Current Issues in Mathematical Linguistics
C. Martín-Vide (Editor)



# BABY-SIT: Towards a situation-theoretic computational environment

Erkan Tın and Varol Akman

Department of Computer Engineering and Information Science,
Bilkent University, Bilkent, Ankara 06533, Turkey

Questions of what it means to do computation with situations and what aspects of situation theory makes this suitable as a novel programming paradigm have not been fully discussed in the literature. This is what we hope to initiate here.

## 1. INTRODUCTION

*Situation theory* was first formulated in detail by Jon Barwise and John Perry in 1983 [5] and has matured over the last decade [8]. Various versions of the theory have been applied to a number of linguistic issues, resulting in what is commonly known as *situation semantics* [2, 3, 4, 7]. Individuals, properties, relations, spatio-temporal locations, and situations are the basic constructs of situation theory. The world is viewed as a collection of objects, sets of objects, properties, and relations. *Infons* ('unit' facts) are discrete items of information and *situations* are first-class objects which describe parts of the real world. Information flow is made possible by a network of abstract 'links' between high-order uniformities, viz. *situation types*. One of the distinguishing characteristics of situation theory is that information content is *context-dependent* (where a *context* is a situation) [1].

In this paper, a computational approach to situation theory and its associated environment (called BABY-SIT) are proposed. (The environment is dubbed BABY-SIT because we believe that presently it includes far too many provisional, make-shift design decisions. We trust, on the other hand, that as the 'baby' grows up, these haphazard dimensions will be trimmed and a natural, settled, and balanced kernel will endure as a truer representative of a situation-theoretic computational framework.) The proposed approach especially adopts the ontological features which were originally put forward in [5]. Existing approaches towards a computational account of situation theory unfortunately incorporated only some of these [6, 12]; the remaining features were omitted for the sake of achieving particular goals. This has caused conceptual and philosophical divergence from the ontology of the original theory—a dangerous side effect.

In the past, the development of a *mathematical* situation theory has been held back by a lack of availability of appropriate technical tools. But by now, the theory has assembled its foundations. With a remarkably original view of information [9], a 'logic,' based not on truth but on information, is being developed [8].

## 2. SITUATIONS: A COMPUTATIONAL PERSPECTIVE

Intelligent agents generally make their way in the world by being able to pick up certain information from a situation, process it, and react accordingly [8, 9, 10]. Being in a (mental) situation, such an agent has information about situations it sees, believes in, hears about, etc. Situations can be of the same type. Among the invariants across situations are not just objects and relations, but also aggregates of such. Realization of some type of situation causes the agent to acquire more information about that situation as well as other situation types, and behave accordingly.

An important phenomenon in situation theory is that of *structured* (*nested*) information. Assuming the possession of prior information and/or awareness of other constraints, the acquisition by an agent of an item of information can also provide the agent with an additional item of information. Reaping information from a situation is not the only way an agent processes information. It can also act in accordance of the obtained information to change the environment. Creating new situations to arrive at new information and conveying information it already had to other agents are the primary functions of its activities.

In short, an intelligent agent has the ability to acquire information about situations, obtain new information about them by being attuned to assorted constraints, and act accordingly to alter its environment. All these are ways of processing information about situations. An information processing environment for such an agent should have the following properties:

- Partitioning of information into situations.
- Parametrization of objects to give a proper treatment of abstraction over individuals, situations, etc.
- Structuring of situations in such a way that they allow nested information, and access to information partitioned in this way.
- Access to information in one situation from another situation connected to the former via some relation.
- Constraint satisfaction to control flow of information within and between situations.

These properties would naturally define the underlying mechanisms for a situation-theoretic computational environment. But what constructs are provided by situation theory to build such an environment?

Abstraction can be captured in a primitive level by allowing *parameters* in *infons*. Parameters are abstractions or generalizations over classes of non-parametric objects (e.g., individuals, spatial locations). Parameters of a parametric object can be associated with objects which, if they were to replace the parameters, would yield one of the objects in the class that parametric object abstracts over. The parametric objects actually define types of objects in that class. Hence, letting parameters in infons results in what is called *parametric infons*. *Parameter-free infons* are the basic items of information about the world (i.e., 'facts') while parametric infons are the basic units that are utilized in a computational treatment of information flow.

To construct a computational model of situation theory, it is convenient to have available abstract analogs of objects. As noted above, by using parameters we can have

abstracts which are parametric objects including parametric situations, parametric individuals, etc. This yields a rich set of data types. Abstract situations can be viewed as models of real situations. They are set-theoretic entities that have only some of the features of real situations, but are amenable to computation. We define abstract situations as structures consisting of sets of parametric infons. Information can be partitioned into situations by defining a hierarchy between situations. A situation can be larger, having other situations as its subparts. Being in this larger situation gives the ability of having information about its subsituations. The *part-of* relation of situation theory can be used to build such hierarchies among abstract situations and the notion of nested information can be accommodated.

Being in a situation, one can derive information about other situations connected to it in some way. For example, from an utterance situation it is possible to obtain information about the situation it describes. Accessing information both via a hierarchy of situations and explicit relationships among them requires a computational mechanism. This mechanism will put information about situation types related in some way into the comfortable reach of the agent and can be made possible by a proper implementation of the *supports relation*, $\models$, of situation theory. Given an infon $\sigma$ and a situation $s$, this relation holds if $\sigma$ is made true by $s$, i.e., $s \models \sigma$.

Constraints enable one situation to provide information about another and serve as links. (They actually link the types of situations.) Constraints can be treated as inference rules. When viewed as a backward-chaining rule, a constraint can provide a channel for information flow between types of situations, from the antecedent to the consequent. This means that such a constraint behaves as a 'definition' for its consequent part. Another way of viewing a constraint is as a forward-chaining rule. This approach enables an agent to alter its environment.

## 3. THE PROPOSED COMPUTATIONAL MODEL

The proposed computational model for BABY-SIT is composed of seven major parts: *programmer/user interface*, *environment*, *background situation*, *anchoring situation*, *constraint set*, *inference engine*, and *interpreter* (Figure 1).

The interface allows interaction of the user with the system. One can develop and test his own program, and enter queries about situations. The environment of the proposed model initially consists of static situation structures and their relationships. These structures can be dynamically changed and new relationships among situation types can be defined as the computation proceeds. Information conveyance among situations is basically made possible by defining a *part-of* relation among them. In this way, a situation $s$ can have information about another situation $s'$ which is part of $s$.

The background situation contains infons which are inherited by all situation structures in the environment. However, a situation can inherit an infon from the background situation only if it does not cause a contradiction in that situation.

A situation in the environment can only be realized if its parameters are anchored to objects in the real world. This is made possible by the anchoring situation which allows a parameter to be anchored to an object of appropriate type—an individual, a situation, a parameter, etc. But a parameter must be anchored to a unique object, i.e.,

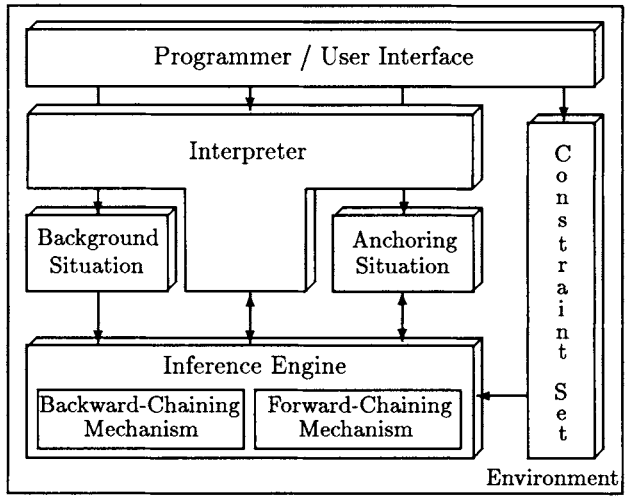


Figure 1: The architecture of the proposed model.

it is anchored once. On the other hand, more than one parameter may be anchored to the same object. Restrictions on parameters are kept in the anchoring situation as well. These restrictions assure anchoring of one parameter to an object having the same qualifications as the parameter.

In addition to the *part-of* relation among situations, constraints are potent means of information conveyance between and within situations. They link various types of situations. Constraints may be physical laws, linguistic rules, law-like correspondences, conventions, etc. In BABY-SIT they are realized as forward-chaining constraints, backward-chaining constraints, or both.

Assertion of a new object into BABY-SIT activates the forward-chaining mechanism. Consequent parts of forward-chaining constraints, if their antecedents are satisfied, asserted into BABY-SIT unless this yields a contradiction. In case of contradiction, the backward-chaining mechanism is activated to resolve it. The interpreter forms the core of execution in BABY-SIT. Anchoring of parameters, evaluation of constraints, etc. are controlled by this part of the system.

## 4. CURRENT VERSION OF BABY-SIT

A prototype for BABY-SIT is currently being developed in KEE$^{TM}$ (Knowledge Engineering Environment) [11] on a SPARCstation$^{TM}$. This interactive environment will help one to develop and test his program, observe its behavior vis-à-vis extra (or missing) information, make inferences, and issue queries [14].

The computational model underlying the current version of BABY-SIT consists of nine primitive domains: *individuals*, *times*, *places*, *relations*, *polarities*, *parameters*, *infons*, *situations*, and *types*. Each primitive domain carries its own internal structure and is

< *proposition* > ::= <*situation-proposition*> | <*parameter-type-proposition*> |
<*situation/object-type-proposition*> | <*infon-proposition*> |
<*type-of-type-proposition*> | <*relation-proposition*>

<*situation-proposition*> ::= <*constant*> "|=" <*infonic-set*>

<*parameter-type-proposition*> ::= <*parameter*> "=" {<*basic-type*>, <*type-name*>,
<*restricted-parameter-type*>}

<*situation/object-type-proposition*> ::= <*constant*> ":"
{<*basic-type*>, <*type-abstraction*>,
<*type-name*>} ["[" <*constant*> "]"]

<*infon-proposition*> ::= <*constant*> "=" <*infon*>

<*type-of-type-proposition*> ::= <*type-name*> "=" {<*basic-type*>, <*type-abstraction*>}

<*relation-proposition*> ::= "<" <*relation*> ["|" <*type-specifier*>
(“,” <*type-specifier*>)*] ">"

<*type-specifier*> ::= <*basic-type*> | <*type-name*> |
"{" {<*basic-type*>, <*type-name*>}
(“,” {<*basic-type*>, <*type-name*>})*] "}"

<*type-abstraction*> ::= "[" <*parameter*> "|" { <*constant*>, <*parameter*>}
"|=" <*infonic-set*> "]"

<*restricted-parameter-type*> ::= <*parameter*> " ^ " <*infonic-set*>

<*basic-type*> ::= "~LOC" | "~TIM" | "~IND" | "~REL" | "~SIT" |
"~INF" | "~TYP" | "~PAR" | "~POL"

<*infonic-set*> ::= "{" <*infon*> (“,” <*infon*>)* "}" | <*infon*>

<*infon*> ::= "<<" <*relation*> (“,” <*argument*>)* [“,” <*polarity*>] ">>"

<*relation*> ::= <*special-relation*> | <*constant*>

<*argument*> ::= <*constant*> | <*parameter*> | <*basic-type*> | <*type-name*>

<*polarity*> ::= "0" | "1"

<*constant*> ::= {<*digit*>, <*lower-case-letter*>} ({<*lower-case-letter*>, <*digit*>})*

<*parameter*> ::= <*upper-case-letter*> ({<*upper-case-letter*>, <*digit*>})*

<*type-name*> ::= "~" <*upper-case-letter*> ({<*upper-case-letter*>, <*digit*>})*

<*lower-case-letter*> ::= "a" | "b" | ... | "z" | "-"

<*upper-case-letter*> ::= "A" | "B" | ... | "Z"

<*digit*> ::= "0" | "1" | ... | "9"

Table 1: Syntax of the assertion mode in extended BNF.

now briefly described:

- Individuals: Unique atomic entities in the model which correspond to real objects in the world.
- Times: Individuals of distinguished type, representing temporal locations.
- Places: Similar to times, places are individuals which represent spatial locations.
- Relations: Various relations hold or fail to hold between objects. A relation has argument roles which must be occupied by appropriate objects.
- Polarities: The 'truth values' 0 and 1.
- Infons: Discrete items of information of the form $<< rel, arg_1, \ldots, arg_n, pol >>$, where $rel$ is a relation, $arg_i$, $1 \leq i \leq n$, is an object of the appropriate type for the $i$th argument role, and $pol$ is the polarity.
- Parameters: 'Place holders' for objects in the model; they are used to refer to arbitrary objects of a given type.
- Situations: (Abstract) situations are set-theoretic constructs, e.g., a set of *parametric infons* (comprising relations, parameters, and polarities). A parametric infon is the basic computational unit. By defining a hierarchy between them, situations can be embedded via the special relation *part-of*. A situation can be either (spatially and/or temporally) *located* or *unlocated*. Time and place for a situation can be declared by *time-of* and *place-of* relations, respectively.
- Types: Higher-order uniformities for individuating or discriminating uniformities in the world.

Each object in the environment must be declared to be of some type, i.e., it must be from one of the above primitive domains. In the existing version, the user can work in the *assertion mode* and can define objects and their types. There are nine basic types corresponding to nine primitive domains; ~IND (individuals), ~TIM (times), ~LOC (places), ~REL (relations), ~POL (polarities), ~INF (infons), ~PAR (parameters), ~SIT (situations), and ~TYP (types). For instance, if $l$ is a place, then $l$ is of type ~LOC, and the infon $<<$*of-type*, $l$, ~LOC, 1$>>$ is a fact in the background situation. Note that type of all types is ~TYP. For example, the infons $<<$*of-type*, ~LOC, ~TYP, 1$>>$ and $<<$*of-type*, ~TYP, ~TYP, 1$>>$ are facts in the background situation by default.

The syntax of the assertion mode is the same as in [8] (Table 1). A syntactic and semantic parser analyses each proposition provided by the user (Figure 2). Suppose *john* is an individual, *see* is a relation, and *sit1* is a situation. Then, these objects can be declared as:

> *john*: ~IND
> *see*: ~REL
> *sit1*: ~SIT

The definition of relations includes the *appropriateness conditions* for their argument roles. Appropriateness conditions define the domains to which arguments of a relation belong. Each argument can be declared to be from one or more of the primitive domains above. Consider the seeing relation above. If we like it to have two arguments, the former being of type individual and the latter being of type situation, we can write:

> <*see* | ~IND, ~SIT>

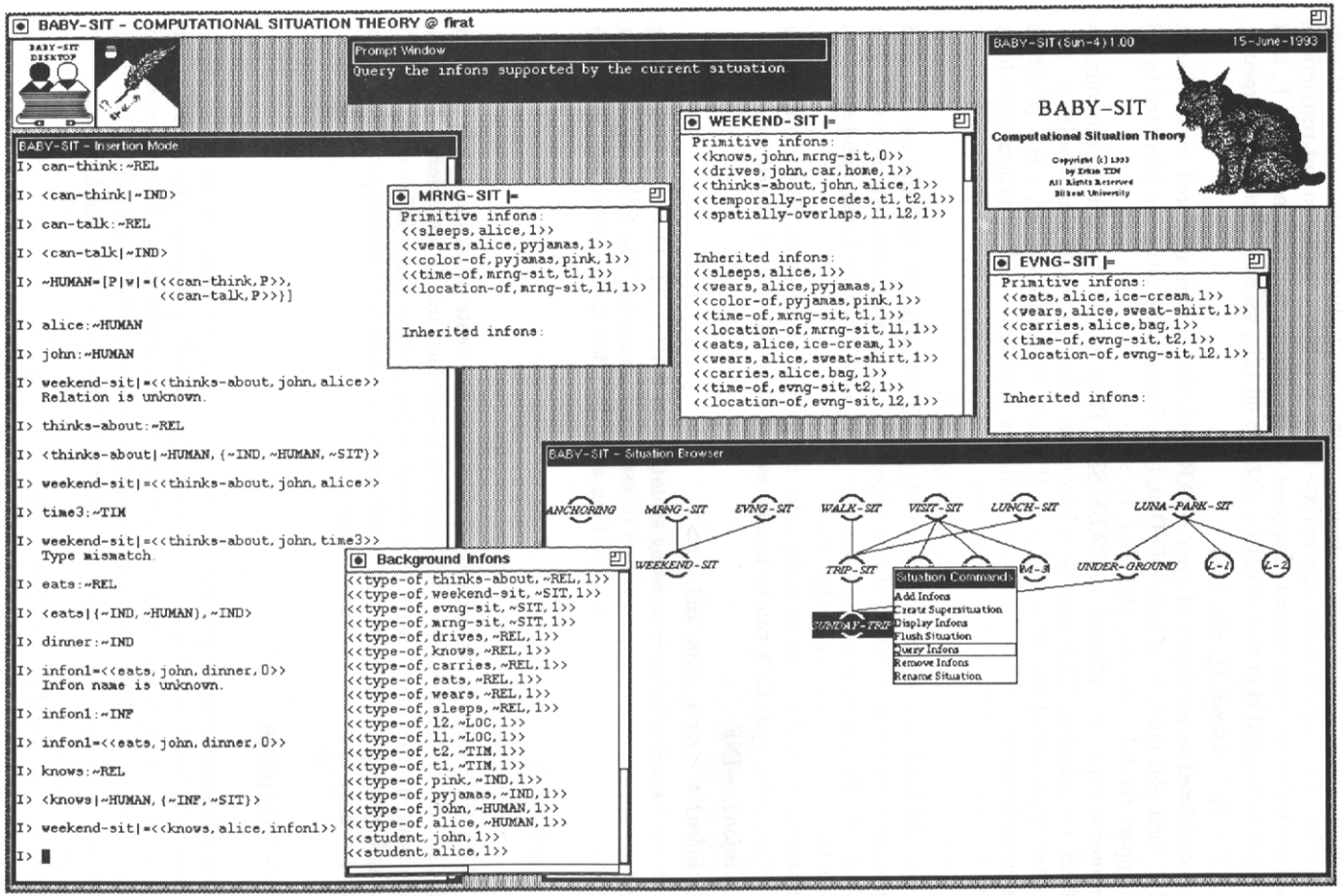


Figure 2: A view from the current version of the BABY-SIT desktop.

Parameters must be declared to be from only one of the primitive domains so that they can be anchored to objects of the appropriate type. It is also possible to put restrictions on a parameter in the environment. Suppose we want to form a parameter, say P, that denotes any individual that sees situation *sit1*. To do this, we can assert the following:

> P = Q ^ <<*see*, Q, *sit1*, 1>>

P is considered as an object of type ~PAR such that if it is anchored to an object, say *obj1*, then *obj1* must be of type ~IND and the background situation (denoted by $w$) must support the infon <<*see*, *obj1*, *sit1*, 1>>.

*Parametric* types are allowed in BABY-SIT. They can be formed by obtaining a type from a parameter. This process is known as *(object-) type-abstraction* [8, p. 60]. Parametric types are of the form $[P \mid s \models I]$ where $P$ is a parameter, $s$ is a situation (i.e., a *grounding* situation), and $I$ is a set of infons. The type of all situations that John sees can be defined in BABY-SIT as follows:

> ~SITALL = [Q | $w \models$ <<*see*, *john*, Q, 1>>]

Hence, ~SITALL is seen as an object of type ~TYP in BABY-SIT and can be used as a type specifier for declaration of new objects in the environment. An object of type ~SITALL, say *obj2*, is an object of basic type ~SIT such that the background situation supports the infon <<*see*, *john*, *obj2*, 1>>.

Naming infons enables one to easily refer to them in expressions. For instance, the infon <<*see*, *john*, *sit1*, 1>> can be named *infon1* by a sequence of assertions:

> *infon1*: ~INF

> *infon1* = <<*see*, *john*, *sit1*, 1>>

Infons can be added into the existing situations in BABY-SIT. One way of achieving this, other than via *situation browser*, is to use assertion mode. The following sequence of assertions creates a situation *sit2* and then adds the infon <<*see*, *john*, *sit1*, 0>> into it:

> *sit2*: ~SIT

> *sit2* $\models$ <<*see*, *john*, *sit1*, 0>>

In BABY-SIT, a situation browser enables one to create situations, browse them graphically, add or delete infons, and establish hierarchies among situations. The user can also issue queries about whether a situation supports a given infon or not. The syntax and operational semantics of a full query and inference mechanism have been defined, but have not been put into use yet.

## 5. CONCLUSIONS

Serious thinking about the computational aspects of the situation theory is just starting. There have been only a few proposals [6, 12, 13] which mainly offer a Prolog- or Lisp-like programming environment with varying degrees of divergence from the ontology of situation theory. In this paper, we proposed a medium (called BABY-SIT) based on bona fide situation-theoretic constructs. BABY-SIT accommodates the following basic features of situation theory:

- Objects: The world is viewed as a collection of objects. The basic objects include individuals, times, places, labels, situations, relations, and parameters.

- Situations: Situations are first-class 'citizens' which represent limited portions of the world.
- Partiality: Infons can be made true or false, or may be left unmentioned by some situation.
- Coherence: A situation cannot support an infon and its dual at the same time.
- Circularity: A situation can contain infons which have the former as arguments.
- Constraints: Information flow is made possible via coercions that link various types of objects.

Compared to existing approaches [6, 12, 13], BABY-SIT enhances the basic features of situation theory. Situations are viewed at an abstract level. This means that situations are modeled as sets of parametric infons, but they may be non-well-founded [3]. Parameters are place holders and can be anchored to unique individuals in the anchoring situation. The anchoring situation is required to cohere. A situation can be realized if its parameters are anchored, either partially or fully, by the anchoring situation. Each relation has 'appropriateness conditions' which determine the type of its arguments. The basic computation regime is unification. Situations (and hence infons they support) have spatio-temporal dimensions. A hierarchy of situations can be defined both statically and dynamically. A situation can have information about another which is part of the former. Moreover, situations can be grouped to form a whole which provides a computational context [1]. Such a whole has its own set of constraints which can be globally applied to the situations collected under it. Partial nature of situations facilitates computation with incomplete information. Constraints can be violated. This aspect is built directly into the computational mechanism.

BABY-SIT allows the use of contextual information which plays a critical role in all forms of behavior and communication. Constraints enable one situation to provide information about another and serve as links between representations and the information they represent. Computation over situations occurs via constraints and is context-sensitive. In BABY-SIT, the abstract nature of situations make it possible to form abstractions without asserting facts into them. The salient contributions of BABY-SIT are listed below:

- An interactive environment helps one to develop and test his program, observe its behavior vis-à-vis extra (or missing) information, make inferences, and issue queries.
- Objects in the environment and the attainment of information flow are compatible with the ontology of situation theory.
- The mode of computation is built upon conveyance and inheritance of information, consistency of the anchoring situation, and constraint satisfaction.
- Various types of constraints (i.e., necessary, nomic, conventional, conditional, and unconditional) [5] can be effectively utilized. Conditional constraints come with a set of *background conditions* which must be satisfied for the constraint to be applied. Each background condition is in the form of a $\models$ relation between a situation and an infon. The following classes of constraints can also be easily modeled:
  - Situation constraints: Constraints between situation types.
  - Infon constraints: Constraints between infons (of a situation).
  - Argument constraints: Constraints on argument roles (of an infon).

- Inheritance of information (from the background situation) is supported by BABY-SIT. This, together with the information conveyance among situations, enables one to use contextual information whenever necessary.

## REFERENCES


1. V. Akman and E. Tın. "What is in a Context?" in E. Arıkan, editor, *Proceedings of the 1990 Bilkent International Conference on New Trends in Communication, Control, and Signal Processing, Volume II*, Elsevier, Amsterdam, Holland, 1990, 1670–1676.
2. J. Barwise. "Noun Phrases, Generalized Quantifiers, and Anaphora," in P. Gärdenfors, editor, *Generalized Quantifiers*, Reidel, Dordrecht, Holland, 1987, 1–29.
3. J. Barwise. *The Situation in Logic*, CSLI Lecture Notes Number 17, Center for the Study of Language and Information, Stanford, CA, 1989.
4. J. Barwise and J. Etchemendy. *The Liar: An Essay on Truth and Circularity*, Oxford University Press, New York, N.Y., 1987.
5. J. Barwise and J. Perry. *Situations and Attitudes*, MIT Press, Cambridge, MA, 1983.
6. A. W. Black. "ASTL—A Language for Computational Situation Semantics," Manuscript, Department of Artificial Intelligence, University of Edinburgh, Edinburgh, U.K., 1992.
7. R. Cooper, K. Mukai, and J. Perry, editors. *Situation Theory and Its Applications, Volume 1*, CSLI Lecture Notes Number 22, Center for the Study of Language and Information, Stanford, CA, 1990.
8. K. Devlin. *Logic and Information*, Cambridge University Press, Cambridge, U.K., 1991.
9. F. Dretske. *Knowledge and the Flow of Information*, MIT Press, Cambridge, MA, 1981.
10. D. Israel and J. Perry. "What is Information?" in P. P. Hanson, editor, *Information, Language, and Cognition*, The University of British Columbia Press, Vancouver, Canada, 1990, 1–28.
11. *KEE™ (Knowledge Engineering Environment) Software Development System*, Version 3.01, IntelliCorp, Inc., Mountain View, CA, 1988.
12. H. Nakashima, H. Suzuki, P.-K. Halvorsen, and S. Peters. "Towards a Computational Interpretation of Situation Theory," in *Proceedings of the International Conference on Fifth Generation Computer Systems*, Institute for New Generation Computer Technology, Tokyo, Japan, 1988, 489–498.
13. H. Schütze. "The PROSIT Language v0.4," Manuscript, Center for the Study of Language and Information, Stanford University, Stanford, CA, 1991.
14. E. Tın and V. Akman. "Computational Situation Theory," Manuscript, Department of Computer Engineering and Information Science, Bilkent University, Ankara, Turkey, 1993.