Using Situation Theory in a computational language for natural language processing

Alan W Black[1]
ATR Interpreting Telecommunications Laboratories
2-2 Hikaridai, Seika-cho, Soraku-gun, Kyoto 619-02, JAPAN
awb@itl.atr.co.jp

Abstract

This paper discusses a computational language based on situation theory called ASTL. ASTL is designed as a formal specification language for contemporary semantic theories of natural language. Descriptions of key aspects of semantic theories may be specified in ASTL and then directly executed to derive semantic representations for natural language utterances.

We will outline the basic syntax and semantics of ASTL and show how situation theory offers a rich base for a logic programming language appropriate for natural language processing. Examples will be given showing how situation theory and in particular *situations* offer the sort of structured objects required in semantic knowledge representation.

The basic properties of ASTL are identified as the basic properties required in a computational language for description of semantic theories. Although these properties may be found in other frameworks these properties are also fundamental to situation theory and hence it seems reasonable to present them in that form.

1 Background

ASTL was primarily developed as a computational environment in which contemporary theories of natural language semantics may be described and compared. Recent work in computational semantics has led to a number of theories for semantic representation (and construction of these representations). Although these address similar semantic phenomena they often use apparently very different forms for their representations. This can best be shown by looking at the representations for a simple sentence in different theories. The following is the representation for the utterance *"A man walks"* in a form of situation semantics (as in [Gawron & Peters 90]), dynamic logic ([Groenendijk & Stokhof 91]) and discourse representation theory ([Kamp 81]):

Situation Semantics	$S \models \ll man, X; 1 \gg$
	$S \models \ll walk, X; 1 \gg$
Dynamic Logic	$\exists x\ [man(x)] \land walk(x)$
DRT	`X` / `man(X)` `walk(X)`

[1] Most of this work was carried out while the author was at the Dept. of Artificial Intelligence and the Centre for Cognitive Science, University of Edinburgh. This work was funded by an SERC postgraduate studentship (number 89313458) and by Esprit Basic Research Action Project 6852 (DYANA-2). I am also grateful for the comments from two anonymous referees.

Note that only the dynamic logic representation has an explicit existential quantifier but even it differs from standard first order logic. The semantics of dynamic logic is such that the x in $walk(x)$ actually does fall within the scope of the preceding existential quantifier.

In order to give a detailed comparison between these semantic theories it would be useful to have an environment in which these theories (or at least key aspects of them) may be formally specified and tested. Also as we are interested in semantics, that environment should have a firm formal (and theoretical) basis.

Situation Semantics ([Barwise & Perry 83]) offered an alternative to possible world semantics for natural language semantics. Later work has introduced the distinction between *situation theory*: the mathematical, logical and philosophical aspects of that theory; and *situation semantics*: theories of natural language semantics described within situation theory. As there is already this distinction between mathematical theory and natural language theory described in it we would like to extend that idea but describe theories other than situation semantic ones within a situation theoretic framework.

Formal theoretical specification is not enough though. As we are interested in computational semantics it is necessary that the framework we choose can also be used for computation.

Given the above criteria, the language ASTL was developed.

2 ASTL – language definition

ASTL is still quite conservative and there are still more features which could be added (both from situation theory and logic programming paradigms in general). ASTL allows the following simple (atomic) terms:

individuals, parameters, relations, variables.

From these the following complex terms may be built:

i-terms: consisting of a relation, arguments (terms) and a polarity (1 or 0). e.g.

```
<<sleeps,hanako,1>>
```

situations: a situation name (optionally) followed by a situation type. For example

```
SIT1::[S ! S != <<sleeps,hanako,1>>
            S != <<dreams,hanako,1>>]
```

That is the situation named by SIT1 is of the type that supports the facts that Hanako dreams and sleeps in that situation.

situation types: an abstraction of the facts that a situation supports as shown in the description of situations.

Situations offer a way to view parts of the world. A situation may support facts such that something may be true in one situation but not in another. Importantly, situations are first class objects in the language. They are terms and may be used as arguments in i-terms. Self-reference is also not a problem. The following situation is valid where the situation itself is referenced in its type.

```
SIT2 :: [S ! S != <<has,hanako,3-diamonds,1>>
             S != <<sees,hanako,SIT2,1>>].
```

That is SIT2 is a situation in which Hanako has the (playing card) three of diamonds, it also supports the fact that Hanako can see that situation. Thus she can see that she has the three of diamonds and can see that she can see that she has the three of diamonds etc.

The above static objects may be enough to give a rich knowledge representation language but are not really sufficient for any form of computation. We also have the following two sentences in ASTL.

propositions: these consist of a situation name followed by a situation type. If that situation is of that type then the proposition is true, otherwise it is false.

constraints: these are defined between situations. The simplest way to illustrate their syntax (and semantics) is to give an example.

```
*S : [S ! S != <<happy,*Y,1>>]
<=
   *S : [S ! S != <<smiles,*Y,1>>].
```

By convention variables are written prefixed by an asterisk. The above constraint states that in all situations which are of the type that supports a fact where something (`*Y`) smiles, that same thing is also happy. Because we use the same variable at either side of the constraint (`*S`) we are stating that the thing that is smiling is happy in the *same* situation but we could easily use a different variable on the left hand side thus removing that restriction.

ASTL has a formal semantics with respect to a situation theoretic model. This model contains complex structured objects such as (situation) types and facts. It will not be presented here but can be read in full in [Black 93].

Still, we need a little more before we have defined a computational situation theoretic language. A set of inference rules is also defined so that we can derive propositions from systems of basic situations and constraints. Four inference rules are sufficient to provide a system capable of useful computation. *Type reduction* can be illustrated by the following example. When a proposition of a situation with a compound type is true, as in

```
SIT1 : [S ! S != <<sleeps,hanako,1>>
            S != <<dreams,hanako,1>>].
```

the following propositions are also true

```
SIT1 : [S ! S != <<sleeps,hanako,1>>].
SIT1 : [S ! S != <<dreams,hanako,1>>].
```

That is types may be broken down. The second inference rule is *Type Combination* which is the reverse of Type Reduction and states that the types of propositions about the same situations may have their types combined to form new propositions. The third rule is *Modus Ponens* which conventionally states the proving of propositions from positions and constraints. The fourth and perhaps least conventional inference rule is *Argument Promotion*. This rule allows a treatment of (typed) situations as arguments in facts.

$$\frac{\begin{array}{l}\sigma_0\!:\!\tau_0 \texttt{ <= } \sigma_1\!:\![\pi \texttt{ ! } \pi \texttt{ != } \texttt{<<}rel, arg_{0..i},\ \sigma_2\texttt{::}\tau_2\ , arg_{i+2..n}, pol\texttt{>>}] \\ \sigma_1\!:\![\pi \texttt{ ! } \pi \texttt{ != } \texttt{<<}rel, arg_{0..i},\ \sigma_2\ , arg_{i+2..n}, pol\texttt{>>}]\end{array}}{\sigma_0\!:\!\tau_0 \texttt{ <= } \sigma_2\!:\!\tau_2}$$

That is the argument $\sigma_2 :: \tau_2$ is *promoted* from an argument to a clause. The informal motivation is to move conditions about situations out of arguments and into the top level of a constraint. For example from

```
SIT1:[S ! S != <<sees,h,SIT2,1>>].
SIT0:[S ! S != <<happy,h,1>>]
<=
  SIT1:[S ! S != <<sees,h,
                      SIT2::[S ! S != <<happy,t,1>>],
                      1>>].
```

we can deduce

```
SIT0:[S ! S != <<happy,h,1>>]
<=
   SIT2:[S ! S != <<happy,t,1>>]
```

These four inference rules are more formally defined in [Black 93, Ch 3], where a soundness proof is also given for the inferencing method.

Now that we have a formally defined language we can show how it can be used for language processing.

3 Language Processing in ASTL

Given the above simple language we can quite easily encode standard grammar formalisms in it. The feature grammar paradigm neatly translates into this framework.[2] That is to say we are not proposing a new syntactic theory just a method for encoding existing theories in this framework. For example we can represent the utterance of a word as a *situation*. That is there is some situation (i.e. some small part of the world) in which that word was uttered. Given that utterance situation, we can talk about various syntactic (and later semantic) facts about that utterance. For example

“*Hanako*” $\rightsquigarrow$

```
Sit123

cat(SIT123,ProperNoun)
use_of(SIT123,"Hanako")
```

That is we relate a syntactic class to the utterance of the proper noun *“Hanako”*.

We will sometimes use the above box notation for ASTL objects. It is based on EKN (see [Barwise & Cooper 93] for a full definition), a graphical notation for situation theoretic objects. This graphical notation is merely used to make ASTL objects easier to read. Double lined boxes are used for situations, while situation types have a slightly different notation (as we will see below). The current implementation only supports output in EKN form, input must still be specified in the linear form.

An utterance also has a start point and an end point. If, say, a verb starts at the same place that a proper noun ends we may wish to state that the two utterance situations are part of a larger situation representing a sentence. We can represent exactly that in an ASTL constraint. A simple sentence “grammar rule” can be written as

```
*S : [S ! S != <<cat,S,Sentence,1>>
          S != <<start,S,*Start,1>>
          S != <<end,S,*End,1>>]
<=
   *NP : [NP ! NP != <<cat,NP,ProperNoun,1>>
               NP != <<start,NP,*Start,1>>
               NP != <<end,NP,*M,1>>],
   *VP : [VP ! VP != <<cat,VP,VerbPhrase,1>>
               VP != <<start,VP,*M,1>>
               VP != <<end,VP,*End,1>>].
```

In fact ASTL has a built in method for dealing with constraints of this type. Grammar rule constraints in ASTL have their own syntax such that they do not need their start and end points explicitly specified. This allows an implementation to be more efficient. The above rule could be written as (note the different arrow)

```
*S : [S ! S != <<cat,S,Sentence,1>>]
->
   *NP : [NP ! NP != <<cat,NP,ProperNoun,1>>],
   *VP : [VP ! VP != <<cat,VP,VerbPhrase,1>>].
```

[2] There are some low level detailed differences between standard feature systems and the encoding of them in ASTL but they are not important to this presentation.

Of course there is nothing new here. All we are doing is really the same as what Definite Clause Grammars do in Prolog; i.e. offer a more readable syntax for a rule form (in this case grammar rules) which can be described in the basic formalism. However there is one important difference. Here we can easily define constraints for both, facts within the same situations and facts in different situations. Thus as well as simple syntactic structure it is quite easy to define other constraints over the syntactic (and semantic) constituents of an utterance. Feature conventions, as used in theories like GPSG ([Gazdar *et al* 85]) and principles as in GB ([Chomsky 81]) can easily be encoded using both forms of constraints.

The original motivation for ASTL was not simply as a grammar representation language. We originally wished to be able to define and compare contemporary computational semantic theories of natural language. Now that we have shown how syntax can be encoded we must also show how we can use ASTL in semantic processing.

4 Using ASTL for computational semantics

At least initially we will only be concerned with building semantic representations from utterances and put off the actual use of these representations for inference until later. First we will show a very simple treatment of semantics and show how a classical logic-like form may be built up. Again as with the description of syntax above it is not the case that ASTL is a semantic theory. It is merely a tool in which we can describe a theory. Situation Theoretic Grammar ([Cooper 89]) is a situation semantic theory which uses a very simple form for semantic representation. We will base our first example on that. In STG we will relate each utterance situation to two objects which will define the semantics. First via the `sem` relation, each utterance situation is related to a *parametric fact*. That is a fact whose arguments are parameters. Parameters are used to represent parts of the fact that are currently underspecified. Unlike variables in a conventional language parameters denote underspecified objects in the model, they do not need to be bound by quantifiers like variables do. In addition to the parametric fact, utterance situations are also related to an *anchoring environment*. This is represented by a situation that supports facts that anchor parameters in the parametric fact to other objects. This process is analogous to binding variables in a more conventional system. For example an utterance of the transitive verb *"likes"* would be as follows

```
SIT1 :: [S ! S != <<cat,S,TransVerb,1>>
             S != <<sem,S,<<R1,A1,A2,1>>,1>>
             S != <<env,S,
                    AE::[T ! T != <<anchor,R1,likes,1>>
                             T != <<label,R1,pred,1>>
                             T != <<label,A1,subj,1>>
                             T != <<label,A2,obj,1>>],1>>]
```

For this transitive verb there is a parametric fact of arity two. The anchoring environment only anchors the parameter used for the relation (`R1`) as at this stage we do not know what the subject and object are. As well as `anchor` facts the anchoring environment supports *labelling* facts. These are used to identify which grammatical functions are related to which parameters. We could view the parametric fact as a form of abstraction (like a lambda expression). During analysis of an utterance as more information is found parameters can be anchored effectively filling in values. For example the grammar rule for a sentence would be written as

```
[S ! S != <<cat,S,Sentence,1>>
     S != <<sem,S,*Fact,1>>
     S != <<env,S,*SEnv ::
                 *VPEnvType &
                 [Env ! Env != <<anchor,*X,*Y,1>>],1>>]
->
 [NP ! NP != <<cat,NP,NounPhrase,1>>
```

```
       NP != <<sem,NP,*Y,1>>],
  [VP ! VP != <<cat,VP,VerbPhrase,1>>
        VP != <<sem,VP,*Fact,1>>
        VP != <<env,VP,*VPEnv ::
                    *VPEnvType &
                    [Env ! Env != <<label,*X,subj,1>>],
                    1>>].
```

The ampersand in the specification of the (situation) type of the anchoring and labelling environment deserves a little explanation. What is desired is that the environment on the sentence node will support all the facts that are supported by the environment related to the verb phrase node. This is achieved by specifying that the sentence environment has the same type as the verb phrase environment (via the variable `*VPEnvType`). In addition to this the sentence environment has an additional `anchor`-fact that anchors the parameter that is labelled `subj` to the individual representing the semantics of the noun phrase. Thus the sentence utterance situation for *"Hanako likes Taro"* will look like

```
SEN1::[S ! S != <<cat,S,Sentence,1>>
           S != <<sem,S,<<R1,A1,A2,1>>,1>>
           S != <<env,S,
                  AE1::[T ! T != <<anchor,R1,likes,1>>
                            T != <<anchor,A1,h,1>>
                            T != <<anchor,A2,t,1>>
                            T != <<label,R1,likes,1>>
                            T != <<label,A1,subj,1>>
                            T != <<label,A2,obj,1>>],1>>]
```

Of course this is only half the story. In lambda calculus terms we have only specified application but not specified anything about reduction. In ASTL this can be captured by a constraint. We can state that each utterance situation is related to a *described situation*. That is the situation which the utterance describes. The type of that situation can be stated as in the following constraint

```
*S : [S ! S != <<described,S,
              *D :: [T ! T != <<*Rel,*Arg1,*Arg2,1>>],1>>]
<=
   *S : [S ! S != <<cat,S,Sentence,1>>
             S != <<sem,<<*R1,*A1,*A2,1>>,1>>
             S != <<env,S,
                    AE1::[T ! T != <<anchor,*R1,*Rel,1>>
                              T != <<anchor,*A1,*Arg1,1>>
                              T != <<anchor,*A2,*Arg2,1>>],1>>]
```

We use the constraint to do the reduction. The fact specified in the type of the described situation is defined in terms of the objects the parameters in the parametric fact are anchored to.

Of course the above constraint is rather simplified, we really want a general constraint to deal with at least relations of different arity, but the above illustrates the general idea.

Given the above treatment of Situation Theoretic Grammar in ASTL we can now consider interpretation of the result. STG in its original specification does not say anything about the use of the resulting situation semantic representation of an utterance. However as we can derive the described situation from the utterance (via the parametric fact and anchoring environment) we can base our inference on that object. If we have the utterance *"Hanako smiles"* we can derive the type of the described situation. If we also have the constraint that anything that smiles is happy we can derive that Hanako is happy in the described situation. Obviously ideally we need something more complex than this, at least a way of dealing with utterances that are questions but it seems possible that inference on the described situation will be possible within ASTL.

5 Comparing theories in ASTL

As noted at the start of this paper, one of the problems with computational semantics is a number of competing theories are addressing similar semantic issues (e.g. quantification and anaphora) but because of their different notations, comparisons are not easy. Ideally we wish to see their real semantic differences which may be hidden by their different syntactic forms. ASTL has been designed to facilitate such comparisons. In [Black 93] three theories are considered: Situation Theoretic Grammar (STG) [Cooper 89], Discourse Representation Theory (DRT) [Kamp 81] and a form of dynamic semantics based on Dynamic Predicate Logic (DPL) [Groenendijk & Stokhof 91]. However only DRT and DPL are given detailed comparison as those theories specifically address the same issues. It should be noted that the treatments of these theories are not by axiomatizing the theories themselves and showing formal equivalence between that and a description in ASTL. This could be done, but because such semantic theories offer a moving target in terms of formalisation it seems very difficult to come up with an axiomatization that would be generally acceptable (and would always be out of date). Besides, a formalisation of paradigmatic analyses of a theory is sufficient to allow detailed comparison.

In order to compare DRT and DPL we first chose a simple syntactic grammar which was sufficient to include utterances that displayed the semantic phenomena that were considered interesting. Specifically the following simple grammar was taken from [Rooth 87]

$$
\begin{array}{ll}
S & \rightarrow NP,\ VP. \\
VP & \rightarrow V,\ NP. \\
NP & \rightarrow Det,\ N. \\
N & \rightarrow N,\ PP. \\
PP & \rightarrow P,\ NP. \\
D & \rightarrow S. \\
D & \rightarrow D,\ S.
\end{array}
$$

This grammar (with an appropriate lexicon) allows simple sentences (and discourses) of the form

Hanako sings.
A man walks. He talks.
Every man with a donkey beats it.

That is the grammar admits examples of simple declarative sentences (and discourses), simple quantifiers, inter- and intra-sentential anaphora and also donkey anaphora.

In addition to the above grammar which is represented in ASTL grammar rule constraints (as described above) we also add constraints that define the relationship between the syntactic aspects of the utterance and the semantic aspects. An important part of both DRT and DPL is their dynamic aspect. As a discourse progresses information is added to the semantic representation. Specifically each utterance situation can have an input and an output semantic form. The output form is a function of the input form *plus* any information contributed by that part of the utterance. To look at a concrete example, given the discourse *"Hanako sings. Taro dances"*, the discourse representation structure (DRS) from the DRT description after the first sentence would be the following situation type

```
   P1
P1

 named(X,"Hanako")
 sings(X)
```

Thus the input and output DRSs for the utterance situation representing the proper noun *"Taro"* would be

```
DRSIn                                    DRSOut
+----------------------+                 +----------------------+
| P1                   |                 | P2                   |
+----------------------+                 +----------------------+
| P1 |                 |                 | P2 |                 |
|                      |  ⇝ "Taro" ⇝     | named(X,"Hanako")    |
| named(X,"Hanako")    |                 | sings(X)             |
| sings(X)             |                 | named(Y,"Taro")      |
+----------------------+                 +----------------------+
```

This *threading* of information through the discourse can be given an abstract definition so that it may be shared between the DRT and the DPL descriptions. Instead of directly specifying the incoming DRS as the outgoing DRS of the previous part of the discourse we use a `thread` relation to specify the relationship between the utterance situations, and then on top of that threading relation define the DRSs. In the DRT case, discourse representation structures (DRSs) are the objects that are defined over the threading relation while in the DPL case *assignments* of DPL variables (and conditions on them) are threaded. The reusability of parts of each description is not obvious from the basic definition of the theories and it is only when they are described within the same language (i.e. ASTL) that this similarity is identified.

It should be noted that threading is not simply a left to right relation between each word in an utterance. The use of the determiners *"a"* and *"every"* make the threads more complex. Also there may be more than one thread in an utterance (i.e. there may be *sub-threads*). What is important about the threads is that each utterance situation has one and only one incoming thread. Also the semantic structures "passed" along each thread are monotonically increasing in information. This fact is true for both DRSs and DPL assignments.

A second advantage of formally describing semantic theories within the same framework is that as well as finding similarities between theories it may also be possible to copy treatments from one theory to another where one theory deals with a phenomena that another omits. For example Situation Theoretic Grammar in its initial description does not discuss any treatment of pronouns. The DRT description on the other hand gives a treatment for simple inter- and intra-sentential anaphora. Once both DRT and STG are described within ASTL it is easy to copy over the treatment of pronouns from the DRT treatment to the STG one. The DRT treatment includes a notion of *accessibility* for nominals introduced in the discourse (not all nouns are accessible as pronouns referents everywhere in a discourse). Each noun introduces a new *marker* and the accessible markers are threaded through the discourse. When a pronoun is found the accessible markers are checked for suitable referents.

6 Discussion

We have only briefly described the computational situation theoretic language ASTL and its use in dealing with natural language processing, but even with the simple descriptions of DRT and dynamic semantics given in ASTL it has been possible to give a very close comparison of these theories. These theories are designed to cover the same phenomena, with DPL designed to give a more logic-like syntax (at the expense of a more complex semantics) than DRT. Much of the two descriptions in ASTL are shared. The result shows how the DPL representation, as it is currently defined, is forced to carry more information through an analysis than the DRT treatment. Basically DRSs take into account the DRSs they are in, hence their semantics is context sensitive while DPL assignments explicitly contain the whole context. Thus it could be considered computationally more expensive to derive inferences from the DPL representation than from the DRSs—although there are optimization techniques that could remove this difference.

Much of the work in DRT and dynamic semantics has been in construction of representations and the semantics of the expressions themselves. Although they have both worked on increasing their linguistic coverage, little has been done on *using* the resulting expressions in systems and drawing inferences from them. ASTL provides a suitable framework in which such work can be done. The resulting semantic expressions constructed from utterances are themselves situation theoretic objects (usually a form of situation type). ASTL already provides the mechanisms necessary for inference from such situation theoretic objects.

Although we have shown that ASTL is suitable for use in describing aspects of natural language semantic theories there is still the issue of whether ASTL is better than already existing frameworks such as feature grammars or logic programming languages. In order to look at this we need to identify the key aspects that make ASTL a suitable language for its desired tasks.

The fundamental requirements for a language suitable for use in semantic description seem to be:

— The ability to represent complex structured objects and allow general relations between them.

— The ability to have constraints between general objects and draw inferences from them.

— A mechanism for reasoning about "variables" in the object language (as distinct from variables in the meta-language) and describing binding mechanisms for these object language variables.

In situation theoretic terms the above properties relate directly to: situations and abstractions; constraints; and parameters and anchoring. All three of these are fundamental properties of situation theory. So it seems reasonable to place this work in the field of situation theory. But it is wrong to think that ASTL is the only possible way to achieve a computational language as a common framework for describing natural language semantic theories. These properties can be found in other paradigms.

The above properties could easily be found (and exploited) in conventional Prolog. Although Prolog actually has only one "situation" (or database) it is quite easy to encode a partitioning of the database (e.g. by adding an extra argument to predicates identifying the situation in which that fact is supported). General constraints between situations could also be written. But the point we wish to make here is not that such a language could not be defined in Prolog (or any other general programming language) but that the features of ASTL are those that are necessary. There are implementations in Prolog for the three semantic theories that have been described in ASTL but it is not always possible in those implementations to distinguish between aspects of the described theory and aspects of the programming structure of Prolog. ASTL seems to be more suited to the actual task.

In a similar way we could define a feature system (e.g. in the spirit of feature logics [Smolka 88] or attribute value logics [Johnson 88]) which has exactly the properties of ASTL. But if we did so, which is probably quite easy to do, then we would really have an implementation of ASTL in a feature system rather than anything new. Likewise if we restricted ourselves in Prolog to offer just the necessary features of ASTL again it could be rightly viewed as an implementation of ASTL rather than a new language.

There is another computational language based on situation theory ([Nakashima *et al* 88]). PROSIT offers a Prolog like language using aspects of situation theory. However its development has concentrated on its use in knowledge representation rather than as in ASTL's case on natural language processing.

However ASTL is by no means the end. There are aspects of semantic theories that although can be described in ASTL as it is currently defined, really require extensions to ASTL in order make their descriptions more natural. For example the use of reduction of abstractions and anchoring environments can be defined by constraints but it would be better if there was a built in mechanism for reduction (in lambda calculus terms, beta reduction). As reduction will be a common requirement a built in feature of the language would allow a much more efficient implementation.

Situation theory offers effectively a three-valued logic for situations and facts. A situation can support a fact which can specify, via the polarity, whether the relation holds or does not hold for the given arguments; or that the fact is not supported (i.e. unknown). At present nothing is defined in ASTL (ASTL only embodies a part of situation theory) to specifically use this. Although this could be used it would be better to perhaps develop ASTL to support more in this area.

Also the whole concept of defaults is currently ignored. However there is research in the situation theory literature which does point in that direction, this or more general AI literature on defaults and *defeasible constraints* should be picked up and used as a basis for a form of default reasoning. There does not seem to be any more problems in extending ASTL in that direction as there are in extending any other similar language.

7 Conclusions

We have briefly outlined a computational language based on the fundamental aspects of situation theory[3]. The motivation behind it was to develop a common framework in which natural language semantic theories could be described. Contemporary theories of natural language semantics such as Situation Semantics, DRT and dynamic semantics have very different syntax and apparently different semantics. A common language for formal specifications would allow a much more detailed comparison.

ASTL is an initial attempt at such a language. It is formally defined and has been implemented. It can be used to describe conventional natural language grammars and the inference engine is suitable to derive "parses" from utterances. Formal specifications can be made in the language of key aspects of semantics theories that are sufficient to derive semantic representations from utterances. Although no use of that semantic translation is actually made yet the form is of a type suitable for inferences to be drawn.

There are still extensions which would make ASTL more useful but even in its current form it does seem to be adequate for simple comparisons of semantic theories. Already it has been possible to transfer techniques from one theory to another once they were described in ASTL.

The basic properties are abstract and could be found in other formalisms but these properties are also fundamental properties of situation theory so it seems reasonable to place ASTL within that field.

References

[Barwise & Cooper 93] J. Barwise and R. Cooper. Extended Kamp Notation: a graphical notation for situation theory. In *Situation Theory and its Applications, III*, CSLI Lecture Notes. Chicago University Press, forthcoming 1993.

[Barwise & Perry 83] J. Barwise and J. Perry. *Situations and Attitudes.* MIT Press, Cambridge, Mass., 1983.

[Black 93] A.W. Black. *A Situation Theoretic approach to computational semantics.* Unpublished PhD thesis, University of Edinburgh, Edinburgh, UK., 1993.

[Chomsky 81] N. Chomsky. *Lectures on Government and Binding.* Studies in Generative Grammar 9. Foris, Dordrecht, Holland, 1981.

[Cooper 89] R. Cooper. Information and grammar. Technical Report RP No. 438, Dept of Artificial Intelligence, University of Edinburgh, 1989. Also to appear in J. Wedekind, ed. *Proceedings of the Titisee Conference on Unification and Grammar.*

[Gawron & Peters 90] M. Gawron and S. Peters. *Anaphora and Quantification in Situation Semantics.* CSLI Lecture Notes Number 19. Chicago University Press, 1990.

[3]A prototype implementation of ASTL in Common Lisp is available by anonymous ftp from scott.cogsci.ed.ac.uk:/pub/awb/astl-0.4.tar.Z

[Gazdar *et al* 85] G. Gazdar, E. Klein, G. Pullum, and I. Sag. *Generalized Phrase Structure Grammar*. Blackwell, Oxford, 1985.

[Groenendijk & Stokhof 91] J. Groenendijk and M. Stokhof. Dynamic Predicate Logic. *Linguistics and Philosophy*, 14:39–100, 1991.

[Johnson 88] M. Johnson. *Attribute-value logic and the theory of grammar*. CSLI Lecture Notes Number 16. Chicago University Press, 1988.

[Kamp 81] H. Kamp. A theory of truth and semantic representation. In J. Groenendijk, T. Janssen, and M. Stokhof, editors, *Formal Methods in the Study of Language*. Mathematical Center, Amsterdam, 1981.

[Nakashima *et al* 88] 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*, pages 489–498. ICOT, 1988.

[Rooth 87] M. Rooth. Noun phrase interpretation in Montague Grammar, file change semantics and situation semantics. In P. Gardenförs, editor, *Generalised Quantifiers*, Studies in Linguistics and Philosophy, 31. Reidel, Dordrecht, 1987.

[Smolka 88] G. Smolka. A feature logic with subsorts. LILOG Report 33, IWBS, IBM Deutschland, 1988. To appear in: J. Wedekind and C. Rohrer (eds.), Unification in Grammar; The MIT Press, 1991.