
11) Any protocol message sent within $Func(b, Arg)$ is also received within it.

$Server := (((\mathrm{a}^1; Proc)[](\mathrm{b}^1; \delta^1)) >> (\mathrm{b}^3; \delta^3))$ $Proc := (\mathrm{c}^2; Proc)$ $\forall\{c, c'\} : FIFO(c, c')$
$Server_1 := (((\mathrm{a}^1; send_2!1; stop)[](\mathrm{b}^1; \delta)) >> ((send_2!2; \delta)\|\|\|(send_3!2; \delta)))$
$Server_2 := (((rec_1!1; Proc_2)[> (rec_1!2; \delta)))$ $Proc_2 := (\mathrm{c}^2; Proc_2)$
$Server_3 := (rec_1!2; \mathrm{b}^3; \delta)$

Table 7: An example implementation of a service combining finite and infinite alternatives

4 Distributed Implementation of Individual Service Specification Subexpression Types

In this section we describe distributed implementation of individual service specification subexpression types. Table 6 summarizes the rules valid for any subexpression of type b, while the more specific rules are given in separate tables, explained in Sections 4.2 to 4.8. Each of the tables is typically divided into four sections: 1) definition of the syntax of the expression e that is being mapped, 2) attribute calculation rules, 3) (optional) additional restrictions on e or its attributes, and 4) mapping $\mathbf{T}$ for e. Studying $\mathbf{T}_c(b, z)$ in the tables with the specific rules, the reader may with no harm pretend that the only server components are those participating in b. The explanations of the transformations have also been conceived from that viewpoint.

4.1 Rules applying to all behaviour types

The generally applicable rules in Table 6 deserve some explanation.

Rule $((EC(b) = \phi) \Rightarrow (TC^+(b) = \phi))$ prevents server components from interpreting a non-terminating b as terminating (i.e. no c can ever terminate, if $(EC(b) = \phi)$, for that might result in a protocol error, if b is alternative to a terminating b' (Kapus-Kolar et al., 1991). With that rule, *it is no longer necessary to report every process instantiation within the service to every component*, as it is in (Kant et al., 1996). An illustrative example is given in Table 7. In comparison with the solution suggested in (Kant et al., 1996), there is an infinite saving in protocol messages - two per every instantiation of "Proc".

if $(EC(b) = \phi)$ then $TC^+(b) = \phi$ else if $\exists b' : ((b < b') \wedge \neg PC_c(b'))$ then $\neg TC_c^+(b)$ else $TC_c^+(b)$ defined in the corresponding one of the Tables 10,12,13,15,17,20 endif endif
$(DT(b) \wedge ST(b)) \Rightarrow ((\lvert EC(b)\rvert = 1) \wedge (TC^+(b) = EC(b)))$
$EC(b) \subseteq (TC^+(b) \setminus (PC(b) \setminus EC(b))) \subseteq TC(b) \subseteq TC^+(b)$
if $\neg PC_c(b)$ then if $TC_c(b)$ then $\mathbf{T}_c(b,z) := \varepsilon$ else $\mathbf{T}_c(b,z) :=$ *stop* endif else $\mathbf{T}_c(b,z)$ defined in the corresponding one of the Tables 9,10,12,13,15,17-19,20 endif

Table 6: Rules valid for any service subexpression b introduced for the purpose of protocol synthesis

The second row of Table 6 requires that a behaviour b whose termination might be both decision-making and synchronized has a single ending component that is also the only component regarding b as terminating. To understand the requirement, observe the service behaviour

$b = (((\mathrm{a}^1;\delta^1)[>(\mathrm{i}^1;\mathrm{b}^2;\delta^1))|[\mathrm{b}^2]|(\delta^1[](\mathrm{i}^1;\mathrm{b}^2;\delta^1))$

$e = b =$ *stop*
$SC_c(b) = EC_c(b) = PC_c(b) = false$
$IT(b) = DT(b) = false$

Table 8: The specific rules for **stop**

$e = b = \delta^{c'}$	
$SC_c(b) = EC_c(b) = PC_c(b) = (c = c')$	
$IT(b) = true$	$DT(b) = false$
$\mathbf{T}_{c'}(e,z) := b$	

Table 9: The specific rules for $\delta^{c'}$

$e = b = b_1[]b_2$
$SC_c(b) = (SC_c(b_1) \vee SC_c(b_2))$
$EC_c(b) = (EC_c(b_1) \vee EC_c(b_2))$
$PC_c(b) = (PC_c(b_1) \vee PC_c(b_2))$
$IT(b) = (IT(b_1) \vee IT(b_2))$
$ST(b_1) = ST(b_2) = ST(b)$
$DT(b) = (DT(b_1) \vee DT(b_2) \vee IT(b))$
$TC_c^+(b_1) = (TC_c(b) \wedge PC_c(b_1))$ $TC_c^+(b_2) = (TC_c(b) \wedge PC_c(b_2))$
$\exists c' : (SC(b_1) = SC(b_2) = \{c'\})$
$\mathbf{T}_c(e,z) := (\mathbf{Alt}_c(b_1,b,z)[]\mathbf{Alt}_c(b_2,b,z))$

Table 10: The specific rules for choice

of the form "$((b_1[> b_2)|[\mathrm{b}^2]|b_3)$". δ of b_1 is decision-making for $(b_1[> b_2)$ and synchronized with δ of b_3, hence both δ *must be controlled by the same component* (in our case, it is the component 1). Thus

$((\lvert EC(b_1)\rvert = 1) \wedge (TC(b_1) = EC(b_1) = EC(b_3)))$.

Moreover, as δ of $\mathbf{T}_1(b_1,z)$ and the corresponding δ of $\mathbf{T}_1(b_1[> b_2,z)$ are guarded by δ of $\mathbf{T}_1(b_3,z)$, component 1 *can't report them separately* (only upon δ of $\mathbf{T}_1(b,z)$), necessitating

$(TC^+(b_1) = TC^+(b_1[> b_2) = TC(b_1))$.

Hence the considered rule applies to "$b_1[> b_2$". Also, $(\lvert TC^+(b_1[> b_2)\rvert = 1)$ (not to speak of a decision-making δ within b_3) implies

$(TC(b) = EC(b) = EC(b_1))$,

i.e. δ of b must be in all its alternatives controlled by component 1 (and indeed it is). (Kant et al., 1996; Brinksma and Langerak, 1995) ignore the fact that a decision-making δ of b_1 within a "$b_1[> b_2$" might be synchronized, thus they sometimes generate erroneous protocols. The two methods can only be amended by getting rid of the rule $(PC_c(b) \Rightarrow TC_c^+(b))$, as we have done.

In the third row of Table 6, the not yet explained idea is that $\neg PC_c(b)$ should imply $(TC_c^+(b) = TC_c(b))$. That is because a non-participant c of a b should not participate in $\mathbf{Term}_c(b,z)$, just as not in $\mathbf{T}_c(b,z)$, as stated in the last row of the table.

The $\mathbf{T}_c(b,z)$ rule in Table 6 implies that if $\neg PC_c(b)$, $TC_c^+(b')$ for any $b' < b$ can be $false$ by definition.

4.2 Implementation of inaction and termination

The specific rules for **stop** and δ^c are defined in Table 8 and Table 9, respectively.

4.3 Implementation of choice

For implementation of choice (Table 10) we adopt the usual restriction (Bochmann and Gotzhein, 1986) that alternatives must have an unique and common starting place. Provided that distributed implementation of individual alternatives is communication-closed and preserves their starting actions, the choice is entirely local.

Let b_1 be the selected alternative. After its execution, one of its ending components proceeds to informing on the selected alternative the missing terminating participants of b. That is implemented by using function **Alt** instead of **Term**. If no message is sent to a missing participant c of b, $\mathbf{T}_c(b,z)$ is equivalent to $\mathbf{T}_c(b_2,z)$, that is in the case of b_1 not enabled, so that

$b = ((\mathrm{a}^1;\mathrm{b}^2;\delta^2)[](\mathrm{c}^1;\mathrm{d}^3;\mathrm{e}^1;\delta^1))$, $FIFO(c,c') = (\{c,c'\} \in \{1,2\})$, $TC(b) = \{1,2\}$
$\mathbf{T}_1(b,z) := ((\mathrm{a}^1;sync_2!z.1;\delta)[](\mathrm{c}^1;send_3!z.2;rec_3!z.3;\mathrm{e}^1;sync_2!z.4;\delta))$
$\mathbf{T}_2(b,z) := ((sync_1!z.1;\mathrm{b}^2;\delta)[](sync_1!z.4;\delta)$
$\mathbf{T}_3(b,z) := (rec_1!z.2;\mathrm{d}^3;send_1!z.3;stop)$

Table 11: An example implementation of choice

$e = b = b_1 >> b_2$		
$SC_c(b) = SC_c(b_1)$	$EC_c(b) = EC_c(b_2)$	$PC_c(b) = (PC_c(b_1) \vee PC_c(b_2))$
$IT(b) = IT(b_1)$	$DT(b) = DT(b_2)$	$ST(b_1) = false$, $ST(b_2) = ST(b)$
$((\neg PC_c(b_2) \wedge TC_c(b_2)) \vee EC_c(b_1)) \Rightarrow TC_c^+(b_1) \Rightarrow ((EC(b) = \phi) \vee TC_c(b) \vee EC_c(b_1) \vee SC_c(b_2))$		
$TC_c^+(b_2) = TC_c(b)$		
$\mathbf{T}_c(e,z) :=$ if $TC_c^+(b_1)$ then $(\mathbf{Term}_c(b_1,z) >> \mathbf{Exch}_c(EC(b_1), SC(b_2), z.Z(b_1)) >> \mathbf{Term}_c(b_2,z))$ else if $SC_c(b_2)$ then $(\mathbf{Term'}_c(b_1,\{c\},z) >> \mathbf{Term}_c(b_2,z))$ else $(\mathbf{T}_c(b_1,z)[> \mathbf{Term}_c(b_2,z))$ endif endif		

Table 12: The specific rules for sequential composition

the component waits for a disrupting message issued at some point after completion of b.

In the example in Table 11, component 2 is a missing terminating participant of the second alternative, thus it is notified of its execution. On the other hand, component 3 does not participate in the first alternative, but is not required to terminate it, thus it receives no notification.

Initial (decision-making) terminations of b are no longer forbidden (see Table 16), as they are in (Brinksma and Langerak, 1995). That is because each initial δ is now controlled by the component c' making the choice and guards termination of $\mathbf{T}_c(b,z)$ for any other $c \in PC(b)$.

4.4 Implementation of sequential composition and action prefix

Implementation of sequential composition is specified in Table 12. Proper sequencing of b_1 and b_2 requires that each $c \in EC(b_1)$ reports (by a message $z.Z(b_1)$) termination of $\mathbf{T}_c(b_1,z)$ to each $c' \in SC(b_2)$ (Kant et al., 1996). An analogous solution is employed for implementing action prefix (Table 13).

1) A $c \in TC^+(b_1)$ executes $\mathbf{Term}_c(b_1,z)$, then its part of the exchanges of message $z.Z(b_1)$ sent from $EC(b_1)$ to $SC(b_2)$, and finally $\mathbf{Term}_c(b_2,z)$.
2) If $(\neg TC_c^+(b_1) \wedge SC_c(b_2))$, the receptions of $z.Z(b_1)$ at c are implemented by upgrading $\mathbf{T}_c(b_1,z)$ into $\mathbf{Term'}_c(b_1,\{c\},z)$; afterwards, c proceeds to $\mathbf{Term}_c(b_2,z)$.
3) If $(\neg TC_c^+(b_1) \wedge \neg SC_c(b_2))$, c concludes $\mathbf{T}_c(b_1,z)$ by inaction, that might be disrupted by $\mathbf{Term}_c(b_2,z)$ or some later activity.

The various situations are illustrated by the example in Table 14, where one can also observe a situation of b_1 having both a terminating and a non-terminating alternative.

As terminations of b_1 are not terminations of b, there is no strict rule for $TC_c^+(b_1)$, but we require that

1) $(\neg PC_c(b_2) \wedge TC_c(b_2))$ implies $TC_c^+(b_1)$, for otherwise $\mathbf{Term}_c(b_2,z)$, unguarded in $\mathbf{T}(b,z)$, could prematurely disrupt $\mathbf{T}_c(b_1,z)$, and
2) $TC_c^+(b_1)$ implies $((EC(b) = \phi) \vee TC_c(b) \vee EC_c(b_1) \vee SC_c(b_2))$,

to prevent disruption of $\mathbf{Term}_c(b_1,z)$ by actions upon termination of b in the alternatives where c doesn't participate in b_2.

4.5 Implementation of disabling

Implementing disabling (Table 15) we encounter similar problems as when implementing choice. The starting actions of the disrupting behaviour b_2 are alternatives to the actions (including δ) in the potentially disrupted b_1, thus we require (Brinksma and Langerak, 1995) b_2 to have an unique starting component c' that is also the only participant of b_1. (Kant et al., 1996) has tried to avoid the restriction, but unsuccessfully, with *potential protocol errors* (Kapus-Kolar, 1999). The difficult problem is that it is not sufficient to individually implement b_1 and b_2, if b_1 is terminating. We need a special termination scheme, like the one explained in the following.

If b_2 is activated, a termination of b is always a termination of b_2, and thus properly detected by all the participants of b needing the information, since by the adopted restriction, $(PC(b) = PC(b_2))$. However, if b_1 terminates without being disrupted by b_2, c' must subsequently report, where necessary, the termination to the other participants of b. For a receiving c'', the situation is exactly as in the case of "$b_1[]b_2$". At c', however, the transmission must not be attached sequentially to the b_1 part (as in (Kant et al., 1996)), because in that position it would be disruptable by the b_2 part (Kapus-Kolar, 1999). Hence it must be

$e = b = s^{c'}; b_2$			
$SC_c(b) = (c = c')$	$EC_c(b) = EC_c(b_2)$	$PC_c(b) = ((c = c') \vee PC_c(b_2))$	
$IT(b) = false$	$DT(b) = DT(b_2)$	$ST(b_2) = ST(b)$	$TC_c^+(b_2) = TC_c(b)$
$\mathbf{T}_c(e,z)$:=if $(c = c')$ then $(s^{c'}; (\mathbf{Send}_c(SC(b_2), z.Z(b_1)) >> \mathbf{Term}_c(b_2, z)))$ else if $SC_c(b_2)$ then $(\mathbf{Rec}_c(\{c'\}, z.Z(b_1)) >> \mathbf{Term}_c(b_2, z))$ else $\mathbf{Term}_c(b_2, z)$ endif endif			

Table 13: The specific rules for action prefix

$b = (((\mathrm{a}^1; \mathrm{b}^3; ((\mathrm{c}^1; \delta^1) \|\|\| (\mathrm{d}^2; \delta^2))) [] (\mathrm{e}^1; \mathrm{f}^4; stop)) >> ((\mathrm{g}^1; \mathrm{h}^4; \mathrm{j}^1; \delta^1) \|\|\| (\mathrm{k}^2; \delta^2)))$
$FIFO(c,c') = (\{c,c'\} = \{1,2\})$, $TC(b) = \{1,2,3,4\}$
$\mathbf{T}_1(b,z) := (((\mathrm{a}^1; send_3!z.1; \mathbf{rec}_3!z.2; \mathrm{c}^1; \delta) [] (\mathrm{e}^1; send_4!z.3; stop))$ $>> (sync_2!z.4; \mathrm{g}^1; send_4!z.5; \mathbf{rec}_4!z.6; \mathrm{j}^1; \delta))$
$\mathbf{T}_2(b,z) := (\mathbf{rec}_3!z.2; \mathrm{d}^2; sync_1!z.4; \mathrm{k}^2; \delta)$
$\mathbf{T}_3(b,z) := (\mathbf{rec}_1!z.1; \mathrm{b}^3; ((send_1!z.2; \delta) \|\|\| (send_2!z.2; \delta)))$
$\mathbf{T}_4(b,z) := ((\mathbf{rec}_1!z.3; \mathrm{f}^4; stop) [> (\mathbf{rec}_1!z.5; \mathrm{h}^4; send_1!z.6; \delta))$

Table 14: An example implementation of sequential composition

executed after c' exits both parts, implying that the message is also sent if b_2 is executed (Brinksma and Langerak, 1995). Therefore $\neg TC_{c''}^+(b_1)$, for the message reports completion of b rather than b_1.

If c' is also the only ending component of b_2, c'' needn't be a terminating component of b_2 and interprets the reception as a disruption of $\mathbf{T}_c(b_2, z)$ (see the example in Table 16). In the opposite case, components (including c') must first terminate b_2 (if currently active), before sequentially proceeding to synchronization upon completion of b (Brinksma and Langerak, 1995). However, the second solution is only applicable if for every $c'' \in ((TC(b) \cap PC(b)) \setminus \{c'\})$, all the initial actions of $\mathbf{Term}_{c''}(b_2, z)$ are also receptions of messages sent by c'. The simplest way to secure that is to require $|PC(b)| = 2$, as it is in (Brinksma and Langerak, 1995).

Initial (decision-making) terminations of b_2 are no longer forbidden, as they are in (Brinksma and Langerak, 1995). That is because each initial δ is now controlled by c' and guards termination of $\mathbf{T}_c(b, z)$ for any other $c \in PC(b)$.

4.6 Implementation of parallel composition

Parallel composition of service parts (Table 17), regardless of the extent of their synchronization, introduces no additional protocol messages. Each component simply executes in parallel its local implementations of individual service parts, locally synchronized as specified by the parallel composition operator. To minimize communication costs, we allow separate termination optimization of individual parallel service parts.

If all components communicate synchronously (as for example in process "First" in Table 22), the *cross-cut theorem* (Eijk, 1990) for re-grouping of parallel processes applies: One may pretend that it is not that protocol interactions in the implementations of b_1 and b_2 share internal system channels and differ only in the message contents, but that the message contents is also a part of the channel name, i.e. that the two implementations use different system channels. Hence as far as synchronization between the service actions in b_1 and b_2 is ignored, the distributed server runs the parallel service parts' implementations independently. Moreover, synchronization upon a service action between the parallel service parts is at sole discretion of the component executing the action, i.e. a local matter.

(Kant et al., 1996; Brinksma and Langerak, 1995; Kapus-Kolar, 1997) *erroneously* (Kapus-Kolar, 1999) presume that the cross-cut theorem also applies to asynchronous communication. The simplest way to establish virtual independence between the implementations of b_1 and b_2 in the presence of FIFO channels is to require that there is no asynchronously communicating pair of components belonging to $(PC(b_1) \cap PC(b_2))$. (For example, for process "First" in Table 22 it is crucial that $\neg FIFO(1,2)$.) If the parallel composition is pure interleaving, however, the above restriction is not necessary (as for example in process "Second" in Table 22). That is because the components are known to always be able to receive protocol messages in the global order in which they have been sent, in both the implementation of b_1 and of b_2 (Kapus-Kolar, 1998)

4.7 Implementation of hiding and renaming

Table 13 indicates that the only property of an action s^c that is relevant for mapping $\mathbf{T}$ is its location. Hence hiding commutes with $\mathbf{T}$ and its implementa-

$e = b = b_1[> b_2$	
$SC_c(b) = (SC_c(b_1) \vee SC_c(b_2))$	$EC_c(b) = (EC_c(b_1) \vee EC_c(b_2))$
$PC_c(b) = (PC_c(b_1) \vee PC_c(b_2))$	$ST(b_1) = ST(b_2) = ST(b)$
$IT(b) = (IT(b_1) \vee IT(b_2))$	$DT(b) = ((EC(b_1) \neq \phi) \vee IT(b_2) \vee DT(b_2))$
$TC_c^+(b_1) = EC_c(b_1)$; if $(EC(b) = PC(b_1))$ then $TC_c^+(b_2) = PC_c(b_1)$ else $TC_c^+(b_2) = TC_c(b)$ endif	
$\exists c' : ((PC(b_1) = SC(b_2) = \{c'\}) \wedge$ $((EC(b) \subseteq \{c'\}) \vee (((TC(b) \cap PC(b)) \setminus \{c'\}) = \phi) \vee (\lvert PC(b)\rvert = 2)))$	
$\mathbf{T}_c(e,z) :=$ if $(EC(b_1) = \phi)$ then if $PC_c(b_1)$ then $(b_1[> \mathbf{Term}_c(b_2,z))$ else $\mathbf{Term}_c(b_2,z)$ endif else if $PC_c(b_1)$ then $((b_1[> \mathbf{Term}_c(b_2,z)) >> \mathbf{Send}_c((TC(b) \cap PC(b)), z.Z(b_1)))$ else if $TC_c(b)$ then if $(EC(b_2) = PC(b_1))$ then $(\mathbf{T}_c(b_2,z)[> \mathbf{Rec}_c(PC(b_1), z.Z(b_1)))$ else $(\mathbf{Rec}_c(PC(b_1), z.Z(b_1))[]$ $(\mathbf{Term}_c(b_2,z)$ if $(EC(b_2) \neq \phi)$ then $>> \mathbf{Rec}_c(PC(b_1), z.Z(b_1))$ endif)) endif else $\mathbf{T}_c(b_2,z)$ endif endif endif	

Table 15: The specific rules for disabling

$b = ((a^1; b^1; \delta^1)[> (\delta^1[](d^1; ((e^2; \delta^2) \lvert\lvert\lvert (f^3; \delta^3))) >> (g^1; \delta^1))))$
$\forall \{c, c'\} : FIFO(c,c')$, $TC(b) = \{1,3\}$
$\mathbf{T}_1(b,z) := (((a^1; b^1; \delta)[> (\delta[](d^1; ((send_2!z.2; \delta) \lvert\lvert\lvert (send_3!z.2; \delta)) >> ((rec_2!z.3; \delta) \lvert\lvert\lvert (rec_3!z.3; \delta)) >> (g^1; \delta))))$ $>> (send_3!z.1; \delta))$
$\mathbf{T}_2(b,z) := (rec_1!z.2; e^2; send_1!z.3; stop)$
$\mathbf{T}_3(b,z) := ((rec_1!z.2; f^3; send_1!z.3; stop)[> (rec_1!z.1; \delta))$

Table 16: An example implementation of disabling

tion is trivial (Table 18). To enforce the commuting for renaming, we require that all renamings are local to individual server components (Table 19). An example implementation of hiding and renaming is given in Table 22.

4.8 Implementation of process definition and instantiation

An explicit process p is implemented at a component c as an explicit process $p_c(n)$ (Table 20), where formal parameter n carries the process instance identifier (Table 21). For the main process *Server* we presume that it might be necessary for its termination to be a common action of the server environment and all the server components. An example implementation of multiple concurrent instances of a process is process "Second" in Table 22. Note that in the example, simplification of all $z.Z(b)$ into $Z(b)$, suggested for the example in Table 7, would result in an erroneous protocol.

5 Discussion and Conclusions

We have proposed a correctness-preserving transformation for functionality decomposition based on specifications written in Basic LOTOS (Bolognesi and Brinksma, 1987), the core sublanguage of the standard specification language LOTOS. Given a specification of the required external behaviour (the expected service) of a system and a partitioning of the specified service actions among the system components, the transformation derives behaviour of individual components implementing the service. A correctness proof is provided in (Kapus-Kolar, 1998).

$e = b =$ **hide** G **in** b_1	
$SC_c(b) = SC_c(b_1)$	$EC_c(b) = EC_c(b_1)$
$PC_c(b) = PC_c(b_1)$	$IT(b) = IT(b_1)$
$ST(b_1) = ST(b)$	$DT(b) = DT(b_1)$
$TC_c(b_1) = TC_c(b)$	
$\mathbf{T}_c(e,z) :=$ *hide* $\mathbf{Proj}_c(G)$ **in** $\mathbf{T}_c(b_1,z)$	

Table 18: The specific rules for hiding

Our algorithm enhances and integrates the algorithms of (Kant et al., 1996; Brinksma and Langerak, 1995). As the two algorithms are themselves a synthesis of the earlier similar approaches, and thoroughly compared to them, in the following we only compare our algorithm to the two algorithms.

- Unlike (Brinksma and Langerak, 1995), our algorithm is applicable to *multi-party* servers.
- It is applicable to servers with *both synchronous and asynchronous inter-component channels*, while (Kant

$e = b = b_1 \vert G \vert b_2$	
$SC_c(b) = (SC_c(b_1) \vee SC_c(b_2))$	$EC_c(b) = (EC_c(b_1) \vee EC_c(b_2))$
$PC_c(b) = (PC_c(b_1) \vee PC_c(b_2))$	$IT(b) = (IT(b_1) \wedge IT(b_2))$
$ST(b_1) = ST(b_2) = true$	$DT(b) = (DT(b_1) \vee DT(b_2))$
$TC_c^+(b_1) = TC_c^+(b_2) = TC_c(b)$	
$(G = \phi) \vee \neg\exists\{c, c'\} \subseteq (PC(b_1) \cap PC(b_2)) : FIFO(c, c')$	
$\mathbf{T}_c(e, z) := (\mathbf{Term}_c(b_1, z) \vert \mathbf{Proj}_c(G) \vert \mathbf{Term}_c(b_2, z))$	

Table 17: The specific rules for parallel composition

$Server := (First \vert [a^1, c^2] \vert Second)$ $First :=$ **hide** b^2 **in** $((((a^1; \delta^1) \vert\vert\vert (b^2; \delta^2)) >> (c^2; \delta^2)) \vert [b^2] \vert (d^1; b^2; \delta^2))$ $Second := ((ren\ A^1 \rightarrow a^1\ B^3 \rightarrow b^3\ C^2 \rightarrow c^2\ in\ Proc) \vert\vert\vert (ren\ A^1 \rightarrow x^1\ B^3 \rightarrow y^3\ C^2 \rightarrow z^2\ in\ Proc))$ $Proc := (((A^1; \delta^1) \vert\vert\vert (B^3; \delta^3)) >> (C^2; \delta^2))$ $FIFO(c, c') = (\{c, c'\} \neq \{1, 2\})$
$Server_1 := (First_1(1) \vert [a^1] \vert Second_1(2))$ $First_1(z) := ((a^1; sync_2!z.1; \delta) \vert\vert\vert (d^1; sync_2!z.2; \delta))$ $Second_1(z) := ((ren\ A^1 \rightarrow a^1\ in\ Proc_1(z.1)) \vert\vert\vert (ren\ A^1 \rightarrow x^1\ in\ Proc_1(z.2)))$ $Proc_1(z) := (A^1; sync_2!z.1; \delta)$
$Server_2 := (First_2(1) \vert [c^2] \vert Second_2(2))$ $First_2(z) :=$ **hide** b^2 **in** $((b^2; sync_1!z.1; c^2; \delta) \vert [b^2] \vert (sync_1!z.2; b^2; \delta))$ $Second_2(z) := ((ren\ C^2 \rightarrow c^2\ in\ Proc_2(z.1)) \vert\vert\vert (ren\ C^2 \rightarrow z^2\ in\ Proc_2(z.2)))$ $Proc_2(z) := (((sync_1!z.1; \delta) \vert\vert\vert (rec_3!z.1; \delta)) >> (C^2; \delta))$
$Server_3 := Second_3(2)$ $Second_3(z) := ((ren\ B^3 \rightarrow b^3\ in\ Proc_3(z.1)) \vert\vert\vert (ren\ B^3 \rightarrow y^3\ in\ Proc_3(z.2)))$ $Proc_3(z) := (B^3; send_2!z.1; \delta))$

Table 22: An example implementation of parallel composition, hiding, renaming and process instantiation

$e = b =$ **ren** R **in** b_1	
$SC_c(b) = SC_c(b_1)$	$EC_c(b) = EC_c(b_1)$
$PC_c(b) = PC_c(b_1)$	$IT(b) = IT(b_1)$
$ST(b_1) = ST(b)$	$DT(b) = DT(b_1)$
$TC_c(b_1) = TC_c(b)$	
$((u_1^c \rightarrow g) \in R) \Rightarrow \exists u_2 : (g = u_2^c)$	
$\mathbf{T}_c(e, z) :=$ **ren** $\mathbf{Proj}_c(R)$ **in** $\mathbf{T}_c(b_1, z)$	

Table 19: The specific rules for renaming

$e = (p := b_1)$	
$SC_c(p) = SC_c(b_1)$	$EC_c(p) = EC_c(b_1)$
$PC_c(p) = PC_c(b_1)$	$IT(p) = IT(b_1)$
$ST(b_1) = ST(p)$	$DT(p) = DT(b_1)$
$TC_c^+(b_1) = TC_c(p)$	
$\mathbf{T}_c(e, z) := (p_c(n) := \mathbf{Term}_c(b_1, n))$	

Table 20: The specific rules for process definition

et al., 1996) only supports asynchronous communication. As such, our algorithm has wide applicability (see the Introduction) and is also suitable for hardware/software co-design.

- The algorithm *corrects* an error identified in (Brinksma and Langerak, 1995) and several identified in (Kant et al., 1996). It is also more general, in the sense that it supports *implementation of decision-making terminations.*
- The algorithm provides means for the generation of *more efficient protocols* (with less intercomponent communication), based on the following observation: If there are two consecutive service parts b_1 and b_2, the only server components that really must detect the termination of b_1 are those executing its ending actions. Other participants of b_1 can as well conclude its execution by inaction that is later disrupted by actions announcing execution of b_2, for it is the start of b_2 - not the formal termination of b_1 - that is relevant to the service users. Even if there is no b_2 following b_1, it might still be more efficient for a non-ending participant of b_1 not to care about its termination, but rather conclude its part of b_1 with inaction later disrupted by termination-signalling messages from the ending participants of b_1.
- The algorithm is *more flexible*, for it allows one to employ the above communication-reduction principle to an extent best meeting his/her various optimization criteria, reduction of inter-component communication often being just one of them. If the principle is only employed where mandatory for protocol correctness,

$e = b = p$	
$SC_c(b) = SC_c(p)$	$EC_c(b) = EC_c(p)$
$PC_c(b) = PC_c(p)$	$DT(b) = DT(p)$
if $(p = Server)$ then $ST(p) = true$ else $ST(p) = (ST(p) \vee ST(b))$	
if $(p = Server)$ then $TC_c(p) = true$ else $TC_c(p) = TC_c(b)$	
$\mathbf{T}_c(e, z) := p_c(z.Z(b))$	

Table 21: The specific rules for process instantiation

the algorithm reduces to a corrected version of (Kant et al., 1996; Brinksma and Langerak, 1995) adapted for multi-party servers with synchronous and/or asynchronous inter-component channels.

It seems that one should prefer our algorithm to (Kant et al., 1996; Brinksma and Langerak, 1995), though it could be further improved by more exact computation of service specification subexpression attributes and restrictions, that would widen its applicability and/or increase the efficiency of the derived protocols. Other items *for further study* are the same as for the two former algorithms:

- introduction of inter-component co-ordination schemes that would render the various restrictions on the service specification structure unnecessary, i.e. allow *distributed decision-making*, as for a very limited setting suggested in (Langerak, 1990). There is presently no adequate solution that would not ruin the compositionality of the algorithm, thereby making the service/protocol relationship difficult to understand.
- extension to service actions with *data parameters* and to *timed service actions*. Our experience (Kapus-Kolar, 1991a,b) shows that it is typically possible to convey data and timing information piggybacked in the protocol messages already present if data and time are ignored, i.e. in the messages introduced by the above presented algorithm. Of course, provided that the messages are sent at appropriate points of service execution. Hence again the message-scheduling flexibility of our algorithm proves convenient, particularly in the presence of real-time requirements and protocol channels with substantial transit delay.
- generalization to *unreliable protocol channels*, though it presently seems that it would be better to solve the problem *below* the application layer of the system. For recovery from errors requires returning to previous states, but the concept of an explicit state is not defined in LOTOS.

The algorithm as it is now is useful as a set of *hints on how to systematically design correct and efficient distributed service implementations*. To complete the work, it would be desirable to implement the algorithm within a CAD tool. The mapping itself is trivial to implement, but the communication optimization part is complicated, if one wants to give the algorithm the best possible performance. Even optimization criteria are not well known; it would be convenient to have them derived automatically from a more detailed specification of the system and the service, particularly from requirements regarding action parameters and quantitative timing, and the information on the invocation probability for individual service parts. Thus we decided to postpone the implementation of the algorithm till completion of a thorough study on the subject.

References

[1] Bochmann, G. v., Gotzhein, R., Deriving Protocol Specifications from Service Specifications, in *Proc. ACM SIGCOMM'86 Symp.*, ACM, 1986, 148-156.

[2] Bolognesi, T., Brinksma, E., Introduction to the ISO Specification Language LOTOS, *Computer Networks & ISDN Systems 14(1)*, 25-59 (1987).

[3] Brinksma, E., Langerak, R., Functionality Decomposition by Compositional Correctness Preserving Transformation, *SACJ/SART 13*, 2-13 (1995).

[4] Eijk, P. v., Tools for LOTOS Specification Style Transformation, in *Formal Description Techniques II* (S. T. Vuong, ed.), North-Holland, 1990, 43-51.

[5] Kant, C., Higashino, T., Bochmann, G. v., Deriving Protocol Specifications from Service Specifications Written in LOTOS, *Distributed Computing 10*, 29-47 (1996).

[6] Kapus-Kolar, M., Deriving Protocol Specifications from Service Specifications Including Parameters, *Microprocessing and Microprogramming 32*, 731-738 (1991).

[7] Kapus-Kolar, M., Deriving Protocol Specifications from Service Specifications with Heterogeneous Timing Requirements, in *Proc. 3rd IEE Int. Conf. on Software Engineering for Real-Time Systems*, IEE, London, 1991, 266-270.

[8] Kapus-Kolar, M., Employing Disruptions for More Efficient Functionality Decomposition in LOTOS, in *Proc. 22nd EUROMICRO Conf.*, IEEE Computer Society, 1997, 464-471.

[9] Kapus-Kolar, M., *Employing Disruptions for More Efficient Functionality Decomposition in LOTOS*, Technical Report 7878, Jožef Stefan Institute, Ljubljana, 1998.

[10] Kapus-Kolar, M., Comments on Deriving Protocol Specifications from Service Specifications Written in LOTOS, to appear in *Distributed Computing 12(4)*, 1999.

[11] Kapus-Kolar, M., Rugelj, J., Bonač, M., Deriving Protocol Specifications from Service Specifications, in *Proc. 9th IASTED Int. Symp. Applied Informatics* (M. H. Hamza, ed.), Acta Press, Anaheim-Calgary-Zürich, 1991, 375-378.

[12] Langerak, R., Decomposition of Functionality: A Correctness-Preserving LOTOS Transformation, in *Protocol Specification, Testing and Verification X* (L. Logrippo, R. Probert, H. Ural, eds.), North-Holland, 1990, 229-242.

[13] Saleh, K., Synthesis of Communication Protocols: An Annotated Bibliography, *Computer Communication Review 26(5)*, 40-59 (1996).

[14] Turner, K. J. (ed.), *Using Formal Description Techniques - An Introduction to ESTELLE, LOTOS and SDL*, John Wiley, New York, 1993.

[15] *WELL - World-wide Environment for Learning LOTOS*, "http://www.cs.stir.ac.uk/~kjt/research/-well/well.html".

An Application-Level Dependable Technique for Farmer-Worker Parallel Programs

Vincenzo De Florio, Geert Deconinck and Rudy Lauwereins
Katholieke Universiteit Leuven, Electrical Engineering Dept, ACCA Group
Kard. Mercierlaan 94, B-3001 Heverlee, Belgium
Phone: +32 16 32 1142, Fax: +32 16 32 1986
E-mail: Vincenzo.DeFlorio@esat.kuleuven.ac.be

Keywords: parallel computing, farmer-worker applications, fault tolerance

Edited by: Rudi Murn
Received: April 14, 1998 **Revised:** February 4, 1999 **Accepted:** May 15, 1999

An application-level technique is described for farmer-worker parallel applications which allows a worker to be added or removed from the computing farm at any moment of the run time without affecting the overall outcome of the computation. The technique is based on uncoupling the farmer from the workers by means of a separate module which asynchronously feeds these latter with new "units of work" on an on-demand basis, and on a special feeding strategy based on bookkeeping the status of each work-unit. An augmentation of the LINDA model is finally proposed to exploit the bookkeeping algorithm for tuple management.

1 Introduction

Parallel computing is nowadays the only technique that can be used in order to achieve the impressive computing power needed to solve a number of challenging problems; as such, it is being employed by an ever growing community of users in spite of what we feel as two main disadvantages, namely:

1. harder-to-use programming techniques, programming models and development tools—if any,—which sometimes translate into programs that don't match as efficiently as expected with the underlying parallel hardware, and
2. the inherently lower level of dependability that characterizes any such parallel hardware, *i.e.*, a higher probability for events like a node's permanent or temporary failure.

A real, effective exploitation of any given parallel computer asks for solutions which take into a deep account the above outlined problems.

Let us consider for example the synchronous farmer-worker algorithm, *i.e.*, a well-known model for structuring data-parallel applications: a master process, namely the farmer, feeds a pool of slave processes, called workers, with some units of work. The slave processes then execute some job on their units. The master then polls the slaves until they return their partial results that are eventually recollected and saved. Though quite simple, this scheme may give good results, especially in homogeneous, dedicated environments.

But how does this model react to events like a failure of a worker, or more simply to a worker's performance degradation due, *e.g.*, to the exhaustion of any vital resource? Without substantial modifications, this scheme is not able to cope with these events—they would seriously affect the whole application or its overall performances, regardless the high degree of hardware redundancy implicitly available in any parallel system. The same unflexibility prevents a failed worker to re-enter the computing farm once it has regained the proper operational state.

As opposed to this synchronous structuring, it is possible for example to implement the farmer-worker model by de-coupling the farmer from the workers by means of an intermediate module, a dispatcher which asynchronously feeds these latter and supplies them with new units of work on an on-demand basis. This strategy guarantees some sort of a dynamic balancing of the workload even in heterogeneous, distributed environments, thus exhibiting a higher matching to the parallel hardware. The Live Data Structure computational paradigm, known from the LINDA context, makes this particularly easy to set up (see for example De Florio, Murgolo and Spinelli, 1994).

With this approach it is also possible to add a new worker at run-time without any notification to both the farmer and the intermediate module—the newcomer will simply generate additional, non-distinguishable requests for work. But again, if a worker fails or its performances degrade, the whole application may fail or its overall outcome be affected or seriously delayed. This is particularly important

when one considers the inherent loss in dependability of any parallel, (*i.e.*, replicated) hardware.

Next sections introduce and discuss a modification to the above sketched asynchronous scheme, which inherits the advantages of its parent and offers new ones, namely:

- it allows a non-solitary, temporarily slowed down worker to be left out of the processing farm as long as its performance degradation exists, and
- it allows a non-solitary worker which has been permanently affected by some fault to be definitively removed from the farm,

both of them without affecting the overall outcome of the computation, and dynamically spreading the workload among the active processors in a way that results in an excellent match to various different MIMD architectures.

2 The Technique

For the purpose of describing the technique we define the following scenario: a MIMD machine disposes of $n + 2$ identical "nodes" ($n > 0$), or processing entities, connected by some communication line. On each node a number of independent sequential processes are executed on a time-sharing basis. A message passing library is available for sending and receiving messages across the communication line. A synchronous communication approach is used: a sender blocks until the intended receiver gets the message. A receiver blocks waiting for a message from a specific sender, or for a message from a number of senders. When a message arrives, the receiver is awaken and is able to receive that message and to know the identity of the sender. Nodes are numbered from 0 to $n + 1$. Node 0 is connected to an input line and node $n + 1$ is connected to an output line.

- Node 0 runs:
  - a Farmer process, connected by the input line to an external producer device. In the following we consider a camera as the producer device. A control line wires again the Farmer to the camera, so that this latter can be commanded to produce new data and eventually send this data across the input line;
  - a Dispatcher process, yet to be described.
- Node $n + 1$ runs a Collector process, to be described later on, connected by the output line to an external storage device, *e.g.*, a disk;
- Each of the nodes from 1 to n is purely devoted to the execution of one instance of the Worker process. Each Worker is connected to the Dispatcher and to the Collector processes.

2.1 Interactions Between the Farmer and the Dispatcher

On demand of the Farmer process, the camera sends it an input image. Once it has received an image, the Farmer performs a predefined, static data decomposition, creating m equally sized sub-images, or blocks. Blocks are numbered from 1 to m, and are represented by variables $b_i, 1 \leq i \leq m$.

The Farmer process interacts exclusively with the camera and with the Dispatcher process.

- Three classes of messages can be sent from the Farmer process to the Dispatcher (see Fig. 1):
  - **class-1 message:** a `NEW_RUN` message, which means: "a new bunch of data is available";
  - **class-2 message:** a `STOP` message, which means that no more input is available so the whole process has to be terminated;
  - **class-3 message:** a couple $(k, b_k), 1 \leq k \leq m$, *i.e.*, an integer which identifies a particular block (it will be referred from now on as a "block-id"), followed by the block itself.
- The only type of message that the Dispatcher process sends to the Farmer process is a block-id, *i.e.*, a single integer in the range $\{1, \ldots, m\}$ which expresses the information that a certain block has been fully processed by a Worker and recollected by the Collector (see §2.3.)

At the other end of the communication line, the Dispatcher is ready to process a number of events triggered by message arrivals. For example, when a class-3 message comes in, the block is stored into a work buffer as follows:

```
receive (k, b_k)
s_k ← DISABLED
w_k ← b_k
```

(Here, `receive` is the function for receiving an incoming message, $\vec{s}$ is a vector of m integers pre-initialized to `DISABLED`, which represents some status information that will be described later on, and $\vec{w}$ is a vector of "work buffers", *i.e.*, bunches of memory able to store any block. `DISABLED` is an integer which is not in the set $\{1, \ldots, m\}$. The "$\leftarrow$" sign is the assignment operator.)

As the Farmer process sends a class-1 message, that is, a `NEW_RUN` signal, the Dispatcher processes that event as follows:

$\vec{s} \leftarrow 0$
`broadcast RESUME`

that is, it zeroes each element of $\vec{s}$ and then broadcasts the `RESUME` message to the whole farm.

When the first image arrives to the Farmer process, it produces a series $(b_i)_{1 \le i \le m}$, and then a sequence of messages $(i, b_i)_{1 \le i \le m}$. Finally, the Farmer sends a `NEW_RUN` message.

Starting from the second image, and while there are images to process from the camera, the Farmer performs the image decomposition in advance, thus creating a complete set of (k, b_k) couples. These couples are then sent to the Dispatcher on an on-demand basis: as soon as block-id i comes in, couple (i, b_i) is sent out. This is done for anticipating the transmission of the couples belonging to the next run of the computation. When eventually the last block-id of a certain run has been received, a complete set of "brand-new" blocks is already in the hands of the Dispatcher; at that point, sending the one `NEW_RUN` message will simultaneously enable all blocks.

2.2 Interactions Between the Dispatcher and the Workers

The Dispatcher interacts with every instance of the Worker process.

- Four classes of messages can be sent from the Dispatcher to the Workers (see Fig. 1):
  1. a `SLEEP` message, which sets the receiver into a wait condition;
  2. a `RESUME` message, to get the receiver out of the waiting state;
  3. a `STOP` message, which makes the Worker terminate;
  4. a (k, w) couple, where w represents the input data to be elaborated.
- Worker j, $1 \le j \le n$, interacts with the Dispatcher by sending it its worker-id message, *i.e.*, the j integer. This happens when Worker j has finished dealing with a previously sent w working buffer and is available for a new (k, w) couple to work with.

In substance, Worker j continuously repeats the following instructions in a loop:

```
send j to Dispatcher
receive message from Dispatcher
process message
```

Clearly, **send** transmits a message. The last instruction, in dependence with the class of the incoming message, results in a number of different operations:

- if the message is a `SLEEP`, the Worker waits until the arrival of a `RESUME` message, which makes it resume the loop, or the arrival of any other message, which means that an error has occurred;
- if it is a `STOP` message, the Worker breaks the loop and exits the farm;
- if it is a (k, w) couple, the Worker starts computing the value $f(w)$, where f is some user-defined function, *e.g.*, an edge detector. If a `RESUME` event is raised during the computation of f, that computation is immediately abandoned and the Worker restarts the loop. Contrarywise, the output couple $(k, f(w))$ is sent to the Collector process.

When the Dispatcher gets a j integer from Worker j, its expected response is a new (k, w) couple, or a `SLEEP`. What rules in this context is the $\vec{s}$ vector—if all entries of $\vec{s}$ are `DISABLED`, then a `SLEEP` message is sent to Worker j. Otherwise, an entry is selected among those with the minimum non-negative value, say entry l, and a (l, b_l) message is then sent as a response. s_l is finally incremented by 1.

More formally, considered set

$$S = \{i \mid 1 \le i \le m \ \wedge \ s_i \neq \texttt{DISABLED}\},$$

if S is non-empty it is possible to partition S according to the equivalence relation R such that:

$$[i]_R = \{j \mid 1 \le j \le m \ \wedge \ s_i = s_j\}.$$

Then let us consider function $f : S/R \to \mathbb{N}$ such that $\forall [i] \in S/R : f([i]) = s_i$. Now, first let us consider $a = \min f(S/R)$; then we choose $l \in f^{-1}(a)$ in any way, *e.g.*, pseudo-randomly; finally, message (l, b_l) is sent to Worker j, s_l is incremented, and the partition is reconfigured accordingly. If S is the empty set, a `SLEEP` message is generated.

In other words, entry s_i when greater than or equal to 0 represents some sort of a priority identifier (the lower the value, the higher the priority for block b_i). The block to be sent to a requesting Worker process is always selected among those with the highest priority; after the selection, s_i is updated incrementing its value by 1. In this way, the content of s_i represents the degree of "freshness" of block b_i: it substantially counts the number of times it has been picked up by a Worker process; fresher blocks are always preferred.

As long as there are "brand-new" blocks, *i.e.*, blocks with a freshness attribute of 0, these are the blocks which are selected and distributed. Note that this means that as long as the above condition is true, each Worker deals with a different unit of work; on the contrary, as soon as the last brand-new block is distributed, the model admits that a same block may be assigned to more than one Worker.

This is tolerated up to a certain threshold value; if any s_i becomes greater than that value, an alarm event is raised—too many workers are dealing with the same input data, which might mean that they are all affected by the same problem, *e.g.*, a software bug resulting in an error when b_i is being processed. We won't deal with this special case. Another possibility is that two or more Workers had finished their work almost at the same time thus bringing rapidly a flag to the threshold. Waiting for the processing time of one block may supply the answer.

A value of DISABLED for any s_i means that its corresponding block is not available to be computed. It is simply not considered during the selection procedure.

2.3 Interactions Between the Workers and the Collector

Any Worker may send just one class of messages to the Collector; no message is sent from this latter to any Worker (see Fig. 1).

The only allowed message is the couple (k, o) in which o is the fully processed output of the Worker's activity on the k^{th} block.

The Collector's task is to fill a number of "slots", namely $p_i, i = 1, \ldots, m$, with the outputs coming from the Workers. As two or more Workers are allowed to process a same block thus producing two or more (k, o) couples, the Collector runs a vector of status bits, *viz.* $\vec{f}$, which records the status of each slot: if f_i is FREE then p_i is "empty," *i.e.*, it has never been filled in by any output before; if it is BUSY, it already holds an output. $\vec{f}$ is firstly initialized to FREE.

For each incoming message from the Worker, the Collector repeats the following sequence of operations:

```
receive (k, o) from Worker
if f_k is equal to FREE
   then
          send k to Dispatcher
          p_k ← o
          f_k ← BUSY
          check-if-full
   else
          detect
endif
```

where:

check-if-full checks if, due to the last arrival, all entries of $\vec{f}$ have become BUSY. In that case, a complete set of partial outputs has been recollected and, after some user-defined post-processing (for example, a polygonal approximation of the chains of edges produced by the Workers), a global output can be saved, and the flag vector re-initialized:

```
if f⃗ is equal to BUSY
   then
          post-process p⃗
          save p⃗
          f⃗ ← FREE
endif
```

detect is a user-defined functionality—he/she may choose to compare the two o's so to be able to detect any inconsistency and start some recovery action, or may simply ignore the whole message.

Note also that an acknowledgment message (the block-id) is sent from the Collector to the Dispatcher, to inform it that an output slot has been occupied, *i.e.*, a partial output has been gathered. This also means that the Farmer can anticipate the transmission of a block which belongs to the next run, if any.

2.4 Interactions Between the Collector and the Dispatcher

As just stated, upon acceptance of an output, the collector sends a block-id, say integer k, to the Dispatcher—it is the only message that goes from the Collector to the Dispatcher.

The Dispatcher then simply acts as follows:

```
s_k ← DISABLED
send k to Farmer
```

that is, the Dispatcher "disables" the k^{th} unit of work—set S as defined in §2.2 is reduced by one element and consequently partition S/R changes its shape; then the block-id is propagated to the Farmer (see Fig. 1).

On the opposite direction, there is only one message that may travel from the Dispatcher to the Collector: the STOP message that means that no more input is available and so processing is over. Upon reception of this message, the Collector stops itself, like any other receiver in the farm does.

3 Discussions and Conclusions

The just proposed technique uses asynchronicity in order to efficiently match to a huge class of parallel and distributed architectures. It also uses the redundancy which is inherent to parallelism to make an application able to cope with events like, *e.g.*, a failure of a node, or a node being slowed down, temporarily or not.

- If a node fails while it is processing block k, then no output block will be transferred to the Collector. When no more "brand-new" blocks are available, block k will be assigned to one or more Worker processes, up to a certain limit. During this phase the replicated processing modules of

the parallel machine may be thought of as part of a hardware redundancy fault-tolerant mechanism. This phase is over when any Worker module delivers its output to the Collector and consequently all others are possibly explicitly forced to resume their processing loop or, if too late, their output is discarded;

- if a node has been for some reason drastically slowed down, then its block will be probably assigned to other possibly non-slowed Workers. Again, the first who succeeds, its output is collected; the others are stopped or ignored.

In any case, from the point of view of the Farmer process, all these events are completely masked. The mechanism may be provided to a user in the form of some set of basic functions, making all technicalities concerning both parallel programming and fault tolerance management transparent to the programmer.

Of course, nothing prevents the concurrent use of other fault tolerance mechanisms in any of the involved processes, *e.g.*, using watchdog timers to understand that a Worker has failed and consequently reset the proper entry of vector $\vec{f}$. The ability to re-enter the farm may also be exploited committing a reboot of a failed node and restarting the Worker process on that node.

3.1 Reliability Analysis

In order to compare the original, synchronous farmer-worker model with the one described in this paper, a first step is given by observing that the synchronous model depicts a *series system* (Johnson 1989), *i.e.*, a system in which each element is required not to have failed for the whole system to operate. This is not the case of the model described in this paper, in which a subset of the elements, namely the Worker farm, is a *parallel system* (Johnson 1989): if at least one Worker has not failed, so it is for the whole farm subsystem. Note how Fig. 1 may be also thought of as the reliability block diagram of this system.

Considering the sole farm subsystem, if we let $C_i(t), 1 \leq i \leq n$, be the event that Worker on node i has not failed at time t, and we let $R(t)$ be the reliability of any Worker at time t then, under the assumption of mutual independency between the events, we can conclude that:

$$R_s(t) \stackrel{\text{def}}{=} P(\bigcap_{i=1}^{n} C_i(t)) = \prod_{i=1}^{n} R(t) = (R(t))^n \quad (1)$$

being $R_s(t)$ the reliability of the farm as a series system, and

$$R_p(t) \stackrel{\text{def}}{=} 1 - P(\bigcap_{i=1}^{n} \overline{C_i}(t)) = 1 - \prod_{i=1}^{n}(1 - R(t)) = 1 - (1 - R(t))^n \quad (2)$$

where $R_p(t)$ represents the reliability of the farm as a parallel system. Of course failures must be independent, so again data-induced errors are not considered. Figure 2 shows the reliability of the farm in a series and in a parallel system as a Worker's reliability goes from 0 to 1.

3.2 An Augmented LINDA Model

The whole idea pictured in this paper may be implemented in a LINDA tuple space manager (see for example Carriero & Gelernter, 1989). Apart from the standard functions to access "common" tuples, a new set of functions may be supplied which deal with "book-kept tuples," *i.e.*, tuples that are distributed to requestors by means of the algorithm sketched in §2.2. As an example:

`fout` (for fault-tolerant `out`) may create a book-kept tuple, *i.e.*, a content-addressable object with book-kept accesses;

`frd` (fault-tolerant `rd`) may get a copy of a matching book-kept tuple, chosen according to the algorithm in §2.2;

`fin` (fault-tolerant `in`) may read-and-erase a matching book-kept tuple, chosen according to the algorithm in §2.2,

and so on. The ensuing augmented LINDA model results in an abstract, elegant, efficient, dependable, and transparent mechanism to exploit a parallel hardware.

3.3 Future Directions

The described technique is currently being implemented on a Parsytec CC system with the EPX/AIX environment (Parsytec GmbH, 1996) using PowerPVM/EPX (Genias GmbH, 1996), a homogeneous version of the PVM message passing library (Beguelin *et al.*, 1994); it will also be tested in heterogeneous, networked environments managed by PVM. Some work towards the definition and the development of an augmented LINDA model is currently being done.

Acknowledgments. This project is partly supported by an FWO Krediet aan Navorsers, by the ESPRIT-IV Project 21012 EFTOS, and by COF/96/11. Vincenzo De Florio is on leave from the Tecnopolis CSATA Novus Ortus science park. Geert

Deconinck is a Postdoctoral Fellow of the Fund for Scientific Research - Flanders (Belgium) (FWO). Rudy Lauwereins is a Senior Research Associate of FWO.

The authors wish to thank Dr. Stefano D. L. Campanozzi whose helpful comments greatly improved the mathematical coherence of this work.

References

[1] Beguelin, A., Dongarra, J. J., Geist, G. A., Manchek, R. & Sunderam, V.S. (1994) *PVM: Parallel Virtual Machine — A Users' Guide and Tutorial for Networked Parallel Computing.* Cambridge: MIT Press.

[2] Carriero, N. & Gelernter, D. (1989) How to write parallel programs: a guide to the perplexed. ACM Comp. Surv. 21, p. 323–357.

[3] Carriero, N. & Gelernter, D. (1989) LINDA in context. Communications of the ACM 32 4, p. 444–558.

[4] Johnson, B. W. (1989) *Design and analysis of fault-tolerant digital systems.* New York: Addison-Wesley.

[5] De Florio, V., Murgolo, F.P. & Spinelli, V. (1994) PvmLinda: Integration of two different computation paradigms. *Proc. of the First EuroMicro Conf. on Massively Parallel Computing Systems*, Ischia, Italy.

[6] Parsytec GmbH (1996) Embedded Parix Programmer's Guide. *Parsytec CC Series Hardware Documentation.* Aachen: Parsytec GmbH.

[7] Genias GmbH (1996) *PowerPVM/EPX for Parsytec CC Systems.* Neutraubling: Genias Software GmbH.

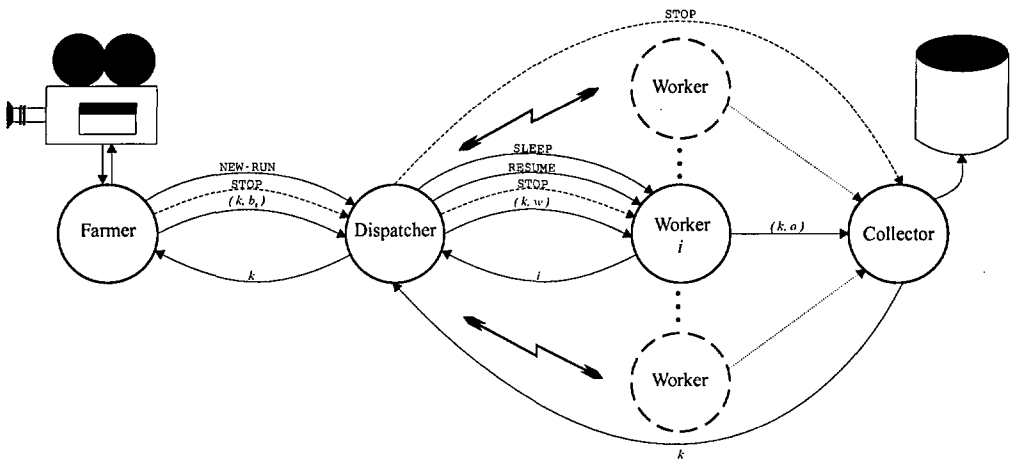


Figure 1: Summary of the interactions among the processes.

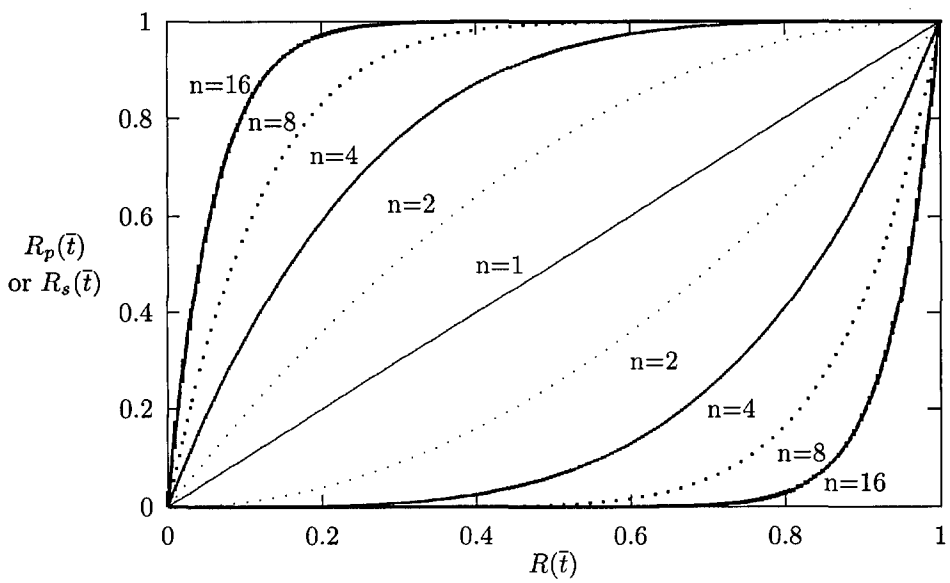


Figure 2: For a fixed value $\bar{t}$, a number of graphs of $R_p(\bar{t})$ (the reliability of the parallel system) and $R_s(\bar{t})$ (the reliability of the series system) are portrayed as functions of $R(\bar{t})$, the reliability of a Worker at time $\bar{t}$, and n, the number of the components. Each graph is labeled with its value of n; those above the diagonal portray reliabilities of parallel systems, while those below the diagonal pertain series systems. Note that for $n = 1$ the models coincide, while for any $n > 1$ $R_p(\bar{t})$ is always above $R_s(\bar{t})$ except when $R(\bar{t}) = 0$ (no reliable Worker) and when $R(\bar{t}) = 1$ (totally reliable, failure-free Worker).

Agent Properties In Multi-Agent Systems

Mirko Maleković
University of Zagreb, Faculty of Organization and Informatics
Varaždin, Croatia
E-mail: mmalekov@foi.hr

Keywords: agent properties, knowledge operators, knowledge theory, multi-agent systems, temporal operators.

Edited by: Matjaž Gams
Received: May 25, 1998 **Revised:** April 24, 1999 **Accepted:** May 6, 1999

We consider agent properties in multi-agent systems. These properties are characterized by knowledge operators and temporal operators (the past and future temporal operators). The properties relate the knowledge of two agents, and can be very helpful in analyzing the respective multi-agent system.

1 Introduction

The[1] theory of multi-agent systems is described in [1]. Incorporating knowledge and time in multi-agent system is given in [1], [2], and [4]. The temporal operators are described in [3]. In this paper, we consider in detail some new agent properties in multi-agent systems. These properties relate the knowledge of two agents, and can be very useful in analyzing the respective multi-agent system. The paper consists of five sections and an Appendix containing all the proofs of the stated propositions. In Section 2, we introduce the basic notions of multi-agent systems. In Section 3, we define the knowledge operators and the past and future temporal operators. Section 4 contains Proposition (more knowledgeable) the past propositions: Proposition ($\bullet$), Proposition ($\blacklozenge$), Proposition ($\blacksquare$), Proposition (S), and Proposition (B). In addition, Section 4 contains the future propositios: Proposition ($\circ$), Proposition ($\Diamond$), Proposition ($\Box$), Proposition ($\mathcal{U}$), and Proposition ($\mathcal{W}$). The propositions state the relationship between the knowledge of two agents. Conclusions are given in Section 5. The Appendix contains the proofs of all the propositions mentioned above.

2 Basic notions

In this section, we introduce some basic concepts and notations. Suppose we have a group consisting of m agents, named $1, 2, .., m$. An agent may be a man (a real agent), a software module or a communicated robot (an artificial agent). Even, an agent may be a component of a computer system (a wire or a message buffer). We assume these agents wish to reason about a world that can be described in terms of a nonempty set P of primitive propositions. A language is just a set of formulas, where the set of formulas LK of interest to us is defined as follows.
(1) The primitive propositions in P are formulas,
(2) If F and G are formulas, then so are $\neg F, (F \vee G), (F \wedge G), (F \Rightarrow G), (F \Leftrightarrow G)$, and $K_i(F)$ for all $i \in \{1, 2, .., m\}$, where K_i is a modal operator. A Kripke structure M for an agent group $\{1, 2, .., m\}$ over P is a tuple $M = (S, I, k_1, k_2, .., k_m)$, where S is a set of possible worlds, I is an interpretation that associates with each world in S a truth assignment to the primitive propositions in P, and k1, k2,.., km are binary relation on S, called the possibility relations for agents 1, 2,.., m, respectively. Given $p \in P$, the expression $I[w](p) = true$ means that p is true in a world w in a structure M. The fact that p is false, in a world v of a structure M, is indicated by the exspression $I[v](p) = false$ The expression $(u, v) \in k_i$ means that an agent i considers a world v possible, given his information in a world u. Since k_i defines what worlds an agent i considers possible in any given world, k_i will be called the possibility relation of the agent i. We now define what it means for a formula to be true at a given world in a structure. Let $(M, w) \models F$ means that F holds or is true at (M, w). Definition of $\models$ is as follows:
(a) $(M, w) \models p$ iff $I[w](p) =$ true, where $p \in P$,
(b) $(M, w) \models F \wedge G$ iff $(M, w) \models F$ and $(M, w) \models G$,
(c) $(M, w) \models F \vee G$ iff $(M, w) \models F$ or $(M, w) \models G$,
(d) $(M, w) \models F \Rightarrow G$ iff $(M, w) \models F$ implies $(M, w) \models G$,
(e) $(M, w) \models F \Leftrightarrow G$ iff $(M, w) \models F \Rightarrow G$ and $(M, w) \models G \Rightarrow F$,
(f) $(M, w) \models \neg F$ iff $(M, w) \not\models F$, that is, $(M, w) \models F$ does not hold,
(g) $M \models F$ iff $(M, w) \models F$ for all $w \in S$.
Finally, we shall define a modal operator K_i, where

[1] LaTeXTech Support: Mr. Mirko Varga, Faculty of Organization and Informatics, Varaždin, CROATIA

$K_i(F)$ is read: Agent i knows F.
(h) $(M,w) \models K_i(F)$ iff $(M,t) \models F$ for all $t \in S$ such that $(w,t) \in k_i$.
In (h) we have that agent i knows F in a world w of a structure M exactly if F holds at all worlds t that the agent i considers possible in w .

Multi-Agent Systems
A multi-agent system is any collection of interacting agents. In dealing with the complexity of a system, we focus attention on only a few of details, and hope that these cover everything that is relevant for our analysis. Next, we find good ways to think about a situation in order to minimize its complexity. It is known that reasoning about systems in terms of knowledge can be very helpful in this regard. To do that, we need a formal model of multi-agent systems. Our key assumption is that if we look at the system at any point in time, each of the agents is in some state. We refer to this as the agent's local state. We assume that an agent's local state encapsulates all the information to which the agent has access. For each agent has a local state, it is very naturally to think of the whole system as being in some (global) state. The global state includes the local states of the agents and the local state of an environment. Latter does play an important role. For instance, if we are considering a system of sensors observing a region, we might need to include features of the region in a description of the global state of the system. Accordingly, we devide a system into two components: the environment and the agents, where we view the environment as everything else that is relevant. Also, the environment can be viewed as just another agent. We need to say that a given system can be modeled in many ways. How to devide the system into agents and environment depends on the system being analyzed.
Let L_e be a set of possible local states for the environment and let L_i be a set of possible local states for agent $i, i = 1,..,n$. We define $G = L_e \times L_1 \times .. \times L_n$ to be the set of global states. A global state describes the system at a given point in time. For a system constantly changes (it is not a static entity), we are interested in how systems change over time. We take time to range over the natural numbers, that is, the time domain is the set of the natural numbers, N . A run over G is a function $r : N \to G$. Thus, a run over G can be identified with a sequence of global states in G. The run r represents a complete description of how the system's global state evolves over time. $r(0)$ describes the initial global state of the system in a possible execution, $r(1)$ describes the next global state, and so on. If $r(m) = (s_e, s_1, .., s_n)$, then we define $r[e](m) = s_e$ and $r[i](m) = s_i$, for $i = 1,..,n$. A pair (r,m), where r is a run and m is time, will be called a point. We shall say that a global state $r(m) = (s_e, s_1, .., s_n)$ is the global state at the point (r,m). We shall also identified the point (r,m) with the global state $r(m)$. Note that $r[i](m) = s_i$ is the local state of the agent i at the (global) state $r(m)$. A system R over G is a set of runs over G. The system R models the possible behaviors of the system being modeled. The intuition that the system being modeled has some behaviors can be captured by the requirenment that the set of runs R be nonempty.

3 Knowledge and temporal operators

We assume that we have a set P of primitive propositions, which we can think of as describing basic facts about a system R . Let I be an interpretation for the propositions in P over G , which assigns thruth values to the primitive propositions at the global states. Thus, for every $p \in P$ and $s \in G, I[s](p) \in \{true, false\}$. An interpreted system IS is a pair (R, I) . Now, we define knowledge in an interpreted system IS . Let $IS = (R,I)$ be an interpreted system. A Kripke structure for IS , denoted $M(IS) = (S, I, k_1, .., k_n)$, is defined in a straightforward way. $S = \{r(m) | r \in R, m \in N\}$, that is, S is the set of the global states at the points (r,m) in the system R. The possibility relations $k_1, k_2, .., k_n$ are defined as follows. Let $r(m) = (s_e, s_1, .., s_n)$, $r'(m') = (s'_e, s'_1, .., s'_n)$ be global states in S. We say that $r(m)$ and $r'(m')$ are indistinguishable to agent i iff $s_i = s'_i$. Thus, the agent i has the same local state in both $r(m)$ and $r'(m')$. We define $k_i = \{(r(m), r'(m')) \in S \times S \mid r(m)$ and $r'(m')$ are indistinguishable to agent $i\}, i = 1,2,..,n$. Accordingly, $(r(m), r'(m')) \in k_i$ iff $s_i = s'_i, i = 1,2,..,n$. There is no the possibility relation k_e for the environment because we are not usually interested in what the environment knows. Now, it is evident what it means for a formula F in LK to be true at a state r(m) in an interpreted system IS. For instance, we have $(IS, r(m)) \models p$ iff $I[r(m)](p) = true$, for all $p \in P$.
$(IS, r(m)) \models K_i(F)$ iff $(IS, r'(m')) \models F$ for all $r'(m') \in S$ such that $(r(m), r'(m')) \in k_i$. The modal operators $K_i, i = 1,..,n$ are called knowledge operators. We say that a formula F in LK is valid in an interpreted system IS, denoted $IS \models F$, iff $(IS, r(m)) \models F$ for all $r(m) \in S$. Let us note that we do not assume that the agents compute their knowledge in any way, or that they can necessarily answer questions based on their knowledge. We interpret knowledge as an external one, ascribed to the agents by someone reasoning about the system. To be able to make temporal statements, we extend our language LK by adding temporal operators, which are new modal operators for talking about time. This language will be denoted by LKT , and

will be used for reasoning about events that happen along a single run r in the system R. We define here five temporal operators for the future: $\circ$ (next), $\Box$(always), $\Diamond$ (eventually), U (until), W (waiting-for or unless); and five temporal operators for the past: $\bullet$ (previously), $\blacksquare$ (has always been), $\blacklozenge$ (once), S (since), and B (back-to).

Future Operators

The Next Operator $\circ$
If $F \in LKT$, then so is $\circ F$.
$\circ F$, read next F, is defined by
$(IS, r(m)) \models \circ F$ iff $(IS, r(m+1)) \models F$.
Thus, $\circ$ F holds at state $r(m)$ iff F holds at the next state $r(m+1)$.

The Always Operator $\Box$
If $F \in LKT$, then *so* is $\Box F$.
$\Box F$, read always F, is defined by
$(IS, r(m)) \models \Box F$ iff $(IS, r(m')) \models F$ for all $m' \geq m$.
Accordingly, $\Box F$ holds at state $r(m)$ iff F holds at state $r(m)$ (now) and at all later states.

The Eventually Operator $\Diamond$
If $F \in LKT$, then *so* is $\Diamond F$.
$\Diamond F$, read eventually F, is defined by $(IS, r(m)) \models \Diamond F$ iff $(IS, r(m')) \models F$ for some $m' \geq m$. Thus, $\Diamond F$ holds at state $r(m)$ iff F holds at state $r(m)$ or some state in the future.

The Until Operator $\mathcal{U}$
If $F \in LKT$, then *so* is $F \mathcal{U} F$.
$F \mathcal{U} F_1$, read F until F_1, is defined by $(IS, r(m)) \models F \mathcal{U} F_1$ iff $(IS, r(m')) \models F_1$ for some $m' \geq m$ and $(IS, r(m'')) \models F$ for all m'' with $m \leq m'' < m'$.
The until formula $F \mathcal{U} F_1$ predicts the eventual occurrence of F_1 and states that F holds continuously at least until the first occurrence of F_1.

The Unless (Waiting-for) Operator W
If $F \in LKT$, then *so* is $F W F_1$.
$F W F_1$, read F unless F_1, has the following semantics.
$(IS, r(m)) \models F W F_1$ iff
$(IS, r(m)) \models F \mathcal{U} F_1$ or $(IS, r(m)) \models \Box F$.
Thus, the formula $F W F_1$ expresses the property that F holds continuously either until the next occurrence of F_1 or throughout the sequence of states. Note that our interpretation of $\circ F$ makes sense because our notion of time is discrete. All the other temporal operators make perfect sense even for continuous notions of time.

Past Operators

We have seen that a future fomula describes a property holding at a suffix of the state, lying to the right of the current state, that is, a future formula at the state $r(m)$ describes a property of the states $r(m), r(m+1), ...$ Each of the future operators has a symmetric counterpart called the past temporal operator. A past formula describes a property of a prefix of the state, lying left to the current position, that is, a past formula at the state $r(m)$ describes a property of the states $r(0), r(1), .., r(m-1), r(m)$.

The Previous Operator $\bullet$
If $F \in LKT$, then so is $\bullet F$, read as previously F. Its semantics is defined by
$(IS, r(m)) \models \bullet F$ iff $m > 0$ and
$(IS, r(m-1)) \models F$.
Thus, $\bullet F$ holds at state $r(m)$ iff $r(m)$ is not the first state in the run r and F holds at state $r(m-1)$. In particular, $\bullet F$ is false at state $r(0)$. This operator makes sense because our notion of time is discrete. All the other past temporal operators make perfect sense even for continuous notions of time.

The Has-always-been Operator $\blacksquare$
$\blacksquare F \in LKT$ if $F \in LKT$. It is read has always been F, and defined by
$(IS, r(m)) \models \blacksquare F$ iff (for all $m', 0 \leq m' \leq m)[(IS, r(m')) \models F]$
Thus, $\blacksquare F$ holds at state $r(m)$ iff F holds at state $r(m)$ and all preceding positions.

The Once Operator $\blacklozenge$
If $F \in LKT$, then so is $\blacklozenge F$, read once F. Its semantics is defined by $(IS, r(m)) \models \blacklozenge F$ iff (for some $m', 0 \leq m' \leq m)[(IS, r(m')) \models F]$. Accordingly, $\blacklozenge F$ holds at state $r(m)$ iff F holds at state $r(m)$ or some preceding state.

The Since Operator S
$FSF1 \in LKT$ if $F, F_1 \in LKT$. It is read as F since F1 and defined by $(IS, r(m)) \models FSF1$ iff (for some $m', 0 \leq m' \leq m)[(IS, r(m')) \models F_1]$ and (for all $k, m' \leq k \leq m)[(IS, r(k)) \models F]$. Thus, FSF_1 states that F_1 has happened in the past and F has held continuously from the state following the last occurrence of F_1 to the present.

The Back-to Operator B
$FBF1 \in LKT$ if $F, F_1 \in LKT$. It is read F back to F_1 and defined by $(IS, r(m)) \models FBF_1$ iff $(IS, r(m)) \models FSF_1$ or $(IS, r(m)) \models \blacksquare F$. The operator B provides a weaker version of the since operator S. Thus, the formula FBF_1 states that F holds continuously at all states preceding and including the present, or F_1 has happened in the past and F has held continuously from the state of the last occurrence of F_1 to the present.

4 Agent Properties

In this chapter, we consider some important agent properties. These properties relate the knowledge of two agents. In the following propositions, we shall use the set $S[j,r](m')$ defined by $S[j,r](m') = \{r_i(m_i) | (r(m'), r_i(m_i)) \in k_j\}$. Thus, $S[j,r](m')$ is the set of the states in S that agent j considers possible in state $r(m')$. Also, $F \in LKT$ is an arbitrary formula in LKT.

Proposition (more knowledgeable)
If $S[j,r](m) \subseteq S[i,r](m)$, then
$(IS, r(m)) \models K_i(F) \Rightarrow K_j(F)$.

Proposition (more knowledgeable) says that at state $r(m)$ agent j knows F if agent i knows F, under the condition that $S[j,r](m) \subseteq S[i,r](m)$ holds. We can say that agent j is more knowledgeable than agent i if the stated premise is true.
In the rest of this chapter, we first state past propositions, and after that future propositions.

Past Propositions

Proposition $(\bullet)$
If $S[j,r](m-1) \subseteq S[i,r](m)$, then
$(IS, r(m)) \models K_i(F) \Rightarrow \bullet(K_j(F))$.

The proposition states that at state $r(m)$ agent j previously knew F if agent i knows F, under the condition that $S[j,r](m-1) \subseteq S[i,r](m)$ holds.

Proposition $(\blacklozenge)$
If (for some $m', 0 \leq m' \leq m)[S[j,r](m') \subseteq S[i,r](m)]$, then $(IS, r(m)) \models K_i(F) \Rightarrow \blacklozenge K_j(F)$

Proposition $(\blacklozenge)$ says that at state $r(m)$ agent j has known F if agent i knows F, under the condition that (for some m', $0 \leq m' \leq m)[S[j,r](m') \subseteq S[i,r](m)]$ holds.

Proposition $(\blacksquare)$
If (for all m', $0 \leq m' \leq m$)
$[S[j,r](m') \subseteq (S[i,r](m)]$, then
$(IS, r(m)) \models K_i(F) \Rightarrow \blacksquare K_j(F)$.

Proposition $(\blacksquare)$ says that at state $r(m)$ agent j has always known F if agent i knows F, under the condition that (for all m', $0 \leq m' \leq m)[S[j,r](m') \subseteq S[i,r](m)]$ is true.

Proposition (S)
(for all $r(m) \in S)[(IS, r(m)) \models K_i(F) \Rightarrow K_j(F)SK_i(F)$.

Proposition (S) states that the formula $K_i(F) \Rightarrow K_j(F)SK_i(F)$ is valid in $IS = (R, I)$.

Proposition (B)
(for all $r(m) \in S)[(IS, r(m)) \models K_i(F) \Rightarrow K_j(F)BK_i(F)$.

Thus, the formula $K_i(F) \Rightarrow K_j(F)BK_i(F)$ is valid in $IS = (R, I)$.

Future Propositions

Proposition $(\circ)$
If $S[j,r](m+1) \subseteq S[i,r](m)$, then
$(IS, r(m)) \models K_i(F) \Rightarrow \circ K_j(F)$.

Proposition $(\circ)$ states that at state $r(m)$ agent j will know F in the next step if agent i knows F, under the condition that $S[j,r](m+1) \subseteq S[i,r](m)$ holds.

Proposition $(\Diamond)$
If (for some $m' \geq m)[S[j,r](m') \subseteq S[i,r](m)]$,
then $(IS, r(m)) \models K_i(F) \Rightarrow \Diamond(K_j(F))$.

Proposition $(\Diamond)$ says that at state $r(m)$ agent j will eventually know F if agent i knows F, under the condition that (for some $m' \geq m)[S[j,r](m') \subseteq S[i,r](m)]$ holds.

Proposition $(\Box)$
If (for all $m' \geq m)[S[j,r](m') \subseteq S[i,r](m)]$,
then $(IS, r(m)) \models K_i(F) \Rightarrow \blacksquare K_j(F)$.

Proposition $(\Box)$ says that at state $r(m)$ agent j will always know F if agent i knows F, under the condition that (for all $m' \geq m)[S[j,r](m') \subseteq S[i,r](m)]$ holds.

Proposition $(\mathcal{U})$
If (for some $m' \in \{m, m+1\})[S[j,r](m') \subseteq S[i,r](m)]$, then $(IS, r(m)) \models K_i(F) \Rightarrow K_i(F)\mathcal{U}K_j(F)$.

Thus, the formula $K_i(F) \Rightarrow K_i(F)\mathcal{U}K_j(F)$ holds in $(IS, r(m))$ if the condition (for some $m' \in \{m, m+1\})[S[j,r](m') \Rightarrow S[i,r](m)]$ holds.

Proposition (W)
If [(for some $m' \in \{m, m+1\}$)
$[S[j,r](m') \subseteq S[i,r](m)]$ or
(for all $m' \geq m)[S[j,r](m') \subseteq (S[i,r](m)]]$, then
$(IS, r(m)) \models K_i(F) \Rightarrow K_i(F)WK_j(F)$.

Thus, the formula $K_i(F) \Rightarrow K_i(F)WK_j(F)$ holds in $(IS, r(m))$ if the condition [(for some $m' \in \{m, m+1\})[S[j,r](m') \subseteq S[i,r](m)]$ or (for all $m' \geq m)[S[j,r](m') \subseteq S[i,r](m)]]$ holds.

5 Conclusions

We have considered some important agent properties in multi-agent systems. These properties relate the knowledge of two agents, and can be very helpful if

we analyze the respective multi-agent system. We have stated Proposition (more knowledgeable), the past propositions: Proposition ($\bullet$), Proposition ($\blacklozenge$), Proposition ($\blacksquare$), Proposition (S), and Proposition (B) ; and the future proposition: Proposition ($\circ$), Proposition ($\Diamond$), Proposition ($\Box$), Proposition ($\mathcal{U}$), and Proposition ($\mathcal{W}$). The proofs of all the propositions are given in the Appendix.

APPENDIX

Proof (Proposition (more knowledgeable))
Assume $U : S[j,r](m) \subseteq S[i,r](m)$. We would like to show $(IS, r(m)) \models K_i(F) \Rightarrow K_j(F)$. Suppose $V : (IS, r(m)) \models K_i(F)$. We have to show $(IS, r(m)) \models K_j(F)$. Let $r_i(m_i) \in S$ be an arbitrary state such that $(r(m), r_i(m_i)) \in k_j$. We need to prove $(IS, r_i(m_i)) \models F$. From $(r(m), r_i(m_i)) \in k_j$ it follows $r_i(m_i) \in S[j,r](m)$. We have (by the assumption U) $r_i(m_i) \in S[i,r](m)$. Therefore, $(r(m), r_i(m_i)) \in k_i$. Because V holds, we obtain $(IS, ri(mi)) \models F$. Accordingly, we have $(IS, r(m)) \models K_j(F)$, as desired.

Proof (Proposition ($\bullet$))
Assume $U1 : S[j,r](m-1) \subseteq S[i,r](m)$. We would like to show $(IS, r(m)) \models K_i(F) \Rightarrow \bullet K_j(F)$. Assume $V1 : (IS, r(m)) \models K_i(F)$. We have to show $(IS, r(m)) \models \bullet K_j(F)$, that is, $(IS, r(m-1)) \models K_j(F)$. Let $r_i(m_i) \in S$ be an arbitrary state such that $(r(m-1), r_i(m_i)) \in k_j$. It follows $r_i(m_i) \in S[j,r](m-1)$. We have (by the assumption U1) $r_i(m_i) \in S[i,r](m)$. Therefore, $(r(m), r_i(m_i)) \in k_i$. Because V1 holds, we have $(IS, r_i(m_i)) \models F$. Consequently, we have $(IS, r(m-1)) \models K_j(F)$, as we wanted to show.

Proof (Proposition ($\blacklozenge$))
Assume U2: (for some $m', 0 \leq m' \leq m)[S[j,r](m') \subseteq S[i,r](m)]$. We would like to show $(IS, r(m)) \models K_i(F) \Rightarrow \blacklozenge K_j(F)$. Assume $V2 : (IS, r(m)) \models K_i(F)$. We have to show $(IS, r(m)) \models \blacklozenge K_j(F)$. Let $m', 0 \leq m' \leq m$, be such a point that (by the assumption U2) $U2' : S[j,r](m') \subseteq S[i,r](m)$ holds. We shall prove $(IS, r(m')) \models K_j(F)$. Let $r_i(m_i) \in S$ be an arbitrary state such that $(r(m'), r_i(m_i)) \in k_j$, that is, $r_i(m_i) \in S[j,r](m')$. It follows (by the assumption U2) $r_i(m_i) \in S[i,r](m)$. Therefore, $(r(m), ri(m_i)) \in k_i$. We obtain (by the assumption V2) $(IS, r_i(m_i)) \models F$. Consequently, we have $(IS, r(m')) \models K_j(F)$, that is, $(IS, r(m)) \models \blacklozenge K_j(F)$., as desired.

Proof (Proposition ($\blacksquare$))
Assume U3: (for all $m', 0 \leq m' \leq m)[S[j,r](m') \subseteq S[i,r](m)]$. We would like to show $(IS, r(m)) \models K_i(F) \Rightarrow \blacksquare K_j(F)$. Assume V3: $(IS, r(m)) \models K_i(F)$. We have to show $(IS, r(m)) \models \blacksquare K_j(F)$. Let m' be an arbitrary point such that $0 \leq m' \leq m$. We need to prove $(IS, r(m')) \models K_j(F)$. Let $r_i(m_i) \in S$ be an arbitrary state such that $(r(m'), r_i(m_i)) \in kj$. We have to prove $(IS, r_i(m_i)) \models F$. From $(r(m'), r_i(m_i) \in kj$, it follows $r_i(m_i) \in S[j,r](m')$. We have (by U3) $r_i(m_i) \in S[i,r](m)$. Therefore, $(r(m), r_i(m_i)) \in k_i$. We have (by V3) $(IS, r_i(m_i)) \models F$. Consequently, we obtain $(IS, r(m')) \models K_j(F)$, that is, $(IS, r(m)) \models \blacksquare K_j(F)$, as we wanted to show.

Proof (Proposition (S))
We shall prove that the formula $K_i(F) \Rightarrow K_j(F)SK_i(F)$ is valid in $IS = (R, I)$. Let $r(m) \in S$ be an arbitrary state. We have to prove $(IS, r(m)) \models K_i(F) \Rightarrow K_j(F)SK_i(F)$ Assume V4: $(IS, r(m)) \models K_i(F)$. We need to show $(IS, r(m)) \models K_j(F)SKi(F)$, that is, (for some $m', 0 \leq m' \leq m)[(IS, r(m')) \models K_i(F)]$ and (for all $m'', m' < m'' \leq m)[(IS, r(m'')) \models K_j(F)]$. It is easy to see that if we take $m' = m$, then we have $(IS, r(m)) \models K_i(F)$ (by V4), and (for all $m'', m < m'' \leq m)[(IS, r(m'')) \models K_j(F)]$ (because the requirenment that $K_j(F)$ holds at all states $r(m'')$, such that $m < m'' \leq m$, is fulfilled vacuously.

Proof (Proposition (B))
We would like to show that the formula $K_i(F) \Rightarrow K_j(F)BK_i(F)$ is valid in $IS = (R, I)$. Because we have proved that the formula $K_i(F) \Rightarrow K_j(F)SK_i(F)$ is valid in $IS = (R, I)$, and because $(IS, r(m)) \models K_j(F)BK_i(F)$ iff $(IS, r(m)) \models K_j(F)SK_i(F)$ or $(IS, r(m)) \models \blacksquare K_j(F)$, it follows that the formula $K_i(F) \Rightarrow K_j(F)BK_i(F)$ is valid in $IS = (R, I)$ too.

Proof (Proposition ($\circ$))
Assume U5: $S[j,r](m+1) \subseteq S[i,r](m)$. We would like to show $(IS, r(m)) \models K_i(F) \Rightarrow \circ K_j(F)$. Assume V5: $(IS, r(m)) \models K_i(F)$. We have to prove $(IS, r(m)) \models (K_j(F)$, that is, $(IS, r(m+1)) \models K_j(F)$. Let $r_i(m_i) \in S$ be an arbitrary state such that $(r(m+1), r_i(m_i)) \in kj$. It follows $r_i(m_i) \in S[j,r](m+1)$. We have (by U5) $r_i(m_i) \in S[i,r](m)$. Therefore, $(r(m), r_i(m_i)) \in k_i$. We obtain (by V5) $(IS, r_i(m_i)) \models K_j(F)$. Accordingly, we have $(IS, r(m+1)) \models K_j(F)$, that is, $(IS, r(m)) \models (K_j(F)$, as we wanted to show.

Proof (Proposition ($\Diamond$))
Assume U6: (for some $m' \geq m)[S[j,r](m') \subseteq S[i,r](m)]$. We would like to show $(IS, r(m)) \models K_i(F) \Rightarrow \Diamond K_j(F)$. Assume V6: $(IS, r(m)) \models K_i(F)$. We have to show $(IS, r(m)) \models (K_j(F)$. Let $m' \geq m$ be such a point that (by U6) U6':

$S[j,r](m') \subseteq S[i,r](m)$ holds. We shall prove $(IS, r(m')) \models K_j(F)$. Let $r_i(m_i) \in S$ be an arbitrary state such that $(r(m'), r_i(m_i)) \in k_j$, that is, $r_i(m_i) \in S[j,r](m')$. We have (by U6') $r_i(m_i) \in S[i,r](m)$. Therefore, $(r(m), r_i(m_i)) \in k_i$. ¿From V6 it follows $(IS, r_i(m_i)) \models F$. Consequently, we have $(IS, r(m')) \models K_j(F)$, that is, $(IS, r(m)) \models \Diamond K_j(F)$, as desired.

Proof (Proposition $(\Box)$**)**
Assume U7: (for all $m' \geq m)[S[j,r](m') \subseteq S[i,r](m)]$. We would like to show $(IS, r(m)) \models K_i(F) \Rightarrow \Box K_j(F)$. Assume V7: $(IS, r(m)) \models K_i(F)$. We have to prove $(IS, r(m)) \models \Box K_j(F)$. Let $m' \geq m$ be an arbitrary point such that U7': $S[j,r](m') \subseteq S[i,r](m)$ holds. We need to show $(IS, r(m')) \models K_j(F)$. Let $r_i(m_i) \in S$ be an arbitrary state such that $(r(m'), r_i(m_i)) \in k_j$, that is, $r_i(m_i) \in S[j,r](m')$. We have (by U7') $r_i(m_i) \in S[i,r](m)$. Therefore, $(r(m), r_i(m_i)) \in k_i$. We obtain (by V7) $(IS, r_i(m_i)) \models F$. Thus, we have $(IS, r(m')) \models K_j(F)$, that is, $(IS, r(m)) \models \Box K_j(F)$, as desired.

Proof (Proposition $(\mathcal{U})$**)**
Assume U8: (for some $m' \in \{m, m+1\})[S[j,r](m') \subseteq S[i,r](m)]$. We would like to show $(IS, r(m)) \models K_i(F) \Rightarrow K_i(F) \mathcal{U} K_j(F)$. Assume V8: $(IS, r(m)) \models K_i(F)$. We need to show $(IS, r(m)) \models K_i(F) \mathcal{U} K_j(F)$, that is, (for some $m' \geq m)[(IS, r(m')) \models K_j(F)]$ and (for all $m'', m \leq m'' < m')[(IS, r(m'')) \models K_i(F)]$. Let $m' \in \{m, m+1\}$ be such a point that (by U8) U8': $S[j,r](m') \subseteq S[i,r](m)$ holds. We shall show that $(IS, r(m')) \models K_j(F)$ and (for all $m'', m \leq m'' < m')[(IS, r(m'')) \models K_i(F)]$. Let $r_i(m_i) \in S$ be an arbitrary state such that $(r(m'), r_i(m_i)) \in k_j$, that is, $r_i(m_i) \in S[j,r](m')$. We have (by U8') $r_i(m_i) \in S[i,r](m)$, that is, $(r(m), r_i(m_i)) \in k_i$. It follows (by V8) $(IS, r_i(m_i)) \models F$. It means that $(IS, r(m')) \models K_j(F)$ holds. Because $m' \in \{m, m+1\}$, that is, $m' = m$ or $m' = m+1$, we have that VR: (for all $m'', m \leq m'' \leq m')[(IS, r(m'')) \models K_i(F)]$ holds. Namely, if $m' = m$, then VR is fulfilled vacuously, and if $m' = m+1$, then VR is reduced to V8.

Proof (Proposition (W)**)**
Assume U8 (from Proposition $(K_i \Rightarrow K_i \mathcal{U} K_j)$) or U9: (for all $m' \geq m)[S[i,r](m') \subseteq S[i,r](m)]$. We would like to show $(IS, r(m)) \models K_i(F) \Rightarrow K_i(F) W K_j(F)$. Assume V9: $(IS, r(m)) \models K_i(F)$. We have to prove (W): $(IS, r(m)) \models K_i(F) W K_j(F)$. Because (W) iff $(IS, r(m)) \models K_i(F) \mathcal{U} K_j(F)$ or $(IS, r(m)) \models \Box K_i(F)$, we have (by U8 or U9) that (W) holds, as we wanted to show.

ACKNOWLEDGMENT
I am grateful to the anonymous referees for their careful reading and useful comments.

References

[1] Fagin, R. et al. (1995). Reasoning about Knowledge. The MIT Press.

[2] Maleković, M. (1998) Multi-Agent Systems: Incorporating Knowledge and Time, International Conference on Information and Intelligent Systems IIS '98.

[3] Manna, Z. and A. Pnueli (1992). The Temporal Logic of Reactive and Concurrent Systems. Springer-Verlag.

[4] Moses, Y. and B. Bloom (1994). Knowledge, timed precedence and clocks. In Proc. 13^{th} ACM Symp. on Principles of Distributed Computing, 294-303.

Extended Predicate Logic and Its Application in Designing MKL Language

Honghua Dai
School of Mathematics and Computer Sciences,
The University of New England
Armidale, NSW 2351, Australia
hdai@cs.une.edu.au

Keywords: Extended Predicate Logic, MKL Language, Knowledge Representation, Expert System Tools, Weather Forecasting Systems

Edited by: Xindong Wu
Received: July 30, 1997 **Revised:** June 17, 1998 **Accepted:** August 8, 1997

In some cases, we may find difficulties or inconveniences in representing rules in conventional logic, or sometimes the logical expressions are too complex. This paper gives two different kinds of new approaches to solve this problem. Firstly, we integrate the conjunction and disjunction using only one OA predicate $\mathcal{A}_m$. Secondly, by using extended predicates. As a consequence, an easier inference mechanism is introduced. Compared to the conventional logic and Prolog knowledge representation method, this approach enables us to greatly simplify an expression, and save a considerable amount of time and space in knowledge processing. It had been successfully applied in developing a specific knowledge representation language: MKL (Meteorological Knowledge representation Language).
The MKL has been applied in building a number of practical weather forecasting expert systems including the WMES I and II, the Wuhan heavy rain forecasting expert systems. Some of the systems have been in operation since 1985. On average, the predictive accuracy rate of the systems are very competitive with that made by a human domain expert. In all of these systems, the knowledge is provided by human experts and encoded using MKL. We expect that an MKL-based learning system can be developed for the automatic discovery of weather forecasting rules, and an MKL-based comprehensive system integrating learning, forecasting, data and knowledge applications can be further built.

1 Introduction

Knowledge representation (KR) has traditionally been thought of as the heart of artificial intelligence[2], and first order logic(FOL) is the key to the heart. Unfortunately, in some cases, it was not always convenient to represent some kinds of rules. This caused a new extension to the first order logic.

First order logic, also called predicate calculus, is a formal language used to represent relations amongst objects and to infer new relations from existing ones. It may be viewed as a formal language used to translate English sentences and to derive new sentences from known ones[11]. FOL is also called a classical, conventional or formal logic.

Usually first order logic plays a basic role in formal approaches to knowledge representation (KR). Especially, FOL is very important in applications of artificial intelligence, such as rule based expert systems, theorem proving, and natural language understanding.

However, many difficulties arise in expressing knowledge with FOL. Therefore, several extended logics based on FOL were developed [12] [15, 16] [14, 3, 6]. These extended logics include *multi-valued logic, fuzzy logic, modal logic, matrix logic* etc. Some recent work in automated reasoning has further developed description logics[10, 1].

For the purpose of improving knowledge representation in logical expressions, we introduced another extension to conventional logic, which integrates two FOL connection operations, conjunction and disjunction, with a single OA predicate $\mathcal{A}_m$. In the term OA predicate the O stands for logical connective OR and the A for AND. This new extended logic is called *Argumented Predicate logic* which can be applied to solve some common logical representation problems in many areas such as, knowledge representation, information retrieval, functional programming and logical programming. The original contribution of this approach is that it can (1) integrate two FOL connection operations as one, (2) extend the logical operations AND and OR to the level where the logical opera-

tion lies between disjunction and conjunction, and (3) significantly reduce logical operation time, and speed up logical inference. This approach also uses various functional representations to represent logical operations with arguments. In this way, it is easy to handle problems, such as, to satisfy m conditions amongst many conditions.

In Section 2 we pose two problems which are difficult to handle with the conventional FOL. In Section 3 we describe a new method which uses Argumented FOL to solve the proposed problems, and we present the corresponding laws, lemmas and theorems of the argumented predicate approach. In Section 4 we look at an implemented application for the purpose of domain knowledge representations in Meteorological Knowledge Language(MKL). In section 5 we give a comparison between our Argumented predicate approach and other knowledge representation methods, e.g. conventional FOL. In the conclusion, we point out some further researches and applications.

2 Special Needs in Domain Knowledge Representation

Using deep domain knowledge is one of the key points in the research frontier of knowledge based systems[2]. Domain knowledge representation is the kernel of domain knowledge discovery and its applications. The two problems we will indicate in this section are the two typical problems in some domain knowledge representation.

Problem 1. m-out-of-n problem

We also call the m-out-of-n problem the Condition-Arbitrary-Selection problem.

Definition 2.1 **m-out-of-n problem**
A problem is called an m-out-of-n problem if it is required to represent a rule that for n given goals, if at least any m ($m \leq n$) out of n are true, then the final goal is achieved.

In general, this problem could be described as: *succeed if at least any m of n goals are true.* Suppose that we have n logical expressions, say,

$$e_1, e_2, ..., e_n$$

and we expect to represent a rule say,

Rule 2.1 *If any $m(m \leq n)$ of the n expressions hold true, then r_1 holds.*

where, r_1 is the consequence. If we represent **RULE 2.1** in conventional first order logic, it could be represented as,

$$(e_1 \wedge e_2 \ldots e_{m-1} \wedge e_m) \vee (e_1 \wedge e_2 \ldots e_{m-1} \wedge e_{m+1}) \vee \ldots$$

$$(e_1 \wedge e_2 \ldots e_{m-1} \wedge e_n) \vee (e_2 \wedge e_3 \ldots e_m \wedge e_{m+1}) \vee \ldots$$

$$(e_2 \wedge e_3 \ldots e_m \wedge e_n) \vee \ldots \vee$$

$$(e_{n-m+1} \wedge e_{n-m+2} \ldots e_{n-1} \wedge e_n) \Longrightarrow r_1 \qquad (1)$$

This expression is found to be very inconvenient and too complex as there are too many redundancies. In this expression, the connective $\wedge$ occurs $(m-1)\binom{m}{n}$ times. Each e_i $(1 \leq i \leq e)$ occurs $\binom{m-1}{n-1}$ times and connective $\vee$ occurs $\binom{m}{n} - 1$ times.

An improved expression is available which is called *Polish Prefix Notion* (PPN). In PPN, the m-out-of-n problem could be represented as,

$$\vee(\wedge(e_1, e_2, \ldots, e_m), \wedge(e_1, e_2, \ldots, e_{m-1}, e_{m+1}),$$

$$\ldots, \wedge(e_1, e_2, \ldots, e_{m-1}, e_n),$$

$$\wedge(e_2, e_3, \ldots, e_m, e_{m+1}), \ldots, \wedge(e_2, e_3, \ldots, e_m, e_n), \ldots,$$

$$\wedge(e_{n-m+1}, e_{n-m+2}, \ldots, e_{n-1}, e_n)) \Longrightarrow r_1 \qquad (2)$$

In the expression (2), the connective $\wedge$ occurs $\binom{m}{n}$ times. Each e_i occurs the same times as in expression (1), and the connective $\vee$ occurs once only.

Now we consider this problem first: is it possible to find an equivalent expression, in which only one predicate is used and each e_i $(1 \leq i \leq e)$ only occurs once? The answer is positive, and this will be dealt with in the following sections.

In practice, let us take the logical programming language PROLOG as an example. In PROLOG, by using *at_least(M, [Goals...])* expression, which is true when at least M of the Goals are true. In this way the *m_out_of_n* problem could be represented as,

```
at_least(M, Goals) :-
        integer(M),
        M >= 0,
        length(Goals, N),
        M =< N,
        at_least(M, Goals, N).

at_least(0, _, N) :- !.
at_least(M, [Goal|Goals], N) :-
        call(Goal),
        M1 is M - 1,
        N1 is N - 1,
        at_least(M1, Goals, N1).
at_least(M, [_|Goals], N) :-
        N1 is N - 1,
        M =< N1,
        at_least(M, Goals, N1).
```

To insist that at least 2 of three goals succeed, we might write

```
r1(X) :-
        at_least(2, [p(X), q(X)]).
```

It is obvious that the representation is not as simple and clear as we desired. So we should find a better way to solve this problem.

Problem 2. Two-Bound-Relation Problem

Definition 2.2 Two-Bound-Relation problem *A problem is called a Two-Bound-Relation problem if it is required to represent that for two given bounds a and b, there are n variables which lie between them.*

Suppose that we have n arguments, say, $x_1, x_2, \ldots, x_n$, and two constants (or arguments) a and b. We are required to represent a rule as,

Rule 2.2

```
If ( a <= x1 <= b) and
        ( a <= x2 <= b) and
        ..........
        ( a <= xn <= b)
 Then r2.
```

In first order logic, it could be represented as,

$$((a \leq x_1) \wedge (x_1 \leq b)) \wedge \ldots$$

$$\ldots((a \leq x_n) \wedge (x_n \leq b)) \Longrightarrow r_2. \qquad (3)$$

In this expression the connective $\wedge$ occurs $2n-1$ times, the relational operator $\leq$ occurs $2n$ times and the two bonds a and b occur n times respectively. In PPN approach, it could be represented as,

$$\wedge(\leq(a, x_1), \leq(x_1, b), \ldots, \leq(a, x_n), \leq(x_n, b))$$

$$\Longrightarrow r_2 \qquad (4)$$

In this PPN expression, the connective $\wedge$ occurs only once, but the relational operator $' \leq'$ occurs $2n$ times and the a and b happen n times each.

Now, a similar question arises: can we use just one predicate instead of $2n-1$ times of $\wedge$, and just 2 relational operators instead of $2n$ relational operators, and two bounds a, b only once, instead of n times? This is the problem we are going to solve.

How can this *two_bond problem* be processed in recent knowledge processing tools then? We will give a method of meeting these requirements. For the specific example a = 1, b = 10, and 3 variables, in PROLOG, it could be represented as,

```
r2(X1, X2, X3) :-
        1 =< X1, X1 =< 10,
        1 =< X2, X2 =< 10,
        1 =< X3, X3 =< 10.
```

This expression is almost the same as expression (3). There is a library predicate called *between/3.*

$$between(Variable, LowerBound, UpperBound)$$

This PROLOG expression means that

$$LowerBound \leq Variable \leq UpperBound$$

in which, *LowerBound, UpperBound* and *Variable* are all integers. By using *between(Variable, LowerBound, UpperBound)* , the problem can be solved for its first argument. So,

```
r2(X1, X2, X3) :-
          between(X1, 1, 10),
          between(X2, 1, 10),
          between(X3, 1, 10).
```

has 1000 possible answers. Or alternatively, it could be represented as,

```
r2 :-
      var1(X1), between(X1, 1, 10),
      var2(X2), between(X2, 1, 10),
      var3(X3), between(X3, 1, 10).
```

In this expression the 3-ary relational predicate *between* occurs n times, so do the lower bound and the upper bound. And the logical connective *and* occurs implicitly $n-1$ times.

Obviously, these expressions are not what we expected. None of them could express either of the two problems using only one operator, and either of the operands could appear only once. In the next section, we will give a method which meets these requirements.

3 Argumented Predicate Approach (APA)

To solve the problems which we raised in §2, we suggest a new method of handling the difficulties. First, let us give the definition of the argumented predicate.

Definition 3.1 Argumented Predicate. *A predicate is called an Argumented predicate if there is at least one argument attached to the predicate.*

In the Argumented predicate approach each of the above two rules can be simply represented as one very short and clear statement.

For the **m-out-of-n problem**, the expression (1) could be simply represented as,

$$\mathcal{A}_m(e_1, e_2, \ldots e_n) \Longrightarrow r1. \qquad (5)$$

For the second problem, i.e, the *Two – Bound – Relation* Problem, we may write the expression (3) as follows,

$$\mathcal{A}_2(\tilde{G}e_a, \tilde{L}e_b)(x_1, x_2, \ldots, x_n) \Longrightarrow r_2. \quad (6)$$

In the specific example for n=3 , we have the following expression,

$$\mathcal{A}_2(\tilde{G}e_1, \tilde{L}e_{10})(x_1, x_2, x_3) \Longrightarrow r_2. \quad (7)$$

3.1 Integration of disjunction and conjunction

Definition 3.2 n-overlapped predicate
An Argumented predicate is called an n-overlapped predicate, if there are n arguments(some of them may be constants) attached to the Argumented predicate.

An 0-overlapped predicate is a conventional predicate.

In formula (5), $\mathcal{A}_m$ is a 1-overlapped Argumented predicate in which $\mathcal{A}$ is a predicate and m is an attached argument to the predicate $\mathcal{A}$. For instance, in the problem of m out n, the one attached argument is m(in the general case, it is a constant). $>_\beta (y)$ and $>_\beta (x_1, x_2, x_3)$ are all 1-overlapped Argumented predicate functions and β is an attached argument. In which, $>_\beta (y)$ is a 1-ary 1-overlapped Argumented predicate function, and $>_\beta (x_1, x_2, x_3)$ is a 3-ary 1-overlapped Argumented predicate function. $BT_{\alpha,\beta}(x)$ is a 2-overlapped Argumented Predicate function.

Lemma 3.1 Transformation Law
Let P(x,y) be a 2-ary conventional predicate function. It can be transformed into a 1-ary Argumented predicate function, denoted as B_β (α). Such that,

$$P(x, y) = B_\beta(\alpha) \quad (8)$$

where the B_β is a 1-overlapped Argumented predicate with its argument β, and the $\alpha \in \{x, y\}$. In the special case, β could be a constant. In general, $\beta \in \{\{x, y, r, \} - \{\alpha\}\}$, r is a constant.

In conventional predicate logic, "John is the son of James" is represented as

$$Son\_of(John, James)$$

By 1-overlapped Argumented predicate, it could be represented as,

$$Son\_of_{James}(John)$$

Theorem 3.1 Extended Transformation Law *An n-ary conventional predicate function $P(x_1, x_2, \ldots, x_n)$ can be transformed into an (n-1)-overlapped 1-ary argumented predicate function $\mathbf{B}_{\alpha_1,\alpha_2,\ldots,\alpha_{n-1}}(x)$, such that,*

$$P(x_1, x_2, \ldots, x_n) = \mathbf{B}_{\alpha_1,\alpha_2,\ldots,\alpha_{n-1}}(x) \quad (9)$$

where $x \in \{x_1, x_2, \ldots, x_n\}$.

Proof:

Suppose that α possesses B feature, according to the predicate logic, it could be represented as $B(\alpha)$.

For a 2-ary conventional predicate function P(x,y), first, we let x stipulate n specific objects

$$\beta_1, \beta_2, \ldots \beta_n$$

Thus, $\forall \beta_i \in A = \{\beta_1, \beta_2, \ldots, \beta_n\}, (1 \leq i \leq n)$, we get $B(\alpha, \beta_i)(i = 1, 2, \ldots, n)$. We take β_i out of $B(\alpha, \beta_i)$ and form an argumented predicate B_{β_i}. Then, we have an argumented predicate function

$$B_{\beta_i}(\alpha) \quad (i = 1, 2, \ldots, n) \quad (10)$$

This new expression is equivalent to $B(\alpha, \beta_i)$. We only change the form of the expression. Assume that the object domain of β is A_0. Let $A \to A_0$, thus,

$$B_{\lim_{A \to A_0} \beta_i}(\alpha) = B_\beta(\alpha) \quad (11)$$

Thus, we transformed a 2-ary predicate function $B(\alpha, \beta_i)$ into a 1-ary argumented predicate function $B_\beta(\alpha)$.

According to the inductive principle, repeat the procedure as above, we get,

$$P(x_1, x_2, \ldots, x_n) = \mathbf{B}_{\alpha_1,\alpha_2,\ldots,\alpha_{n-1}}(x) \quad (12)$$

□

Corollary 3.1 Transformation Law.
An n-ary conventional predicate function $P(x_1, \ldots, x_n)$ can be transformed into an m-overlapped (n-m)-ary argumented predicate function in the form of $\mathbf{B}_{\alpha_1,\ldots,\alpha_m}(\beta_1, \ldots, \beta_{n-m})$, such that,

$$P(x_1, \ldots, x_n) = \mathbf{B}_{\alpha_1,\ldots,\alpha_m}(\beta_1, \ldots, \beta_{n-m}) \quad (13)$$

□

Among the Argumented predicates we introduced, there are two special cases which are the most significant. The first one is the one-overlapped Argumented predicate $\mathcal{A}_i$, which is called an *OA* predicate. The second one is the combined Argumented relational predicate in the general form of $\mathcal{A}_i(OP^{(1)}{}_\alpha, OP^{(2)}{}_\beta)(i = 1, 2)$,where the $OP^{(i)}, i = 1, 2$ are argumented relational predicates, such as $\mathcal{A}_2(\geq 16, \leq 90)$.

Similarly we can extend Lemma 3.1 to any $n - ary$ conventional predicate function. We can transform it from $1 - overlapped$ argumented predicate function to $(n-1) - overlapped$ argumented predicate functions.

Definition 3.3 *OA* Predicate
An argumented predicate $\mathcal{A}_i$ is called an OA predicate if,

1. *$\mathcal{A}_i$ can and only can act upon n conditional expressions $e_1, e_2, ..., e_n$ i.e.,*

$$\mathcal{A}_i(e_1, e_2, ..., e_n)$$

which is denoted as L_i. In which, i is any integer. L_i is a logical expression. The case when i is a negative integer will be discussed after we introduce the defective restriction *in Definition 3.5.*

2. *The truth value of the expression $L_i = \mathcal{A}_i(e_1, e_2, ..., e_n)$ is defined as follows,*

$$L_i = \begin{cases} true & iff\ at\ least\ any\ i\ of\ n\ expressions \\ & e_1, e_2, ..., e_n\ are\ true \\ false & otherwise \\ & when\ \ i \in \{1, 2, \ldots, n\} \end{cases} \tag{14}$$

Lemma 3.2 Properties
According to the definition of OA Predicate, we can derive four significant special properties of the OA predicate $\mathcal{A}_i$:

1. *$i \leq 0$, $\mathcal{A}_i(e_1, ..., e_n) = 1$, in this case, none of the conditions is necessary.*
2. *$i = 1$, $\mathcal{A}_i(e_1, ..., e_n) = \mathcal{A}_1(e_1, ..., e_n) = \vee(e_1, ..., e_n)$, any one of the conditions is needed.*
3. *$i = n$, $\mathcal{A}_n(e_1, ..., e_n) = \mathcal{A}_n(e_1, ..., e_n) = \wedge(e_1, ..., e_n)$, all the conditions must be met.*
4. *$i > n$, $\mathcal{A}_i(e_1, ..., e_n) = 0$, all the n conditions plus $i - n$ hidden conditions must be satisfied.*

¿From the above Definition 3.3 and Lemma 3.2, we find that the disjunction $\vee$ and conjunction $\wedge$ are only two special cases of OA predicate $\mathcal{A}_i$ when i=1 and i=n. That is to say, $\mathcal{A}_1 = \vee$, $\mathcal{A}_n = \wedge$. In the following example we will use $\mathcal{A}_1$ and $\mathcal{A}_2$, or any $\mathcal{A}_i$. Therefore, the OA predicate $\mathcal{A}_i$ is regarded not only as an integration but also as an extension of disjunction and conjunction.

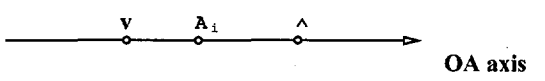


Figure 1: Illustration of OA axis

If we denote $\Re(\alpha)$ the restriction degree of predicate α, then we have the following relation

$$\Re(\vee) \leq \Re(\mathcal{A}_i) \leq \Re(\wedge) \tag{15}$$

where $1 \leq i \leq n$

No	Predicate p	Restriction Value $\Re(p)$
1	Not	n
2	$\vee$	1
3	$\wedge$	n
4	$\mathcal{A}_i$	$i(1 \leq i \leq n)$

Table 1: Table of Restriction Value

Definition 3.4 Restriction value
For a given predicate p, the restriction value $\Re(p)$ is the value which describes the restriction degree of the predicate p.

In the Argumented predicate approach, the restriction values of some predicates are defined in Table 1. In the table, n is the number of possible logical expressions which are acted on by the predicates, e.g., in the expression, $\wedge(e_1, e_2, e_3)$ and $\mathcal{A}_2(e_1, e_2, e_3)$, the restriction values are $\Re(\wedge) = 3$ and $\Re(\mathcal{A}_2) = 2$.

Lemma 3.3 Property
The restriction value $\Re(p)$ of a predicate p IS expression dependent. It may vary in inferences. Therefore, it is also a value for describing the dynamical state of a predicate.

Corresponding to the four significant special cases of the OA predicate $\mathcal{A}_m$, we can have four particular restrictions.

Definition 3.5 *Let $\mathcal{A}_m$ be an OA predicate in an OA predicate function $\mathcal{A}_m(e_1, \ldots, e_n)$. This function is said to be*

1. *Defective Restriction, iff $m \leq 0$;*
2. *Unique Restriction, iff $m = 1$;*
3. *Global Restrictions, iff $m = n$;*
4. *Over Restriction, iff $m \geq n$.*

By this definition, the logical connective $\vee$ is a unique restriction operation, i.e. in the expression

$$\vee(e_1, e_2, ..., e_n) = \mathcal{A}_1(e_1, e_2, ..., e_n)$$

the logical connective $\vee$ or argumented predicate $\mathcal{A}_1$ need and only need to find any one logical expression e_i which is true if there is one. That is to say among the n logical expressions $e_1, e_2, ..., e_n$, we need to have one and only one expression e_i to be true in order to achieve the final goal. In the expression

$$\wedge(e_1, e_2, ..., e_n) = \mathcal{A}_n(e_1, e_2, ..., e_n)$$

the logical connective $\wedge$ or the equivalent argumented predicate $\mathcal{A}_n$ has to act upon all the logical expressions $e_i(1 \leq i \leq n)$ that is why $\mathcal{A}_n(or\ \wedge)$ is called a global restriction predicate. This shows that the operator

$\mathcal{A}_1 = \vee$ and $\mathcal{A}_n = \wedge$. In the same reason, by the way of the usage in our approach, the logical connective $\neg$ is also a global restriction predicate because in the expression

$$\neg(e_1, e_2, ..., e_n)$$

the connective $\neg$ will act and must act upon all the logical expression $e_i (1 \leq i \leq n)$. i.e.,

$$\neg(x_1, x_2, x_3, x_4, \ldots, x_n) = (\neg x_1, \neg x_2, \neg x_3, \ldots, \neg x_n)$$

the defective restriction predicate $\mathcal{A}_i$ $(i \leq 0)$ can have two physical explanations. First, for $i = 0$, it means in a rule,

$$\mathcal{A}_0(e_1, e_2, \ldots, e_n) \Longrightarrow r_1$$

the "r_1 holds true" has nothing to do with $e_j (1 \leq j \leq n)$. Second, for $i < 0$, it means that except for thesen expressions, there are i more expressions, for which no matter whether they are true or no, r_1 will always hold. This expression is therefore also called a valid formula. The over restriction predicate $\mathcal{A}_i (i > n)$ means that the over restriction predicate function,

$$\mathcal{A}_i(e_1, e_2, ..., e_n)$$

will be true if there are more than these n logical expressions $e_1, e_2, ..., e_n$ which hold true. In other words,

$$\mathcal{A}_i(e_1, e_2, ..., e_n) = 1$$

if and only if,

$$\mathcal{A}_2(\mathcal{A}_n(e_1, ..., e_n), \mathcal{A}_{n-m}(\tilde{e_1}, ..., \tilde{e_{m-n}}))$$

where $\tilde{e_1}, \tilde{e_2}, ..., \tilde{e_{m-n}}$ are some other logical expressions. Becase, in fact, we do not know what $\tilde{e_1}, \tilde{e_2}, ..., \tilde{e_{m-n}}$ are. Here $\mathcal{A}_m(e_1, e_2, ..., e_n)$ is always false, which therefore, is called an invalid formula.

3.2 Properties of Argumented Predicate Logic

The laws for this extended logic are as follows,

Basic Laws

$$\mathcal{A}_1(x, 0) = x \quad \mathcal{A}_2(x, 0) = 0 \tag{16}$$

$$\mathcal{A}_1(x, 1) = 1 \quad \mathcal{A}_2(x, 1) = x \tag{17}$$

$$\mathcal{A}_n(x_1, \ldots, x_{n-1}, 1) = \mathcal{A}_{n-1}(x_1, \ldots, x_{n-1}) \tag{18}$$

$$\mathcal{A}_n(x_1, x_2, \ldots, x_{n-1}, 0) = 0 \tag{19}$$

Associative laws

$$\mathcal{A}_1(x, \mathcal{A}_1(y, z)) = \mathcal{A}_1(\mathcal{A}_1(x, y), z) = \mathcal{A}_1(x, y, z). \tag{20}$$

$$\mathcal{A}_2(x, \mathcal{A}_2(y, z)) = \mathcal{A}_2(\mathcal{A}_2(x, y), z) = \mathcal{A}_3(x, y, z) \tag{21}$$

Eliminative Laws

1. First Eliminative Laws

$$\mathcal{A}_1(x_1, \mathcal{A}_1(x_2, \mathcal{A}_1(\ldots, x_n)) \ldots)$$

$$= \mathcal{A}_1(x_1, x_2, \ldots, x_n) \tag{22}$$

$$\mathcal{A}_2(x_1, \mathcal{A}_2(x_2, x_3)) = \mathcal{A}_3(x_1, x_2, x_3)$$

$$\mathcal{A}_2(x_1, \mathcal{A}_2(x_2, \mathcal{A}_2(x_3, x_4))) = \mathcal{A}_2(x_1, \mathcal{A}_3(x_2, x_3, x_4)) = \mathcal{A}_4(x_1, x_2, x_3, x_4)$$

In general, we have

2. Second Eliminative Laws

$$\mathcal{A}_2(x_1, \mathcal{A}_2(x_2, \mathcal{A}_2(x_3, \ldots, x_n)) \ldots))$$

$$= \mathcal{A}_n(x_1, x_2, x_3, x_4, \ldots, x_n) \tag{23}$$

Commutative Laws

$$\mathcal{A}_1(x_1, x_2) = \mathcal{A}_1(x_2, x_1) \tag{24}$$

$$\mathcal{A}_2(x_1, x_2) = \mathcal{A}_2(x_2, x_1) \tag{25}$$

Extended Commutative Law

$$\mathcal{A}_m(x_1, x_2, x_3, x_4, \ldots, x_n)$$

$$= \mathcal{A}_m(x_{t_1}, x_{t_2}, x_{t_3}, \ldots, x_{t_n}) \tag{26}$$

Where the $x_{t_1}, x_{t_2}, \ldots, x_{t_n}$ is any combination of the n subscripts 1, 2, 3, ... , n.

Distributive Laws

$$\mathcal{A}_2(x_1, \mathcal{A}_1(x_2, x_3)) = \mathcal{A}_1(\mathcal{A}_2(x_1, x_2), \mathcal{A}_2(x_1, x_3)) \tag{27}$$

$$\mathcal{A}_1(x_1, \mathcal{A}_2(x_2, x_3)) = \mathcal{A}_2(\mathcal{A}_1(x_1, x_2), \mathcal{A}_1(x_1, x_3)) \tag{28}$$

Absorptive Laws

$$\mathcal{A}_2(x, \mathcal{A}_1(x, y)) = x \tag{29}$$

$$\mathcal{A}_1(x, \mathcal{A}_2(x, y)) = x \tag{30}$$

Extended Absorptive Laws

$$\mathcal{A}_1(x, \mathcal{A}_i(x, x_1, x_2, x_3, x_4, \ldots, x_n)) = x \tag{31}$$

$$\mathcal{A}_2(x, \mathcal{A}_1(x, x_1, x_2, x_3, x_4, \ldots, x_n)) = x \tag{32}$$

De Morgan's Law

$$\neg\mathcal{A}_1(x, y) = \mathcal{A}_2(\neg x, \neg y) = \mathcal{A}_2\neg(x, y) \tag{33}$$

$$\neg\mathcal{A}_2(x, y) = \mathcal{A}_1(\neg x, \neg y) = \mathcal{A}_1\neg(x, y) \tag{34}$$

¿From De Morgan's Law, we can get the following extensions,

$$\neg\mathcal{A}_1(x_1, x_2, \ldots, x_n) =$$

$$\mathcal{A}_n(\neg x_1, \neg x_2, \ldots, \neg x_n) =$$

$$\mathcal{A}_n\neg(x_1, x_2, \ldots, x_n)$$

$$\neg\mathcal{A}_2(x_1, x_2, \ldots, x_n) =$$

$$\mathcal{A}_{n-1}(\neg x_1, \neg x_2, \ldots, \neg x_n) =$$

$$\mathcal{A}_{n-1}\neg(x_1, x_2, \ldots, x_n)$$

¿From the above two formulas, we can prove the following extended De Morgan's Law is true, i.e.,

Extended De Morgan's Law

$$\neg\mathcal{A}_m(x_1, x_2, \ldots, x_n)$$

$$= \mathcal{A}_{n-m}(\neg x_1, \ldots, \neg x_n)$$

$$= \mathcal{A}_{n-m}\neg(x_1, \ldots, x_n) \tag{35}$$

3.3 Argumented Relational Predicates

An Argumented Relational Predicate, sometimes, is simply called an AR predicate. If there are n arguments attached to an Argumented relational predicate, it is called an n-overlapped Argumented relational predicate (n=0,1,2,...). In particular, the 0-overlapped relational predicates are conventional relational predicates.

Definition 3.6 Argumented Relational Predicates.
A relational predicate is called an Argumented Relational Predicate if there is at least one argument(or constant) attached to the relational predicate.

A conventional relational expression $e > a$ can be represented as an Argumented relational expression $\tilde{G}t_a e$. But in conventional predicate logic, it should be represented as GT(e,a). Here, the predicate GT is a 0-overlapped relational predicate, i.e., conventional relational predicate. $\tilde{G}t_a$ is an 1- overlapped relational predicate with an argument a attached to the 0-overlapped relational predicate $\tilde{G}t$.

Basic Argumented Relational Predicates

There are five basic argumented relational predicates, $\tilde{G}t_b, \tilde{L}t_b, \tilde{G}e_b, \tilde{L}e_b$ and $\tilde{E}q_b$. The following is a list of the expressions of these five basic argumented relational predicates and their equivalents in conventional logic and in predicate logic.

1. $a > b \Longleftrightarrow GT(a, b) \Longleftrightarrow \tilde{\mathbf{G}}\mathbf{t_b a}$
2. $a < b \Longleftrightarrow LT(a, b) \Longleftrightarrow \tilde{\mathbf{L}}\mathbf{t_b a}$
3. $a \geq b \Longleftrightarrow GE(a, b) \Longleftrightarrow \tilde{\mathbf{G}}\mathbf{e_b a}$
4. $a \leq b \Longleftrightarrow Le(a, b) \Longleftrightarrow \tilde{\mathbf{L}}\mathbf{e_b a}$
5. $a = b \Longleftrightarrow EQ(a, b) \Longleftrightarrow \tilde{\mathbf{E}}\mathbf{q_b a}$

Argumented Two-Bound-Relational Predicates

Suppose that [a,b] is a real interval and x,y are real variables. The general form of the combined Argumented predicates is as follows,

$$\mathcal{A}_i(\left\{ \begin{array}{c} \tilde{G}t_\alpha \\ \tilde{G}e_\alpha \end{array} \right\} \left\{ \begin{array}{c} \tilde{L}t_\beta \\ \tilde{L}e_\beta \end{array} \right\}) \tag{36}$$

i=1,2, ...

So, the general form of an argumented Two-Bound-Relational Predicate function is,

$$\mathcal{A}_i(\left\{ \begin{array}{c} \tilde{G}t_\alpha \\ \tilde{G}e_\alpha \end{array} \right\} \left\{ \begin{array}{c} \tilde{L}t_\beta \\ \tilde{L}e_\beta \end{array} \right\})(e_1, e_2, \ldots\ldots, e_n) \tag{37}$$

$$i=1,2,\ldots$$

The above general form of the predicates can be rewritten as the following eight separated formulas,

1. $a \leq x \leq b \Longleftrightarrow \mathcal{A}_2(\tilde{G}e_a, \tilde{L}e_b)x$
2. $a < x \leq b \Longleftrightarrow \mathcal{A}_2(\tilde{G}t_a, \tilde{L}e_b)x$
3. $a \leq x < b \Longleftrightarrow \mathcal{A}_2(\tilde{G}e_a, \tilde{L}t_b)x$
4. $a \leq x \leq b \Longleftrightarrow \mathcal{A}_2(\tilde{G}t_a, \tilde{L}t_b)x$
5. $x < a \; or \; x > b \Longleftrightarrow \mathcal{A}_1(\tilde{G}t_b, \tilde{L}t_a)x$
6. $x \leq a \; or \; x > b \Longleftrightarrow \mathcal{A}_1(\tilde{G}t_b, \tilde{L}e_a)x$
7. $x < a \; or \; x \geq b \Longleftrightarrow \mathcal{A}_1(\tilde{G}e_b, \tilde{L}t_a)x$
8. $x \leq a \; or \; x \geq b \Longleftrightarrow \mathcal{A}_1(\tilde{G}e_b, \tilde{L}e_a)x$

Corresponding to conventional relational predicates, there are several equivalent relations, such as,

1. $\tilde{G}t_\alpha e \Longleftrightarrow e > \alpha$
2. $\tilde{G}t_\alpha(e_1, e_2, \ldots\ldots, e_n) \Longleftrightarrow (e_1 > \alpha) \wedge (e_2 > \alpha) \wedge (e_3 > \alpha) \ldots \wedge (e_n > \alpha) \Longleftrightarrow \bigwedge_{i=1}^{n}(e_i > \alpha)$
3. $\tilde{L}e_\alpha(e_1, e_2, \ldots\ldots, e_n) \Longleftrightarrow \bigwedge_{i=1}^{n}(e_i \leq \alpha)$
4. $\mathcal{A}_1(GT\ \alpha\ \ LT\ \beta)(e_1, e_2, \ldots\ldots, e_n) \Longleftrightarrow \mathcal{A}_1(\tilde{G}t_\alpha, \tilde{L}e_\beta)(e_1, e_2, \ldots\ldots, e_n) \Longleftrightarrow \bigwedge_{i=1}^{n}((e_i > \alpha) \vee (e_i < \beta))$
5. $\mathcal{A}_2(GE\ \alpha\ \ LT\ \beta)e \Longleftrightarrow \mathcal{A}_2(\tilde{G}e_\alpha, \tilde{L}t_\beta)e \Longleftrightarrow ((e \geq \alpha) \wedge (e < \beta))$
6. $\mathcal{A}_2(GE\ \alpha\ \ LT\ \beta)(e_1, e_2, \ldots\ldots, e_n) \Longleftrightarrow \mathcal{A}_2(\tilde{G}t_\alpha, \tilde{L}e_\beta)(e_1, e_2, \ldots\ldots, e_n) \Longleftrightarrow ((e_1 > \alpha) \wedge (e_1 \leq \beta)) \wedge ((e_2 > \alpha) \wedge (e_2 \leq \beta)) \ldots \wedge ((e_n > \alpha) \wedge (e_n \leq \beta)) \Longleftrightarrow \bigwedge_{i=1}^{n}((e_i > \alpha) \vee (e_i \leq \beta))$

4 MKL Language

The Meteorological Knowledge Representation Language(MKL)[3, 5] is a functional knowledge representation programming language which is based on the argumented predicate logic we discussed above. A full discussion about the application of MKL appeared in [3, 4, 7, 9]. The theoretical part of MKL is represented in [3, 8, 6, 5]. In MKL, a statement is composed of two parts, the operator(s) and the operand(s). Thus, a rule in MKL could be represented in the general form,

$$r \Leftarrow O(P). \tag{38}$$

where O is an operator list, P is an operand list and r is the consequence part of the rule which may include a confidence.

This representation method has several advantages which will be discussed later. In this article, we are not going to give a full description of MKL. We only list some examples to show how the argumented predicates are applied in the implementation and application of the MKL language.

The first problem, i.e, the m-out-of-n problem, in MKL, is simply represented as

$$arb * m(e_1, e_2, \ldots, e_n) \Longrightarrow r_1 \tag{39}$$

The second problem, i.e, the Two-Bound-Relation Problem, in MKL[5, 3] can be represented in the following general form

$$\mathcal{A}_m(\left\{ \begin{array}{c} GT \\ GE \end{array} \right\}\alpha, \left\{ \begin{array}{c} LT \\ LE \end{array} \right\}\beta)(x_1, x_2, \ldots, x_n) \tag{40}$$

$$m = 1, 2, \ldots$$

That is,

$$arb * m(\left\{ \begin{array}{c} GT \\ GE \end{array} \right\}\alpha, \left\{ \begin{array}{c} LT \\ LE \end{array} \right\}\beta)(x_1, x_2, \ldots, x_n) \tag{41}$$

$$m = 1, 2, \ldots$$

For example, to represent a rule, if all the n variables $x_1, x_2, \ldots, x_n$ are greater than a and less than b, then r_2 holds. In MKL, we simply represent it as

$$\mathcal{A}_2(GE\ a\ \ LE\ b)(x_1, \ldots, x_n) \Longrightarrow r_2. \tag{42}$$

Specifically, let a be 1, b be 10 and n=3, then in MKL, we have,

$$\mathcal{A}_2(GE\ 1\ \ LE\ 10))(x_1, x_2, x_3) \Longrightarrow r_2. \tag{43}$$

One limitation of this new representation is that if x_i have different ranges such representation would not be very helpful. However, in the case that x_i have the same range this representation would be very helpful.

Given

Rule 4.1 **If** *any two of the differences of the geopotential height on the 500 hPa between the stations 58847 and 58456, 55585 and 59463 are greater than or equal to 9 geopotential deca-meter height,* **THEN** **NTR** *occurs with confidence 0.86.*

It could be represented in MKL as,

$$\mathbf{NTR}(0.86) \Leftarrow$$

$$\mathcal{A}_2\tilde{G}e_9 - H5(58847, 58456, 55585, 59463). \tag{44}$$

Rule 4.2 **If** *the wind directions on the 500hPa layer over 29612 and 36003 stations are all greater than* 270^0 *or, less than or equal to* 90^0, *and the wind directions on same layer over any two of the three stations 29231, 29634 and 29838 are greater than* 90^0 *and less than or equal to* 270^0, **THEN** *the weather event NTCS occurs.*

Comparison items	FOL	PPN	APA
$\wedge$	$(m-1)\binom{m}{n}$	$\binom{m}{n}$	-
$\vee$	$\binom{m}{n}-1$	1	-
OA predicate	-	-	1
each e_i	$\binom{m-1}{n-1}$	$\binom{m-1}{n-1}$	1

Table 2: Costs in representing m-out-of-n problem

In MKL, it is represented as,

$$\textbf{NTCS} \Leftarrow$$

$$arb * 2(arb * 1(GT\ 270\ \ LE\ 90)D5(29612, 36003);$$

$$arb * 2(arb * 2(GT\ 90\ \ le\ 270)D5(29231, 29634, 29838)). \quad (45)$$

or, in the extended predicate logic, it is represented as,

$$\textbf{NTCS} \Leftarrow$$

$$\mathcal{A}_2(\mathcal{A}_1(\tilde{G}t_{270}, \tilde{L}e_{90})D5(29612, 36003);$$

$$\mathcal{A}_2(\tilde{G}t_{90}, \tilde{L}e_{270})D5(29231, 29634, 29838)). \quad (46)$$

5 Analysis and Comparison

First of all, we compare the argumented predicate approach(APA) with the conventional first order logic method(FOL) and the Polish Prefix Notation(PPN). The following tables show how many times the operators and operands are used in dealing with the two problems.

Compared with the FOL expression and the PPN expression, the differences are listed in Tables 2 and 3.

Compared to the traditional logic, our extended logic has three main advantages.

1. **expressed succinctly**
   The following is the same rule in different expressions. (47) is in extended logic expression.

$$\textbf{NGHR} \Leftarrow$$

$$\mathcal{A}_2(\tilde{L}t_{589}H5(47936, 58847, 59316, 59289);$$

$$\tilde{G}t\_{}_3vh5(55299, 55591, 56004, 56029, 56046,$$

$$56080, 56096, 56137)). \quad (47)$$

In conventional logic, it could be represented as,

$$\textbf{NGHR} \Leftarrow$$

Comparison items	FOL	PPN	APA
$\wedge$	2n-1	1	-
$\leq$	2n	2n	-
AR predicate	-	-	1
a	n	n	1
b	n	n	1
x_i	2	2	1

Table 3: Costs in representing two-bound-relation

$$((H5(47936) < 589) \wedge (H5(58847) < 589) \wedge$$

$$(H5(59316) < 589) \wedge (H5(59287) < 589)) \wedge$$

$$(((H5(55299) - H51(55299)) > -3) \wedge$$

$$((H5(55591) - H51(55591)) > -3) \wedge$$

$$((H5(56004) - H51(56004)) > -3) \wedge$$

$$((H5(56029) - H51(56029)) > -3) \wedge$$

$$((H5(56046) - H51(56046)) > -3) \wedge$$

$$((H5(56080) - H51(56080)) > -3) \wedge$$

$$((H5(56096) - H51(56096)) > -3) \wedge$$

$$((H5(56137) - H51(56137)) > -3). \quad (48)$$

It is very obvious that the extended logic expression (47) is much shorter and clearer than the conventional logical expression (48).

2. **Integration and extension of disjunction and conjunction**
   The OA predicate $\mathcal{A}_i$ is an integration and extension of disjunction $\vee$ and conjunction $\wedge$ in conventional logic. As mentioned earlier, disjunction, $\vee$ and conjunction, $\wedge$ in conventional logic are only two special cases of the OA predicate when i=1 or i=n, that is,

$$\mathcal{A}_1(e_1, e_2, ..., e_n) = \vee(e_1, e_2, ..., e_n)$$

$$\mathcal{A}_n(e_1, e_2, ..., e_n) = \wedge(e_1, e_2, ..., e_n)$$

We have $\mathcal{A}_1 = \vee$ and $\mathcal{A}_n = \wedge$.

3. **Saving time and space**
   Obviously, the Argumented predicate approach improves substantially on the conventional FOL method. From Tables 2 and 3, we can see that for processing the same rule, in APA only one operator is needed. But in FOL $m\binom{m}{n} - 1$ operators are needed. And in PPN $\binom{m}{n} + 1$ operators are needed. We find from the computational point of view, the argumented method can reduce relational operations, retrieval time, and logic operations. For example, in the problem of m out of n, the average operation is $O(n\binom{m}{n})$, however, in our approach it is only $O(n)$.

However, MKL is a special language for the representation of meteorological knowledge. Whereas, FOL, APA, Prolog are the tools for more general purposes. The MKL language is interpreted by the interpreter written in C. The idea , the design strategy and implementation technology of MKL are applicable to many other problems. But to the language itself, the introduced predicates and the functions are mainly for convenient of meteorological knowledge representation and processing. As the MKL interpreter is written in the high language C, the size of the interpreter is very small. As a specific purpose language, it well fit the needs of meteorological knowledge representation, and is convenient for meteorologists. The language adopts logic construction method. It considers both the needs in solving meteorological problems and the advantage of logical expressions. At the meantime it also takes the advantage of Prolog in inference. MKL focuses on the processing of the representation of index type knowledge and the character type knowledge. A meteorological expert system can directly accept, recognize and process the knowledge represented in MKL. Such processing include understanding, explanation and applications in solving real world problems in the area of weather forecasting.

The MKL language has been applied in building a number of weather forecasting expert systems which include the WMES I and II, the Wuhan heavy rain forecasting expert systems. These systems have been in operation since 1995. On average, the predictive accuracy rate of the systems are very competitive with that made by a human domain expert. It was very interesting that in the summer of 1985, two heavy rain events were predicted correctly by the system, but were failed by human experts due to the careful analysis carried out by the system. Later on we discovered the same case. In all of these systems, the knowledge is provided by human experts and encoded in MKL. We expect that a learning system based on the MKL knowledge representation can be developed for the discovery of weather forecasting rules from observational data. Further, an integrated comprehensive system can be built using MKL representation which combines learning, forecasting, data and knowledge analysis and applications.

6 Conclusion

This paper introduced a new approach for handling specific domain knowledge representation difficulties and inconveniences using Argumented predicates. This method can significantly speed up the logical operation process and save a considerable time and space. The idea of the approach is to transform conventional first logic into a flexible Argumented predicate representation, which can greatly reduce the descriptive complexity of a rule and thereby greatly reduce the times of logical operations. We also provide a number of definitions and corresponding lemmas and corollaries.

This research demonstrated some potential benefits to further research in knowledge processing. In particular, for other informal logics, such as fuzzy logic, matrix logic and model logic. Future research will extend the argumented logic to other informal logics, and apply them to solve real application problems. The uncertainty problem [13, p415-467] processing strategies can also be incorporated in the approach. We expect that the application of the OA predicate in machine learning and automatic reasoning could be very fruitful.

References

[1] Liviu Badea. Reifying concepts in description logics. In Martha E. Pollack, editor, *Proceedings of the 15th International Joint Conference on Artificial Intelligence (IJCAI'97)*, pages 142–147, Nagoya, Japan, August 23-29, 1997.

[2] Ronald J. Brachman. The furture of knowledge representation. *AAAI-90 Proceedings*, 1(1):1082–1092, 1990.

[3] Honghua Dai. The development of the WMES system and a study to its theoretical problems. *M.Sc. Thesis, IAP of Academia Sinica*, 1985.

[4] Honghua Dai. A regional heavy rain forecasting expert system— WMES. *China Meteorology*, 3(2), 1986.

[5] Honghua Dai. A study on the WMES system and its theoretical problems. Technical report, IAP, Academia Sinica, 1986.

[6] Honghua Dai. *The Foundations of Meteorological Expert Systems*, volume I & II. IAP, Academia Sinica, Beijing, China, 1986 (in Chinese).

[7] Honghua Dai. The general design scheme of VSRFES project. Technical report, Wuhan National Regional Weather Centre, 1987.

[8] Honghua Dai. Several properties of meteorological knowledge used for expert systems. *Advances in Atmospheric Sciences*, 5(4):515–521, 1988.

[9] Honghua Dai, Qi Song Zheng, and Zhao Xi Zhao. An expert system for predicting heavy rain. *Advances in Atmospheric Sciences*, 4(4):496–505, 1987.

[10] F. Donini, D. Nardi, and R. Rosati. Autoepistemic description logics. In Martha E. Pollack, editor, *Proceedings of the 15th International Joint Conference on Artificial Intelligence (IJCAI'97)*, pages 136–141, Nagoya, Japan, August 23-29, 1997.

[11] Patrick Valduriez Gorges Gardarin. *Relational Database and Knowledge Bases.* Addison-Wesley Publishing Company, Inc., *Reading, Massachusetts*, 1989.

[12] W. Marek and M. Truszczynski. Modal logic for default reasoning. *Annals of Mathematics and Artificial Intelligence*, 1:275–302, 1990.

[13] Stuart Russell and Peter Norvig. *Artificial Intelligence: A Mordern Approach.* Prentice Hall, Englewood Cliffs, New Jersey 07632, 1995.

[14] August Stern. *Matrix Logic.* ElsevierScience Publishers B.V., P.O.Box 1991, 1000 BZ Amsterdam, The Netherlands, 1988.

[15] L. A. Zadeh. Fuzzy sets. *Information and Control*, 8:338–353, 1965.

[16] L. A. Zadeh. Fuzzy sets as a theory of possibility. *Int. J. Man-Machine Stud.*, 1(1):3–28, 1978.

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

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:

http://ai.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 23 Number 2 May 1999 ISSN 0350-5596

Informatica

An International Journal of Computing and Informatics