BABY-SIT: A Computational Medium Based on Situations

Erkan Tın and Varol Akman
Computer Engineering Dept., Bilkent University, Ankara

1 Introduction

Following its inception (Barwise and Perry, 1983), *situation theory* has quickly matured (Cooper et al., 1990; Devlin, 1991) and under the familiar name of *situation semantics* has been applied to a number of linguistic issues (Barwise, 1987; Barwise, 1989; Barwise and Etchemendy, 1987; Cooper, 1986; Cooper, 1991; Cooper et al., 1990; Fenstad et al., 1987), including quantification and anaphora (Gawron and Peters, 1990). 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 mathematical foundations based on intuitions basically coming from set theory and logic (Aczel, 1988; Barwise, 1989; Cooper et al., 1990). With a remarkably original view of information (which is fully adapted by situation theory) (Dretske, 1981), a 'logic,' based not on truth but on information, is being developed (Devlin, 1991). This logic will probably be an extension of first-order logic (Barwise, 1977) rather than being an alternative to it.

While situation theory and situation semantics provide an appropriate framework for a realistic model-theoretic treatment of natural language, serious thinking on their 'computational' aspects has just started (Black, 1992; Nakashima et al., 1988). Existing proposals mainly offer a Prolog- or Lisp-like programming environment with varying degrees of divergence from the ontology of situation theory. In this paper, we introduce a computational medium (called BABY-SIT) based on situations. The primary motivation underlying BABY-SIT is to facilitate the development and testing of programs in domains ranging from linguistics to artificial intelligence in a unified framework built upon situation-theoretic constructs.

2 Situations and Natural Language Semantics

Activities pertaining to language include talking, listening, reading, and writing. These activities are *situated*; they occur in situations and they are about situations (Austin, 1961). What is common to these situated activities is that they convey information (Devlin, 1991; Dretske, 1981). When uttered at different times by different speakers, a statement can convey different information to a hearer and hence can have different meanings.[1]

Situation semantics makes simple assumptions about the way natural language works. Primary among them is the assumption that language is used to convey information about the world (the so-called *external significance* of language). Even when two sentences have the same interpretation, i.e., they describe the same situation, they can carry different information.[2]

[1]Consider the sentence "That really attracts me." Depending on the reference of the demonstrative, interpretation (and hence meaning) would change. For example, this sentence would be uttered by a boy referring to a cone of ice cream or by a cab driver referring to fast driving, meaning absolutely different things (Grice, 1968).

[2]For example, "Bob went to the theater" and "The father of Carol went to the theater" both describe the same situation in which Bob (an individual) went to the theater, assuming that Bob is Carol's father. However, while the first sentence says that this individual is Bob, the second

Classical approaches to semantics underestimate the role played by *context-dependence*; they ignore pragmatic factors such as intentions and circumstances of the individuals involved in the communicative process (Austin, 1961; Grice, 1968). But, indexicals, demonstratives, tenses, and other linguistic devices rely heavily on context for their interpretation and are fundamental to the way language conveys information (Akman and Tın, 1990). Context-dependence is an essential hypothesis of situation semantics. A given sentence can be used over and over again in different situations to say different things (the so-called *efficiency* of language). Its interpretation, i.e., the class of situations described by the sentence, is therefore subordinate on the situation in which the sentence is used. This context-providing situation, *discourse situation*, is the speech situation, including the speaker, the addressee, the time and place of the utterance, and the expression uttered. Since speakers are usually in different situations, having different causal connections to the world and different information, the information conveyed by an utterance will be relative to its speaker and hearer (the so-called *perspectival relativity* of language).

Besides discourse situations, the interpretation of an utterance depends on the speaker's connections with objects, properties, times and places, and on the speaker's ability to exploit information about one situation to obtain information about another. Therefore, context supports not only facts about speakers, addressees, etc. but also facts about the relations of discourse-participants to *resource situations*. Resource situations are contextually available and provide entities for reference and quantification (Barwise and Perry, 1983).[3]

Another key assumption of situation semantics is the so-called *productivity* of language: we can use and understand expressions never before uttered. Hence, given a finite vocabulary, we can form a potentially infinite list of meaningful expressions. The underlying mechanism for such an ability seems to be *compositionality*.

Situation semantics closes another gap of traditional semantic approaches: the neglect of *subject matter* and *partiality of information*. In traditional semantics, statements which are true in the same models convey the same information (van Benthem, 1986). Situation semantics takes the view that logically equivalent sentences need not have the same subject matter: they need not describe situations involving the same object and properties. The notion of partial situations (partial models) leads to a more fine-grained notion of information content and a stronger notion of logical consequence that does not lose track of the subject matter.

The *ambiguity* of language is taken as another aspect of the efficiency of language. Natural language expressions may have more than one meaning. There are factors such as intonation, gesture, the place of an utterance, etc. which play a role in interpreting an utterance (Fenstad et al., 1987). Instead of throwing away ambiguity and contextual elements, situation semantics tries to build up a full theory of linguistic meaning by initially isolating some of the relevant phenomena in a formal way and by exploring how the rest helps in achieving the goal (Barwise and Perry, 1983).

---

sentence conveys the information that Carol (another individual) has a father who went to the theater.

[3] Consider the following example adapted from (Barwise and Etchemendy, 1987). There are two card games going on, one across town from the other: Max is playing cards with Emily and Claire is playing cards with Dana. Suppose Bob watching the former game mistakes Emily for Claire, and utters the sentence "Claire has the three of clubs." According to the classical (Russelian) theories (Evans, 1991), if Claire indeed has 3♣, this claim would be true since the definite noun phrases "Claire" and "the three of clubs" are used to pick out, among all the things in the world, the unique objects satisfying the properties of being an individual named Claire and being a 3♣, respectively. In contrast, situation semantics identifies these objects with respect to some limited situation—the resource situation exploited by Bob. The claim would then be wrong even if Claire had 3♣ across town.

According to situation semantics, meaningful expressions convey information not only about the external world but also about our minds (the so-called *mental significance* of language). Situation semantics differs from other approaches in that we do not, in attitude reports, describe our mind directly (by referring to states of mind, ideas, senses, thoughts, etc.) but indirectly (by referring to situations that are external).

With these underlying assumptions and features, situation semantics provides a fundamental and appropriate framework for studying the semantics of natural language (Barwise and Etchemendy, 1989). The ideas emerging from research in situation semantics have also been coalesced with well-developed linguistic theories such as *lexical-functional grammar* (Sells, 1985) and led to rigorous formalisms (Fenstad et al., 1987). On the other hand, situation semantics has been compared to other influential mathematical approaches to the theory of meaning, viz., *Montague grammar* (Cooper, 1986; Dowty et al., 1981; Rooth, 1986) and *Discourse Representation Theory* (DRT) (Kamp, 1981).

3 Constructs for Situated Processing

Intelligent agents generally make their way in the world as follows: pick up certain information from a situation, process it, and react accordingly (Devlin, 1991; Dretske, 1981; Israel and Perry, 1990). Being in a (mental) situation, such an agent has information about situations it sees, believes in, hears about, etc. Awareness of some type of situation causes the agent to acquire more information about that situation as well as other situation types, and to act accordingly. Assuming the possession of prior information and knowledge of some constraints, the acquisition of an item of information by an agent can also provide the agent with an additional item of information.[4]

In situation theory, *infons* are the basic units of information. 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 the parametric object abstracts over. The parametric objects actually define types of objects in that class. For example, $\langle$*see*, X, *Alice*; 1$\rangle$ and $\langle$*see*, X, Y; 1$\rangle$ are parametric infons where X and Y are parameters over individuals. *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 capture only some of the features of real situations, but are amenable to computation. We define abstract situations as structures consisting of a set 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. For example, an utterance situation for a sentence consists of the utterance situations

[4]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.

for each word forming the sentence. The *part-of* relation of situation theory can be used to build hierarchies among 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.[5]

Constraints enable one situation to provide information about another and serve as links. (They actually link 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.

4 Computational Situation Theory

4.1 PROSIT

PROSIT (PROgramming in SItuation Theory) is a situation-theoretic programming language (Nakashima et al., 1988). PROSIT is tailored more for general knowledge representation than for natural language processing. One can define situation structures and assert knowledge in particular situations. It is also possible to define relations between situations in the form of constraints. PROSIT's computational power is due to an ability to draw inferences via rules of inference which are actually constraints of some type. PROSIT can deal with self-referential expressions (Barwise and Etchemendy, 1987).

One can assert facts that a situation will support. For example, if the situation S1 supports the fact that Bob is a young person, this can be defined in the current situation S as:

S: ($\models$ S1 (young "Bob")).

Note that the syntax is similar to that of Lisp and the fact is in the form of a predicate. The supports relation, $\models$, is situated so that whether a situation supports a fact depends on within which situation the query is made. Queries can be posed about one situation from another, but the results will depend on where the query is made.

Constraints can be specified as forward chaining constraints, backward chaining constraints, or both. Backward chaining constraints are activated at query-time while forward-chaining constraints are activated at assertion-time. By default, all the tail facts of an activated forward-chaining constraint are asserted to the situation, which may in turn activate other forward-chaining constraints recursively.

For a constraint to be applicable to a situation, the situation must be declared to 'respect' the constraint. Constraints in PROSIT are about local facts within a situation rather than about situation types. That is, the interpretation of constraints does not allow direct specification of constraints between situations, only between infons within situations.

[5] Given an infon σ and a situation s, this relation holds if σ is made true by s, i.e., $s \models \sigma$.

Situated constraints offer an elegant solution to the treatment of *conditional constraints* which only apply in situations that obey some condition.[6] This is actually achieved in PROSIT since information is specified in the constraint itself. Situating a constraint means that it may only apply to appropriate situations and is a good strategy to achieve *background conditions*. However, it might be required that conditions are set not only within the same situation, but also between various types of situations. Because constraints have to be situated in PROSIT, not all situations of the appropriate type will have a constraint to apply. PROSIT does not provide an adequate mechanism for specifying *conventional constraints*, i.e., constraints which can be violated.[7]

Parameters, variables, and constants are used for representing entities in PROSIT. Variables match any expression in the language and parameters can be equated to any constant or parameter. That is, the concept of *appropriateness conditions* is not exploited in PROSIT. Appropriateness conditions, in fact, specify restrictions on the types of arguments a relation can take, and any restrictions between these arguments (Devlin, 1991). It is more useful to have parameters that range over various classes rather than to work with parameters ranging over all objects. Some treatment of parameters is given in PROSIT with respect to *anchoring*. Given a parameter of some type (individual, situation, etc.), an anchor is a function which assigns an object of the same type to the parameter (Devlin, 1991). Hence, parameters work by placing restrictions on anchors. There is no appropriate anchoring mechanism in PROSIT since parameters are not typed.

Set operations are possible on sets of facts supported by a situation. As mentioned before, situations are closed under constraints and rules of inference. PROSIT has been used to show how problems involving cooperation of multiple agents can be solved, especially by combining reasoning about situations (Nakashima et al., 1987).

4.2 ASTL

Black's ASTL (A Situation Theoretic Language) is another programming language based on situation theory (Black, 1992). ASTL is aimed at natural language processing. The primary motivation underlying ASTL is to figure out a framework in which semantic theories such as situation semantics and DRT (Kamp, 1981) can be described and possibly compared. One can define in ASTL constraints and rules of inference over the situations. An interpreter passes over ASTL definitions to answer queries about the set of constraints and basic situations.

ASTL ontology incorporates individuals, relations, situations, parameters, and variables. These form the basic terms of the language. Complex terms are in the form of i-terms (to be defined shortly), situation types, and situations. Situations can contain facts which have those situations as arguments. Sentences in ASTL are constructed from terms in the language and can be constraints, grammar rules, or word entries.

An *i-term* is simply an infon[8] $\langle rel,\ arg_1,\ \ldots,\ arg_n, pol\rangle$ where rel is a relation of arity n, arg_i is a term, and pol is either 0 or 1. A *situation type* is given in the form $[param|cond_1 \ldots cond_n]$ where $cond_i$ has the form $param \models$ *i-term*. If S1

[6] For example, when Alice throws a basketball, she knows it will come down—a constraint to which she is attuned, but which would not hold if she tried to play basketball on the moon.

[7] An example of this sort of constraint is the relation between the ringing of the bell and the end of class. It is not necessary that the ringing of the bell should mean the end of class.

[8] We use Black's notation almost verbatim rather than adapting it to the 'standard' notation of our paper.

supports the fact that Bob is a young person, this can be defined as:

S1: [S | S $\models$ ⟨young, bob, 1⟩].

The single colon indicates that S1 supports the situation type on its right-hand side. The supports relation in ASTL is global rather than situated. Consequently, query-answering is independent of the situation in which the query is issued.

Constraints are actually backward-chaining constraints. For example, the constraint that every man is a human being can be written as follows:

*S: [S | S $\models$ ⟨human, *X, 1⟩] $\Leftarrow$ *S: [S | S $\models$ ⟨man, *X, 1⟩].

*S, *X are variables and S is a parameter. Another interesting property of ASTL is that constraints are global. Thus, a new situation of the appropriate type need not have a constraint explicitly added to it. Grammar rules are yet another sort of constraints with similar semantics.

Although one can define constraints between situations in ASTL, the notion of a background condition for constraints is not available. Similar to PROSIT, ASTL cares little about coherence within situations. This is left to the user's control. Accordingly, there is no mechanism in ASTL to specify constraints that can be violated.

Declaring situations to be of some type allows abstraction over situations to some degree. But, the actual means of abstraction over objects in situation theory, viz., parameters, carry little significance in ASTL.

As in PROSIT, variables in ASTL have scope only within the constraint they appear. They match any expression in the language unless they are declared to be of some specific situation type in the constraint. Hence, it is not possible to declare variables as well as parameters to be of other types such as individuals, relations, etc. Moreover, ASTL does not allow definition of appropriateness conditions for arguments of relations.[9]

ASTL does not have a mechanism to relate two situations so that one will support all the facts that the other does. This might be achieved via constraints, but there is no built-in structure between situations (as opposed to the hierarchy of situations in PROSIT).

4.3 Situation Schemata

Situation schemata have been introduced (Fenstad et al., 1987) as a theoretical tool for extracting and displaying information relevant for semantic interpretation from linguistic form. A situation schema is an attribute-value system which has a choice of primary attributes matching the primitives of situation semantics. In this sense, it is just another knowledge representation mechanism. The boundaries of situation schemata are however flexible and, depending on the underlying theory of grammar, are susceptible to amendment.

A simple sentence φ has the situation schemata shown in Figure 1(a). Here r can be anchored to a relation, and a and b to objects; $i \in \{0,1\}$ gives the polarity. LOC is a function which anchors the described fact relative to a discourse situation d, c. LOC will have the general format in Figure 1(b). IND.α is an indeterminate for a location, r denotes one of the basic structural relations on a relation set R, and loc_0 is another location indeterminate. The notation $[\ ]_\alpha$ indicates a repeated

[9] 'Speaking' relation, for example, might require its speaker role to be filled by a human. Such a restriction could be defined only by using constraints of ASTL. However, this requires writing the restriction each time a new constraint about 'speaking' is to be added.

$$
\mathrm{SIT}.\varphi \begin{bmatrix} \mathrm{REL} & r \\ \mathrm{ARG.1} & a \\ \mathrm{ARG.2} & b \\ \mathrm{LOC} & - \\ \mathrm{POL} & i \end{bmatrix}
\qquad
\begin{bmatrix} \mathrm{IND} & \mathrm{IND}.\alpha \\ \mathrm{COND} & \begin{bmatrix} \mathrm{REL} & r \\ \mathrm{ARG.1} & [\,]_{\alpha} \\ \mathrm{ARG.2} & loc_d \\ \mathrm{POL} & i \end{bmatrix} \end{bmatrix}
$$

(a) (b)

Figure 1: (a) A prototype situation schema, (b) the general format of LOC in (a).

reference to the shared attribute value, IND.α. A partial function g anchors the location of SIT.φ, viz. SIT.φ.LOC, in the discourse situation d, c if

$g(loc_0) = loc_d$ and
$c(r)$, $g(\mathrm{IND}.\alpha)$, loc_d; 1

where loc_d is the discourse location and $c(r)$ is the relation on R given by the speaker's connection c. The situation schema corresponding to "Alice saw the cat" is given in Figure 2.

Situation schemata can be adopted to various kinds of semantic interpretation.[10] One could give some kind of operational interpretation in a suitable programming language, exploiting logical insights. But in its present state, situation schemata do not go further than being a complex attribute-value structure. They allow representation of situations within this structure, but do not use situation theory itself as a basis. Situations, locations, individuals, and relations constitute the basic domains of the structure. Constraints are declarative descriptions of the relationships holding between aspects of linguistic form and the semantic representation itself.

5 BABY-SIT

5.1 Computational Model

The computational model underlying the current version of BABY-SIT consists of nine primitive domains: *individuals* (I), *times* (T), *places* (L), *relations* (R), *polarities* (O), *parameters* (P), *infons* (F), *situations* (S), and *types* (K). Each primitive domain carries its own internal structure:

- 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.

[10] Theoretical issues in natural language semantics have been implemented on pilot systems employing situation schemata. The grammar described in (Fenstad et al., 1987), for example, has been fully implemented using a lexical-functional grammar system (Fenstad, 1987) and a fragment including prepositional phrases has been implemented using the DPATR format (Colban, 1987).

$$
\text{SIT.1}\begin{bmatrix}
\text{REL} & \text{`see'} \\
\text{ARG.1} & \begin{bmatrix} \text{IND} & \text{`Alice'} \end{bmatrix} \\
\text{ARG.2} & \begin{bmatrix} \text{IND} & \text{IND.2} \\ \text{SPEC} & \text{`the'} \\ \text{COND} & \begin{bmatrix} \text{REL} & \text{`cat'} \\ \text{ARG.1} & [\,]_2 \\ \text{POL} & 1 \end{bmatrix} \end{bmatrix} \\
\text{LOC} & \begin{bmatrix} \text{IND} & \text{IND.3} \\ \text{COND} & \begin{bmatrix} \text{REL} & \text{`}<\text{'} \\ \text{ARG.1} & [\,]_3 \\ \text{ARG.2} & loc_d \\ \text{POL} & 1 \end{bmatrix} \end{bmatrix} \\
\text{POL} & 1
\end{bmatrix}
$$

Figure 2: Situation schemata for “Alice saw the cat.”

- 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 ith 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.

The structure of the model, M, is a tuple $<I, T, L, R, O, P, F, S, K>$. This structure is shared by all components of the system. *Description* of a model, D_M, consists of a definition of the structure M and a set of *constraints*, C. The computational model is then defined as a tuple $<D_M, A, A', U>$ where A is an anchor for parameters, A' is an assignment for variables, and U is an interpretation for D_M. A is provided by the anchoring situation while A' is obtained through unification.

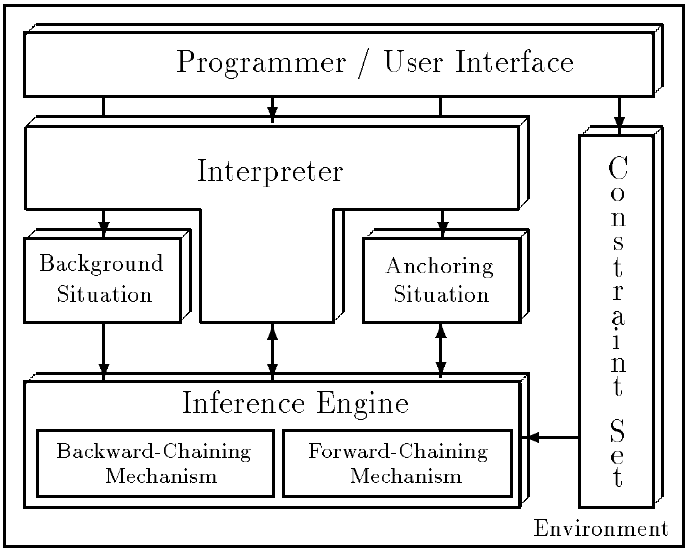

Figure 3: The architecture of BABY-SIT.

U will be defined by the operational semantics of the computation. Each object in the environment must be declared to be of some type.

5.2 Architectural Considerations

The architecture of BABY-SIT is composed of seven major parts: *programmer/user interface*, *environment*, *background situation*, *anchoring situation*, *constraint set*, *inference engine*, and *interpreter* (Figure 3).

The interface allows interaction of the user with the system. The environment 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 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 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. A parameter must be anchored to a unique object. On the other hand, more than one parameter may be anchored to the same object. Restrictions on parameters 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 or backward-chaining constraints, or both. Assertion of a new object into BABY-SIT activates the forward-chaining mechanism. Once their antecedent parts are satisfied, consequent parts of the forward-chaining constraints are asserted into BABY-SIT, unless this yields a contradiction. In case of a contradiction, the backward-chaining mechanism is activated to resolve it. The interpreter is the control authority in BABY-SIT. Anchoring of parameters, evaluation of constraints, etc. are all controlled by this part of the system.

5.3 Modes of Computation

A prototype of BABY-SIT is currently being developed in KEE^{TM} (Knowledge Engineering Environment) (KEE, 1993) on a SPARCstation^{TM}. Some of the available modes of computation in this evolving project are described below.

5.3.1 Constraints

Barwise and Perry identify three forms of constraints (Barwise and Perry, 1983). *Necessary constraints* are those by which one can define or name things, e.g., "Every dog is a mammal." *Nomic constraints* are patterns that are usually called natural laws, e.g., "Blocks drop if not supported." *Conventional constraints* are those arising out of explicit or implicit conventions that hold within a community of living beings, e.g., "The first day of the month is the pay day." They are neither nomic nor necessary, i.e., they can be violated. All types of constraints can be *conditional* and *unconditional*. Conditional constraints can be applied to situations that meet some condition while unconditional constraints can be applied to all situations.

Some constraints can be defined as forward-chaining constraints, some as backward-chaining constraints, others as both forward- and backward-chaining constraints. In BABY-SIT, 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.

In BABY-SIT, a constraint becomes a candidate for activation when its background conditions, if any, are satisfied. A candidate forward-chaining constraint is activated whenever its antecedent part is satisfied. All the consequences are asserted if they do not yield a contradiction in the situation into which they are asserted. New assertions may in turn activate other candidate forward-chaining constraints. If consequences cause contradictions within themselves, backward-chaining constraints are used to decide which one(s) will be successfully asserted. Candidate backward-chaining constraints are activated either when a query is entered explicitly or is issued by the forward-chaining mechanism.

In BABY-SIT, the following classes of constraints can be easily modeled (Black, 1991):

- Situation constraints: Constraints between situation types.
- Infon constraints: Constraints between infons (of a situation).
- Argument constraints: Constraints on argument roles (of an infon).

5.3.2 Assertions

Assertion mode provides an interactive environment in which one 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 (Devlin, 1991) (cf. Table 1).

Suppose *aynur* is an individual, *syntactic-entity* is a relation, and *u1* is an utterance situation for the word 'aynur.' Then, these objects can be declared as:

I> *aynur*: ~IND
I> *syntactic-entity*: ~REL
I> *u1*: ~SIT

The definition of relations includes the *appropriateness conditions* for their argument roles. Each argument can be declared to be from one or more of the primitive domains above. Consider *syntactic-entity* above. If we like it to have only one argument of type individual, we can write:

I> <*syntactic-entity* | ~IND>

In order for the parameters to be anchored to objects of the appropriate type, parameters must be declared to be from only one of the primitive domains. It is also possible to put restrictions on a parameter in the environment. Suppose we want to have a parameter E that denotes any syntactic entity. This can be done by asserting:

I> E = IND1 ^ <<*syntactic-entity*, IND1, 1>>

IND1 is a default system parameter of type ~IND. E 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 <<*syntactic-entity*, *obj1*, 1>>.

Parametric types are also allowed in BABY-SIT. They can be formed by obtaining a type from a parameter. 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 syntactic entities can be defined as follows:

I> ~CATEGORY = [IND1 | *w* ⊨ <<*syntactic-entity*, IND1, 1>>]

~CATEGORY is seen as an object of type ~TYP and can be used as a type specifier for declaration of new objects in the environment. For instance:

I> noun: ~CATEGORY

yields an object, *noun*, which is of type ~IND such that the background situation supports the infon <<*syntactic-entity*, *noun*, 1>>.

Infons can be added into situations in BABY-SIT. The following sequence of assertions adds <<*category*, *u1*, *noun*, 1>> into *u1*:

I> *category*: ~REL
I> <*category* | ~SIT, ~CATEGORY>
I> *u1* ⊨ <<*category*, *u1*, *noun*, 1>>

5.3.3 Querying

Query mode enables one to issue queries about situations. BABY-SIT's response depends on its understanding of the intention of the user. There are several possible actions which can be further controlled by the user:

```
< 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> “ ˆ ” <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.

< *query* > ::= <*situation-query*> | <*oracle-query*>
<*situation-query*> ::= <*situation*> {"|=", "|=/="} <*query-infonic-set*>
("," <*situation*> {"|=", "|/="} <*query-infonic-set*>)*
["[" <*anchoring-situation*> "]"]

<*oracle-query*> ::= <*constant*> "=" "@" "(" <*constant*> ")" [<*issue-set*>]

<*situation*> ::= <*constant*> | <*query-variable*>

<*issue-set*> ::= "{" <*issue-infon*> ("," <*issue-infon*>)* "}"

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

<*query-infon*> ::= "<<" {<*relation*>, <*query-variable*>}
("," {<*argument*>, <*query-variable*>})* ","
{<*polarity*> ">>"

<*issue-infon*> ::= "<<" <*relation*> ("," <*argument*>)* ["," <*polarity*>] ">>"

<*query-variable*> ::= "?" <*parameter*>

<*anchoring-situation*> ::= <*constant*>

Table 2: Syntax of the query mode.

- Replacing each parameter in the query expression by the corresponding individual if there is a possible anchor, either partial or full, provided by the given anchoring situation for that parameter.
- Returning solutions. (Their number is determined by the user.)
- Returning all solutions.
- Displaying a solution with its parameters replaced by the individuals to which they are anchored by the given anchoring situation.
- For each solution, displaying infons anchoring any parameter in the solution to an individual in the given anchoring situation.
- Displaying a trace of anchoring of parameters in each solution.

The computation upon issuing a query is done either by direct querying through situations or by the application of backward-chaining constraints. This is also under the control of the user. A situation, s, *supports* an infon if the infon is either explicitly asserted to hold in s, or it is supported by a situation s' which is part of s, or it can be proven to hold by application of backward-chaining constraints.[11] The syntax of the query expressions is given in Table 2. Given *anchor1* as the anchoring situation (Figure 4), a query and the system's response to it are as follows:

Q> u1 ⊨ {<<?X, ?Y, *nominative*, 1>>, <<*time-of*, u1, ?Z, 1>>},
u2 ⊭ {<<*category*, u2, *pronoun*, 1>>

u1 ⊨ {<<*case*, u1, *nominative*, 1>>, <<*time-of*, u1, T1, 1>>},
u2 ⊭ {<<*category*, u2, *pronoun*, 1>>

[11] Remember that given an infon σ and a situation s, if s supports σ, then this is denoted by $s \models \sigma$. Otherwise, we say $s \not\models \sigma$.

with the anchoring on parameters:

anchor1 $\models$ {<<*anchor*, T1, t1, 1>>

In addition to query operations, a special operation, *oracle*, is allowed in the query mode. An *oracle* is defined over an object and a set of infons (*set of issues*) (Devlin, 1991). The oracle of an object enables one to chronologically view the information about that object from a particular perspective provided by the given set of infons. One may consider oracles as 'histories' of specific objects. Given an object and a set of issues, BABY-SIT anchors all parameters in this set of issues and collects all infons supported by the situations in the system under a specific situation, thus forming a 'minimal' situation which supports all parameter-free infons in the set of issues.

6 Conclusion

BABY-SIT accommodates the basic features of situation theory. The world is viewed as a collection of objects. The basic objects include individuals, times, places, labels, situations, relations, and parameters. Situations are 'first-class citizens' which represent limited portions of the world. Infons can be made true or false, or may be left unmentioned by some situation. A situation cannot support an infon and its dual at the same time. A situation can contain infons which have the former as arguments. Information flow is made possible via coercions that link various types of objects.

BABY-SIT enhances these features. Situations are viewed at an abstract level. This means that situations are sets of parametric infons, but they may be non-well-founded (Aczel, 1988; Barwise and Etchemendy, 1987). Parameters are place holders, hence they can be anchored to unique individuals in the anchoring situation. 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. Situations (and hence infons they support) may have spatio-temporal dimensions. A hierarchy of situations can be defined both statically and dynamically. A built-in structure allows one situation to have information about another which is part of the former. Grouping of situations provides a computational context. Partial nature of situations facilitates computation with incomplete information. Constraints can be violated. This aspect is built directly into the computational mechanism: a constraint can be applied to a situation only if it does not cause an incoherence.

References

Aczel, P.: 1988, *Non-Well-Founded Sets*, CSLI Lecture Notes Number 14, Center for the Study of Language and Information, Stanford, California

Akman, V. and Tın, E.: 1990, What is in a context? in E. Arıkan (ed.), *Proceedings of the 1990 Bilkent International Conference on New Trends in Communication, Control, and Signal Processing, Volume II*, Elsevier, Amsterdam, 1670–1676

Austin, J. L.: 1961, Truth, in J. O. Urmson and G. J. Warnock (eds.), *Philosophical Papers of J. L. Austin*, Oxford University Press, Oxford, 117–133

Barwise, J.: 1977, An introduction to first-order logic, in J. Barwise (ed.), *Handbook of Mathematical Logic*, North-Holland, Amsterdam, 5–46

Barwise, J.: 1987, Noun phrases, generalized quantifiers, and anaphora, in P. Gärdenfors (ed.), *Generalized Quantifiers*, Reidel, Dordrecht, 1–29

Barwise, J.: 1989, *The Situation in Logic*, CSLI Lecture Notes Number 17, Center for the Study of Language and Information, Stanford, California

Barwise, J. and Etchemendy, J.: 1987, *The Liar: An Essay on Truth and Circularity*, Oxford University Press, New York

Barwise, J. and Etchemendy, J.: 1989, Model-theoretic semantics, in M. I. Posner (ed.), *Foundations of Cognitive Science*, MIT Press, Cambridge, Massachusetts, 207–243

Barwise, J. and Perry J.: 1983, *Situations and Attitudes*, MIT Press, Cambridge, Massachusetts

Benthem, J. v.: 1986, *Essays in Logical Semantics*, Reidel, Dordrecht

Black, A. W.: 1991, Constraints in computational situation semantics, Lecture Notes circulated during the *Logic, Language, and Information Summer School*, Saarbrücken, Germany

Black, A. W.: 1992, ASTL—A language for computational situation semantics, Ph.D. thesis, Department of Artificial Intelligence, University of Edinburgh, Edinburgh

Colban, E.: 1987, Prepositional phrases in situation schemata, Appendix A in Fenstad et al.

Cooper, R.: 1986, Tense and discourse location in situation semantics, *Linguistics and Philosophy* 9, 17–36

Cooper, R.: 1987, Meaning representation in Montague grammar and situation semantics, in B. G. T. Lowden (ed.), *Proceedings of the Alvey Sponsored Workshop on Formal Semantics in Natural Language Processing*

Cooper, R.: 1991, Three lectures on situation theoretic grammar, in M. Filgueiras et al. (eds.), *Natural Language Processing*, Lecture Notes in Artificial Intelligence, Vol. 476, Springer-Verlag, Berlin, 102–140

Cooper, R., Mukai, K., and Perry, J. (eds.): 1990, *Situation Theory and Its Applications, Volume 1*, CSLI Lecture Notes Number 22, Center for the Study of Language and Information, Stanford, California

Devlin, K.: 1991, *Logic and Information*, Cambridge University Press, Cambridge

Dowty, D., Wall, R., and Peters, S.: 1981, *Introduction to Montague Semantics*, Reidel, Dordrecht

Dretske, F.: 1981, *Knowledge and the Flow of Information*, MIT Press, Cambridge, Massachusetts

Evans, G.: 1991, *The Varieties of Reference*, Oxford University Press, New York

Fenstad, J. E., Halvorsen, P.-K., Langholm T., and Benthem, J. v.: 1987, *Situations, Language, and Logic*, Reidel, Dordrecht

Fenstad, J. E.: 1987, Natural language systems, in R. T. Nossum (ed.), *Advanced Topics in Artificial Intelligence: 2nd Advanced Course*, Lecture Notes in Artificial Intelligence, Vol. 345, Springer-Verlag, Berlin, 189–233

Gawron, J. M. and Peters, S.: 1990, *Anaphora and Quantification in Situation Semantics*, CSLI Lecture Notes Number 19, Center for the Study of Language and Information, Stanford, California

Grice, H. P.: 1968, Utterer's meaning, sentence-meaning, and word-meaning, *Foundations of Language* 4, 1–18

Israel D. and Perry, J.: 1990, What is information? in P. P. Hanson (ed.), *Information, Language, and Cognition*, The University of British Columbia Press, Vancouver, 1–28

Kamp, H.: 1981, A theory of truth and semantic representation, in J. Groenendijk, T. Janssen, and M. Stokhof (eds.), *Formal Methods in the Study of Language*, Mathematical Center Tract 135, Amsterdam, 277–322

KEE^{TM}: 1993, *(Knowledge Engineering Environment) Software Development System*, Version 4.1, IntelliCorp, Inc., Mountain View, California

Nakashima, H., Peters, S., and Schütze, H.: 1987, Communication and inference through situations, in *Proceedings of the Third Conference on Artificial Intelligence Applications*, IEEE Computer Society Press, Washington, D.C., 76–81

Nakashima, H., Suzuki, H., Halvorsen, P.-K., and Peters, S.: 1988, 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, 489–498

Rooth, M.: 1986, *Noun Phrase Interpretation in Montague Grammar, File Change Semantics, and Situation Semantics*, Report No. CSLI–86–51, Center for the Study of Language and Information, Stanford, California

Sells, P.: 1985, *Lectures on Contemporary Syntactic Theories*, CSLI Lecture Notes Number 3, Center for the Study of Language and Information, Stanford, California