Embedding DRT in a Situation Theoretic Framework

Alan W Black
Dept of Artificial Intelligence, University of Edinburgh,
80 South Bridge, Edinburgh EH1 1HN, UK.
awb@ed.ac.uk

Abstract

This paper proposes the use of situation theory as a basic semantic formalism for defining general semantic theories. ASTL, a computational situation theoretic language, is described which goes some way to offering such a system. After a general description of Discourse Representation Theory an encoding of DRT in ASTL is given. Advantages and disadvantages of this method are then discussed.

Topic: computational formalisms in semantics and discourse

Introduction

The purpose of this paper is to show how a computational language based on situation theory can be used as a basic formalism in which general semantic theories can be implemented. There are many different semantic theories which, because of their different notations, are difficult to compare. A general language which allows those theories to be implemented within it would offer an environment where similar semantic theories could be more easily evaluated.

Situation Theory (ST) has been developed over the last ten years [2]. Much work has been done in both the formal aspects of situation theory and its use in natural language semantics (situation semantics), however so far little work has been done in its computational aspects. It is the eventual goal of the work presented here to show how situation theory can be used computationally and how a computational situation theoretic language can provide an environment in which different semantic theories can be easily compared.

Because there are so many variants of ST we must define our own here. The language ASTL [3] has been defined. Although it uses surprisingly few features of situation theory, it seems powerful enough to act as a basic language for semantics. It has been considered that some extension to "classical" feature structures be made and use those to encode semantic forms. Features systems augmented with set values, cyclicity and other extensions may be powerful enough but the method described here takes an existing semantic theory and refines it rather than building a new one.

This paper is basically split into two sections. The first discusses how ST can be used in a computational system, and introduces the language ASTL. The second half of this paper discusses Discourse Representation Theory (DRT) as a theory in itself and shows how it can be encoded with ASTL.

ST and Computation

The view according to situation theory is that parts of the "world" can be described as situations. Situations support facts. Facts can be true, false, or undefined in some situation. A fact's truth value may be different in different situations. Situations are first class objects in the theory, and hence they can be used as arguments to facts so that relations can be defined between situations. Situations are useful in translations for *naked infinitives* (e.g. "*see*"). Situations make ST different from more conventional logical theories although there have been proposals to add situation-like objects to more classical theories like Montague grammar [8].

As well as situations and partiality, situation theory offers many other intensional objects, including abstractions, types, and parameters (partially determined objects). These form a rich formalism for describing semantic phenomena. However these features alone do not constitute a computational system, with the addition of constraints and rules of inference we can have the basis for a computational system. The idea of a computational situation theoretic language has been considered elsewhere. Most notable is the language PROSIT [9] which offers a Prolog-like language based on situation theory rather than first order logic. Other systems (e.g. [5]) allow the representation of situations etc. within some other formalism (e.g. feature structures) but do not use situation theory itself as the basis for the language.

ASTL

ASTL is a language based on situation theory. It takes a very conservative view of situation theory, admitting only some basic parts. Although ASTL may need to be extended later, it already can be used to describe simple versions of semantic theories (such as situation semantics and DRT). Rather than use, or extend, PROSIT it was decided to develop a new language. ASTL includes some built-in support for natural language parsing based on the ideas of Situation Theoretic Grammar [4] while PROSIT is designed more for knowledge representation than direct language processing.

ASTL allows the following *basic terms*:

Individuals : e.g. `a`, `b`, `c`.
Parameters : e.g. `X`, `Y`, `Z`.
Variables : e.g. `*X`, `*Y`, `*Z`.
Relations : e.g. `see/2`. Relation name and arity.
i-terms : consisting of a relation, arguments and a polarity (0 or 1), e.g. `<<sing,h,1>>`.

types : consisting of an abstraction over propositions. For example

```
[S ! S != <<sing,h,1>>
     S != <<see,h,S,1>>]
```

That is the type of situation which supports the fact that h sings and h sees that situation.

Situations : written as names optionally followed by a type. e.g.

```
S1::[T ! T != <<run,t,1>>].
S2::[S ! S != <<see,h,S1,1>>].
```

In addition to terms there are the following *sentences*:

Propositions : consisting of a situation and a type e.g.

```
Sit1:[S ! S != <<see,h,S,1>>
          S != <<dance,h,1>>]
```

Constraints : are defined between propositions, they consist of a proposition following by <= followed by a list of propositions. For example

```
Sit1:[S ! S != <<happy,h,1>>]
 <=  Sit1:[S ! S != <<smile,h,1>>].
```

The semantics of ASTL (defined fully in [3]) are defined in terms of a model consisting of individuals, relations, parameters, situations and a set consisting of pairs of situations and facts. Informally, a proposition is true if the denotation of the situation supports all of the facts in the type. A constraint is true if when all the propositions in the right hand side of the constraint are true, the left hand proposition is true also. As it is currently defined ASTL has no built-in definition with respect to *coherence*, that is there is no built-in mechanism that stops a situation supporting both a fact and its dual (the fact with the opposite polarity).

Constraints can be generalised using variables. An example will help to illustrate this. If we define the following basic situation and constraint:

```
Sit1:[S ! S != <<smile,t,1>>].
*S:[S ! S != <<happy,*Y,1>>]
 <=  *S:[S ! S != <<smile,*Y,1>>].
```

Informally the constraint states that in any situation where something smiles it is also happy (in that same situation). From the above basic axioms we can derive that the following is true:

```
Sit1:[S ! S != <<happy,t,1>>
          S != <<smile,t,1>>]
```

Rather than just use the linear forms for displaying ASTL objects, an extension has been added for output. Based on EKN [1] ASTL objects can be displayed as boxes, making complex objects much easier to view. In this notation we write situations as boxes with their names in a top left inset with facts written (in a more conventional predicate argument form) inside the box.

Using the work of Cooper [4] we can process language in a situation theoretic way. Situation Theoretic Grammar takes the view that utterances can be represented by situations. For example

```
               | SIT123 |
"Hanako" ~>    |  cat(SIT123,ProperNoun)
               |  use_of(SIT123,"Hanako")
```

That is, the use of the phrase "*Hanako*" gives rise to a situation that supports the facts that it (the situation) is a ProperNoun and it is a use of the word "*Hanako*". We call these ***utterance situations.*** As an utterance happens at a particular time and location this fact should also be recorded in the situation. In ASTL this temporal aspect is built-in to the language. A special form of constraint, *grammar rules*, can be used to constrain utterance situations. General constraints apply to any form of situation (utterance or otherwise) while grammar rules only apply to utterance situations. A grammar rule between utterance situations such as

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

takes into account that the two utterance situations occur next to each other. It is possible to model all of this within the standard constraint system by adding facts about start and end points of utterances (in a similar way that DCGs are interpreted in Prolog) but as one of the main uses of ASTL is language processing it was felt more efficient to build utterance situations (and constraints on them) directly into the language.

A basic implementation has been made within Common Lisp which takes ASTL *descriptions* (definitions, basic situations and constraints) and allows queries to be made about their sets of constraints and basic situations.

Discourse representation theory

Given a simple language like ASTL there is now the question about how it can be used in representing other semantic theories. DRT [7] offers a representation for discourses. A ***discourse representation structure*** (DRS) is defined at each stage in a discourse describing the current state of the analysis. A DRS consists of two parts; a set of ***domain markers***, which can be bound to objects introduced into the current discourse, and a set of ***conditions*** on these markers. DRSs are typically written as boxes with the markers in the top part and conditions below. For example a DRS for the utterance "***a man sings***" is

```
|    X     |
|----------|
|  man(X)  |
|  sing(X) |
```

The following description of DRT in ASTL is based on the DRT definition in [6]. First we need a syntactic backbone to be able to discuss the construction of a DRS for a discourse. As seen (briefly) above ASTL offers a basic grammar formalism. That is, grammar rules are specified as constraints between utterance situations.

Given such a backbone we need to define an ASTL representation for DRSs. DRSs have two parts. Discourse markers can be represented as parameters in ASTL. In situation theory parameters denote partially determined objects. Parameters can be ***anchored*** to other objects as information about their denotation is found. DRS conditions are represented by i-terms. A DRS itself is represented as a parametric situation—a situation whose type contains parameters. Discourse markers are not explicitly listed in the DRS representation. An ASTL representation of the DRS for "*a man sings*" is

```
    Sit345::[S ! S != <<man,X,1>>
                 S != <<sing,X,1>>]
```

where `X` is a parameter.

This allows a simple semantics close to that of a conventional DRS. That is an ASTL DRS will be true for some situation (i.e. a model) if there exists an anchoring for the parameters in it which make it a type of the model-situation. A special definition will be needed for the condition `every` (and possibly others if extensions to basic DRT are included). It may be better to think of the situation name also as a parameter which gets anchored to the model-situation. But as the semantics of ASTL relates situations names to situations (i.e. two situation names can denote the same situation) there is still a level of indirection.

DRSs are objects which are related to utterance situations. They are not themselves representations of the utterances but representations of what the utterances describe.

Threading

An important aspect of DRT is how a DRS is constructed from a discourse. Here (and in [6]) we use the technique of *threading*. The general idea is that a DRS gets passed through a discourse being added to as the discourse progresses.

In this description, a discourse consists of a set of utterance situations which can be viewed through a number of different structural relations. The first is through the relation `daughter` which defines the syntactic structure of the discourse as defined by the grammar rules (immediate dominance and linear precedence). Secondly the `thread` relation defines an ordering of the utterance situations used in the generation of the DRSs. Lastly there are two relations, `range` and `body` used in defining the logical structure of the discourse.

The threading relation is a binary relation between utterance situations. We will say the first argument is *threaded* to the second. Each utterance situation appears exactly once as the second argument in the `thread` relation (i.e. it has exactly one incoming thread). There is one exception, a special situation called `DStart` which does not have an incoming thread (it is used to represent the null context at the start of a discourse), but does appear as an incoming thread for one or more utterance situations. There are no cycles in threading but as we shall see there may be more than one linked thread of utterances within a discourse. The actual construction of the threading relations is discussed later.

Each utterance situation is related to two DRSs, through the relations `DRSIn` and `DRSOut`. A `DRSIn` DRS is the `DRSOut` DRS of the incoming thread. This constraint can be written in ASTL as

```
  *S:[S ! S != <<DRSIn,S,*DRS,1>>]
  <=    TS:[TS ! TS != <<thread,*S1,*S,1>>],
        *S1:[S1 ! S1 != <<DRSOut,S1,*DRS,1>>].
```

The relation between the two DRSs related to an utterance is also constrained. This is a core part of DRT. Basically the outgoing DRS contains the same information as the incoming DRS *plus* any information the utterance adds to the discourse. In the case of a proper noun utterance situation we can capture this relation with the following constraint:

```
*S:[S ! S != <<DRSout,S,*DRSout::
               *DRSInType &
               [D ! D != <<name,*X,*N,1>>],1>>]
<=
 *S:[S ! S != <<cat,S,ProperNoun,1>>
         S != <<use_of,S,*N,1>>
         S != <<sem,S,*X,1>>
         S != <<DRSIn,S,*DRSin::*DRSInType,1>>]
```

Information is monotonically increasing in DRSs as we traverse along a thread. We are not destructively modifying a DRS as the discourse progresses but constructing a new DRS which supports the same conditions as the incoming DRS. The constraint above forms the outgoing DRS from the type (`*DRSInType`) of the incoming one, which will contain all the conditions of the incoming DRS, plus a new condition introducing the parameter for the proper noun and a condition on its name.

We also have the constraint that any argument or relation that appears in the conditions of a DRS must be related to some utterance situation by the relation `sem` previously in that thread. This condition means that arguments are threaded before predicates. For example both the subject NP and object NP of a simple sentence will be threaded before the VP. In contrast in [6] the VP comes before a object NP which means a DRS is created with an argument in a condition which is not yet determined (i.e. a free variable).

The other structural relations are `range` and `body`. Each determiner utterance situation appears in exactly one `range`-relation and exactly one `body`-relation. The second argument to these relations are utterance situations that do not appear as first arguments in any threading relation (i.e. they are ends of threads). The `DRSOut` of a determiner utterance situation is a function of the `DRSIn` DRS plus information from the `range` and `body` related threads. In the `every` determiner case the `DRSOut` constraint is

```
*S:[S ! S != <<DRSOut,S,*DRSOut::
               *DRSInType &
               [DS ! DS != <<every,*RangeDRS,
                             *BodyDRS,1>>],1>>]
```

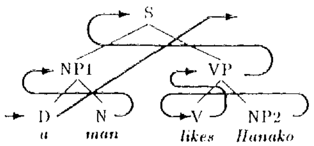
```
<=
 *S:[S ! S != <<cat,S,Determiner,1>>
         S != <<DRSIn,S,*DRSIn::*DRSInType,1>>
         S != <<sem,S,every,1>>],
 TS:[TS !
     TS != <<body,*S,*Body::
      [S ! S != <<DRSOut,S,*BodyDRS,1>>],1>>
     TS != <<range,*S,*Range::
      [S ! S != <<DRSOut,S,*RangeDRS,1>>],1>>]
```

While for the indefinite determiner the DRSOut simply contains all the conditions from the DRSin, range and body related utterances.

```
*S:[S ! S != <<DRSOut,S,*DRSOut::
                *DRSInType & *DRSRType &
                *DRSBType,1>>]
<=
 *S:[S ! S != <<cat,S,Determiner,1>>
         S != <<DRSIn,S,*DRSIn::*DRSInType,1>>
         S != <<sem,S,some,1>>],
 TS:[TS !
    TS != <<body,*S,*Body::
     [S ! S != <<DRSOut,S,
                *BodyDRS::*DRSBType,1>>],1>>
    TS != <<range,*S,*Range::
     [S ! S != <<DRSOut,S,
                *RangeDRS::*DRSRType,1>>],1>>].
```

But how is threading built? The grammar rules specify the basic syntactic structure (via the daughter relations). At the same time the threading information can be constructed. Each utterance situation is related to two others by the relations need and out. The need relation identifies the utterance situation (either itself or one of its daughters) which requires an incoming thread while out identifies which situation is to be threaded on to the next part of the discourse. Although the need and out relations are determined at the time a grammar rule is realised the actual thread, range and body relations may not be determined locally. The utterance to be threaded to the need of an NP cannot be realised until the NP is put in context. In contrast with [6] instead of passing up the utterance that needs a thread, they pass down the "utterance" that is to be threaded in. Here we give a bottom up definition rather than as in [6] a top down one.

As seen in the constraints above the structural facts whose relations are thread, range and body are collected in a situation called TS. Below is an example sentence shown as a syntax tree with the thread relation drawn as arrows to show the flow of information through the discourse



In addition, DStart is threaded to D, N and NP2. The main discourse thread will go through D. There are two other threads ending at NP1 and S. D will be related to NP1 by the relation range and to S by the relation body. Hence the output DRS from the sentence (from the determiner "*a*"— by the constraints given above) is built from the incoming DRS *plus* the outgoing DRSs from NP1 and S (which are related to D via the range and body relations).

Pronouns and Accessibility

Unlike other utterance situations, pronouns do not just add new information to a DRS. They also require existence of some referent already introduced in the context. To put it simply there must be a suitable object in the incoming DRS that the pronoun can match. A condition can be written as

```
*S:[S ! S != <<DRSout,S,*DRSout::
               *DRSInType &
               [DS ! DS != <<is,*X,*Y,1>>],1>>]
<=
 *S:[S ! S != <<cat,S,Pronoun,1>>
         S != <<type,S,*TYPE,1>>
         S != <<sem,S,*X,1>>
         S != <<DRSIn,S,*DRSIn::*DRSInType,1>>
         S != <<accessible,S,*A::
               [A ! A != <<*TYPE,*Y,1>>],1>>].
```

Where *TYPE will be one of male, female or neuter. However, it is not sufficient to simply check the conditions in the incoming DRS for some marker of the right type.

The accessible relation is also defined over the threading relations. Each utterance situation is related to a situation that supports the facts about which markers are accessible at that point in the discourse. The accessible markers for an utterance situation U are defined (informally) as follows:

If U is a noun (or propernoun) the accessible markers are from that noun plus the accessible markers from the incoming thread.

if U is the start of a thread whose end is related to a determiner by the relation body then the accessible markers are those from the end of that determiner's range thread.

if U is the start of a range thread, the accessible markers are those from the incoming thread of the related determiner.

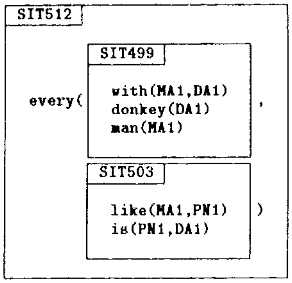
if U is an indefinite determiner the accessible markers are those of the end of the body thread

if U is an every determiner the accessible markers are those from its incoming thread (i.e. does *not* include the markers introduced in the range and body threads).

otherwise the accessible markers are those of the incoming thread

These conditions can easily be represented by ASTL constraints.

Given the above descriptions: a syntactic backbone, a DRS representation, threading and definition for accessibility, we can form DRSs for simple discourses. The coverage is that of [6]. This still allows an example of donkey anaphora as in "*every man with a donkey likes it*". The DRSOut for the discourse utterance situation is.



Discussion

Although translation of DRT into ASTL is possible there are some important consequences. The semantics of an ASTL DRS, briefly described above, requires that it is possible to tell the properties of every object in the situation. As situations are partial it may not be defined for everything whether it is a man or not, thus it is not possible to define "all men." (Note, lack of information does *not* imply falsity.) This is perhaps unfair to consider this as a problem as in the standard definitions of DRT it is required that the model be complete (all properties are defined on all objects) — so it seems no worse to require this of the situation in which we are finding the truth conditions of a DRS. However we could include further definitions for the **every** relation and require that there be some resource situation that identifies actual objects that fall in its scope. This technique has been used by [4].

There is the question of compositionality. It could be said that the threading relations are only partially determined compositionally. But this seems exactly what the theory states and the intuition behind it. We cannot define a DRS for a noun phrase unless we know what context the NP is in. All that can be determined is partial definition with conditions on the context.

An important aspect of DRT is that there is a left to right dependency on DRSs. This does not necessarily mean that parsing must be left to right, though normally it will be. A definition of DRT should include this dependency and not rely on how a implementation happens to order processing. The ASTL definition does include a left to right dependency, without specifying a processing order on the inference mechanism.

Summary

This paper has introduced the notion of using situation theory as a basic formalism in which other semantic theories might be defined. A computational situation theoretic language called ASTL is discussed. Situation theory is suitable as basis for a metatheory because a representation of situations allows the representation of higher order objects necessary for describing other semantic theories. A possible translation of DRT in ASTL is given. The coverage is that of [6].

This translation is interesting because first it shows that situation theory is not some opposing semantic theory but that it can be used in discussing other theories. However perhaps it is not surprising that a language such as ASTL is powerful enough to give this translation. A feature system, with sets (or some definition), cycles and constraints is close to what ASTL is, but it is interesting that these properties can be found as the basis in a current semantic theory without introducing a new theory. Finally a situation theoretic description of DRT allows extensions of DRT to use the properties of situation theory. Situations which are useful in describing various natural language semantic phenomena (e.g. naked infinitives) are now readily available to be included in extensions of DRT.

Acknowledgements: This work was supported by an SERC studentship award number 89313458. I would also like to thank Robin Cooper, Ian Lewin and Graeme Ritchie for comments and guidance on this work.

References

[1] J. Barwise and R. Cooper. Simple Situation Theory and its graphical representation. In *Partial and Dynamic Semantics III*, DYANA R2.1.C, Cognitive Science, University of Edinburgh, 1991.

[2] J. Barwise and J. Perry. *Situations and Attitudes*. MIT Press, 1983.

[3] A. Black. A situation theoretic approach to computation semantics. forthcoming PhD thesis, Dept of AI, University of Edinburgh, 1992.

[4] R. Cooper. Information and grammar. Technical Report RP No. 438, Dept of AI, University of Edinburgh, 1989.

[5] J. Fenstad, P-K. Halvorsen, T. Langholm, and J. van Bentham. *Situations, Language, and Logic*. Reidel, Dordrecht, 1987.

[6] M. Johnson and E. Klein. Discourse, anaphora and parsing. In *COLING86*, Bonn, 1986.

[7] 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.

[8] R. Muskens. Meaning and Partiality. PhD thesis, University of Amsterdam, 1989.

[9] H. Nakashima, H. Suzuki, P-K. Halvorsen, and S. Peters. Towards a computational interpretation of situation theory. In *FGCS*, ICOT, 1988.