Describing the Approaches

The FraCaS Consortium
Robin Cooper, Richard Crouch, Jan van Eijck, Chris Fox,
Josef van Genabith, Jan Jaspers, Hans Kamp,
Manfred Pinkal, Massimo Poesio, Stephen Pulman, Espen Vestre

FraCaS

A Framework for Computational Semantics
LRE 62-051

Deliverable D8

December 1994
A Framework for Computational Semantics

CWI Amsterdam

University of Edinburgh
Centre for Cognitive Science and Human Communication Research Centre

Universität des Saarlandes
Department of Computational Linguistics

Universität Stuttgart
Institut für Maschinelle Sprachverarbeitung

SRI Cambridge

For copies of reports, updates on project activities and other FRACAS-related information, contact:

The FRACAS Project Administrator
University of Edinburgh
Centre for Cognitive Science
2 Buccleuch Place
Edinburgh EH8 9LW, Scotland, UK
fracas@cogsci.ed.ac.uk

Copies of reports and other material can also be accessed via anonymous ftp from ftp.cogsci.ed.ac.uk, directory pub/FRACAS.

©1994, The Individual Authors
No part of this document may be reproduced or transmitted in any form, or by any means, electronic or mechanical, including photocopy, recording, or any information storage and retrieval system, without permission from the copyright owner.
Contents

1 Discourse Representation Theory ........................................ 8
  1.1 Semantic Tools ...................................................... 8
    1.1.1 A Dynamic and Representational Account of Meaning ........ 8
    1.1.2 A Simple DRS Language and its Interpretation .............. 17
    1.1.3 DRS Construction Algorithm, Accessibility and Resolution ... 20
    1.1.4 Compositionality ............................................. 22
    1.1.5 Extensions to the Basic Framework ............................ 23
    1.1.6 Lexical Semantics ............................................. 46
    1.1.7 Inferencing .................................................... 54
    1.1.8 Underspecification ........................................... 61
  1.2 Syntax-semantics Interface ........................................ 66
    1.2.1 The Top-Down Construction Algorithm ......................... 67
    1.2.2 A Bottom-Up Version .......................................... 71
    1.2.3 A Declarative Reformulation in Terms of Equation Solving ... 73
    1.2.4 A HPSG-style UDRS Syntax-Semantics Interface ............... 76

2 Update and Dynamic Semantics ........................................ 80
  2.1 Semantic Tools ...................................................... 80
2.1 Dynamic horizons ............................................. 80
2.2 Dynamic Parameters ........................................... 81
2.3 Existing Systems ............................................... 90
2.4 Towards a Common Framework for Dynamic Semantics .... 102

2.2 Syntax-semantics Interface .................................. 107
2.2.1 Syntactic Component ....................................... 107
2.2.2 Compositional Extensional Semantics ................... 108
2.2.3 Compositional Intensional Semantics .................. 111

3 Situation Semantics ............................................. 114

3.1 Semantic Tools .................................................. 115
3.1.1 Introduction: Syntax and Semantics of a Situation-Theoretic Language 115
3.1.2 An Information-Based, Algebraic Approach to Semantics ........... 115
3.1.3 Predicate Structures ....................................... 118
3.1.4 Infon Structures .......................................... 119
3.1.5 Infon Algebras ............................................. 120
3.1.6 Situations .................................................. 123
3.1.7 Truth and Propositions ................................... 127
3.1.8 Parameters and Quantification ............................ 133
3.1.9 Context-Dependency ...................................... 142

3.2 Syntax-semantics Interface ................................. 144
3.2.1 A Sample Grammar Illustrating the syntax-semantics interface .... 144
3.2.2 Notation and Operations .................................. 150
3.2.3 The Gawron & Peters Fragment ............................ 157
4 Property Theory

4.1 Semantic Tools ........................................ 167
  4.1.1 The Basic Theory\(^1\) .......................... 172
  4.1.2 Definitions of Types ............................... 176
  4.1.3 Model\(^2\) .................................. 178
4.2 Syntax-semantics Interface ............................ 180
  4.2.1 PTQ-like Interpretation ............................ 181
  4.2.2 Underspecified Semantics .......................... 182

5 Monotonic Semantics .................. 184

5.1 Semantic Tools ........................................ 184
  5.1.1 Introduction and Motivation ....................... 184
  5.1.2 Interpretation and Monotonicity ................... 186
  5.1.3 QLF Syntax .................................. 195
  5.1.4 QLF Semantics ................................ 200
  5.1.5 QLF Semantics: Comments and Criticisms ........ 206
  5.1.6 Comparisons and Alternatives to QLF ........... 213
5.2 Syntax-semantics Interface .................. 216
  5.2.1 Syntax-Semantics Rules ........................... 216
  5.2.2 Reference Rules and Resolution .................. 216
  5.2.3 Reversibility ................................ 217
Describing the Approaches

The FraCaS Consortium
Robin Cooper, Richard Crouch, Jan van Eijck, Chris Fox,
Josef van Genabith, Jan Jaspers, Hans Kamp,
Manfred Pinkal, Massimo Poesio, Stephen Pulman, Espen Vestre
Chapter 1

Discourse Representation Theory

In the following two sections we give a brief introduction into Discourse Representation Theory.

In the first section we retrace some of the basic motivations for the development of DRT and give a brief outline of the standard formulation of the theory as well as some of its more recent extensions.

In the second section we consider four alternative formulations of DRS construction: the standard formulation in terms of a top down construction algorithm, a semi-compositional bottom up version, a version based on equation solving and a HPSG-style principle based specification for Underspecified Discourse Representation Structures (UDRSs).

The present notes cannot claim to be comprehensive in any reasonable sense of the word. There is much further work in DRT which should have been included here but due to limits of time and space could not have been covered to the extent required to do justice to these proposals.

1.1 Semantic Tools

1.1.1 A Dynamic and Representational Account of Meaning

Natural language texts are highly structured objects with a considerable amount of inter- and intrasentalential cohesion. Much of this cohesion can be traced back to anaphoric properties of natural language expressions, that is their capacity to refer back to (or point forward to) other expressions in the text\(^1\). Pronominals and tense and aspect are but two examples of

\(^1\text{DRT and other dynamic semantic theories focus on textual anaphora. This is not meant to indicate that deictic and common ground etc. anaphora are in any sense considered less important.}\)
anaphoric devices – devices whose anaphoric nature was realized many years ago but which, it turned out, were difficult to capture with the machinery available within formal semantics in the 60’s and 70’s. Traditionally, formal approaches to natural language semantics have focused on the sentential level. If such an approach is extended to capture intersentential anaphoric phenomena, it soon becomes evident that (i) the narrow conception of meaning in terms of truth conditions has to give way to a more dynamic notion and (ii) the traditional analysis of (NP) anaphora in terms of bound variables and quantificational structures has to be modified. Below we briefly retrace some of the basic and by now often rehearsed arguments.

DRT is probably still best known for its treatment of the inter- and intrasentential anaphoric relations between indefinite NPs and personal pronouns. In this first section we will concentrate on this part of the theory and somewhat arbitrarily refer to this part as “core DRT”.

In predicate logic the following two expressions are equivalent

\[ \exists x \Phi \iff \neg \forall x \neg \Phi \]

If \( \Phi = (\text{man}(x) \land \text{walk}_{in\text{park}}(x)) \), then the two formulas are approximate representations of

(2) A man is walking in the park.

and

(3) It is not the case that every man is not walking in the park.

However, while (2) can be extended into the mini-discourse

(4) A man, is walking in the park. He, is enjoying himself,

where coreference is indicated by subscripts, its truth-conditionally equivalent counterpart (3) does not admit of any such extension:

(5) * It is not the case that every man, is not walking in the park. He, is enjoying

---

2C.f. the introductory sections of [Kamp, 1981], [Groenendijk and Stokhof, 1991a], [Groenendijk and Stokhof, 1990] and textbooks such as [Kamp and Reyle, 1993] and [Gamut, 1991]. Some of our presentation will be based on these sources.

3Historically this is somewhat inaccurate since the original motivation for the development of DRT was provided by accounts of temporal anaphora. Here it should also be mentioned that DRT did not come completely “out of the blue”. Some of the central ideas in DRT were in some form or other already present and/or being developed at about the same time as the original formulation of DRT in e.g. the work of [Karttunen, 1976], [Heim, 1982] and [Seuren, 1986].
himself;

This example provides a simple illustration that truth conditions do not fully capture the contextual role of a sentence.


(6) If Pedro; owns a donkey, he; likes it.

(7) Every farmer; who owns a donkey; likes it.

It has been widely agreed that the truth conditions associated with (6) correspond to the truth conditions associated with the following predicate logic formula:

(8) \( \forall x[(\text{donkey}(x) \land \text{own}(\text{pedro}, x)) \rightarrow \text{like}(\text{pedro}, x)] \)

A Montague-style quantifying-in approach [Montague, 1973] would result in

(9) \( \exists x[(\text{donkey}(x) \land \text{own}(\text{pedro}, x)) \rightarrow \text{like}(\text{pedro}, x)] \)

while a direct insertion\(^4\) approach would produce an open formula like

(10) \( \exists x(\text{donkey}(x) \land \text{own}(\text{pedro}, x)) \rightarrow \text{like}(\text{pedro}, x) \)

Neither (9) nor (10) are adequate representations of the perceived meaning of (6). Furthermore, in the predicate logic representation (8) of the perceived meaning of (6) the indefinite NP a donkey surfaces as a universally quantified expression in the representation taking wide scope over the material implication operator while in (2) the indefinite a man has existential import. Indefinite expressions, however, are usually uniformly associated with existentially quantified terms.

The occurrence of the indefinite noun phrase a donkey inside the relative clause in (7) poses similar problems. The perceived meaning of (7) corresponds to the predicate logic formula

(11) \( \forall x \forall y[(\text{farmer}(x) \land \text{donkey}(y) \land \text{own}(x, y)) \rightarrow \text{like}(x, y)] \)

where again an indefinite NP, this time located inside a relative clause modifying a universally quantified NP, surfaces as a universally quantified expression with wide scope.

\(^4\)That is a derivation where quantifiers are interpreted in situ.
A third illustration of the fact that transsentential anaphora cannot be adequately treated by means of quantifying-in is provided by (12) and (13). It is not possible to analyze (12) by treating the full stop between the two sentences as conjunction and then quantifying in the phrase exactly one boy. For this yields the truth conditions of (13), as given in (15), whereas the truth conditions of (12) are rather those in (14).

(12) Exactly one boy walks in the park. He whistles.

(13) Exactly one boy walks in the park and whistles.

(14) \( \exists x (\forall y [(\text{boy}(y) \land \text{walkinpark}(y)) \rightarrow x = y] \land \text{whistle}(x)) \)

(15) \( \exists x \forall y [(\text{boy}(y) \land \text{walkinpark}(y) \land \text{whistle}(y)) \rightarrow x = y] \)

Examples (2), (3), (4), (5), (6), (7), (12) and (13) illustrate the need to extend the narrow conception of meaning as truth conditions to a more dynamic conception of meaning relative to context and a reconsideration of the traditional quantificational and bound variable approach to nominal anaphora on the intra- and intersentential level.

DRT provides a dynamic conception of meaning which is based on the observation that a human recipient of a discourse is able to process discourse on-line in an incremental fashion and the fact that new pieces of discourse are interpreted against the context established by the already processed discourse. In its original formulation on the one hand DRT tries to do justice to a conception prevalent in a number of AI, Cognitive Science and Linguistics [Fodor, 1975] approaches according to which the human mind can be conceived of as an information processing device and that meaning can best be viewed as an instruction to dynamically construct a mental representation which the mind can thus employ in further processing (such as theoretical and practical reasoning). On the other hand DRT is inspired by traditional truth-conditional-semantical approaches to meaning.

In DRT interpretation - i.e. the identification of meaning - involves a two stage process: first, the construction of semantic representations, referred to as Discourse Representation Structures (DRSs) from the input discourse and second, a model-theoretic interpretation of those DRSs. The dynamic part of meaning resides in how the representations of new pieces of discourse are integrated into the representation of the already processed discourse and what effect this has on the subsequent integration of the representations of subsequent pieces of discourse. Put differently, a new piece of discourse updates the representation of the already processed discourse and the meaning of a linguistic expression consists both in its update potential and its truth-conditional import in the resulting representation. This can be defined as a function which takes us from one context (available in terms of the already constructed representation) to a new context (the updated representation) which serves as context to discourse yet to come.

The dynamic view of meaning in terms of updates of representations and the attempt at
a rational reconstruction of the on-line and incremental character of discourse processing by human agents naturally leads to an algorithmic specification of DRS-construction in the standard formulation of DRT. To process a sequence of sentences $S_1, S_2, \ldots, S_n$ the construction algorithm starts with the first sentence and transforms it in a roughly top-down left-to-right fashion with the help of DRS construction rules into a $DRS_1$ which serves as the update context for the processing of the second sentence $S_2$ etc. The nature of the models in which such structures are interpreted depends on the natural language fragment covered. Before launching into a precise specification of the level of representation, the models and the construction procedure, we briefly and informally describe some of the basic tools which are characteristic for the DRT enterprise.

The level of representation in DRT is specified in terms of the language of DRSs. DRSs are pairs consisting of a set of discourse referents $U$ - often referred to as the "universe" of the DRS - and a set of conditions $Con$.

\[
DRS = \langle U, Con \rangle
\]

The DRS construction procedure maps sentence (1) into the DRS

\[
DRS = \{\{x\}, \{ \text{man}(x), \text{walkinpark}(x) \}\}
\]

which is often represented pictorially in the box notation

\[
\begin{array}{|c|}
\hline
x \\
\hline
\text{man}(x) \\
\text{walkinpark}(x) \\
\hline
\end{array}
\]

Here we will use both the box and the linear notation. The box notation ensures better readability especially in the case of complex DRSs while the linear notation is better suited for the formal definitions of the syntax and semantics of the language.

Briefly, the indefinite a man contributes the discourse referent $x$ into the universe of the DRS in (18) and the atomic condition $\text{man}(x)$ to its set of conditions. The VP walks in the park contributes the atomic condition $\text{walkinpark}(x)$\(^\text{5}\). The associated semantics ensures that the DRS will be true just in case there exists a mapping from the discourse referents in this DRS into a model such that all the conditions in the set of conditions will come out true. In this way discourse referents in the top box of a DRS are endowed with existential force.

The mini-discourse in (4) which constitutes a simple extension of (1) will give rise to the following anaphorically unresolved representation:

\(^\text{5}\)This is, of course, a simplification.
Informally, the personal pronoun he in the second sentence introduces a new discourse referent \( y \) into the universe of the DRS and an unresolved atomic condition \( y = ? \) into the set of conditions. The VP whistles gives rise to the atomic condition \( \text{whistle}(y) \). At this stage the DRS resolution component takes over and resolves the newly introduced discourse referent \( y \) against the previously introduced discourse referent \( x \). The resulting structure is

\[
\begin{array}{|c|c|}
\hline
x & y \\
\hline
\text{man}(x) & \\
\text{walkinpark}(x) & \\
y = ? & \\
\text{whistle}(y) & \\
\hline
\end{array}
\]

which is associated with the truth conditions which intuitively obtain of discourse (4).

Discourse referents have a double function. On the one hand they serve as antecedents for anaphoric expressions such as pronouns, on the other they act as the bound variables of a quantification theory. This second function entails that discourse referents must be able to stand to each other in certain scope relations. To mark these relations we must introduce the concept of a “SubDRS”: DRSs can occur as constituents of larger DRSs. As it turns out, this mechanism provides a natural explanation of the chameleonic quantificational import of indefinite NPs like those in (1), (6) or (7). SubDRSs always occur as part of complex DRS conditions. Two examples of complex DRS conditions are those involving implication and negation.

The conditional construction if \( S_1 \) then \( S_2 \) in (6) introduces a complex condition of the following form:

\[
\begin{array}{|c|c|}
\hline
S_1 & \Rightarrow & S_2 \\
\hline
\end{array}
\]

which consists of two DRSs joined by the \( \Rightarrow \) operator.

Similarly negation introduces a complex condition of the form

\[
\begin{array}{|c|}
\hline
\neg & S \\
\hline
\end{array}
\]
which contains a DRS as its subconstituent.

Sentence (6) gives rise to the following DRS

\[
\begin{array}{c|c|c}
|x, y| & |z, w| \\
\hline
\text{predro}(x) & \text{beat}(z, w) \\
\text{donkey}(y) & z = x \\
\text{own}(x, y) & w = y \\
\end{array}
\]

The truth conditions associated with (6) as represented by the predicate logical formula in (8) involve a wide scope universal quantification over the variable associated with a donkey. Intuitively the interpretation of the conditional sentence (6) says that whenever a situation obtains that satisfies the description provided by the antecedent of the conditional, then a situation as described by the consequent obtains as well. In other words, the consequent is interpreted and evaluated in the context established by the antecedent. The natural language paraphrase of the truth conditions associated with (6) expresses the universal force with which the indefinite a donkey in (6) is endowed. Furthermore, since the consequent is interpreted in the context set by the antecedent, the truth-conditional requirement that situations in which the antecedent is true be accompanied by situations in which the consequent is true is tantamount to situations of the former kind being part of (possibly more comprehensive) situations in which antecedent and consequent are true together. This is the informal justification of why discourse referents introduced in the antecedent of the conditional are available for resolution of anaphors in the consequent but not vice versa. It also explains why the universal quantifier expressed by the conditional is conservative in the sense of generalized quantifier theory. The conservativity of other natural language quantifiers follows in the same way. The DRS construction for a universal NP with a relative clause and an embedded indefinite NP like in (7) proceeds in a similar manner.

The semantics of conditional DRS conditions, then, is based on the principle that the interpretation of the antecedent can be extended to an interpretation of the consequent. This principle entails that a pronoun in the consequent can be interpreted as anaphoric to a constituent in the antecedent, i.e. the pronoun’s discourse referent can be linked to the one introduced by this constituent - or, as it is put in DRT, for the purposes of anaphora resolution the antecedent discourse referent must be accessible from the position of the pronoun. On the other hand, discourse referents from the consequent of a conditional are in general not accessible to pronouns in the antecedent. So there is an asymmetry in the accessibility relation here: discourse referents introduced by constituents in the antecedent are accessible to the consequent but not vice versa (unless they are allowed to “escape” to a higher position in the DRS, c.f. the discussion on proper names below). The accessibility relation turns out to play a central role in the DR-theoretical account of when anaphora is possible and when not. How DRS-constructors - which, like those of (21) and (22), create complex DRS conditions - affect accessibility, is an essential aspect of the semantic analysis of the natural language.

6In the jargon of the trade: the consequent is updated by the antecedent.
constituents (if .. (then) .. , not etc.) which they are used to represent. It can be argued, along lines similar to the argument we have given for conditionals above, that the discourse referents within the scope of a negation operator ¬ are not accessible from outside the Sub-DRS which is in the scope of the negation operator and similarly for discourse referents in the scope of a conditional operator ⇒ (again, unless they can “escape”, see below). As long as ⇒ and ¬ are the only complex DRS condition constructors, the accessibility relation can be graphically described in terms of the geometrical configurations of the box representation of the DRS language as going up and left.

The particular quantificational force of discourse referents in a DRS depends on the structure of the DRS. This structure determines, via its model theoretic interpretation, the quantificational import of discourse referents it contains. In this way indefinites are interpreted as referential terms which receive different quantificational import depending on where the discourse referents they introduce end up within the DRS. In a sense therefore, the role of quantifiers (variable binders) in traditional predicate logic or within the higher type Intensional Logic used in Montague grammar style representations, and in particular their scope and binding properties, has been replaced in DRT by the DRS universes, which in effect act as quantifier prefixes which bind all discourse referents they contain, and by the structure of DRSs which defines the scope and binding properties of these universe or discourse referent quantifiers.

We are now in a position to account for the difference between (2) and (3) manifest in (4) and (5). (2) and (3) will be mapped into the following representations, respectively:

(24) \[
\begin{array}{c}
x \\
\text{man}(x) \\
\text{walkinpark}(x)
\end{array}
\]

(25) \[
\begin{array}{c}
\neg \\
\begin{array}{c}
x \\
\text{man}(x)
\end{array}
\Rightarrow
\begin{array}{c}
\neg \\
\text{walkinpark}(x)
\end{array}
\end{array}
\]

(24) and (25) are truth-conditionally equivalent, as can be verified against the definitions in (33). However, (24) can be extended to an anaphorically resolved DRS

(26) \[
\begin{array}{c}
x, y \\
\text{man}(x) \\
\text{walkinpark}(x) \\
y = x \\
\text{whistle}(y)
\end{array}
\]

15
representing (4) while (25) can only be extended to the unresolvable

\[
\neg \frac{x}{\text{man}(x)} \Rightarrow \neg \text{walkinpark}(x)
\]

(27)

\[\begin{array}{c}
\text{whistle}(y) \\
y = ?
\end{array}\]

where unresolvability corresponds to the impossibility of the intended anaphoric relations as indicated in (5).

Finally let us consider the pair of sentences in (12) and (13). An analysis of sentence sequencing as conjunction together with a quantifying-in approach as the last step in the derivation would ascribe a complex property \(\lambda x. (\text{walkinpark}(x) \land \text{whistle}(x))\) to the quantifying NP exactly one boy in one fell swoop resulting in the formula \(\exists x \forall y [ (\text{boy}(y) \land \text{walkinpark}(y) \land \text{whistle}(x)) \rightarrow x = y ]\). In contrast, in the DRT approach a discourse referent \(x\) is set up by the NP in the first sentence and then incrementally constrained by the additions of further conditions. In this way we obtain the truth conditions which are associated with the predicate logic formula \(\exists x (\forall y [ (\text{boy}(y) \land \text{walkinpark}(y)) \rightarrow x = y ] \land \text{whistle}(x))\) which are those intuitively associated with (12).

In speaking of accessibility we have twice used the qualification “unless (the discourse referents) escape to a higher position”. What is meant by this is the following. In the examples shown so far the discourse referent introduced by a given sentence constituent (i.e. an NP) is always introduced in the universe of the DRS as part of which the constituent is interpreted by the DRS-construction algorithm. But in some cases this is not really adequate. Consider, for example, the DRS (23) for sentence (6). In (23) the discourse referent for Pedro is a member of the universe of the antecedent DRS. Under this representation the interpretation of (6) becomes “whenever there is a \(x\) who is Pedro and a \(y\) which is a donkey and \(x\) owns \(y\) then \(x\) beats \(y\)”. Although this cannot be considered as wrong as far as the truth conditions of (6) are concerned it fails to do justice to the intuition that in (6) the existence of Pedro is, unlike that of any donkeys he might own, presupposed: the use of the proper name Pedro, as opposed to, say, the phrase someone named “Pedro”, carries the implication that Pedro is already part of the available context.

To do justice to this intuition, DRT assumes that the discourse referents for proper names are always part of the highest DRS universe (the highest DRS universe contains those discourse referents which represent entities that can be considered as elements of the current context of interpretation, as it has been established by the interpretation of the already processed parts of the text). Note that there is a certain tension between the claim we just made that the use of a name presupposes that its bearer is already represented in the context,
and the stipulation that the name introduces a discourse referent representing its bearer into the context. This apparent contradiction can be resolved according to the following lines. Indeed, by using the name the speaker presupposes familiarity with its bearer, and thus that the bearer is (saliently) represented in the context. But this is a presupposition that can easily be accommodated: if, as far as the recipient knows, the bearer is not part of the context, then s/he will readily comply with what s/he takes to be the speaker’s assumptions concerning the context, and act as if the bearer is part of the context - i.e. the assumption will be accommodated. But accommodation of the name’s bearer amounts to introducing a discourse referent representing the bearer.

As this discussion implies we must distinguish between two kinds of occurrences of names, those where the bearer has not yet been introduced into the context and where accommodation is thus required and those where the bearer has been introduced and where the name occurrence acts as quasi-anaphorical: its discourse referent can be identified with the one that represents the bearer already. In “classical DRT” only the accommodation case was considered. The interpretation rule for this case is simple: introduce a discourse referent for the name at the highest DRS-level. Using this rule the DRS for (6) is not (23) but rather

\[
\begin{array}{c}
  x \\
  \text{predro}(x) \\
\end{array} \\
\begin{array}{c}
  y \\
  \text{donkey}(y) \\
  \text{own}(x, y) \\
\end{array} \Rightarrow \\
\begin{array}{c}
  z \ w \\
  \text{beat}(z, w) \\
  z = x \\
  w = y \\
\end{array}
\]

(28)

1.1.2 A Simple DRS Language and its Interpretation

Up to now we have given an informal presentation of some of the distinctive characteristics of DRT (such as discourse referents, resolution, notion of accessibility, the \(\Rightarrow\) operator, the construction algorithm etc.) for a simple extensional fragment and we have discussed some of the basic motivations for these distinctive characteristics.

Now we will give formal definitions for a simple DRS language, its interpretation, the notion of accessibility and the construction algorithm for the simple fragment considered so far. The fragment will then be gradually extended to other phenomena tackled with the DRT framework.

The vocabulary of the simple DRS language we have been looking at so far consists of

\begin{enumerate}
  \item[(i)] a set \(\text{Const}\) of individual constants
  \item[(ii)] a set \(\text{Ref}\) of discourse referents
  \item[(iii)] a set \(\text{Pred}\) of predicate constants
  \item[(vi)] a set \(\text{Sym}\) of logical symbols like \{=, \neg, \Rightarrow, \ldots\}
\end{enumerate}
The set of terms \( \text{Terms} = \text{Const} \cup \text{Ref} \). DRSs and DRS-conditions are defined by simultaneous recursion:

(i) if \( U \) is a (possibly empty) set of discourse referents \( x_i \in \text{Ref} \) and \( \text{CON} \)

a (possibly empty) set of conditions \( \text{con}_j \), then \( \langle U, \text{CON} \rangle \) is a DRS

(ii) if \( x_i, x_j \in \text{Ref} \), then \( x_i = x_j \) is a condition

(iii) if \( c_i \in \text{Const} \) and \( x \in \text{Ref} \), then \( c_i = x \) is a condition

(iv) if \( P \) is a \( n \)-place relation name in \( \text{Rel} \) and \( t_1, \ldots, t_n \in \text{Term} \),

then \( P(t_1, \ldots, t_n) \) is a condition

(v) if \( K \) is a DRS, then \( \neg K \) is a condition

(vi) if \( K_1 \) and \( K_2 \) are DRSs, then \( K_1 \lor K_2 \) is a condition

(vii) if \( K_1 \) and \( K_2 \) are DRSs, then \( K_1 \Rightarrow K_2 \) is a condition

DRS are defined in (i), atomic conditions in (ii), (iii) and (iv), and complex conditions in (v), (vi) and (vii).

The model-theoretic interpretation of the core DRS language defined above can be stated as follows: intuitively a DRS \( K = \langle U, \text{CON} \rangle \) can be conceived of as a “partial” model representing the information conveyed by some discourse \( D \); \( K \) is true if and only if \( K \) can be embedded into the “total” model \( M = \langle U, \Im \rangle \) in such a way that all the discourse referents in the universe \( U \) of \( K \) are mapped into elements in the domain \( U \) of \( M \) such that under this mapping all the conditions \( \text{con}_i \in \text{CON} \) in \( K \) come out true in \( M \). In other words \( K \) is true if and only if there is a homomorphism from \( K \) into \( M \). In DRT parlance, such a homomorphism is called a verifying embedding for \( K \) into \( M \). Models for the simple core DRT language defined above are simple extensional first-order models consisting of a domain \( U \) of individuals and an interpretation function \( \Im \) which maps constant names in \( \text{Const} \) into elements in \( U \), and \( n \)-ary relation names in \( \text{Rel} \) into elements of the set \( \text{P}(U^n) \).

The conception of a DRS \( K \) as a partial model makes straightforward sense only in those cases where all conditions of \( K \) are atomic. As soon as the DRS contains complex conditions, of the form (21), say, or of the form (22), the notion becomes problematic for the very same reasons that negation and implication are problematic in Situation Semantics. Take negation: should the condition

\[
\begin{array}{|c|c|}
\hline
y & \text{donkey}(y) \\text{own}(x, y) \\
\hline
\end{array}
\]

be understood as giving partial information in the sense that (the value of) \( x \) does not own any of the donkeys that can be found in some limited set or should it be taken as an absolute denial that \( x \) owns any donkeys whatever? The view adopted by classical DRT is that (31) is to be interpreted absolutely in the sense that an embedding (assignment) \( f \) with \( f(x) = a \) into a model \( M = \langle U, \Im \rangle \) verifies (31) iff there is no \( b \in U \) such that \( b \in \Im(\text{donkey}) \) and \( (a, b) \in \Im(\text{own}) \); or to put it into slightly different terms, and assuming that \( f \) is not defined for \( y \): \( f \) verifies (31) in \( M \) iff there is no function \( g[y]f \) (i.e. no extension \( g \) of \( f \) such that
\( \text{Dom}(g) = \text{Dom}(f) \cup \{ y \} \) which verifies

<table>
<thead>
<tr>
<th>( y )</th>
<th>donkey(y)</th>
</tr>
</thead>
<tbody>
<tr>
<td>( x )</td>
<td>own(x,y)</td>
</tr>
</tbody>
</table>

in \( \mathcal{M} \).

A similar verification clause is adopted for conditional conditions of the form \( K_1 \Rightarrow K_2 \), where \( K_1 = \langle U_{K_1}, \text{Con}_{K_1} \rangle \) and \( K_2 = \langle U_{K_2}, \text{Con}_{K_2} \rangle \) are DRSs. \( K_1 \Rightarrow K_2 \) is verified by \( f \) in \( \mathcal{M} \) iff for every \( g \) such that \( g[U_{K_1}]f \) which verifies \( K_1 \) there exists an \( h \) such that \( h[U_{K_2}]g \) such that \( h \) verifies \( K_2 \). Putting these considerations together we come to the following definitions of verification and truth:

\[
\begin{align*}
(\text{i}) & \quad h \models_{\mathcal{M}, g} \langle U, \text{CON} \rangle \text{ iff } h[U]g \text{ and for all } \text{con}; \in \text{CON} : h \models_{\mathcal{M}, h} \text{con}; \\
(\text{ii}) & \quad \models_{\mathcal{M}, g} x_i = x_j \text{ iff } g(x_i) = g(x_j) \\
(\text{iii}) & \quad \models_{\mathcal{M}, g} c_i = x \text{ iff } \exists c_i = g(x) \\
(\text{iv}) & \quad \models_{\mathcal{M}, g} P(x_1, \ldots, x_n) \text{ iff } \langle g(x_1), \ldots, g(x_n) \rangle \in \exists(P) \\
(\text{v}) & \quad \models_{\mathcal{M}, g} (\neg K) \text{ iff there does not exist an } h \text{ such that } h \models_{\mathcal{M}, g} K \\
(\text{vi}) & \quad \models_{\mathcal{M}, g} (K_1 \lor K_2) \text{ iff there is some } h \text{ such that } h \models_{\mathcal{M}, g} K_1 \text{ or } h \models_{\mathcal{M}, g} K_2 \\
(\text{vii}) & \quad \models_{\mathcal{M}, g} (K_1 \Rightarrow K_2) \text{ iff for all } h \text{ such that } h \models_{\mathcal{M}, g} K_1 \text{ there exists a } k \text{ such that } k \models_{\mathcal{M}, h} K_2
\end{align*}
\]

A DRS \( K \) is true in a model \( \mathcal{M} \) with respect to an assignment \( g \) iff

\[
\text{there exists a verifying embedding } h \text{ for } K \text{ in } \mathcal{M} \text{ with respect to } g. \\
\text{We write: } \models_{\mathcal{M}, g} K \text{ iff } h \models_{\mathcal{M}, g} K
\]

The effect of the definition of truth for a DRS in a model given in (34), together with the definition of a verifying embedding for DRSs (33), ensures that the discourse referents in the universe of a DRS which is not a constituent of any complex condition (often referred to as the main DRS while DRSs in complex conditions are referred to as SubDRSs) are interpreted as existentially quantified variables with widest possible scope. In the literature, this is sometimes referred to as existential closure of the main DRS. Clause (i) in (33) effects a conjunctive interpretation of the conditions in a DRS. Existential quantification and conjunction are thus absorbed into the structure of DRSs. Likewise, the definition of truth of a complex condition of the form \( K_1 \Rightarrow K_2 \) in (33) (vii) has the effect that the discourse referents in the universe of \( K_1 \) are universally quantified. In addition the definition ensures that because of the passing on to \( k \) of assignments of elements in \( \mathcal{U} \) to discourse referents in the top box of the antecedent in the conditional by \( h \), discourse referents in the consequent box can be "bound" by discourse referents in the universe of the antecedent box. This extended binding capacity\(^7\) accounts for the dynamics of this simple DRT language on the semantic level.\(^8\)

\(^7\)Compared to predicate logic.

\(^8\)Other parts of the dynamics of DRT are related to the construction procedure and the resolution
The DRSs in the first order fragment defined in (30) and (33) can be mapped straightforwardly into corresponding FOPL formulae in terms of a function $\phi^D$ defined by simultaneous recursion (following the clauses in definition (30) above):

\[
\begin{align*}
(i) & \quad \phi^D(\{x_1, \ldots, x_n\}, \{\gamma_1, \ldots, \gamma_m\}) := \\
& \quad \exists x_1 \ldots \exists x_n(\phi^D(\gamma_1) \land \ldots \land \phi^D(\gamma_m)) \\
(ii) & \quad \phi^D(x_i = x_j) := (x_i = x_j) \\
(iii) & \quad \phi^D(c_i = x) := (c_i = x) \\
(iv) & \quad \phi^D(\pi(t_1, \ldots, t_n)) := \pi(t_1, \ldots, t_n) \\
(v) & \quad \phi^D(\neg K) := \neg(\phi^D(K)) \\
(vi) & \quad \phi^D(K_1 \lor K_2) := \phi^D(K_1) \lor \phi^D(K_2) \\
(vii) & \quad \phi^D(\{\{x_1, \ldots, x_n\}, \{\gamma_1, \ldots, \gamma_m\} \Rightarrow K_2\}) := \\
& \quad \forall x_1 \ldots \forall x_n[\phi^D(\gamma_1) \land \ldots \land \phi^D(\gamma_m)) \rightarrow \phi^D(K_2)]
\end{align*}
\]

1.1.3 DRS Construction Algorithm, Accessibility and Resolution

In the formulation of [Kamp, 1981] and [Kamp and Reyle, 1993] DRS construction is defined algorithmically in terms of an iteration over a sequence of input sentences with an embedded recursion over reducible DRS conditions. Each iteration produces a syntactic analysis of the currently processed sentence in the input discourse sentence sequence, which is then inserted as a reducible condition into the appropriate place in the currently constructed DRS. The recursion defined over DRS conditions will then try to reduce reducible conditions until all conditions in the DRS have been reduced. If this is the case the algorithm will go into the next iteration step and process the next sentence in the input sentence sequence. The recursion over DRS conditions is defined in terms of DRS construction rules. DRS construction rules are specified in terms of triggering tree domain configurations (syntactic “templates”) and sequences of instructions to perform decorated tree-to-DRS transductions in the form of transformation operations. These transductions include sequences of deletion, insertion, substitution and where anaphoric elements can be resolved in more than one way, certain choice operations. The application of construction rules is ordered according to the configurational hierarchy of the parse tree which represents syntactic structure. In case there is no unique highest triggering configuration the construction algorithm is non-deterministic. Likewise, if a DRS contains more than one reducible condition the construction algorithm is non-deterministic. In the deterministic case, the specification of the DRS construction algorithm results in a top-down, left-to-right analysis tree traversal procedure. Non-deterministic settings will result in minor variations of this general processing strategy. Further details of the construction procedure will be described in section 2 Syntax-Semantics Interface below.

Anaphora resolution is partially constrained in terms of accessibility. The accessibility relation is a relation which holds between discourse referents and DRS.“positions”. It can be defined component.

\footnote{Strictly speaking in order to ensure that $\phi^D$ is functional we have to define it for a certain canonical order on the sets of discourse referents and conditions in a given DRS. The definition given above maps a DRS into a set of equivalent FOPL formulae.}
in terms of the configurational notion of subordination between DRSs.

A discourse referent \( x \) is accessible from the SubDRS \( K' \) of \( K \) iff there is a \( K'' \in SubDRS(K) \) such that \( x \in U_{K''} \) and \( K'' \) is subordinate to \( K' \).

We first define the set of SubDRSs for a given DRS \( K \):

\[
\text{(36) The set of SubDRSs of a DRS } K, SubDRS(K), \text{ is defined by the following, straightforward recursion: } SubDRS(K) \text{ is the smallest set such that}
\]

\[
\begin{align*}
(i) & \quad K \in SubDRS(K) \\
(ii) & \quad \text{if } K' \in SubDRS(K) \text{ and } K_1 \Rightarrow K_2 \in Con_{K'} \quad \text{then } K_1, K_2 \in SubDRS(K) \\
(iii) & \quad \text{if } K' \in SubDRS(K) \text{ and } \neg K_1 \in Con_{K'} \quad \text{then } K_1 \in SubDRS(K)
\end{align*}
\]

The notion of accessibility of a discourse referent from a certain DRS position can be defined formally in terms of subordination between the members of \( SubDRS(K) \): the discourse referent \( x \) is accessible from the SubDRS \( K' \) iff there is a SubDRS \( K'' \) such that \( x \in U_{K''} \) and \( K'' \) stands in the accessibility relation to \( K' \).

\[
\text{(37) The accessibility relation } Acc(K) \text{ on } SubDRS(K)^2 \text{ is defined as the reflexive transitive closure of the set of all pairs } (K', K'') \text{ such that either}
\]

\[
\begin{align*}
(i) & \quad \neg K'' \in Con_{K'} \text{ or} \\
(ii) & \quad K'' \Rightarrow K''' \in Con_{K'} \text{ for some } K''' \in SubDRS(K) \text{ or} \\
(iii) & \quad K' \Rightarrow K'' \in Con_{K''} \text{ for some } K'' \in SubDRS(K)
\end{align*}
\]

Anaphora resolution is defined in terms of the interplay between the sequential behaviour of the DRS construction procedure and the accessibility relation. In particular, the accessibility relation is only ever invoked for the condition currently processed by the DRS construction algorithm. Hence resolution only actually occurs between certain subsets of accessible discourse referents. To give a simple example: the definition of the accessibility relation as stated above makes all discourse referents in the same universe indiscriminately accessible to each other. The DRS construction procedure, however, effectively defines an ordering on the discourse referents in the same universe such that discourse referents associated with pronouns can only ever be resolved against previously introduced discourse referents.

In [Kamp, 1981] and [Kamp and Reyle, 1993] anaphora resolution is effectively constrained with respect to person, number and gender agreement. Recency, reiteration, parallelism, grammatical function compatibility and other constraints on saliency are modelled in a computational setting in e.g. [Asher and Wada, 1988].
1.1.4 Compositionality

Montague Grammar ([Montague, 1973], [Dowty et al., 1981] etc.) has arguably been the single most influential paradigm for formal semantics since its inception in the nineteen-sixties and early seventies. In Montagovian approaches natural language sentences are interpreted model theoretically and the meaning of a sentence is ultimately explicated in terms of its truth conditions.\(^\text{10}\) Intermediate levels of representation, such as Montague’s IL, are both constructed and interpreted compositionally and hence are in principle expendable. The mapping between the syntactic and the semantic algebra and the mapping between the semantic algebra and the model are homomorphisms. The combination of these two homomorphisms is again a homomorphism, hence the semantic algebra, though convenient, does not constitute a crucial ingredient of the theory.

DRT departs in several important respects from classical Montague Grammar. Apart from its dynamic conception of meaning and its focus on multi sentence discourses rather than on isolated sentences in the standard formulation of DRT the level of representations (i.e. the level of DRSs) has been regarded as an essential component of the theory. DRSs play a crucial part in the dynamics of the system in that such representations literally provide the context against which further sentences in a discourse are interpreted: the structure of DRSs determines the resolution potential of anaphors via the accessibility relation and antecedents for anaphors may be constructed (e.g. in terms of summation and abstraction operations in the construction of plural antecedents) for resolution purposes. From this perspective DRSs can be seen as data structures which in a very real sense are manipulated by computations effecting antecedent construction, anaphora resolution and (general) inferencing. This is in line with the procedural conception of meaning as an instruction to the recipient to construct a representation which is one the ingredients of the conception of meaning in DRT.

More abstract notions of discourse context are possible. A discourse context can, for example, be defined in terms of embedding conditions (of the corresponding DRS), update of a context in terms of a relational setting with input - output embeddings and the notion of truth can be defined from that. The dynamic conception of meaning in DRT together with its decidedly non-compositional setup has inspired (or rather provoked) the development of alternatives to standard DRT such as the approaches proposed by [Barwise, 1987b], Dynamic Predicate Logic (DPL) [Groenendijk and Stokhof, 1991a], Dynamic Montague Grammar (DMG) [Groenendijk and Stokhof, 1990], dynamic interpretation in higher order logic [Muskens, 1991], Dynamic Type Theory (DTT) [Chierchia, 1991a], constructive mathematics based approaches [Ranta, 1991] and (semi-) compositional reformulations of DRT like [Zeevat, 1989], [Muskens, 1994] and [Asher, 1993] to mention but a few. For all their differences, in contrast to the original formulation of DRT the approaches mentioned above subscribe to the Montagovian tenet of compositionality as a fundamental methodological (rather than an empirical) principle.

In the standard formulation the DRS-construction algorithm does not have an associated semantics. Apart from its intuitive appeal as an approximate rational reconstruction of on-line interpretation its justification is that it gets things right in sense that (i) the resulting representations are associated with the intuitively right truth conditions and (ii) the representations

\(^{10}\)More precisely as sets of possible world: that is functions from possible worlds to truth values.
together with the construction algorithm provide an account of the non-truthconditional discourse function of sentences in terms of a specification of an update potential of a sentence with respect to an already existing representation.

In compositional approaches each syntactic constituent in a derivation is associated with a semantic representation and the meaning of a constituent is a function of the meanings of its subconstituents. In the standard version of DRT an explicit semantics is given for only for fully constructed DRSs and their component parts, hence a semantics is given only indirectly for the syntactic component parts of a discourse. In contrast to the standard formulation of DRT in most of its compositional alternatives anaphora resolution is not part of the construction of semantic representations. Like in classical Montague Grammar it is assumed that anaphoric relations are fully determined by some other (usually unspecified) component of the grammar before syntactic structures are interpreted.

1.1.5 Extensions to the Basic Framework

In this section we look at some of the extensions to core DRT as outlined above not so much from the point of view of empirical coverage but from the point of view of the general import of these extensions on the setup of the theory.

1.1.5.1 Generalized Quantifiers

Clauses (i) and (vii) in (33) define an unselective notion of quantification for the simple fragment discussed above. Selective quantification is introduced in terms of generalized quantifiers represented in terms of duplex conditions of the form

\[(38) \quad K_1 \quad \Box \quad Q_x \quad K_2 \]

where $Q$ is interpreted as a generalized quantifier i.e. as relation between two sets and where the DRS $K_1$ on the left of the diamond denotes the set of objects $u \in \mathcal{U}$ which can be assigned to $x$ which satisfy $K_1$ while the DRS $K_2$ on the right of the diamond is interpreted as the set of of objects $u$ assigned to $x$ which satisfy both $K_1$ and $K_2$.\(^{11}\)

Definition (30) is “updated” by

\(^{11}\)Such quantifiers are referred to as conservative quantifiers or quantifiers which live on their first argument.
(39) (viii) if $R$ and $S$ are DRSs, then $R \rightarrow_{x} Q \rightarrow S$ is a condition

and definition (33) by

$$\begin{align*}
(33) & \quad \models_{M,j} R \rightarrow_{x} Q \rightarrow S \text{ iff } (A, B) \in \exists(Q) \\
& \quad \text{where} \\
& \quad A = \{ u \mid \text{there exists an } f(U_{R})(g \cup \{ \langle x, u \rangle \}) \text{ and } \models_{M,f} R \} \\
& \quad B = \{ u \mid \text{there exists an } f(U_{R})(g \cup \{ \langle x, u \rangle \}) \text{ and } \models_{M,f} R \} \\
& \quad \text{and there exists an } h(U_{S})f \text{ and } \models_{M,h} S \\
\end{align*}$$

If the determiner every is interpreted in terms of the generalized quantifier approach outlined above the "donkey-sentence" in (7) is no longer associated with the intuitively correct truth conditions derived from the unselective binding interpretation of the $\Rightarrow$ operator in clause (vii) in definition (33). Instead of quantifying over $(\text{farmer, donkey})$ pairs and requiring that for (7) to be true for all such pairs the first member beats the second member under the $\Rightarrow$ approach, in the generalized quantifier approach we quantify over donkey owning farmers and consider pairs of sets of the form \{(donkey owning farmer), (donkey owning farmer who beats this donkey)\} and under this approach (7) comes out true iff \{donkey owning farmer\} $\subseteq$ \{donkey owning farmer who beats this donkey\} which does not require that every farmer beats every donkey he owns.\(^{12}\)

In order to account for the "donkey sentence" type of examples like in (7) which involve anaphoric reference to an indefinite inside a relative clause modifying a universally quantified NP the truth conditions in (40) could be revised into

$$\begin{align*}
(40) & \quad \models_{M,j} R \rightarrow_{x} Q \rightarrow S \text{ iff } (A, B) \in \exists(Q) \\
& \quad \text{where} \\
& \quad A = \{ u \mid \text{there exists an } f(U_{R})(g \cup \{ \langle x, u \rangle \}) \text{ and } \models_{M,f} R \} \\
& \quad B = \{ u \mid \text{there exists an } f(U_{R})(g \cup \{ \langle x, u \rangle \}) \text{ and } \models_{M,f} R \} \\
& \quad \text{and for all } f(U_{R})(g \cup \{ \langle x, u \rangle \}) \text{ and } \models_{M,f} R \\
& \quad \text{there exists an } h(U_{S})f \text{ and } \models_{M,h} S \\
\end{align*}$$

However it has been argued that (40) and not (41) provides the right truth conditions for Most farmers who own a donkey beat it. Native speakers’ intuitions on these issues differ widely. DRT usually opts for the version in (41).

\(^{12}\)This is kind of like the proportion problem (c.f. [Kadmon, 1987]) in reverse.
1.1.5.2 Plurals

The DRT account of plurality is based on the plural count noun part of the lattice theoretical approach by [Link, 1983] and is particularly shaped by its central concern for plural pronominal anaphora. In particular it distinguishes between different types or sorts of discourse referents and provides simple inferencing mechanisms for the construction of plural antecedents.

Individual and group denoting discourse referents $x$ are marked as $at(x)$ and $nonat(x)$, respectively, atomic members of groups are represented as $x \in y$ and cardinality statements are expressed as $|x| = n$. A summation operation $\oplus$ is used to construct plural discourse referents from already introduced discourse referents. An abstraction operation $\Sigma$ is used to construct plural discourse referents from duplex conditions representing quantificational structures discussed in (39) and (41) above. The syntactic definitions in (30) and (39) are extended by the following clauses

\[
\begin{align*}
\text{(ix)} & \quad \text{if } x \in \text{Ref}, \text{ then } at(x) \text{ is a condition} \\
\text{(x)} & \quad \text{if } x \in \text{Ref}, \text{ then } nonat(x) \text{ is a condition} \\
\text{(xi)} & \quad \text{if } x, y \in \text{Ref}, \text{ then } x \in y \text{ is a condition} \\
\text{(xii)} & \quad \text{if } x \in \text{Ref}, \text{ then } |x| = n \text{ is a condition where } n \in \mathbb{N} \\
\text{(xiii)} & \quad \text{if } x, y_1, \ldots, y_n \in \text{Ref}, \text{ then } x = y_1 \oplus \ldots \oplus y_n \text{ is a condition} \\
\text{(xiv)} & \quad \text{if } x, y \in \text{Ref} \text{ and } K \text{ a DRS, then } x = \Sigma y : K \text{ is a condition}
\end{align*}
\]

Models for the core DRT language defined in (30) and (33) are simply pairs of unstructured sets with associated interpretation functions. The models for the extended DRT language are tuples of the form $\mathcal{M} = \langle \mathcal{U}, 3 \rangle$ where $\mathcal{U}$ is a complete, atomic, free, upper semilattice with a bottom element $\bot$. That is $\mathcal{U} = \langle \mathcal{U}, \leq \rangle$ where $\mathcal{U}$ is a set, $\leq$ a partial ordering relation on $\mathcal{U}$ such that for all $X \subseteq \mathcal{U}$ the least upper bound $\bigvee X$ exists ($\mathcal{U}$ is complete), for all $a, b \in \mathcal{U}$ if $a \not\leq b$ then there exists an atom $c$ such that $c \leq a$ and $c \not\leq b$ ($\mathcal{U}$ is atomic), for all $a \in \mathcal{U}, X \subseteq \mathcal{U}$, if $a$ is atomic and $a \leq \bigvee X$, then there exists a $b \in \mathcal{U}$ such that $a \leq b$ ($\mathcal{U}$ is free). Each such complete, atomic, free, upper semilattice with a bottom element $\bot$ is isomorphic to $\langle \mathcal{P}(A), \subseteq \rangle$ that is the structure defined by the powerset of some set $A$ and the subset relation $\subseteq$ defined on that set.

The interpretation for the new constructs listed in (42) can now be given as

\[
\begin{align*}
\text{(ix)} & \quad \mathcal{M}_g \models at(x) \text{ iff } g(x) \text{ is atomic element in } \mathcal{U} \\
\text{(x)} & \quad \mathcal{M}_g \models nonat(x) \text{ iff } g(x) \text{ is nonatomic element in } \mathcal{U} \\
\text{(xi)} & \quad \mathcal{M}_g \models x \in y \text{ iff } g(x) \leq g(y) \\
\text{(xii)} & \quad \mathcal{M}_g \models |x| = n \text{ iff } \{a | a \text{ is atomic element in } \mathcal{U} \text{ and } a \leq g(x)\} = n \\
\text{(xiii)} & \quad \mathcal{M}_g \models x = y_1 \oplus \ldots \oplus y_n \text{ iff } g(x) = \bigvee \{g(y_1), \ldots, g(y_n)\} \\
\text{(xiv)} & \quad \mathcal{M}_g \models x = \Sigma yK \text{ iff } g(x) = \bigvee \{a | a \text{ is an element in } \mathcal{U} \text{ and } \mathcal{M}_g, \{y,a\} \models R\}
\end{align*}
\]

Footnote: Examples illustrating the use of these conditions and operations will be given in section 3 Semantic Phenomena below.
1.1.5.3 Tense and Aspect

Temporal and aspectual phenomena in natural language did in fact provide one of the original motivations for the development of DRT [Kamp, 1979]. The DRT approach outlined below is inspired by and extends approaches based on [Davidson, 1967], [Reichenbach, 1947] and [Vendler, 1967].

The vocabulary of the extended DRS language for tense and aspectual phenomena consists of a set $Const$ of individual constants, a set $Ref$ of discourse referents, a set $Fun$ of one-place function symbols and a set $Rel$ of relation symbols:

(i) $Const = \{c_1, ..., c_n\}$ of individual constants

(ii) $Ref$ contains 5 different sorts of discourse referents which are distinguished typographically as follows:

- $Ind = \{x_1, ..., x_n\}$, a set of individual referents
- $Time = \{t_1, ..., t_n\}$, a set of referents for times
- $Event = \{e_1, ..., e_n\}$, a set of referents for events
- $State = \{s_1, ..., s_n\}$, a set of referents for states
- $Amount = \{mt_1, ..., mt_n\}$, a set of referents for amounts of time

(iii) $Fun$ is a set of 1-place function symbols which come in two sorts:

- $beg$, $end$, $loc$ are function symbols taking event or state discourse referents as arguments
- $dur$ is a function symbol taking event, state or time discourse referents as arguments

The set of terms consists of the set of discourse referents $Ref$ closed under application of $Fun$ as follows:

(i) if $d_i \in Ref$ then $d_i$ is a term
(ii) if $ev_i \in Event \cup State$ then $beg(ev_i)$ and $end(ev_i)$ are event denoting terms
(iii) if $ev_i \in Event \cup State$ then $loc(ev_i)$ is a time denoting term
(iv) if $\tau_i \in Event \cup State \cup Time$ then $dur(\tau_i)$ is an amount of time denoting term

The set $Rel$ of relation symbols contains

14This is further detailed by way of examples in sections 8.1 in Deliverable 9 Temporal Reference and 9.1 in Deliverable 9 Verbs (Aspect and Intensional) below.
1-place relation symbols taking individual discourse referents as arguments \( \{ R_1, \ldots, R_m \} \)

\( n \)-place event relation symbols taking event discourse referents \( e \) as referential arguments \( \{ e : R_1, \ldots, e : R_m \} \)

\( n \)-place state relation symbols taking state discourse referents \( s \) as referential arguments \( \{ s : R_1, \ldots, s : R_m \} \)

(46) \( n \)-place state relation symbols formed with a progressive \( PROG \) operator from \( n \)-place event relation symbols
\( \{ PROG(e : R_1), \ldots, PROG(e : R_m) \} \)

2-place relation symbols over events, states and times \( \{ <, \bigcirc, \sqsubset, \subseteq \} \)

1-place relation symbols over times \( \{ T_1, \ldots, T_n \} \)

1-place relation symbols over amounts of times \( \{ MT_1, \ldots, MT_n \} \)

The DRS-conditions and DRS are again defined in terms of a simultaneous recursion. Here we only list the conditions which are defined with elements of the new vocabulary introduced in (44) above.

\[
\text{(i) if } e_i \in \text{Event, } x_1, \ldots, x_n \in \text{Ind and } e : R \in \text{Rel an } n \text{-place event relation then } e_i : R(x_1, \ldots, x_n) \text{ is a condition} \\
\text{(ii) if } s_i \in \text{State, } x_1, \ldots, x_n \in \text{Ind and } s : R \in \text{Rel an } n \text{-place state relation then } s_i : R(x_1, \ldots, x_n) \text{ is a condition} \\
\text{(iii) if } s_i \in \text{State, } e_j \in \text{Event, } x_1, \ldots, x_n \in \text{Ind and } PROG(e : R) \in \text{Rel an } n \text{-place state relation then } s_i : PROG(e_j : R(x_1, \ldots, x_n)) \text{ is a condition} \\
\text{(iv) if } \tau, \sigma \in \text{Event} \cup \text{State} \cup \text{Time}, R \text{ a two place relation over events, states and times then } \tau R \sigma \text{ is a condition.} \\
\text{(v) if } \tau \in \text{Time}, R \text{ a one place relation over times then } R(\tau) \text{ is a condition.} \\
\text{(vi) if } \tau \in \text{Amount}, R \text{ a one place relation over amounts of times then } R(\tau) \text{ is a condition.} \\
\text{(vii) if } \tau \text{ and } \sigma \text{ are terms of the same type then } \tau = \sigma \text{ is a condition.} 
\]

The model theory for the DRT language defined above raises a number of foundational issues about the status of entities such as events, states, instants, intervals, their identity conditions and interrelations. Should events and states be regarded as first degree citizens in the model theory or should they be derivative notions defined in terms of e.g., instants and intervals or vice versa. A further option would be to take events as a primitive notion and define states as pre- and post-conditions of events. Alternatively one could assume states as the primitive notion and define events as transitions between states. These and still further options for temporal ontologies have been discussed in the literature. In what follows we can only point out some of the motivations for the design decisions in DRT model theory for tense and aspectual phenomena without giving due attention to the philosophical issues at stake.
States and events are prominent even if seriously underdetermined conceptualisations in the natural language metaphysical reconstruction of “reality”. In addition, an event-based semantics avoids some of the technical problems and conceptual inadequacies inherent in instant-based (e.g. with respect to iteration of tense operators) and interval-based (e.g. no definite truth values for certain intervals) temporal semantics approaches. The decision to have both states and events as first degree citizens in DRT model theory is motivated by the observation that natural language data do not seem to provide conclusive evidence to single out either of the two as somehow more basic.

In rough outline, the model theory for the language defined in (44) is based on two structures, which are assumed to be basic: first an *eventuality structure*\(^{15}\) \(\mathcal{E}V\) which contains both events and states and a precedence \(<\) and an overlap relation \(\bigcap\) defined on them; second a *time structure* \(\mathcal{T}\) isomorphic to the real numbers. The two structures are related in terms of a function \(\text{LOC}\) which “locates” eventualities by assigning time intervals to eventualities. To this end we define an *interval structure* \(\text{Int}(\mathcal{T})\) derived from \(\mathcal{T}\) and an *instant structure* \(\mathcal{I}(\mathcal{E}V)\) derived from \(\mathcal{E}V\).

(48) An eventuality structure \(\mathcal{E}V\) is a triple \(\langle EV, <, \bigcap\rangle\) with \(EV = E \cup S\) where \(E\) is a set of events and \(S\) a set of states. \(\mathcal{E}V\) satisfies:

\[
\begin{align*}
1. & \forall e_1 \forall e_2 ((e_1 < e_2) \rightarrow \neg (e_2 < e_1)) \\
2. & \forall e_1 \forall e_2 \forall e_3 ((e_1 < e_2 \land e_2 < e_3) \rightarrow (e_1 < e_3)) \\
3. & \forall e (e \bigcap e) \\
4. & \forall e_1 \forall e_2 (e_1 \bigcap e_2) \rightarrow (e_2 \bigcap e_1) \\
5. & \forall e_1 \forall e_2 (e_1 < e_2) \rightarrow \neg (e_2 \bigcap e_1) \\
6. & \forall e_1 \forall e_2 \forall e_3 \forall e_4 ((e_1 < e_2 \land e_2 \bigcap e_3 \land e_3 < e_4) \rightarrow (e_1 < e_4)) \\
7. & \forall e_1 \forall e_2 (e_1 < e_2 \lor e_1 \bigcap e_2 \lor e_2 < e_1)
\end{align*}
\]

The second irreducible component in the models for the extended DRT language is a *time (instant) structure* \(\mathcal{T}\) isomorphic to the structure \(\mathcal{R}\) of the real numbers:

(49) A time structure \(\mathcal{T}\) is a tuple \(\langle T, <\rangle\) where \(<\) is a dense linear ordering.

\(\mathcal{T}\) is a triple \(\langle T, <\rangle\) where \(<\) is a dense linear ordering.

From \(\mathcal{T}\) we obtain an *interval structure* \(\text{Int}(\mathcal{T})\):

(50) An interval structure \(\text{Int}(\mathcal{T})\) derived from a time structure \(\mathcal{T} = \langle T, <\rangle\) is a triple \(\langle \text{Int}, <_{\text{int}}, \bigcap_{\text{int}}\rangle\) where \(\text{Int}\) is the set of all convex subsets of \(T\) and \(\text{Int}(\mathcal{T})\) satisfies

\[
\begin{align*}
1. & \forall X \in \text{Int} \forall Y \in \text{Int} (X <_{\text{int}} Y) \iff \forall x \in X \forall y \in Y (x <_T y) \\
2. & \forall X \in \text{Int} \forall Y \in \text{Int} (X \bigcap_{\text{int}} Y) \iff X \cap Y \neq \emptyset \\
3. & \forall X \in \text{Int} \forall Y \in \text{Int} (X \subseteq_{\text{int}} Y) \iff \forall x (x \in X \rightarrow x \in Y)
\end{align*}
\]

\(\mathcal{E}V\) and \(\mathcal{T}\) are related indirectly in terms of a function \(\text{LOC}\) from \(\mathcal{E}V\) to \(\text{Int}\) which assigns

\(^{15}\)Eventuality is a cover term for events and states.
to each eventuality ev in EV a closed interval of T, that is an element i ∈ Int, which is the smallest closed interval which temporarily includes ev. To this end we first define an instant structure I(EV) derived from an eventuality structure EV:

\[ I(EV) \text{ is the instant structure } \langle I(EV), <, \Diamond \rangle \text{ derived from an eventuality structure } EV = \langle EV, <, \emptyset \rangle \text{ as follows:} \]

\( I_i(EV) \text{ if } \)

\( i \subseteq EV \)

\( ev_1, ev_2 \in i \Rightarrow ev_1 \cap ev_2 \)

\( i \subseteq H \text{ and for all } ev_1, ev_2 \in H, ev_1 \circ ev_2, \text{ then } H \subseteq i \)

\( (2) \text{ for all } i_1, i_2: i_1 < i_2 \text{ iff there are } ev_1 \in i_1, ev_2 \in i_2 \text{ such that } ev_1 < ev_2 \)

Intuitively, an instant i ∈ I(EV) is a maximal set of pairwise overlapping eventualities ev ∈ EV. <; is a strict linear order on I(EV).

With this LOC is defined as:

\[ LOC : EV \rightarrow Int \text{ such that:} \]

\( \forall ev_1 \in EV \forall ev_2 \in EV [(ev_1 < ev_2) \rightarrow LOC(ev_1) < LOC(ev_2)] \)

\( \forall ev_1 \in EV \forall ev_2 \in EV [(ev_1 \cap ev_2) \rightarrow LOC(ev_1) \cap LOC(ev_2) \neq \emptyset] \)

\( \forall i \in I(EV) \\cap \{LOC(e) | e \in i\} \neq \emptyset \)

LOC induces a homomorphic contraction \( T' = \langle T' , <, \Diamond \rangle \text{ of } T \text{ where } T' \text{ consists of the equivalence classes under the relation } \equiv_i \text{ defined by} \)

\[ t_i \equiv t_j \text{ iff } \forall e (t_i \in LOC(e) \rightarrow t_j \in LOC(e)) \]

We still need to define the part of the model which interprets amounts of time. Amounts of time can be identified with equivalence classes \([i]_{\equiv_{Int}}\) of intervals i of equal duration in Int(T). The equivalence classes generated by \( \equiv_{Int} \) are ordered by a weak linear ordering relation ≤:

\[ \text{for all } i, j \in Int [i] \leq [j] \text{ iff there exist } l, k \in Int \text{ such that } i \equiv l \subseteq k \equiv j. \]

≤ is a weak linear ordering:

\( [i] \leq [i] \)

\( [i] \leq [j] \land [j] \leq [i] \rightarrow [i] = [j] \)

\( [i] \leq [j] \land [j] \leq [k] \rightarrow [i] \leq [k] \)

\( [i] \leq [j] \lor [j] \leq [i] \)

The constraints in (54) (1)-(4) are entailed by
Models for the DRS-language defined in (44) are structures of the form $\langle U, \mathcal{E}V, \mathcal{T}, \text{LOC}, \equiv_{\text{Int}}, \exists \rangle$ where

(56) \[ \mathcal{M} = \langle U, \mathcal{E}V, \mathcal{T}, \text{LOC}, \equiv_{\text{Int}}, \exists \rangle \]

$\mathcal{M}$ is an interpretation function such that:

- for $c_i \in \text{Const}$, $\exists(c_i) \in U$;
- for $f_j \in \{\text{beg, end}\}$, $\exists(f_j) : \mathcal{E}V \to E$, $\exists(\text{loc}) = \text{LOC}$, $\exists(\text{dur}) = []^{\equiv_{\text{Int}}}$;
- for 1-place relation symbols $R_i$ taking individual discourse referents $x_j \in \text{Ind}$ as arguments, $\exists(R_i) \in \mathcal{P}(U)$;
- for $n$-place relation symbols $\epsilon : R_i$ taking event discourse referents $e_j \in \mathcal{E}V$ as referential arguments, $\exists(\epsilon : R_i) \in \mathcal{P}(E \times_1 U \times_2 \cdots \times_n U)$;
- for $n$-place state relation symbols $s : R_i$ taking state discourse referents $s_j \in \text{State}$ as referential arguments, $\exists(s : R_i) \in \mathcal{P}(S \times_1 U \times_2 \cdots \times_n U)$;
- for $n$-place state relation symbols $\text{PROG}(\epsilon : R_i)$ formed with a progressive $\text{PROG}$ operator from $n$-place event relation symbols of the form $\epsilon : R_i$, $\exists(\text{PROG}(\epsilon : R_i)) \in \mathcal{P}(S \times_1 U \times_2 \cdots \times_n U)$ such that $s_j, u_1, \ldots, u_n \in \exists(\text{PROG}(\epsilon : R_i))$ iff $\epsilon(e_k, u_1, \ldots, u_n) \in \exists(\epsilon : R_i)$ and $s_j \subseteq e_k$;
- for 2-place relation symbols over events, states and times $\{<, \bigcirc, \subset, \subseteq\}$, $\exists(<) = <_{\text{Int}}$, $\exists(\bigcirc) = \bigcirc_{\text{Int}}$, etc.
- for 1-place relation symbols over times $\{T_1, \ldots, T_n\}$, $\exists(T_i) \in \mathcal{P}(\text{Int})$;
- for 1-place relation symbols over amounts of times $\{MT_1, \ldots, MT_n\}$, $\exists(MT_i)$ a subset of the set of equivalence classes generated by $\equiv_{\text{Int}}$;
The denotation of terms relative to a model $\mathcal{M}$ and an assignment $g$ is defined in terms of an a function $\llbracket \cdot \rrbracket_{\mathcal{M},g}$:

\begin{align}
(57) \quad \text{(i)} & \quad \text{if } d_i \in R e f \text{ then } \llbracket d_i \rrbracket_{\mathcal{M},g} = g(d_i) \\
(\text{ii}) & \quad \text{if } d_i \in R e f \text{ and } f_j \in F u n \text{ then } \llbracket f_j(d_i) \rrbracket_{\mathcal{M},g} = \exists(f_j)(\llbracket d_i \rrbracket_{\mathcal{M},g})
\end{align}

The interpretation of the conditions specified in (47) is then defined as:

\begin{align}
(58) \quad \llbracket e_i \rrbracket_{\mathcal{M},g} : R(x_1, \ldots, x_n) & \iff \llbracket \llbracket e_i \rrbracket_{\mathcal{M},g}, [x_1]_{\mathcal{M},g}, \ldots, [x_n]_{\mathcal{M},g} \rrbracket \in \mathcal{S}(R) \\
\llbracket s_i \rrbracket_{\mathcal{M},g} : R(x_1, \ldots, x_n) & \iff \llbracket \llbracket s_i \rrbracket_{\mathcal{M},g}, [x_1]_{\mathcal{M},g}, \ldots, [x_n]_{\mathcal{M},g} \rrbracket \in \mathcal{S}(R) \\
\llbracket P R O G(e_i : R) \rrbracket_{\mathcal{M},g} & \iff \llbracket \llbracket P R O G(e_i : R) \rrbracket_{\mathcal{M},g}, [x_1]_{\mathcal{M},g}, \ldots, [x_n]_{\mathcal{M},g} \rrbracket \in \mathcal{S}(P R O G(e_i : R)) \\
\llbracket \tau \rrbracket_{\mathcal{M},g} & \iff \llbracket \llbracket \tau \rrbracket_{\mathcal{M},g}, [\sigma]_{\mathcal{M},g} \rrbracket \in \mathcal{S}(R) \\
\llbracket \tau \rrbracket_{\mathcal{M},g} & \iff \llbracket \llbracket \tau \rrbracket_{\mathcal{M},g}, [\sigma]_{\mathcal{M},g} \rrbracket \in \mathcal{S}(R) \\
\llbracket \tau = \sigma \rrbracket_{\mathcal{M},g} & \iff \llbracket \llbracket \tau \rrbracket_{\mathcal{M},g} = [\sigma]_{\mathcal{M},g} \rrbracket_{\mathcal{M},g}
\end{align}

1.1.5.4 Attitudes

DRT offers a variety of formal languages (languages the formulas of which are DRSs) which cover various parts of first or higher order logic. The familiar semantics for these languages is extensional. But it is straightforward to formulate, in analogy with Montague’s Higher Order Intensional Logic, an intensional model theory. In such a semantics a DRS will be assigned a set of pairs consisting of a world and a verifying embedding of the DRS in that world. One can then introduce the intension forming operator on DRSs and predicates and functors which take such intensional objects as arguments. In this way the intensional accounts of attitudinal contexts that are familiar from the work of Montague and others are readily available within a DR-theoretical framework.

However, the problems that are intrinsic to any intensional treatment of attitudinal concepts are too well-known to need rehearsing: Certainly some (and probably most) of those concepts are sensitive to distinctions at the sub-intensional level. This is so in particular with “explicit belief” and other attitudes which we take to be fairly directly connected with assent behaviour towards sentences. Thus, insofar as we take $x$’s explicit belief in what is expressed by $s$ (where $s$ is a sentence in a language that $x$ understands) as dependent on whether $x$ will assent when presented with $s$, then there is no reason to expect that whenever $s$ and $s'$ are sentences of such a language which are intensionally equivalent, then $x$ will explicitly believe what is expressed by $s$ iff he explicitly believes what is expressed by $s'$; and the propositional attitude literature contains plenty of examples which show that this conclusion does not follow.

As a matter of fact, the principal ideas behind DRT suggest a different approach. DRT differs from other forms of model-theoretic semantics in that it aims not only at being a theory of truth conditions but also a theory of interpretation. Its claim to being a theory of interpretation is substantiated primarily in the double role played by the formal devices it employs for describing meaning, viz the DRSs. DRSs serve simultaneously as descriptions of
the content (and in particular, where applicable, of the truth conditions) of what has been interpreted already and as context for the interpretation of what is to be interpreted yet. This double role, which has been summarized as the “unity of content and context”, imposes upon DRSs structural constraints which are not available within other semantic frameworks.

It is tempting to think that the structural features of DRSs which these constraints impose have a measure of cognitive reality and that the contents of at least some of our propositional attitudes are structured in just these ways. In fact, DRT contains a number of suggestions about the structure of attitudinal content and, importantly, of attitudinal change which appear to be extremely plausible, once they have been noted. Here are some of them:

(i) Different attitudes of a person \( x \) at any given time \( t \) - e.g. two different beliefs, or belief and a want, or a want and an intention - may be connected through sharing of discourse referents. Where this is so, the contents of the different attitudes are semantically (and even truth-conditionally) connected in ways reminiscent of the interconnectedness of the DRS components corresponding to the different sentences that make up a cohesive text. Thus two connected beliefs express a joint proposition that cannot be “factored” into a conjunction of two propositions that can be regarded as the individual contents of those beliefs; and similar complications, harder to analyze satisfactorily, arise when the attitudes are of distinct mode (as are, say, a belief and a want). It is evident and important that much of the practical reasoning which leads to the formation of goals and plans depends on the sharing of discourse referents between the different attitudes which such reasoning processes use as premises and conclusions.

(ii) New attitudes, e.g., new beliefs, typically get to be connected with attitudes (especially beliefs) that were part of the subject’s attitudinal state already. For the reason indicated under (i) such additions tend to produce a change in the content of those earlier attitudes.

(iii) Representations of entities - i.e. the discourse referents of the DRSs characterizing attitudinal states - acquire a certain attitude-internal identity through time. Such an entity can “change” over time in that the subject attributes properties \( t_0 \) at time \( t_1 \) which he did not attribute to it at time to (and yet it is the same entity)

(iv) similarly, propositional attitudes are capable of temporal persistence and, therewith, of change over time. Important among the changes are changes in strength and in mode. Thus a belief can change from one to which the subject attaches a certain degree of confidence into one in which he places either more or less trust. Another important change is the one that occurs when a belief that is represented as directed towards the present - a belief about how things are now, as we believe the contents of what we directly perceive or experience - turns into a belief about the past, something we do not longer experience but instead remember. (Some might argue that this is a case of one belief replacing another. What speaks against this is that typically in such cases it is not only the content of the attitude that remains the same but also what is known about its causal origins, about how one came to this information. There is a strong intuition here that that which is first linked to the time of awareness and then to some time preceding it is a token representation, whose identity is partly determined by its causal
connections with other parts of the subject's awareness and with the environment in
which he is embedded.) A third type of case of that where something which one plans
or expects becomes a reality - either through one's own action, as when one executes a
plan, or through forces not one's own - and then, as in the case just described becomes
something that has happened, lies behind one. Cases of this kind are interesting in
that they involve what appears to be a transition from an attitude that is purely de
dicto to one that is in an important sense de re: At first, when the attitude is one of
expectation, the expected event or situation is given only in terms of the descriptive
conditions under which it is conceived. But when the expectation is at last fulfilled it
is always an actual event or situation which does fulfill it and in most cases the subject
perceives this real event or situation as the one that makes his expectation true. Not
only does this turn his expectation into a belief - arguably this amounts to a change in
attitudinal mode - the content of the attitude is now linked, via its situation or event
discourse referent, to the external situation or event; and thus the belief is with respect
to this discourse referent, de re.

(v) Language is a means of communication, of transferring thoughts from speaker to addresssee. If we assume that the transferred thoughts are structured along the lines indicated,
then we are led also to the hypothesis that which a given statement by S provokes
in H has important structural similarities with the thought S has expressed. In par-
ticular, discourse referents that are structural constituents of S's thought, and which
will usually be realized in S's statement by corresponding noun phrases, will give rise
to corresponding constituents in the thought of H. It is a pervasive feature of verbal
communication that the corresponding thoughts of the participants not only have, as a
matter of fact, corresponding discourse referents as constituents, but that the partici-
pants actually take these discourse referents to be shared - it is this which lies at the
root of the possibility of discussing and disagreeing about things, even when they do
not actually exist.

The notational devices required to represent the aspects of attitudinal states hinted at in (i) -
(v) go in part beyond what is needed to represent texts. Before we go into some detail about
these devices, however, it is important to say a couple of things about the methodology of the
present approach. As should have been transparent from the above remarks, DRSs (or DRS-
like structures) will be used as complex descriptors (or predicates, if you prefer) of attitudinal
states. Thus the approach presupposes a substantially realist position towards attitudinal
states. These states are real, in the sense of being genuine bearers of such predications:
moreover, inasmuch as the DRS-predicates attribute to these states structure that may go
beyond what is directly reflected by the intensions determined by those predicates, the present
approach also treats those structural aspects of the states as real. It is clear therefore that
each additional notational device which we add to our repertoire of DRS-like state predicates
is likely to involve a new ontological commitment, and thus that each such addition must be
treated with circumspection.

A second important consequence of the approach is that as it stands it does not provide us with
a compositional account of attitudinal predicates (such as the English verbs believe or want) in
the traditional sense: believe, say, is no longer analyzed as a relation between individuals and
certain kinds of semantic objects (propositions or sentential intensions). Rather, sentences involving the verb believe are interpreted as predicking the DRS-like predicate induced by the complement of believe to the subject; the contribution of the verb believe itself then is to determine the mode of this attitudinal predicate (besides being instrumental in identifying the subject of the attribution (= the grammatical subject of the verb) and the time of the attribution (which is determined in the usual way as the time of the eventuality the verb describes.

Of course it is possible to derive from this semantics of attitude attributing sentences a relational account of belief: inasmuch as belief can be regarded as a relation between individuals and intensions, a belief attribution is true in this relational sense if the belief relation holds between the denotation of the subject phrase and the intension defined by the predicate. But of course we do not really want this analysis in general for the indicated (and familiar) reasons. One might hope that the present, DRT-based approach would show the way towards a propositional concept that is more refined (and thus unaffected by the familiar counterexamples to the intensional approaches) and yet well-motivated. We consider it unlikely, however, that there is any absolute, once-and-for all characterization of propositions as objects of the attitudes to be had. Because of the dynamics of attitudinal states - in particular, our tendency to recast and combine information at our disposal whenever it is activated in thought - propositional identity of the kind such a characterization would be after is too context-dependent and therefore too variable to allow for such a characterization.

We now turn to the specific representational devices which have been proposed within the context of the present approach to capture the different aspects of attitudinal states and their reports to which we alluded in (i) - (v).

First, we need to say something about the general set-up. As we said, attitude ascriptions to a subject x at a time t - are interpreted as attributions of certain structural properties to the attitudinal state of x at t. For instance, the sentence

\[(59) \quad \text{Fred believes that Susan isn’t at home.}\]

will be analyzed as saying that at the relevant time - let us make the default assumption that this is the utterance time t_o - the attitudinal state that Fred is in has as one of its components a belief whose content is completely or partly described by the state predicate\(^{16}\)

\[(60)\]

\[
\begin{array}{|c|}
\hline
s & y & z & n \\
\hline
\text{susan}(y) & \text{home}(z) & \text{is}(y, z) & n \subseteq s \\
\text{s : at}(y, z) \\
\hline
\end{array}
\]

\(^{16}\)Here the condition “n \subseteq s” is to express that the belief Fred has is “present tense” i.e. is directed towards his concept of the present; the indexical discourse referent is used to capture this direct sense of the present that under normal circumstance we all have.
Let us represent this component of Fred’s attitudinal state by means of the discourse referent \( p \). Then we need to express two things: (i) that \( p \) is part of Fred’s attitudinal state at \( t_0 \) - we will represent this by the relational condition \( \text{PAST}(f, p) \), where \( f \) is a discourse referent representing Fred - and (ii) that \( p \) is a belief characterized by (60) - we represent this by the condition

\[
\begin{array}{c}
\{ s \mid y \in s \\
\text{susan}(y) \\
\text{home}(z) \\
's(y, z) \\
n \subseteq s \\
s : \text{at}(y, z) \}
\end{array}
\]

Note that the present tense character of \( p \) is to be distinguished from the present tense of the verb believe in (59). Even when we turn (59) into the past tense sentence

\[
(62) \quad \text{Fred believed that Susan wasn’t at home.}
\]

the content of the belief can still be interpreted as present tense in the sense just elucidated even though the belief itself is now past rather than present. This latter distinction, between the presentness of the belief reported in (59) and the pastness of the belief reported in (62), we represent in the manner familiar from the treatment of tense in DRT, viz. by locating the state consisting in Fred having the belief lying around the utterance time in the first and before the utterance time in the second case. Thus (62) gets the representation:

\[
\begin{array}{c}
s' \quad f \quad p \quad n \\
\text{fred}(f) \\
n \subseteq s' \\
s' : \text{PAST}(f, p)
\end{array}
\]

\[
\begin{array}{c}
s \quad y \in s \\
\text{susan}(y) \\
\text{home}(z) \\
's(y, z) \\
n \subseteq s \\
s : \text{at}(y, z)
\end{array}
\]

It is important to distinguish between (a) the complete attitudinal state of Fred at the ut-
In (i) we argued that different (component) attitudinal states often share discourse referents. This phenomenon can be captured essentially without introducing any new notational devices. It suffices to allow the characterizing predicates of component states to have discourse referents in common. Thus $f$'s attitudinal state at $t_0$ may have two components, a belief that there is a valuable old US postage stamp in the album on the bookshelf and a desire to have this stamp. Using the same representational format we can represent this attitude as

\[
\begin{align*}
\text{(64)} & \\
& \begin{array}{c}
  \langle \text{BEL}, \text{DES}, \rangle \\
  \langle s', z \rangle \\
  n < s' \\
  s' : \text{have}(f, y)
\end{array}
\end{align*}
\]

In (64) $p$ represents a part of $f$'s attitudinal state which encompasses what we would normally consider two distinct components, a belief that the album contains a valuable old South African stamp and the desire to own “that stamp”. Note well: it is in principle possible for the attitudinal states characterized by representations such as (64) to be more specific, i.e. to contain more information, than the characterizing DRS predicate mentions. Thus $f$'s belief about the stamp may actually involve a notion precisely which stamp it is, how much it is
worth according to the current catalogues, etc.

Note that in (64) an explicit representation of the stamp occurs only in the description of the content of the belief: it is in that description that \( y \) occurs as part of the DRS universe; in the characterization of the desire \( y \) only occurs as argument of a condition, not as member of its universe. In the present case, where the complex involves a belief and a desire, this asymmetry is defensible insofar as it is the belief of there being such a stamp in the album that forms the cognitive basis for the desire, which in a sense is supervenient on it. In many other cases of shared discourse referents, e.g. when a discourse referent is shared between two beliefs, this kind of asymmetry seems unwarranted. In fact, in such cases it appears more plausible to see the representations of the relevant individuals as having a status that is separate from and neutral between the several attitudes that involve it, as in the following representation:

\[
\begin{array}{c}
s' \ s'' \ f \ p \ n \\
\text{fred}(f) \\
\text{old}(y) \\
\text{valuable}(y) \\
\langle \text{EXI}, \text{SouthAfrStamp}(y) \rangle \\
\langle \text{EXI}, \text{the album}(z) \rangle \\
\langle \text{BEL}, \text{on}(z,u) \rangle \\
\langle \text{BEL}, \text{the bookshelf}(f,u) \rangle \\
\langle \text{BEL}, \text{inside}(y,z) \rangle \\
\langle \text{DES}, \text{have}(f,y) \rangle \\
\end{array}
\]

There is a sense in which (65) represents the same total attitudinal content as (64), but it presents it as having a somewhat different fine structure. According to (65) the attitudinal
complex consists of:

(i) two attitudes of a type not yet encountered, which might be informally described as "assuming a thing of type $P$", where $P$ is some type or property;

(ii) two beliefs about the individuals assumed in the first two components - this may be felt to be a gratuitous deviation from what we have in (64), but when the existence assumptions are separated off in the way they have been in the present representation, the valuableness of the stamp and its being inside the mentioned album would seem to have a good claim to constituting distinct beliefs, for one thing, they are likely to spring from different sources of information.

(iii) a desire to have the represented stamp

Neither (64) nor (65) explicitly formalize the dependency relations between attitudes of different mode: the contents of our desires and other bouletic attitudes typically depend on our beliefs in that we tend to have desires only towards objects about which we have certain beliefs, but our beliefs do not depend in this same way on our desires. (Many of us are prone to certain forms of wishful thinking, but that is another matter.) It is not easy to make explicit in a model-theoretic context (using, say, some kind of possible worlds semantics) what these dependencies amount to, though some progress on this topic has been made in recent times (see [Heim, 1992], [Frank, 1994])

In addition to the dependence of bouletic attitudes on doxastic attitudes there typically also exist certain dependency relations between the doxastic attitudes themselves. For instance, beliefs to the effect that certain individuals have certain properties (especially contingent ones) or stand to each other in certain relations tend to depend on - or, in some sense, “presuppose” - the beliefs that these things actually exist. (65) - this is the principal difference between (65) and (64) - makes these existential beliefs explicit, and thus, though it does not clarify the relationship between them and the other (“predicational”) beliefs explicit as it stands, it offers the basis for such a clarification. If the relation between these existence assumptions and the predicational beliefs concerning the assumed individuals is indeed a relation of presupposition, then this could be made explicit by the devices that are used for representing presupposition generally (see below). (One notational possibility is that used in Situation Theory, where presuppositions are placed to the right of the descriptions of the propositions or situation types which presuppose them - a notation which there is used also to indicate the anchoring of parameters.)

The representation of an individual as existing necessarily requires that it be represented as being of some kind, i.e. it must be represented as having certain properties. But which? There can be no absolute answer to this question; what we consider an “identifying” property of a thing which we take to exist and what a “contingent” property of it, the attributing of which constitutes a separate attitudinal state or act, is to some extent a matter of perspective, which can shift from one context of thought to the next. In particular, the conditions included in the existence assumptions of (65) should not be regarded as being irrevocably tied to the discourse referents under which they are subsumed. In a different context one might think of
some of these conditions as detachable of the individuals that these conditions are predications of as identified via conditions that, in the present context, are treated as contingent. It is this flexibility which lies at the root of the so-called “cluster theory” of names (cf. [Searle, 1969]). Also it is responsible for some of the complexities of the concept of “guise” as it has been used in analysis of the paucity of belief and other attitudes.

There is much more that ought to be said about this approach, but we will, in part for reasons of space, leave matters as they now stand. We conclude with two brief hints at two areas in which the approach we have outlined is meant to be applied.

The first area is the semantics of those parts of natural language which we use to report or attribute propositional attitudes or describe their properties (e.g. how vivid or how irrevocable they are) and their relations to other things (such as, say, the relation in which certain beliefs stand to the perceptions from which they arise and to the things therein perceived; or the relations or their effect on action and the things which actions create or affect). As we said at the outset of this section, it is one of the central tenets of the present approach that a theory of attitude reports can be developed only on the basis of an underlying and largely independently motivated theory of attitudinal structure. But “attitude reports”, as this term is usually understood in extant philosophical and linguistic discussions of attitude semantics, constitute only a small part of the multifarious ways in which natural language refers to the attitudes and to their relations to each other and to the non-mental.

A second application area is the theory of verbal communication. There have been several hints of such a theory in earlier publications (See eg. [Kamp, 1983], [Kamp, 1990], [Kamp, 1992]), though a more fully worked out account is still missing. The idea common to the existing hints is to (i) expand the attitude theory we have sketched above into an account of how belief and other attitudinal states are modified through the processing of natural language input, and (ii) to expand this account further into a theory of shared knowledge, common ground and other aspects of intersubjectivity. The basics of an account of attitude modification are comparatively straightforward, as the attitude representations which our attitude theory postulates are well-nigh isomorphic to the DRSs which standard DRT postulates as content representations. This makes it possible to identify - at least in the simplest cases - the representation which embodies the new awareness of content arising through the processing of a verbal input with the DRS which standard DRT assigns to this input as its semantic representation.

The account of shared belief and other attitudes is another matter. Here there is need for fundamentally new concepts, concepts which DRT helps us to focus on, but which go nevertheless well beyond what is clearly implicit in the theory as it has been developed for other purposes. The most important insights in this area to date are - in our view - (i) that sharing is not confined to belief (or “knowledge”), but is at least as important for other attitudinal modes, such as e.g. goals, hopes, grief; (ii) that what is shared are not only full propositions but also some of their constituents, in particular “individuals”: in verbal communication (but also where other forms of interaction take place) the participants typically enter coordinated mental states; the coordination of these states involves the “sharing of discourse referents” in the following sense. Assume for simplicity that there are two participants, A and B and that their respective attitudinal states involve discourse referents \( x_A \) and \( x_B \). Then this pair of
discourse referents may qualify as shared between A and B if there exists a repertoire of referential devices - in the case of verbal communication foremost among these will be referential noun phrases - such that when e.g. A uses one of these devices to refer to the individual he has represented as $x_A$ then this device will produce in B a reference to the individual he has represented as $x_B$, (and similarly, of course, conversely). So, in particular, when A uses one of the referential NPs in the given repertoire to refer to (the individual he has represented as) $x_A$ then B, when processing this NP, will interpreting it as being about (the individual he has represented as) $x_B$ and take the predications A has expressed as predications of $x_B$.

1.1.5.5 Presuppositions

There are several reasons why DRT constitutes a natural setting for a theory of presupposition. First, as a theory of dynamic semantics, it is in a position to distinguish, at an abstract level, between contexts which satisfy the presuppositions of a given sentence and thus can be updated straightforwardly with the new information which that sentence contributes, and contexts in which some presupposition fails and which therefore cannot be updated without further ado. As was first pointed out by [Heim, 1983], presupposition failure may be seen as causing undefinedness of the “fundamental function of dynamic semantics”, the function $\lambda c.\lambda f.c[f]$, which maps pairs consisting of a context $c$ and a sentence $f$ onto $c[f]$, i.e. into the context “$c$ updated with $f$”. Indeed, it might be said that within a dynamic setting the task of presupposition theory is that of analysing the partiality of this function, a point of view that is especially transparent in recent work such as that of [Beaver, 1993].

A second aspect of DRT which makes it a useful framework for presupposition analysis is the degree of detail in which it represents contexts. This degree of detail makes it possible to not only to define abstractly what it means for a context to satisfy the presuppositions of a given sentence, but also to distinguish between different degrees in which a context may fall short of satisfying those presuppositions. It is often the case that part but not all of a complex presupposition is satisfied in a context, so that a comparatively modest accommodation will transform the context into one that is fit for the sentence in question. Such partial accommodations, which suffice when the context “meets the presupposition halfway”, tend to be somewhat easier for the recipient than whole-sale accommodation. For an example of this difference consider the following pair of mini-texts:

(66)  
(i) Kim is disappointed. The job went to another man.  
(ii) Sheila is disappointed. The job went to another man.

The phrase another man in the second sentences of (i) and (ii) carries the presuppositions

---

[17] The partiality of the function $\lambda c.\lambda f.c[f]$ will, when worked out in detail for some formal language or natural language fragment, generate a certain type of three-valued logic for this language. This established the obvious connection with the historically important logical approach to presupposition. We do not think, however, that the formal investigation of three-valued logic - e.g. by providing consistent and complete axiomatizations - has done much to improve our understanding of presupposition and from a linguistic perspective formal investigation of the three-valued logics generated by dynamic accounts of presupposition does not seem to us to be a particularly urgent priority.
that one or more men distinct from the value of the variable introduced by the NP another man itself are contextually salient. The context set by the first sentence of (ii) does not provide any such individual as it is and so the interpreter is invited to assume the existence of some relevant but not explicitly mentioned man or men (perhaps Sheila's husband, or father or some other male person related to her) who she had hoped would get the job. Such an accommodation is equally possible in the case of (i). But there it is also possible to assume that this other man is Kim himself, i.e. that Kim is used as the name of a male person here. And all things being equal there would appear to be a certain preference for this "cheaper" accommodation, which does not need to assume the existence of a new, as yet unmentioned individual but gets by with attributing certain compatible properties to an individual already introduced.

The treatment of presupposition within DRT has highlighted the close connections between presupposition and anaphora. This connection has been especially prominent in the work of Van Der Sandt (see [Sandt, 1989],[Sandt, 1992]) where presupposition is treated as a species of anaphora. The presuppositional aspects of anaphora had been clearly visible within DRT from the start. For instance, according to the standard DR-theoretical accounts of anaphoric pronouns, each such pronoun carries the presupposition that the context contain an antecedent discourse referent with which its own discourse referent can be identified. In this regard, the theory suggests, there is no fundamental difference between pronouns and other noun phrases that can be used anaphorically, such as, in particular, definite descriptions - the main difference between anaphoric pronouns and anaphoric descriptions being that the constraints which the latter impose on their antecedents usually have more descriptive content, which makes it easier to think of them as presuppositions in a more traditional sense, i.e. as certain existential propositions.18

Van Der Sandt goes further in urging that all presuppositions are anaphoric because they should all be seen as demands on the context (viz. as requirements that the context contain certain bits of information, which the presuppositions specify). Presupposition-generating expressions are anaphoric in that they impose such requirements. If the requirements are not met, the expression cannot be interpreted as things stand. In some cases repair is possible through accommodation. When accommodation is impossible, the expression remains uninterpretable.

While the emphasis on the anaphoric aspects of presupposition has been very fruitful, it should not blind us to the fact that important differences between the classical examples of presupposition and anaphora remain. Thus it is generally much easier to accommodate the presupposition carried by a definite description than that carried by a pronoun. To keep this observation in perspective, however, it should also be pointed out that the availability of accommodation also varies significantly within the narrower range of phenomena which have been considered instances of presupposition in the linguistic tradition of the past three decades, which thought of presupposition and anaphora as very different things. For instance, as has been pointed out e.g. by Kripke, it is much harder to accommodate the presupposi-

---

18It is ironic that pronouns and definite descriptions, which have long been considered the paradigmatic cases of anaphora and presupposition by a theoretical tradition in which presupposition and anaphora were treated as very different and unconnected phenomena, end up in the present theory with analyses which differ only in inessential details. For a detailed discussion of this approach see especially [Heim, 1982].
tions carried by “text-oriented” presuppositions such as also than it is to accommodate the presuppositions carried by many definite descriptions.

The first most striking advantage of the dynamic approach towards presupposition pertained to the projection problem. (i.e. the problem of predicting when a presupposition generated by an expression (e.g. by a clause) remains (survives) when the expression is embedded inside a larger one and when it does not (or gets “plugged”, as the jargon has it). As [Heim, 1983] makes clear, the principal cases of presupposition plugging in embedded sentence contexts - that where a presupposition generated by the consequent of a conditional gets plugged out by its antecedent and that where the first conjunct of a conjunction acts as plug to a presupposition generated by the second conjunct - fall out of the dynamic approach (common to her File Change Semantics and DRT) without the need of any special stipulations: In such situations presuppositions get plugged because they are satisfied in the local context which the antecedent of a conditional provides for its consequent, or the first conjunct of a conjunction provides for the second conjunct.

These advantages are equally apparent in the DRT-based presupposition accounts of Van Der Sandt and Zeevat [Zeevat, 1992]. In these accounts it is possible to represent a presupposition as part of the sub-DRS which contains the (representation of the) expression which triggers it and to use the resulting representation as the basis for verification (and if necessary and possible) accommodation of the presupposition. To see how this works, consider the following sentence (this is a slight variant of a much discussed example of Kripke’s):

(67) If John and Mary have pizza on Mary’s birthday, then they won’t have pizza again on John’s birthday.

The point of this example is that it strongly tempts the recipient to infer that John’s birthday is after Mary’s. It is clear, moreover, that this inference is connected with the presence of again, for when this word is eliminated from (67), the pressure to draw it is no longer there. It is also not hard to see that it is the need to justify the presupposition triggered by again which gives rise to the inference. We conclude this section on presupposition by reconstructing this inferential process. To this end we present a DR-theoretical treatment of (67), which combines the insights of Van Der Sandt with proposals made in [Kamp and Roßdeutscher, 1994a] about again. To make the connections with extant DRT transparent, we use the syntax and syntax-semantics interface of [Kamp and Reyle, 1993].

The DRS construction proposed in [Kamp and Reyle, 1993] for a future tense conditional like (67) begins with the introduction of a conditional condition, which contains syntactic analyses of if-clause and main clause in the antecedent and consequent box, respectively. As usual, the proper names John and Mary give rise to discourse referents in the universe of the main DRS. Let us, for the moment, make a similar assumption about the definite descriptions Mary’s birthday and John’s birthday. (We will return to the latter two NPs in the remarks concluding this section.) The pronoun they in the consequent of (67) is naturally interpreted

19As [Kamp and Reyle, 1993] admits, the syntactic treatment of tense, auxiliaries and temporal adjuncts it presents does not meet today’s syntactic standards and convictions; however its shortcomings do not bear on the points that matter here.
as referring to the set consisting of John and Mary. We represent this possibility as in [Kamp and Reyle, 1993], Ch.4, by introducing, using the Principle of Summation a discourse referent \( X \) to represent the mereological sum of these two individuals. Given these assumptions, the processing of the antecedent is unproblematic, so we present the antecedent DRS in its final form. As we are primarily concerned with the presupposition generated by \textit{again}, however, we will need to have a closer look at the syntactic structure of the main clause; so we begin by presenting the consequent box as it appears immediately after the conditional structure has been introduced:\(^{20}\)

\[
\begin{array}{|c|c|}
\hline
\text{\( j \) m \( X \) \( d_1 \) \( d_2 \)} \\
\hline
\text{\( john(j) \)} \\
\text{\( mary(m) \)} \\
\text{\( birthday(d_1) \)} \\
\text{\( s(j, d_1) \)} \\
\text{\( birthday(d_2) \)} \\
\text{\( s(m, d_2) \)} \\
\hline
\end{array}
\]

\[ X = j \uplus m \]

(68)

\[
\begin{array}{|c|c|}
\hline
\text{\( \epsilon_1 \)} \\
\hline
\text{\( n < \epsilon_1 \)} \\
\text{\( \epsilon_1 \subseteq d_1 \)} \\
\text{\( \epsilon_1 : have = \text{pizza}(X) \)} \\
\hline
\end{array}
\]

\[ \Rightarrow \]

(For simplicity \textit{have pizza} is treated here as a single verb.)

As argued in [Kamp and Roß deutsch, 1994a], with a presupposition trigger such as \textit{again}, it is a non-trivial problem to discover the principles according to which the presuppositions triggered by individual occurrences of \textit{again} are computed.\(^{21}\) In particular, what presupposition is generated is a matter of scope: the presupposition generated by a particular token of

\(^{20}\)For simplicity \textit{have pizza} is treated here as a single verb.

\(^{21}\)[Kamp and Roß deutsch, 1994a] argue this for the German equivalent of \textit{again} (the word \textit{wieder}), but the problem arises equally for \textit{again}.
again is to the effect that there has been a previous eventuality which satisfies all conditions that derive from clause material inside that token’s scope. Precisely how the scope of again is determined, appears to be a fairly complicated manner. Here we will simply assume that the scope is given by the syntactic position of again in the syntactic tree of the clause in which it occurs. For the syntactic tree in the right hand side sub-DRS of (68) this means that it is only the condition deriving from the “verb” have pizza that becomes part of the presupposition. On this assumption the presupposition can be represented as in

\[
\begin{array}{c}
e' \\
e' < e_2 \\
e' : \text{have} - \text{pizza}(X)
\end{array}
\]

where \(e_2\) represents the event described by the main clause of (67). The representation (69) is generated in tandem with the representation of the assertoric part of the main clause. (Again we omit details.) In the notation for (locally generated) presuppositions introduced by Van Der Sandt, this joint representation procedure for the assertoric content and the presupposition associated with the main clause transforms (69) into:

\[
\begin{array}{c}
j m X d_1 d_2 \\
\text{john}(j) \\
\text{mary}(m) \\
\text{birthday}(d_1) \\
'\text{s}(j, d_1) \\
\text{birthday}(d_2) \\
'\text{s}(m, d_2) \\
X = j \oplus m
\end{array}
\]

\[
\begin{array}{c}
e_1 \\
n < e_1 \\
e_1 \subseteq d_1 \\
e_1 : \text{have} - \text{pizza}(X)
\end{array}
\Rightarrow
\begin{array}{c}
e_2 Y \\
n < e_2 \\
e_2 \subseteq d_2 \\
Y = X \\
e_2 : \text{have} - \text{pizza}(Y)
\end{array}
\]

In (70) the presupposition (the part within the doubly framed lines) is located inside the sub-DRS within which it is triggered. Verification of this presupposition is now possible at any DRS level that is accessible form this sub-DRS. In the present case verification is “almost” possible at the level of the antecedent DRS, as there we find an event of the right type. The
only bit of information that is missing is that the event $e_1$ represented in the antecedent is before the time of the event $e_2$ described in the main clause. As Kripke rightly observed, in a case like this it is this condition which is naturally accommodated, so that we get presupposition justification at the “local” level of the antecedent (rather than “global” accommodation at the level of the main DRS. The result of this accommodation and, thus, the (indirect) justification of the presupposition, which is then no longer needed in the representation and therefore gets eliminated from it, is given in:

A treatment of this example that is even more in the spirit of the cited work of Van Der Sandt would specify the presuppositions associated with the different definite NPs that occur in (67). For the proper names John and Mary these presuppositions amount to the salient presence in the context of unique individuals going by these two respective names. There is much to say about these presuppositions, which ultimately goes to the heart of the philosophical discussions about how proper names refer, but this is not the place for that and we won’t have any more to say about these presuppositions here. The NPs Mary’s birthday and John’s birthday pose a different problem. Like familiar Christian names such as John and Mary, these NPs do not denote uniquely in any absolute sense, but the mechanism by which they achieve unique reference in context is quite different from those which govern the reference of tokens of ambiguous names. Descriptions of the form $x$’s birthday belong to a category which can be informally characterized as “functional NPs with tacit arguments”. In the case of $x$’s birthday ……

While we believe that the construal of (67) we have just given reflects the interpretation which most speakers will assign to this sentence when it is presented to them without further context, it is not the only one. There is at least one possible alternative, which consists in taking the presupposition generated by again to be that of there having been an earlier event of John and Mary having pizza on John’s birthday and thus to accommodate to a context for the main clause of (67) which contains the information that the two had pizza on some
previous birthday of John’s (most plausibly on his last birthday). That such an interpretation is possible becomes clear when we contemplate a situation of use for (67) in which it is part of the common ground that John’s birthday is before Mary’s.22

The possibility of construing the presupposition generated by again in the way just described shows two points. First, the computation of the presupposition generated by the token of a given presupposition trigger is not as straightforward as our discussion of the first interpretation of (67) may have suggested; in particular, it is not so clear in what way, if at all, the presupposition is determined by syntactic structure. Second, even accommodation of a presupposition triggered by again seems to be possible, and comparatively easy, in cases where virtually all the missing information needs accommodation. Therefore, the hypothesis that the availability of accommodation is a function of the quantity of information that needs to be accommodated is in need of reevaluation and/or refinement.

1.1.6 Lexical Semantics

As in any other theory of syntax and/or semantics, the lexicon is of central importance in DRT. This point deserves to be stressed, since in most published DRT-work the lexicon has been getting short shrift. Only recently have lexical questions connected with DRT been given some of the attention they merit c.f. [Kamp and Roß deutsc her, 1992], [Roßdeutscher, 1993], [Kamp and Roß deutsc her, 1994a], [Kamp and Roß deutsc her, 1994b].

Work on lexical semantics within DRT has been under way only for about four years. It has been motivated by a perspective that is at the root of DRT generally, viz. that the unit of semantic analysis is not the single sentence but the cohesive discourse or text and that the meaning of sentences, discourses and texts are manifest first and foremost through the inferences that they support. Since many of those inferences, and in particular many of those which are needed as part of the process of discourse interpretation itself, depend crucially on lexical information, it is a central task for a theory which (like DRT) has set itself the goal of analyzing this interpretation process, to determine what form and content lexical specifications must have so that they can support such inferences. Thus far investigations guided by this concern have already revealed many complexities in the meanings of, for instance, certain classes of verbs which had previously gone unnoticed within most alternative approaches to lexical semantics, simply because the questions of lexical meaning - i.e. of meaning at the subsentential level - had not previously been connected in any systematic way with semantic questions at the discourse, i.e. supra-sentential, level. In this second part of the present section we discuss a few lexical entries that have come out of these investigations - they all stem from work by Roß deutsc her - indicating in particular how they support certain inferences. Although this is a direction of research that is still in its infancy, we hope that the illustrations below give a hint of where these investigations mean to be going.

Our first example concerns the verbs empty and fill. Compare the two mini-texts (72) (a) and (72) (b):

22In order for the sentence to be interpreted in this way its spoken articulation would require a different prosody from the one which allows the interpretation represented in (71)!
(72) (a) Elmer filled the glass with gin. Then he emptied it.
(b) Elmer poured the gin out of the glass. Then he filled it again.

(72) (a) permits the inference that Elmer emptied the glass of gin. (72) (b), in contrast, does not allow us to infer that Elmer filled the glass with gin. The reason for this difference is intuitively clear - what goes in must come out again; but what has been taken out can be replaced by many kinds of other stuff. It should be equally clear that it is the lexical information associated with the verbs empty and fill that is responsible for this inferential difference and thus that the responsibility for sustaining the one inference while blocking the other lies with the lexicon.

The lexical entries for verbs (and other word classes) that have been used in DRT consist of two parts. The first part specifies the verb's syntax-semantics interface; this part states how many semantic arguments the verb has, which of these are syntactically obligatory, which optional and which syntactically blocked, and how the arguments get syntactically realized (i.e. with what case and/or what preposition). Recently there have been substantial efforts to systematize this kind of information and to discover the underlying linguistic generalizations (in particular those which govern the relationship between thematic roles and their syntactic realizations). This is a cluster of problems with which DRT has not so far been concerned; and indeed, the lexical proposals it has come up with have made no serious attempt to do justice to the systematicity that is to be found here. It should be stressed, however, that neither the spirit of the enterprise nor the actual entries that have been proposed preclude such a systematization.

The second part of the entry contains the specific semantic contribution that is made by the lexical item (here: verb) in question. For those lexical items which permit semantic specification in the form of lexical entries at all, this contribution comes in the form of a DRS-like structure, which gets introduced into the DRS for any sentence, discourse or text in which the item occurs (with the meaning captured by this particular entry). It is here that we should expect to find the difference between empty and fill which is responsible for the inferential discrepancy between (72) (a) and (72) (b).

We begin with the entry for fill:

\[
\begin{array}{|c|c|c|}
\hline
\text{fill} & \text{Nom} & \text{Acc} \\
\hline
\text{cc} & x & y \\
\hline
\end{array}
\]

\[
\begin{array}{|c|}
\hline
e' \\
\hline
\end{array}
\]

\[
\begin{array}{|c|}
\hline
\text{CAUSE} \\
\hline
\end{array}
\]

\[
\begin{array}{|c|}
\hline
\text{BECOME} \\
\hline
\end{array}
\]

\[
\begin{array}{|c|}
\hline
\text{Agent}(e') = x \\
\hline
\end{array}
\]

\[
\begin{array}{|c|}
\hline
\text{Theme}(e) = y \\
\hline
\end{array}
\]

\[
\begin{array}{|c|}
\hline
\text{e} \\
\hline
\end{array}
\]

\[
\begin{array}{|c|}
\hline
\text{P} \\
\hline
\end{array}
\]

\[
\begin{array}{|c|}
\hline
\text{s : full(y,z)} \\
\hline
\end{array}
\]

\[
\begin{array}{|c|}
\hline
\text{P(z)} \\
\hline
\end{array}
\]

\[
\begin{array}{|c|}
\hline
\text{Agent}(e') = x \\
\hline
\end{array}
\]

\[
\begin{array}{|c|}
\hline
\text{Theme}(e) = y \\
\hline
\end{array}
\]
Comments: The arguments of fill are listed at the top as $x$, $y$, and $P$. $x$ is the filler, $y$ the vessel and $P$ the stuff that is put into $y$. The specification "Nom", etc indicate how these arguments are realized. The parentheses around "with + Acc" indicate that this argument is syntactically optional. The use of the letter "$P$" for the third argument indicates that this is not an argument in the usual ("referential") sense. Rather, $P$ stands for the kind of stuff that the vessel is filled with. The motivation for representing the third argument in this way is that the natural complements of with in the context of fill are bare mass nouns and bare plurals (e.g. gin or marbles). Referential NPs are not impossible in this position - e.g. one can say: "He filled the glass with a half pint of milk." - but they seem marked and it is our view that their interpretation requires a form of coercion, along some such lines as "The glass was filled with milk, and it took half a pint to fill it." Those who are not persuaded by these considerations may prefer an entry which specifies a the third argument, like the two others, as a referential element. The issue is not crucial to the central points of the present discussion.

Besides the three arguments mentioned, the verb has a fourth argument $ec$, its so-called "referential argument". $ec$ stands for the event that the verb describes. In the semantic (lower) part of the entry this event is analyzed as a complex consisting of an action $e'$ by the Agent $x$ which causes the Theme $y$ to become filled; i.e. $e$ is a process which leads to $y$ being in a state of being full with stuff of the kind $P$.

After these explanations the entry for empty is almost self-explanatory:

\[
\begin{array}{cccc}
\text{empty} & \text{Nom} & \text{Acc} & (\text{of} + \text{Acc}) \\
ec & x & y & P \\
\hline
\end{array}
\]

\[
\begin{array}{c}
e' \ e \\
\hline
\hline
\end{array}
\]

Only one comment is needed here. The third argument of empty has been represented here by a discourse referent - $z$ - for an ordinary individual, and not by a discourse referent for a property. This difference with the entry for fill reflects the intuition that when a vessel is emptied there is always a definite quantity of stuff which it contains to begin with and it is that quantity which then gets taken out of it. Indeed, "He emptied the glass of the gin." seems less marked than "He filled the glass with the gin." It should be admitted, however, that the difference is a subtle one, here as in the case of fill, one may prefer an alternative treatment for the third argument. But again, the matter is of no consequence here.

Do these entries give us the difference we were after, i.e. do they support the inference licenced by (72) (a), but not the analogous, but unwanted, inference in (72) (b)? The answer is no.
To get the inference we want we need to know two things which the entries do not give us as they stand. The first of these is that \textit{BECOME} is "veridical", in the sense that when an event \textit{ec} of the kind described by the verb occurs, then a result state of the type that occurs as \textit{BECOME}'s second argument does in fact result. This principle is given by the following lexical axiom scheme:

\[
\begin{array}{c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c |
prestate of the process with the result state of the first. (Note that strictly speaking this is an inference by default; the assumption that nothing happened between the end of the first process and the beginning of the second which could have led to the glass being emptied and subsequently filled again, perhaps with some other stuff, is something that could in principle be overruled.)

The reason why a similar inference is not forthcoming in the case of (72) (b) can now easily be identified: whereas the representations of the prestate and of the process of emptying share the discourse referent \( z \), there is no such sharing between the representations of the prestate and the process of filling. Therefore we can in the first case transfer what we know about the stuff that is in the glass during the prestate, viz. that it is gin, to the third argument of the process, whereas in the second case the basis for a similar transfer is missing.

The axioms (76) and (77) present the occurrence of the prestates given on the right hand side of the arrow as ordinary entailments. However, the presence of such states is arguably a matter of presupposition, not of entailment. This can be seen when verbs like fill or empty are negated. For instance, in

\[(78)\] Elmer didn’t fill the glass

there is a clear presumption that at the time that the sentence is understood to be about the glass was empty (even if, as in many other cases of presupposition, this presupposition is easily overruled). If we want to treat prestates as presupposed by such verbs, then we will have to make this clear. The natural solution is to discard (76) and (77) and to modify the entries for empty and fill so that they explicitly contain the respective presuppositions. Using the notation introduced in section 1.1.5.5, we obtain e.g. instead of (73) the entry (79)

\[(79)\]

<table>
<thead>
<tr>
<th>empty</th>
<th>Nom</th>
<th>Acc</th>
<th>(of + Acc)</th>
</tr>
</thead>
<tbody>
<tr>
<td>ee</td>
<td>(x)</td>
<td>(y)</td>
<td>(P)</td>
</tr>
<tr>
<td>(s_0)</td>
<td>(s_0: \text{in}(z, y))</td>
<td>(\text{cc: }\epsilon)</td>
<td>(\text{ec: }\epsilon)</td>
</tr>
<tr>
<td>(\epsilon')</td>
<td>(\epsilon')</td>
<td>(\epsilon')</td>
<td>(\epsilon')</td>
</tr>
<tr>
<td>(\text{CAUSE empty }\epsilon)</td>
<td>(\text{become }\epsilon)</td>
<td>(\text{empty}(y, z))</td>
<td>(P(z))</td>
</tr>
<tr>
<td>Agent((\epsilon')) = (x)</td>
<td>Theme((\epsilon)) = (y)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The next entry we discuss is of the verb fetch. fetch is a verb that, as part of its lexical meaning, carries certain implications concerning the mental state of the agent of the event it describes. That natural languages have many such verbs is of course not exactly exciting news - for one
thing some such verbs, the so-called propositional attitude verbs, which refer explicitly to psychological states such as belief, desire, hope, etc. have long been the subject of intensive scrutiny, both within linguistics and within philosophy. Nevertheless the extent to which lexical meaning is spiced with intentionality has not, we believe, been sufficiently appreciated; nor has there been sufficient appreciation of how complex the intertwining between mental and non-mental aspects of meaning often is.

Events described by fetch are complex in the following sense. In order that we can say that $x$ fetched $y$ from $z$, it must have been the case that (i) $x$ went from some place $a$ to $z$; (ii) $x$ collected $y$ at $z$, and (iii) $x$ took $y$ from $z$ to some place $b$ (which often is, but need not be identical with $a$). But this is not all. Not every sequence of three events (i), (ii) and (iii) can be described as a case of fetch. In order that the sequence qualify for that it must have been the case that $x$ knew that $y$ was at $z$ and that he went to $z$ in order to pick up $y$ there and take it somewhere else - at least this must have been part of his motive for going to $z$ - and finally that $x$ engaged in these three events because of this motive; that they happened, we will say, as an execution of $x$’s plan to realize his goal of getting $y$ to $b$.

In order that the entry for fetch succeeds in capturing these aspects of its meaning, it must be able (i) to represent the pertinent propositional attitudes and (ii) to express the concept of certain real events happening as executions of these attitudes. It is important to note that especially (ii) imposes demands which semantic formalisms have thus far not been thought to be in need of meeting. Here we will have to make use of proposals to fill this hiatus within DRT that have not yet appeared in print, but yet without presenting all deals and motivations here. We will explain what we are doing as well as we can, as the concepts we will need in this connection get introduced.

But first things first. We must begin with an adequate description of the "real world aspect" of the realizations of fetch. An entry, set up along the lines exemplified above, which does just that is given in (80)

(80) fetch Nom Acc (from + Acc)

\[
\begin{array}{ccc}
 & x & y & z \\
 e_1 & e_2 & e_3 & a & b \\
 e_1 \supset & e_2 \supset & e_3 \\
 e_1 : & \text{move}(x, a, z) \\
 e_2 : & \text{take}(x, y) \\
 e_3 : & \text{move}(x + y, z, b) \\
\end{array}
\]

Comment: The use of the abutment relation ($\supset$) between the three events which make up $ec$ may be slightly too strong: there is no need for the taking to seemlessly follow upon the going to $z$, nor for the departure from $z$ to follow immediately after the taking. On the other hand the relation ($<$) of mere precedence seems too weak here. What is needed is for the events to be intentionally related as components of an over-all plan (part of which would be that the realization of each one component enables the execution of the next one. So a more adequate formulation of the relations between $e_1$, $e_2$ and $e_3$ presupposes connecting the events that actually happen with a plan of the agent, which is what we turn to now.
As we said, it is essential to the meaning of fetch that the component events are executed as part of a "plan to fetch y at z". This requires in the first place means to represent what it is to have a plan. But plans are a species of propositional attitude, like goals, desires, wishes, conjectures and beliefs. So the problem of how to represent (the having of) a plan is a subproblem of that of representing propositional attitudes in general. The mode of representation we adopt here is the one sketched in section 1.1.5.4, according to which x's having, say, a certain belief p at time t is represented as the existence at t of a certain state of affairs s to the effect x's psychological state has a component whose mode is BEL and whose content is p. Here the attitudinal mode is not that of belief, but that of, well, "having a plan" - we assume that having a plan is an attitudinal mode sui generis, which, while akin to desire and intention, should nevertheless be distinguished from those. One reason for this assumption is that plans are typically structured in special ways; this is part of what sets them aside as plans. Although the structures of individual plans can differ considerably from each other, there is one feature they all have in common: each plan has a goal, and a structure of events (some or all of which are actions by the agent whose plan it is), which are assumed to lead to the eventual realization of the goal. Both the goal and the "events" which the plan specifies as means are eventuality types, which we present in the form of DRSs with a distinguished eventuality discourse referent. More often than not the eventuality represented by the distinguished discourse referent of the goal is a state, although non-state-like eventuality types, e.g. kinds of activities, are possible goals too. In contrast, we assume that the eventuality types that figure in the plan as means are always types of events.

Here we consider only plans of a very simple structure, which specify a sequence of events $e_1, \ldots, e_n$ of types $E_1, \ldots, E_n$ such that either $e_n$ is itself a realization of the goal (this is so when the goal is the type of an action or activity that involves the agent himself), or else $e_n$ has a result state $s$ which instantiates the goal. For the special case of fetch the goal will be the state type which is instantiated by those states which are to the effect that the Theme $y$ be at the destination $b$ of the move event $e_3$ and the plan's means are events of the types specified of $e_1$, $e_2$ and $e_3$, respectively, in (81)

$$P \equiv \langle \text{Plan}, \rangle \quad \langle \text{MEANS}, \rangle \quad \langle \text{GOAL}, s : \begin{aligned} e_1 & e_2 e_3 s \\
 e_1 < e_2 < e_3 & \sqcup \subseteq s \\
 e_1 & : \text{move}(x, a, z) \\
 e_2 & : \text{take}(x, y) \\
 e_3 & : \text{move}(x + y, z, b) \\
 \text{Poss}(x, y) & \\
 \text{At}(x + y, b) \end{aligned} \rangle \rangle$$

In order that an event complex consisting of events of the three types represented in (80) qualify as a case of fetching the agent x must have had the attitude represented in (81). But as we argued, this is not enough; it must also be the case that the events that instantiate (80) were the result of x executing his plan (81). So we need a component to the entry for fetch
which makes this connection between the two parts (80) and (81) explicit.

The concept of plan execution is quite complex and deserves a much more extensive discussion than it can be given here. (It is a discussion that involves much of what is fundamental to the theory of action and thus it is of the highest importance and from this point is urgently needed; to our knowledge nothing that fully meets what we consider is needed here.)

We shall, then, be very brief. Let us concentrate on the component event $e_1$ of an actual event complex $ee$ realizing (80). Assume that the agent $x$ has, starting at some time before $e_1$, a plan $P$ of the form (81). That $e_1$ results from execution of $P$ is a complex relation between $e_1$ and $P$, to the effect that $x$ tried, when intentionally performing $e$, to realize the first "leg" of $P$ - i.e. the part specifying an event of the type $move(x, a, z)$ - and that he succeeded. There is, we repeat, much that ought to be said here. (Successfully) executing a particular part of a plan means having the intention to perform an action of the type this part specifies, letting this intention guide the action one performs and experiencing the action one performs as an implementation of this intention, so that, when the action is completed it is recognized as a successful realization of it. We simply introduce a new primitive - try, a 3-place predicate which relates individuals $x$, eventuality types $E$ and eventualities $e$, meaning that $e$ is an attempt of $x$ to realize $E$ - and express the complex relation of $e$ being a successful execution of $E$ by $x$ as the conjunction "$\text{try}(x, E, e) \land E(e)$", where the second conjunct expresses that $e$ is of type $E$. (The properties of the predicate try are left to some other occasion.) We extend the try predicate to apply to sequences of eventuality types and sequences of eventualities in the obvious way - e.g. $\text{try}(x, \langle E_1, E_2, E_3 \rangle, \langle e_1, e_2, e_3 \rangle)$ stands for the conjunction $\text{try}(x, E_1, e_1) \land \text{try}(x, E_2, e_2) \land \text{try}(x, E_3, e_3)$ and, another straightforward extension, to sequences of eventualities and plans, where the plan contributes the sequence of eventuality types that constitute its means. By the same token we write, e.g. $\langle E_1, E_2, E_3 \rangle(\langle e_1, e_2, e_3 \rangle)$ to express that the eventualities $e_1, e_2, e_3$ are of the types $E_1, E_2, E_3$ respectively, and similarly interpret $P(\langle e_1, e_2, e_3 \rangle)$.

Using these conventions we can tie the components (80) and (81) of our entry for fetch as follows:

\[
\begin{array}{cccccccc}
(82) & \text{fetch} & \text{Nom} & \text{Acc} & \text{(from + Acc)} \\
& ee & x & y & z & s & P
\end{array}
\]
More work will be needed before we can be confident that this is the most suitable form in which to code the intentional aspects of the meaning that the verb fetch evidently has. It is also not clear at this point, whether other lexical items the meanings of which involve intentional components can be adequately represented with the formal machinery we have sketched. The matter of lexicalized intentionality appears to us to be of such importance, however, that it justifies the quite extensive discussion we have just given to the one example considered here.

1.1.7 Inferencing

In order to obtain a proof system for the first order DRS language presented in section 1.1.2 A Simple DRS Language and its Interpretation, in principle, it is sufficient to map DRS representations into their corresponding FOPL representations as outlined in (35) above and then employ any of the standard calculi developed for FOPL. [Kamp and Reyle, 1991] develop a sound and complete proof system for a first order DRS language which operates directly on DRS representations and obviates the detour through FOPL.\(^{23}\)

Arguments are structured texts. In order to capture the transsentential connections - like

\(^{23}\)Several other proof systems for first order DRS languages have been proposed in the literature. Among them are : [Sedogbo and Eytan, 1988], [Koons, 1988], [Reinhard, 1989], [Saurer, 1993], [Reyle and Gabbay, 1994].
for example pronominal reference - which obtain between sentences in the premise text and sentences in the conclusion text, in DRT logical consequence \( \vdash_{\text{DRS}} \) is defined as a relation between a DRS \( K \) and some extension \( K' \) of \( K \) (thus \( K \subseteq K' \)).²⁴

The definition of logical consequence is as expected. It makes use of a few syntactic definitions. For a given DRS \( K \) we define the set of declared discourse referents \( \text{decl}(K) \):

\[
\begin{align*}
(i) \quad & \text{decl}(\langle U_K, \text{Con}_K \rangle) := U_K \cup \bigcup_{\gamma \in \text{Con}_K} \text{decl}(\gamma) \\
(ii) \quad & \text{decl}(x_i = x_j) := \emptyset \\
(iii) \quad & \text{decl}(P(x_1, \ldots, x_n)) := \emptyset \\
(iv) \quad & \text{decl}(\neg K) := \text{decl}(K) \\
(v) \quad & \text{decl}(K_1 \vee K_2) := \text{decl}(K_1) \cup \text{decl}(K_2) \\
(vii) \quad & \text{decl}(K_1 \Rightarrow K_2)) := \text{decl}(K_1) \cup \text{decl}(K_2)
\end{align*}
\]

(83)

For a given DRS \( K \) the set of free discourse referents \( \text{free}(K) \) is:

\[
\begin{align*}
(i) \quad & \text{free}(\langle U_K, \text{Con}_K \rangle) := \left( \bigcup_{\gamma \in \text{Con}_K} \text{free}(\gamma) \right) - U_K \\
(ii) \quad & \text{free}(x_i = x_j) := \{x_i, x_j\} \\
(iii) \quad & \text{free}(P(x_1, \ldots, x_n)) := \{x_1, \ldots, x_n\} \\
(iv) \quad & \text{free}(\neg K) := \text{free}(K) \\
(v) \quad & \text{free}(K_1 \vee K_2) := \text{free}(K_1) \cup \text{free}(K_2) \\
(vii) \quad & \text{free}(K_1 \Rightarrow K_2)) := \text{free}(K_1) \cup (\text{free}(K_2) - U_{K_1})
\end{align*}
\]

(84)

The set of discourse referents \( \text{dr}(K) \) is simply \( \text{dr}(K) = \text{decl}(K) \cup \text{free}(K) \). A DRS \( K \) is proper if \( \text{free}(K) = \emptyset \). A DRS \( K \) is pure if it does not contain otiose declarations of discourse referents:

\[
\text{A DRS } K \text{ is pure if for every two distinct DRSs } K_1 \text{ and } K_2 \text{ such that } K_1 \text{ is a sub-DRS of } K_2 \text{ and } K_2 \text{ a sub-DRS of } K, U_{K_1} \cap U_{K_2} = \emptyset.
\]

(85)

With these definitions²⁵ in place logical consequence is defined as

\[
\text{For } K \text{ and } K' \text{ pure (but not necessarily proper) DRSs: } K \vdash_{\text{DRS}} K' \text{ holds iff for every model } M = \langle \mathcal{U}, \mathcal{T} \rangle \text{ and function } f \text{ from } U_K \cup \text{free}(K) \cup \text{free}(K') \text{ into } \mathcal{U} \text{ such that } f_{M, f} K, \text{ there is a function } g[U_K]f \text{ such that } f_{M, g} K'.
\]

(86)

The DRT proof system is modelled on the FOPL proof system developed by [Kalish and Montague, 1964]. The syntactic consequence relation \( \vdash_{\text{DRS}} \) is defined in terms of rules of

²⁴Cases where a conclusion DRS is not an extension of a premise DRS are simply reduced to the extension case: \( K' \) is a consequence of \( K \) just in case the merge \( K' \cup_{\text{DRS}} K \) is a consequence of \( K \).

²⁵In the formulation of the deduction rules we will need to take recourse to two further notions: alphabetic variant of a DRS \( K \) and homomorphic copy (embedding) of a DRS \( K \) in a DRS \( K' \). Here the reader will be spared the precise definitions and is referred to [Kamp and Reyle, 1993].
proofs and inference rules. Rules of proof come in two types: direct and indirect rules of proof. Direct proofs do not involve any subproofs while indirect ones do. The system has one direct rule of proof RDP (Rule of Direct Proof) and two indirect rules of proof CP (Conditional Proof) and RAA (Reductio Ad Absurdum). Inference rules apply to some DRS K and extend it to a DRS K' with $K \subseteq K'$. The system without disjunction and identity involves three inference rules DET (Detachment - also referred to as GMP (Generalized Modus Ponens)), DNE (Double Negation Elimination) and NEU (Non-Empty Universe). The full system with disjunction and identity features four additional inference rules MTP (Modus Tollendo Ponens), DI (Disjunction Introduction), SoI (Substitution of Identicals) and SI (Self-Identity). Soundness and completeness theorems relating $\models_{\text{DRS}}$ and $\vdash_{\text{DRS}}$ are proved.

Proofs involve the introduction and cancellation of "show-lines" $\text{Show: } P$ where P is the proposition one wants to prove. When P has been proved the show-line is "cancelled":

$$\text{Show: } P.$$

DRSs with show-lines are referred to as extended DRSs.

The Rule of Direct Proof states that a DRS or DRS-condition $\delta$ is proved if an alphabetic variant $\delta'$ of $\delta$ occurs as part of the DRS which contains the show-line $\text{Show: } \delta$:

$$\text{(RDP Rule of Direct Proof): if a DRS } K \text{ contains a show-line } \text{Show: } \delta \text{ and if } K \text{ contains } \delta' \text{ where } \delta' \text{ is an alphabetic variant of } \delta, \text{ the show-line may be cancelled.}$$

Direct Proofs are proofs involving RDP and the inference rules.

The rule of Detachment applies to conditional structures of the form $K_1 \Rightarrow K_2$ in a DRS K. It states that provided it is possible to homomorphically embed $K_1$ into K we can add an alphabetic variant of $K_2$ to K such that the discourse referents in the variant of $K_2$ do not already occur in K:

$$\text{(DET Detachment (GMP Generalized Modus Ponens)): Given a DRS K, if } K_1 \Rightarrow K_2 \in \text{Com}_K \text{ and if there is a homomorphic embedding } f(K_1) \text{ into } K, \text{ then we may add an alphabetic variant } g(K_2) \text{ to } K \text{ where } g[U_{K_2}], f - f \text{ is one-to-one and } g \text{ maps } U_{K_2} \text{ to a set of discourse referents that do not already occur in } K.}$$

$$\text{(87) and (88) can be illustrated with the following example. In order to show that}$$

$^{26}$In fact this formalises a “backward chaining” strategy.
we add a show-line with the conclusion to the premise in (89):

![Diagram showing the logical structure]

Since the left hand side of the conditional DRS condition in (90) can be homomorphically embedded in the main DRS we can apply DET (88) and add an alphabetic variant of the right hand side of the conditional DRS-condition which extends the homomorphic embedding of the left-hand side to the main DRS.

![Diagram showing the logical structure]

We can now apply RDP to (91) and cancel the show-line, completing the proof of (89):
The rule of Double Negation Elimination applies to structures of the form

In simple cases it amounts to the cancellation of two negation signs. In more complex cases where $K_1$ contains conditions other than $\neg K_2$ DNE can be applied provided that $K_1 - \langle \emptyset, \{\neg K_2\} \rangle$ has a homomorphic embedding in $K$.

DNE Double Negation Elimination: if $\neg K_1 \in \text{Con}_K$ and $\neg K_2 \in \text{Con}_{K_1}$ and $f(K_1 - \langle \emptyset, \{\neg K_2\} \rangle)$ a homomorphic embedding in $K$, then $g(K_2)$ may be added to $K$ where $g[U_{K_2}]f$, $g - f$ is one-to-one and $g$ maps $\text{dec}(K_2)$ to a set of discourse referents new to $K$.

The rule of Non-Empty Universe states that we only consider models with non-empty universes. This means that we can always introduce discourse referents at the highest level of the DRS.

Disjunction is treated in terms of two inference rules: Modus Tollendo Ponens and Disjunction Introduction. Modus Tollendo Ponens states that given a DRS with a disjunctive condition together with the negation of an alphabetic variant of one of the disjuncts we may add a disjunctive condition to the DRS which is like the original disjunction except that the disjunct corresponding to the negated condition is missing.
MTP Modus Tollendo Ponens: given an DRS $K$ with a disjunctive condition of the form $K_1 \lor \ldots \lor K_{i-1} \lor K_i \lor K_{i+1} \lor \ldots K_n$ and a condition of the form $\neg K'_i$ where $K'_i$ is an alphabetic variant of $K_i$ we may add $K_1 \lor \ldots \lor K_{i-1} \lor K_{i+1} \lor \ldots K_n$ to $K$.

Disjunction Introduction permits us to introduce any disjunctive condition into a DRS if the DRS already contains one of the disjuncts.

DI Disjunction Introduction: if $K_i$ is included in $K$ then we may add $K_1 \lor \ldots \lor K_{i-1} \lor K_i \lor K_{i+1} \lor \ldots K_n$ to $K$.

The proof system features two inference rules pertaining to identity: Substitution of Identicals and Self-Identity. We state them without further comment.

SoI Substitution of Identicals: if $K$ contains conditions $x = y$ and $\gamma$ where $x, y \notin \text{decl}(\gamma)$, we may add a condition $\gamma'$ to $K$ where $\gamma'$ results from $\gamma$ by replacing one occurrence of $x$ by $y$.

SI Self-Identity: if $K$ is a DRS, for any $x \in U_K$ we may add $x = x$ to $K$.

As stated the inference rules apply at the level of the “main” DRS only. It can be shown, however, that conveniently the application of the inference rules can be extended to embedded DRSs and furthermore that every proof in the thus extended proof system is also provable in the old system.

The inference rules described above are based entirely on the premise DRS and basically extend the premise DRS until RDP can be applied. Proofs based on RDP and the inference rules are referred to as direct proofs. They do not involve any intermediate proofs and do not introduce any new temporary assumptions. In addition to direct proofs the system features two rules of proof for indirect proofs involving sub-proofs and the introduction of temporary assumptions. The rules are the rule of Conditional Proof and Reductio ad Absurdum.

The rule of Conditional Proof is applied in proofs of conditional structures of the form $K_1 \Rightarrow K_2$ in some premise DRS $K$. It introduces a sub-proof which on the assumption that an alphabetic variant of $K_1$ holds tries to derive a variant of $K_2$. The sub-proof may make use of what is asserted in the premise DRS $K$. If the sub-proof is successful, $K_1 \Rightarrow K_2$ is established and the sub-proof together with the assumption discarded.

CP Conditional Proof: if $\text{Con}_K$ in a premise DRS $K$ contains a show-line $\text{Show}$: $K_1 \Rightarrow K_2$ we may introduce a sub-proof

$$K \quad \| \quad \begin{array}{c} K'_1 \\ \text{Show}: K'_2 \end{array}$$
where $K'_1$ and $K'_2$ are alphabetic variants of $K_1$ and $K_2$, respectively. When the show-line in the sub-proof is cancelled, the show-line $Show: K_1 \Rightarrow K_2$ in the premise DRS $K$ may be cancelled as well.

Suppose we want to show

$$
\begin{array}{c}
\frac{x}{P(x)} \quad \Rightarrow \quad Q(x) \\
\frac{y}{Q(y)} \quad \Rightarrow \quad R(y)
\end{array}
\quad \vdash_{\text{DRS}}
\begin{array}{c}
\frac{z}{P(z)} \quad \Rightarrow \quad R(z)
\end{array}
$$

we add the conclusion in a show-line to the premise DRS and apply CP.

$$
\begin{array}{c}
\frac{x}{P(x)} \quad \Rightarrow \quad Q(x) \\
\frac{y}{Q(y)} \quad \Rightarrow \quad R(y)
\end{array}
\quad ||
\frac{z}{P(z)} \quad \Rightarrow \quad R(z)
$$

Now we can apply DET twice: from $(z, P(z))$ in the CP sub-derivation and the first condition in the premise DRS we get $Q(z)$ and from $(z, Q(z))$ together with the second condition in the premise DRS we get $R(z)$.

$$
\begin{array}{c}
\frac{x}{P(x)} \quad \Rightarrow \quad Q(x) \\
\frac{y}{Q(y)} \quad \Rightarrow \quad R(y)
\end{array}
\quad ||
\begin{array}{c}
\frac{z}{P(z)} \quad \Rightarrow \quad R(z)
\end{array}
\quad \frac{z}{P(z)} \quad \Rightarrow \quad R(z)
\quad \frac{Q(z)}{R(z)}
$$

Now RDP may be applied to the CP sub-derivation cancelling the show-line $Show: R(z)$. 

60
According to the CP rule we may now also cancel the show-line in the premise DRS completing the proof of (101).

The final rule of proof, Reductio ad Absurdum, also opens up a new sub-derivation in which we try to show that the assumption $\neg K_1$ where $K_1$ is a goal in a show-line in the premise DRS leads to a contradiction $\bot$ thus establishing $K_1$.

\begin{equation}
(104) \text{ RAA Reductio ad Absurdum: if } Con_K \text{ of some premise DRS } K \text{ contains a show-line } Show: K_1 \text{ we may introduce a sub-proof}
\end{equation}

\[
K \parallel \begin{array}{|c|}
\hline
\neg K_1 \\
Show: \bot \\
\hline
\end{array}
\]

When the show-line in the sub-proof is cancelled, the show-line $Show: K_1$ in the premise DRS $K$ may be cancelled as well.

\subsection*{1.1.8 Underspecification}

Looked at in isolation, i.e. without reference to any one particular context, natural language sentences (discourses) display a remarkable degree of ambiguity with respect (but not limited) to quantifier scope, distributive, collective, cumulative and generic readings of plural NPs, the scope of negation, coordination, N-bar modification, prepositional phrases, category polymorphism, anaphoric dependencies to mention but a few. Traditionally, this ambiguity is accounted for by mapping a single ambiguous natural language expression into a set of unambiguous formal representations each representing a distinct reading of the expression in question.\textsuperscript{27} Further contextual information may then be used to eliminate readings from this set. The traditional set-up sketched above is not entirely satisfactory from at least two points of view: first, from a computational perspective, the determination of the set of alternative readings might result in a combinatorial explosion; second, from a psycho-linguistic point of view, it does not seem to be an adequate model of human natural language understanding. Underspecified semantic representations are designed to alleviate (avoid) some of these problems.

Expressions in a semantic representation language usually serve a dual purpose: first, they are associated with a formally explicit interpretation (and thus indirectly associate natural language expressions with explicit interpretations) and second, they provide the basic objects manipulated in a syntactic calculus designed to capture the inferential properties of (sets of) such expressions (and thus help to model the inferential potential of natural language expressions). Underspecified semantic representations are most useful when they are associated with a proof theory which operates \textit{directly} on the underspecified representations.\textsuperscript{28}

\textsuperscript{27}For quantifier scope phenomena see for example the "quantifying in" mechanism in Montague Grammar or the "Cooper-storage" mechanism.

\textsuperscript{28}If the proof theory operates on disambiguated semantic representations the complexity problem is simply shifted from the construction of the representation to (the interface to) the associated reasoning component.
Reyle [Reyle, 1993] has developed a sound and complete proof theory for a first order language of Underspecified Discourse Representation Structures (UDRSs). The language allows underspecified representations with respect to the scope inducing operators $\Rightarrow$ and $\neg$. The associated proof theory operates directly on the underspecified representations without considering cases.

The basic idea underlying underspecified representations can best be explained by way of example. In standard DRT scope relations are fully determined in terms of a total ordering on the structural configurations (corresponding to the subordination relation between the boxes) in a DRS. The two readings of the quantificationally ambiguous sentence

(105) Every student passed an exam.

are represented by

(106) $\begin{array}{c}
\text{student}(x) \\
\text{student}(x)
\end{array} \Rightarrow \begin{array}{c}
y \\
\text{exam}(y) \\
\text{pass}(x, y)
\end{array}$

and

(107) $\begin{array}{c}
y \\
\text{exam}(y) \\
\text{student}(x)
\end{array} \Rightarrow \begin{array}{c}
\text{pass}(x, y)
\end{array}$

In (106) the universally quantified NP takes wide scope over the indefinite while (107) represents the specific reading of the indefinite NP. A scopally underspecified representation of (105) somehow has to capture what is common to both (106) and (107) and at the same time leave underspecified the subordination relations between the semantic representations of the scope inducing elements in (105). In both (106) and (107) the semantic contribution of the verb passed is subordinated to the the semantic contributions of the two NPs which in turn are (in the case of the indefinite in (107) trivially) subordinated to the main DRS. (106) and (107) differ with respect to the subordination relation between the two NPs. Graphically what is common to both (106) and (107) can be represented as follows:
(108) depicts the UDRS associated with (105). In general, a UDRS is an upper semi-lattice with a top element $\top$. The lattice represents a partial ordering relation between DRS conditions. Textual definitions of UDRSs are based on a labeling (indexing) of DRS conditions (where the labels index the boxes in the corresponding fully specified DRSs) and an explicit statement of a partial ordering relation between the labels.

The language of UDRSs consists of a set $L$ of labels, a set $\text{Ref}$ of discourse referents, a set $\text{Rel}$ of n-place relation symbols and a set $\text{Sym}$ of logical symbols. It features two types of conditions:

1. if $l \in L$ and $x \in \text{Ref}$ then $l : x$ is a condition
   
   if $l \in L$, $R \in \text{Rel}$ a n-place relation and $x_1, \ldots, x_n \in \text{Ref}$
   
   then $l : P(x_1, \ldots, x_n)$ is a condition

2. if $l_i, l_j \in L$ then $l_i : \neg l_j$ is a condition

   if $l_i, l_j, l_k \in L$ then $l_i : l_j \Rightarrow l_k$ is a condition

   if $l, l_1, \ldots, l_n \in L$ then $l : \forall (l_1, \ldots, l_n)$ is a condition

(109)

If $l_i, l_j \in L$ then $l_i \leq l_j$ is a condition where $\leq$ is a partial ordering defining an upper semi-lattice with a top element.

UDRSs are pairs of a set of type 1 conditions with a set of type 2 conditions.

A UDRS $\mathcal{K}$ is a pair $\langle \mathcal{L}, \mathcal{D} \rangle$ where $\mathcal{L} = \langle L, \leq \rangle$ is an upper semi-lattice of labels and $\mathcal{D}$ a set of conditions of type 1 in (109) above such that if $l_i : l_j \in \mathcal{D}$ then $l_j = l_i \in \mathcal{L}$ and if $l_i : l_j \Rightarrow l_k \in \mathcal{D}$ then $l_j \leq l_i, l_k \leq l_i \in \mathcal{L}.$

The construction of UDRSs, in particular the specification of the partial ordering between labelled conditions in $\mathcal{L}$, is constrained by a set of meta-level constraints (principles). The constraints place upper and lower bounds on the ordering relations. They ensure, e.g., that verbs are always subordinated with respect to their scope inducing arguments, that scope sensitive elements obey the restrictions postulated by whatever particular syntactic theory is

29The definitions in (109) and (110) abstract away from some of the complexities in the formal definitions of the UDRS language. For the full definitions the reader is referred to [Reyle, 1993].

30The full language also contains type 1 conditions of the form $l : \sigma(l_1, \ldots, l_n)$ indicating that $(l_1, \ldots, l_n)$ are contributed by a single sentence.

31This simply closes $\mathcal{L}$ under the subordination relations induced by complex conditions of the form $\neg K$ and $K_i \Rightarrow K_j$.

63
adopted, that potential antecedents are scoped with respect to their anaphoric potential etc. Below we list a few examples:

- **Clause Boundedness**: the scope of genuinely quantificational structures is clause bounded. If $l_q$ and $l_{cl}$ are the labels associated with the quantificational structure and the containing clause, respectively, then the constraint $l_q \leq l_{cl}$ enforces clause boundedness.

- **Scope of Indefinites**: indefinites labelled $l_i$ may take arbitrarily wide scope in the representation. They cannot exceed the top-level DRS $l_{\tau}$, i.e. $l_i \leq l_{\tau}$.

- **Proper Names**: proper names, $\pi$, always end up in the top-level DRS, $l_{\tau}$. This is specified lexically by $l_{\tau} : \pi$

A further principle, the Closed Formula Principle, ensures that discourse referents in argument positions are properly bound. To state the principle we define partial functions $\text{scope}$ and $\text{res}$ on the set of labels in a UDRS $K = \langle L, D \rangle$:

(111) if $(l : -l_i) \in D$ then $\text{scope}(l) = \text{res}(l) = l_i$

if $(l : l_i \Rightarrow l_j) \in D$ then $\text{res}(l) = l_i$ and $\text{scope}(l) = l_j$

if $(l : \forall(l_1, \ldots, l_n)) \in D$ then $\text{scope}(l) = l_i$ for some contextually determined $l_i$

With this the Closed Formula Principle is defined as

- **Closed Formula Principle**: a verb always takes narrow scope with respect to its arguments. If $l$ labels a UDRS condition containing a verb and $l_1 \ldots l_n$ labels labeling its arguments then $l \leq \text{scope}(l_1), \ldots, l \leq \text{scope}(l_n)$.

During the construction of a semantic representation subordination constraints from a variety of sources (syntactic, contextual and global constraints etc.) are simply added successively. The process is monotonic in that (i) addition of further information to an underspecified representation may reduce the set of its readings and (ii) the construction of a semantic representation does not involve any destructive manipulations.

The definition of truth for UDRSs as detailed in [Reyle, 1993] is defined in terms of a disambiguation function for UDRSs mapping a UDRS into a disjunction of fully specified DRSs. The UDRS in (108), for example, is mapped into the disjunction of the two DRSs in (106) and (107). A UDRS is true if and only if one of its disambigurations is. The definition of truth guarantees that the meaning of an underspecified representation corresponds to the disjunction of the meanings of its fully specified DRSs.

The semantic consequence relation $\models_{UDRS}$ for UDRSs is defined classically. It is reflexive, transitive and monotonic.
Reyle [Reyle, 1993] defines a corresponding syntactic consequence relation \( \models_{UDRS} \) which operates directly on the underspecified representations without the need to consider disambiguated cases. Soundness and completeness theorems relating \( \models_{UDRS} \) and \( \models_{UDRS} \) are proved. In rough outline the UDRS-Calculus follows the DRS-Calculus developed in [Kamp and Reyle, 1991] discussed in section 1.1.7 above and extends it to underspecified representations. It distinguishes between direct and indirect inference rules. Direct rules essentially permit an extension of the representation of the premisses but do not involve any sub-proofs. Indirect rules are used to prove goals by way of intermediate sub-proofs.

A goal is proved by means of a direct proof if its representation is embeddable into the premiss set. Direct proofs involve the following inference rules: NeU (non-empty universe), DET (detachment), COLL, DIFF, EFQL (ex falso quod libet), DNE (double negation elimination), MTP (modus tollendo ponens) and DI (disjunction introduction). We will sketch a few of these below.

The NeU rule states that a universe can be extended by any finite collection of discourse referents. It reflects the assumption that only models with non-empty universes are considered.

The rule of detachment DET is a generalisation of modus ponens. Given a DRS \( K \) with a condition of the form \( K_i \Rightarrow K_j \), if \( K_i \) can be embedded into \( K \) by a function \( f \), then we may add \( K'_j \) to \( K \) where \( K'_j \) results from \( K_j \) by replacing the discourse referents in the universe \( U_{K_j} \) by new ones and the discourse referents \( x \) in \( U_{K_i} \) by \( f(x) \). So called extended applications of DET apply to occurrences of \( K_i \Rightarrow K_j \) in subordinated DRSs. In order to extend DET to UDRSs the function \( f \) is defined on labels and discourse referents preserving conditions in which they occur. DET is restricted to conditions which are right monotone increasing, i.e. it is not admissible in the scope of a \( \neg \) operator.

Indirect proofs employ sub-proofs to prove some goal. They involve the following inference rules: weak COND (conditional proofs), weak RAA (reductio ad absurdum), RESTART, and (strong) Reductio ad Absurdum. Weak COND and weak RAA are adaptations of the COND and RAA rules in [Kamp and Reyle, 1991] to the UDRS framework. COND states that a

---

\[ (112) \ \Delta \models_{UDRS} K, \text{ where } \Delta \text{ is a conjunction of UDRSs } K_1 \land \ldots \land K_n \text{ and } K \text{ is a UDRS,} \]

holds if every model that is a model of \( \Delta \) is also a model of \( K \).\(^{32}\)

\(^{32}\)In principle, other options are possible. In a weaker version \( \models_{UDRS} \) may hold if the conclusion is true in some some of the models satisfying the premises. A stronger definition may require that all readings of the conclusion are true in the models that satisfy the antecedent. Both options, however, violate reflexivity.

\(^{34}\)In fact [Reyle, 1993] provides two consequence relations which are shown to be equivalent.

\(^{34}\)\( \sigma \) conditions state that the labels in the condition were induced by the same sentence.
DRS-condition $K_i \Rightarrow K_j$ can be proved by adding $K_i$ to the premisses and proving $K_j$. In the UDRS approach it is assumed that ambiguities occurring in embedded positions like in the antecedent of a conditional or in a relative clause in the restrictor of a universal quantifier are interpreted locally. This means that instead of interpreting the contribution of an ambiguous antecedent in a conditional structure $(\phi \rightarrow \psi)$ as

$$(\phi' \rightarrow \psi) \lor (\phi'' \rightarrow \psi)$$

where $\phi'$ and $\phi''$ are the disambiguated interpretations of the antecedent, the conditional structure is interpreted as equivalent to

$$(\phi' \rightarrow \psi) \land (\phi'' \rightarrow \psi)$$

In contrast to the strong (or standard) version the weak version of COND can also be applied to implicative conditions in subordinate positions. RAA states that a DRS $K$ can be proved if a contradiction can be derived from adding $\neg K$ to the premise set. The weak version may apply only to negative conditions occurring at either top level or embedded positions. RESTART allows us to replace the current goal in a derivation by the original or some perviously introduced goal using results already obtained.

Recently [Reyle, 1994] has extended the UDRS framework to a treatment of plural NPs addressing ambiguities resulting from collective, distributive, cumulative and generic readings as well as plural pronoun resolution. An HPSG-style UDRS syntax-semantics interface based on the work by [Frank and Reyle, 1992] and [Frank and Reyle, 1994] is outlined in section 1.2.4 below.

### 1.2 Syntax-semantics Interface

DRS construction has been specified in a wide variety of ways and integrated in a number of syntactic frameworks such as categorial grammars, phrase structure grammars, HPSG, LFG and GB-type grammar formalisms. Here we give four “prototypical” examples: the standard top-down construction algorithm, a (semi-) compositional bottom-up version, a declarative reformulation based on equation solving and an HPSG-style principle based specification for UDRS representations.\(^{35}\) In each case the description is confined to the simple “core DRT” outlined in sections 1.1.1 and 1.1.2 above.

\(^{35}\)The bottom-up version and the HPSG-style version are declarative reformulations (i.e. not tied to one particular processing strategy) as well. The modification “bottom-up” refers to information flow rather than to the fixing of some particular processing strategy.
1.2.1 The Top-Down Construction Algorithm

In standard DRT discourses are assumed to be single source texts. A text is simply a sequence of sentences $S_1, S_2, S_3, \ldots, S_n$. The DRS construction algorithm is specified in terms of an iteration over the discourse sentence sequence and an embedded recursion over the DRS under construction.

(113) DRS CONSTRUCTION ALGORITHM

\[
\begin{align*}
\text{INPUT} & \quad \text{discourse } D = S_1, \ldots, S_i, S_{i+1}, \ldots, S_n \\
& \quad \text{empty DRS } K_0 \\
\text{REPEAT} & \quad \text{for } i = 1 \text{ to } n \\
(i) & \quad \text{add syntactic analysis of } S_i \text{ to conditions of } K_{i-1} \\
& \quad \text{call this DRS } K_i^* \\
(ii) & \quad \text{INPUT set of reducible conditions of } K_i^* \\
& \quad \text{apply DRS construction principles to reducible conditions in } K_i^* \text{ until a DRS } K_i \text{ is obtained} \\
& \quad \text{which only contains irreducible conditions}
\end{align*}
\]

In [Kamp and Reyle, 1993] a simple CF-PSG fragment with feature structure annotations is assumed. Rules take the form:\footnote{Here we only show part of the NP rules of the fragment to convey the flavour of the rules.}

\[
\begin{align*}
NP & \quad [ \text{num } x ] \rightarrow DET \quad N \quad [ \text{num } x ] \\
& \quad [ \text{gen } y ] \quad [ \text{num } x ] \quad [ \text{gen } y ]\\
NP & \quad [ \text{num } x ] \rightarrow PRO \\
& \quad [ \text{gen } y ] \quad [ \text{gen } z ] \quad [ \text{gen } y ]
\end{align*}
\]

\[
\begin{align*}
NP & \quad [ \text{num } x ] \rightarrow N \\
& \quad [ \text{gen } y ] \quad [ \text{gen } y ] \quad [ \text{gen } y ]
\end{align*}
\]

\[
\begin{align*}
RC & \quad [ \text{num } x ] \rightarrow RCPRO \\
& \quad [ \text{num } x ] \quad [ \text{gen } y ] \quad [ \text{gap } NP_{num=x} ]
\end{align*}
\]
Given syntactic specifications of the form in (114), the DRS construction algorithm as presented above is in fact a decorated tree-sequence-to-DRS transducer which scans an input sentence sequence from left to right and, successively for each sentence in the input stream, first determines its syntactic structure and then transduces the syntactic structure into components of the DRS representing the discourse in its entirety. As it stands the specification in (113) requires that for each sentence in the input stream the syntactic structure has to be determined before the decorated tree-to-DRS transduction can take place. Hence in each case, syntactic analysis has to precede semantic analysis.\(^{37}\)

In [Kamp and Reyle, 1993] the syntactic configurations (triggering configurations) which trigger the application of particular subtree-to-DRS transformation procedures (DRS construction rules) are in fact often subtrees specified in terms of more than one syntactic rule application. In the specification of the construction algorithm triggering configurations are used to specify what is essentially a top-down, left-to-right tree traversal algorithm in terms of an ordering which is defined by the relative height of the triggering configurations in a tree representation. If there is no unique ordering the construction algorithm is non-deterministic.

The DRS construction rules work on reducible DRSs (DRS conditions) and involve destructive manipulations in terms of deletion, insertion, substitution and choice operations. Reducible DRS conditions are simply those conditions to which construction rules apply. Unreducible DRSs (DRS conditions) are those which are defined in (30) and interpreted in (33). Construction rules are best explained by way of example. For the core fragment we have been considering in sections 1.1 and 1.2 we need construction rules for proper names (CR.PN), pronouns (CR.PRO), indefinite descriptions (CR.ID), relative clauses (CR.NRC), conditional sentences (CR.COND) and universal quantification (CR.EVERY). Here we will consider three representative cases:

\[(115)\quad \text{CR.ID}\]

**Triggering configurations } t \subseteq T \in \text{CON}_K:\]

\[
\begin{array}{c}
\text{S} \\
\text{NP} \quad \text{VP} \\
\quad \text{gen=x} \\
\text{DET} \ 	ext{N} \\
\quad \text{a(n)}
\end{array}
\quad
\begin{array}{c}
\text{VP} \\
\quad \text{V} \\
\quad \text{NP} \quad \text{gen=x} \\
\text{DET} \quad \text{N} \\
\quad \text{a(n)}
\end{array}
\]

Introduce into universe of DRS new discourse referent \(u\)
Introduce into condition set of DRS new condition \(\#(u)\)
Substitute in \(T\):

\(^{37}\)This seems to be slightly at odds with the general on-line philosophy which is characteristic of DKT. In fact, it is possible to reformulate the DRS construction procedure in terms of the familiar syntactic and semantic rule to rule pairing characteristic of much of formal semantics as currently pursued, so that the construction of the syntactic and semantic representation is more closely intertwined (c.f. [Asher, 1993]).
The construction rule for indefinite descriptions CR.ID is triggered by the occurrence of indefinite descriptions in either subject or object position in the simple, declarative sentences in the fragment. Insertion operations introduce a new discourse referent \( u \) into the universe and a condition associating the discourse referent with the common noun \( N(u) \) in the indefinite description into the set of conditions in the currently constructed (sub-) DRS. A substitution operation ensures that part of the triggering configuration \( t \) is replaced by the discourse referent introduced.

In contrast to the DRS construction rule above which will only produce atomic conditions the construction rule for conditional sentences CR.COND will introduce complex conditions containing sub-DRSs into the currently processed DRS.

\[
\text{(116) CR.COND}
\]

Triggering configuration \( T \in CON_K \):

The triggering configuration for CR.COND is the syntactic representation of a conditional sentence and the operation performed by the principle is to replace the triggering configuration (in this case the entire condition) by a new complex DRS condition of the form \( K_1 \Rightarrow K_2 \). The new complex condition contains two DRSs with empty universes, the first of which contains the syntactic analysis of \( S_1 \) as its only reducible condition, while the second contains the syntactic analysis of \( S_2 \) as its only reducible condition. We are faced with the situation that the resulting DRS contains a complex condition with two reducible conditions (DRSs) and
the question is which is to be reduced first? In such a case the recursive definition of the
application of the DRS construction principles is intentionally nondeterministic. Without
going into too much detail, this is in order to be able to account for certain cataphoric
phenomena such as

\[(117) \quad \text{If he}_i \text{ likes Buddenbrooks}_j \text{ then Jones}_i \text{ owns it}_j.\]

Finally we consider the construction rule for personal pronouns CR.PRO. This construction
rule is partly responsible for the so called dynamic aspect of meaning in the DRT fragment
presented here. Intuitively, its dynamicity consists in the way personal pronouns are ana-
phorically related to previously introduced discourse referents: anaphora are interpreted with
respect to the previously established representation of the discourse at the time the anaphor in
question is encountered by the DRS construction algorithm. More precisely, personal pronoun
anaphora are required to be resolved in terms of an identification with previously introduced
discourse referents which are accessible. The notion of accessibility is defined in terms of the
structure of DRSs and in many ways is the DRT counterpart of the notions of scope and
binding in traditional predicate logic representations. If some anaphor cannot be resolved by
the DRS construction algorithm the discourse is considered to be ill-formed.

\[(118) \quad \text{CR.PRO}\]

**Triggering configurations } t \subseteq T \in \text{CON}_K:*

\[
\begin{array}{c|c|c}
\text{S} & \text{VP} & \text{NP} \\
\hline
\text{NP} & \text{VP} & \text{V} & \text{NP} \\
\text{gen}=x & \text{gen}=x & | & | \\
| & | & Pro & Pro \\
| & | & \alpha & \alpha \\
\end{array}
\]

Introduce into universe of the DRS new discourse referent \( u \)
Choose suitable antecedent \( v \) such that \( v \) is accessible to \( u \)
Introduce into condition set of the DRS new conditions \( v = u \) and \( \text{gen}(u) = x \)
Substitute in \( T \)

\[
\begin{array}{c|c|c}
\text{S} & \text{NP} & | \\
\hline
u \quad \text{for} \quad \text{NP} & \text{gen}=x & | \\
| & | & Pro \\
| & | & \alpha \\
\end{array}
\]

One of the operations performed by CR.PRO is a choice operation: choose a suitable an-
tecedent \( v \) such that \( v \) is accessible to the discourse referent \( u \) associated with the personal
pronoun currently processed. The term *suitable* indicates that the antecedent has to agree with the pronoun in terms of gender, number and possibly other properties to be specified.\(^{28}\)

### 1.2.2 A Bottom-Up Version

In this section we give an outline of the bottom-up, semi-compositional version of DRS construction detailed in [Asher, 1993]. In this approach lexical entries are associated with *predicative* or *partial* DRSs which are combined according to the syntactic rules in the grammar in terms of a notion of DRS conversion. On the sentential level DRSs are combined via an operation of DRS-union. In this setup, each subconstituent in a derivation is associated with an explicit semantics. The approach is semi-compositional since (i) anaphoric resolution is kept apart as a separate process and (ii) a single syntactic structure can give rise to a set of semantic representations for the scope relations between the scope bearing elements in the natural language source expression.

Predicative DRSs are defined as

\[(119)\] Definition: If \(\alpha_1, \ldots, \alpha_n\) are variables over discourse referents and \(K = \langle U, Con \rangle\) with \(x_1, \ldots, x_n \in U\) then \(\lambda \alpha_1, \ldots, \lambda \alpha_n\langle U - \{x_1, \ldots, x_n\}, Con(\alpha_1/x_1, \ldots, \alpha_n/x_n)\rangle\) is a predicative DRS.

The following two are examples of predicative DRSs

\[(120)\] \(\lambda \alpha \underline{\text{man}(\alpha)}\)

\[(121)\] \(\lambda \alpha_1 \lambda \alpha_2 \underline{\text{call}(\alpha_1, \alpha_2)}\)

(120) denotes a function from discourse referent denotations to the denotation of its argument DRS in the familiar fashion. Likewise (121) denotes a function from discourse referent denotations to a function from discourse referent denotations to the denotation of its argument DRS. Predicative DRSs can be applied to discourse referents and this application can be reduced in terms of DRS-conversion. Unlike \(\lambda\)-conversion, DRS-conversion can apply to any of the \(\lambda \alpha_i\) parts in a \(\lambda \alpha_1, \ldots, \lambda \alpha_n\) prefix of predicative DRS. Which \(\lambda\) binder is affected is determined by linking rules exploiting syntactic and possibly other information which is useful in accounting for the scope relations between quantifiers and other operators.

\[(122)\] Definition: partial DRSs are DRS with one or more predicative DRSs abstracted over.

\(^{28}\)Recency, reiteration, parallelism, grammatical function compatibility and other constraints on saliency are investigated in a computational setting in [Asher and Wada, 1988].
Examples of partial DRSs are determiner and NP translations. The indefinite determiner *a* corresponds to

\[ \lambda P \lambda Q \]
\[ \begin{array}{c}
  x \\
  P(x) \\
  Q(x)
\end{array} \]

while *every* corresponds to

\[ \lambda P \lambda Q \]
\[ \begin{array}{c}
  x \\
  P(x) \\
  Q(x)
\end{array} \]

Prop names translate as

\[ \lambda P \]
\[ \begin{array}{c}
  x \\
  P(x) \\
  \text{john}(x)
\end{array} \]

while pronominal anaphors translate as

\[ \lambda P \]
\[ \begin{array}{c}
  x \\
  P(x) \\
  x = ?
\end{array} \]

The condition \(x = ?\) acts as an instruction to the resolution component to find a suitable antecedent for \(x\) which is accessible in the representation of the already processed discourse. Anaphors are resolved after DRS update operations.

\[ \text{Definition: } \text{DRS-update}(K_1, K_2) = (U_{K_1} \cup U_{K_2}, Con_{K_1} \cup Con_{K_2}) \]

The sentence *A man walks in the park* gives rise to the following multiple application

\[ (\lambda P \lambda Q) \]
\[ \begin{array}{c}
  x \\
  P(x) \\
  Q(x)
\end{array} \]
\[ \left( \lambda \alpha_1 \left[ \begin{array}{c}
  \text{man}(\alpha_1)
\end{array} \right] \right) \left( \lambda \alpha_2 \left[ \begin{array}{c}
  \text{walkinpark}(\alpha_2)
\end{array} \right] \right) \]

which reduces through DRS-conversion.
(129) \[
\begin{array}{c|c}
  x & \text{man}(x) \\
  \hline
  \text{walkinpark}(x) & \\
\end{array}
\]

while the continuation *He whistles* results in

(130) \[
\begin{array}{c|c}
  y & \text{whistle}(y) \\
  \hline
  y = ? & \\
\end{array}
\]

(129) and (130) combine in terms of a DRS-update *plus resolution* into

(131) \[
\text{DRS-update} \left( \begin{array}{c|c}
  x & \text{man}(x) \\
  \hline
  \text{walkinpark}(x) & \text{whistle}(y) \\
  \hline
  y = ? & \\
\end{array} \right) = \begin{array}{c|c}
  \hline
  x & \text{man}(x) \\
  \text{walkinpark}(x) & \text{whistle}(y) \\
  \hline
  y = x & \\
\end{array}
\]

1.2.3 A Declarative Reformulation in Terms of Equation Solving

The authors of [Johnson and Klein, 1986] present a declarative reformulation of core DRT as outlined in section 1.1 above. The reformulation is based on equation solving, membership constraints and the threading technique in logic programming (c.f. [Pereira and Shieber, 1987]).

In the standard formulation of DRT the basic left to right dependencies in discourse are captured in the algorithmic specification of the construction procedure. In the approach presented by [Johnson and Klein, 1986] these dependencies are directly encoded into the grammar.

Discourse relevant aspects of the meaning of a linguistic expression $\alpha$ can be viewed as a relation between the immediately preceding and the subsequent discourse.

(132) **Preceding Discourse** $\mid \alpha \mid$ **Following Discourse**

Referential expressions such as indefinite NPs take the incoming discourse, add reference markers and conditions and make the resulting discourse representation available to the subsequent discourse. By contrast, an NP anaphor (like a personal pronoun) will have to check whether the incoming discourse representation contains a suitable discourse referent against which the anaphor can be resolved. If this is the case the anaphor simply equates the incoming with the outgoing discourse. Otherwise, the discourse is considered ill-formed due to resolution failure:
The threading idea can be outlined as follows: each node in the syntactic representation is decorated with an \textit{in} and an \textit{out} attribute and the incoming discourse is assigned as the value of the \textit{in} attribute while outgoing discourse is assigned as the value of the \textit{out} attribute. In this way discourse representations are threaded through the entire syntactic representation. The tree nodes in the representation can either block, update or simply pass on the discourse representation\footnote{There is quite a striking similarity between this threading idea and assignment passing in Dynamic Predicate Logic [Groenendijk and Stokhof, 1991a] which is probably not entirely accidental.} depending on the discourse contribution of the syntactic constituent dominated by the particular node.

The specifications in (133) can be translated into sets (conjunctions) of equations interpreted as constraints (partial functions \textit{in} and \textit{out} defined) on nodes \(N_i\) in syntactic representations:

\begin{equation}
\begin{aligned}
\text{in}(NP_{[\text{woman}]} &= C] \land \text{out}(NP_{[\text{woman}]} = C \cup f] \\
\text{in}(NP_{[\text{her}]} &= C] \land \text{out}(NP_{[\text{her}]} = C] \land [f \in C]
\end{aligned}
\end{equation}

If such constraints are integrated with CF-PSG rules we get something which resembles very closely a specification expressed in the PATR-II grammar formalism [Shieber, 1986]. In a slightly extended version of this formalism the lexical entry for \textit{woman} is

\begin{equation}
\begin{aligned}
\text{Word woman} \\
W:\text{cat} &= n, \\
W:\text{syn:index} &= w, \\
W:\text{sem:in} &= [\text{Current}|\text{Super}], \\
W:\text{sem:out} &= [[w,woman(w)|\text{Current}|\text{Super}].
\end{aligned}
\end{equation}

In this formalism sets are approximated with lists (here represented in the familiar Prolog list notation) and DRT accessibility relations are represented in terms of ordered sequences of DRS representations where DRS further down the list are accessible while DRS embedded inside members of the list are not. The third equation in (135) separates the incoming discourse into the currently active DRS \textit{Current} and the sequence of superordinate, accessible DRSs \textit{Super}. The fourth equation introduces the condition \textit{woman}(f) and the discourse referent \(f\) into the currently open DRS.

The constraint equations associated with the lexical entry of the determiner \textit{every} push two new empty subspaces onto the currently open DRS, one for the discourse representation provided by the restrictor domain associated with the quantifier and one for the discourse
representation associated with the scope domain of the quantifier. Furthermore, the equations ensure that the incoming discourse is available as the value of the in attribute of the res (restrictor) attribute and that the representation of the preceding discourse together with the restrictor contribution is available as the input to the quantifier’s scope attribute. The resulting representation is then associated with the quantifier’s sem:out attribute.

(136) Word every

\[
\begin{align*}
W:cat &= \text{det}, \\
W:sem:in &= \text{DetSemIn}, \\
W:sem:res:in &= \emptyset | \text{DetSemIn}, \\
W:sem:scope:in &= \emptyset | \text{DetSemResOut}, \\
W:sem:scope:out &= \emptyset | \text{Scope,Res}|\text{Current}|\text{Super}], \\
W:sem:out &= \emptyset | (\text{Res} \Rightarrow \text{Scope})|\text{Current}|\text{Super}.
\end{align*}
\]

The constraints associated with the lexical entry of the pronominal anaphor her are expressed as follows:

(137) Word her

\[
\begin{align*}
W:cat &= \text{np}, \\
W:sem:in &= \text{SemIn}, \\
W:syn:index &= w, \\
\text{member}(\text{Space,SemIn}), \\
\text{member}(w,\text{Space}), \\
\end{align*}
\]

Here the first member/2 constraint selects some superordinate (i.e. accessible) DRS from the preceding discourse representation. The second member/2 constraint attempts to resolve the anaphor against antecedent discourse referents in the DRS picked out by the first member/2 constraint. Note that here member/2 is in fact employed to implement the element relation ‘∈’ specified in (134) above.

Together with the rest of the clauses given in [Johnson and Klein, 1986] clauses (135)-(137) above constitute a declarative reformulation of core aspects of DRT. Anaphoric relations are specified independently of processing algorithms in terms of set membership constraints and a relational view of the anaphorically (and semantically) relevant properties of linguistic expressions in terms of in and out attributes and set constructor relations. The separation of the incoming discourse as the value of the in attribute of the representation of some linguistic expression from the outgoing discourse as the value of the out attribute of that expression in effect achieves the ordering of discourse referents implicitly defined in terms of the DRS construction algorithm in the canonical formulation of DRT ([Kamp, 1981], [Kamp and Reyle, 1993]). The clauses in effect constitute a logical specification of core aspects of DRT which can serve as input to theorem provers. The clauses thus constitute a runnable specification of DRT which does not include destructive operations. The resulting DRS is simply the solution
to the equations plus the solutions to the recursive member/2 constraints in the proof tree associated with a syntactic derivation.

1.2.4 A HPSG-style UDRS Syntax-Semantics Interface

HPSG [Pollard and Sag, 1994] descriptions are based on typed feature structures referred to as signs simultaneously describing phonological, syntactic and semantic information. Types are ordered in an inheritance hierarchy. Signs are required to satisfy the typing regime. Lexical entries (lexical signs) are given in terms of complex typed feature structures featuring \texttt{Pron} (phonology) and \texttt{Synsem} (syntax - semantics) root attributes. Phrasal signs are signs with (lists of) other signs as the value of a \texttt{Dtrs} (daughters) attribute satisfying certain ID (immediate dominance) and LP (linear precedence) constraints. Phrasal signs are required to satisfy a set of principles. The principles are stated in a separate and modular fashion. Amongst other things, they regulate subcategorization requirements, percolation of information from the head daughter of a phrasal sign to the head of the sign and the construction of a semantic representation. In the current version HPSG employs a scaled down version of Situation Semantics. The formalism is sentence oriented and employs a version of the Cooper storage mechanism [Cooper, 1983] for the purposes of scope representation.

In this section we will briefly describe a syntax-semantics interface for the construction of UDRSes in HPSG based on the work by [Frank and Reyle, 1992] and [Frank and Reyle, 1994].\footnote{For a UDRS syntax-semantics interface in an extended categorial grammar framework where categories are complex typed feature structures see [König, 1994].} In standard HPSG on the level of phrasal signs the interface between syntactic and the semantic representation is defined in terms of the interplay between three principles: the Quantifier-Inheritance principle, the Scope Principle and the Semantics Principle implementing a Cooper storage based approach which yield sets of disambiguated signs for scopally ambiguous phrases. In the approach outlined below the Cooper storage mechanism is replaced by an approach based the partial (i.e. underspecified) representations - UDRSes - introduced in section 1.1.8 Underspecification above. Scopally ambiguous phrases are associated with a single underspecified representation and the construction and disambiguation of such underspecified representations is defined in a completely monotonic fashion\footnote{Monotonicity here means that both construction and disambiguation of underspecified representations is simply achieved by the accumulation of information from a variety of sources like syntax, semantics, context, world knowledge etc. In the present framework there is no need for destructive manipulations of representations.} in terms of a Semantics Principle and a set of general and theory specific meta-constraints. The meta-constraints apply to the specification of lexical signs and the construction of phrasal signs. They include conditions on the scope potential of indefinites and proper names, the clause boundedness of genuine quantificational structures (Clause Boundedness) and general well-formedness conditions on UDRSes which guarantee proper binding of discourse referents in argument positions of verbs (Closed Formula Principle). Additional theory specific or contextual meta-constraints may further disambiguate underspecified representations. [Frank and Reyle, 1992] and [Frank and Reyle, 1994] e.g. recode the syntactic constraints on quantifier scoping postulated by [Frey, 1993] and [Frey and Tappe, 1992] originally formulated in a GB framework in the UDRS-HPSG framework. Roughly speaking, the scope principle states that
in a local domain \( L_\alpha \) a phrase \( \alpha \) may have scope over a phrase \( \beta \) if \( \alpha \) c-commands \( \beta \) or one of \( \beta \)'s traces.

UDRSes are encoded as the value of the \texttt{UDRS} feature at the \texttt{SYNSEM}\texttt{LOC} path of a sign. A UDRS consists of a set of labelled conditions, a set of subordination constraints and a pair of distinguished minimal and maximal labels labelling the UDRS. The distinguished labels define upper and lower bounds for the UDRS in the upper semi-lattice defined by an embedding UDRS.

\[
\begin{align*}
\texttt{UDRS} & \left[ \begin{array}{c}
\text{LS} \left[ \begin{array}{c}
\text{L-MAX } l_{\text{max}} \\
\text{L-MIN } l_{\text{min}} \\
\end{array} \right] \\
\text{SUBORD} \left[ \begin{array}{c}
\{ l \in \text{L} \} \\
\end{array} \right] \\
\text{CONDS} \left[ \begin{array}{c}
\end{array} \right] \\
\end{array} \right]
\end{align*}
\]

Genuinely quantificational determiners like e.g. \textit{every} induce a restrictor (\texttt{RES}) and a nuclear scope (\texttt{SCOPE}) attribute into the representation. A new discourse referent is introduced in the restrictor condition. The subordination constraints specify that both restrictor and scope are subordinate to the label \( l_1 \) representing the upper bound (the distinguished maximal label) associated with the quantificational structure in its entirety. Scope relations between restrictor and scope are left unspecified. The lower bound (the distinguished minimal label) of the quantificational structure is identified with the label associated with the nuclear scope \( l_{12} \).\(^{42}\)

\[
\begin{align*}
\texttt{LOC} & \left[ \begin{array}{c}
\text{CAT} \left[ \begin{array}{c}
\text{HEAD } \texttt{quant} \\
\text{SUBCAT} \{ \texttt{LOC} \} \\
\text{LS} \left[ \begin{array}{c}
\text{L-MAX } l_{\text{MAX}} \\
\text{L-MIN } l_{\text{MIN}} \\
\end{array} \right] \\
\text{SUBORD} \{ \texttt{LABEL}[1] \} \\
\text{CONDS} \{ \texttt{LABEL}[3] \} \\
\end{array} \right] \\
\end{array} \right]
\end{align*}
\]

The lexical entry for the indefinite determiner does not distinguish between a restrictor and a scope attribute. The distinguished maximal and minimal labels are identified with the distinguished label in the discourse referent introducing condition. On specific readings indefinite NPs may get arbitrary wide scope in the resulting representation. Hence the lexical entry for the indefinite determiner does not feature subordination constraints.

\[
\begin{align*}
\texttt{LOC} & \left[ \begin{array}{c}
\text{CAT} \left[ \begin{array}{c}
\text{HEAD } \texttt{quant} \\
\text{SUBCAT} \{ \texttt{LOC} \} \\
\text{LS} \left[ \begin{array}{c}
\text{L-MAX } l_{\text{MAX}} \\
\text{L-MIN } l_{\text{MIN}} \\
\end{array} \right] \\
\text{SUBORD} \{ \texttt{LABEL}[\square] \} \\
\text{CONDS} \{ \texttt{LABEL}[\square] \} \\
\end{array} \right] \\
\end{array} \right]
\end{align*}
\]

\(^{42}\)Reentrancies are indicated in terms of the box \( \square \) notation.
Given lexical entries for determiners along the lines outlined above the entries for common nouns are almost trivial.

\[
(141) \quad \begin{array}{c}
\text{LOC} \quad \begin{array}{c}
\text{CAT} \quad \begin{array}{c}
\text{HEAD}_{\text{noun}}
\end{array}
\end{array}
\end{array}
\begin{array}{c}
\text{UDRS} \quad \begin{array}{c}
\text{SUBORD} \{ \}
\end{array}
\end{array}
\begin{array}{c}
\text{COND}_{\text{s}} \quad \{ \text{LABEL}_{i} \text{REL}_{\text{machine}} \}
\end{array}
\end{array}
\]

Proper names introduce a discourse referent and will always end up in the one element \( \top \) (the “top” DRS) of the upper semi-lattice defined by the resulting UDRS.

\[
(142) \quad \begin{array}{c}
\text{LOC} \quad \begin{array}{c}
\text{CAT} \quad \begin{array}{c}
\text{HEAD}_{\text{quant}}
\end{array}
\end{array}
\end{array}
\begin{array}{c}
\text{UDRS} \quad \begin{array}{c}
\text{SUBORD} \{ \}
\end{array}
\end{array}
\begin{array}{c}
\text{COND}_{\text{s}} \quad \{ \text{LABEL}_{i_{\text{r}}} \text{REL}_{\text{many}} \}
\end{array}
\end{array}
\]

Lexical entries for verbs simply ensure the correct mapping of argument positions in the semantic representation of the verb with the semantic representation of elements on the subcategorization list. They do not introduce subordination constraints.\(^{43}\)

Based on lexical specifications of the kind outlined above the construction of semantic representations of phrasal signs is defined in terms of a general Semantics Principle, a Closed Formula Principle which ensures that discourse referents in argument positions are properly bound and possibly other meta-constraints.

The Semantics Principle defines:

- the inheritance of UDRS-conditions of the daughters of a phrasal sign to the condition set of the sign
- the inheritance of subordination constraints from the daughters to the subordination constraint set of the entire sign and
- the projection of lower and upper bounds (distinguished labels) in the semi-lattice from the head daughter to the phrasal sign.

The set of UDRS-conditions of the phrasal sign is simply the union of the sets of UDRS-conditions \( \Gamma_{i} \) of its daughters. Likewise, the subordination constraint set of a phrasal sign is the union of the sets of subordination constraints \( \Sigma_{i} \) of its daughters. Finally, the distinguished labels \( l_{\text{max}} \) and \( l_{\text{min}} \) associated with the phrasal sign are the distinguished labels from the head daughter.

\(^{43}\)Unless the Closed Formula Principle (discussed below) is absorbed into the lexical specifications of verbs.
The Closed Formula Principle ensures that the UDRS-conditions associated with the arguments of a verb are always superordinated to the UDRS-condition associated with the verb.

Alternatively, the Closed Formula Principle could be lexicalized in the entries associated with verbs. If scope inducing adjuncts are allowed to occur between subcategorised elements the grammar can be restricted to binary branching structures and (133) and (134) to the binary case.
Chapter 2

Update and Dynamic Semantics

This contribution on update and dynamic semantics is outlined as follows. First we provide an overview of dynamic parameters in processing. Next we give a brief survey of what existing dynamic semantic theories have to offer, with some remarks as to how they relate to each other. Then we look at the way in which syntax and semantics are linked up in an NLP system with a dynamic semantic component. Finally, we take a list of specific semantic problems or application areas and state in which ways dynamic and update semantics can be helpful in solving the problems or charting the areas.

The text is aimed at co-workers in the field of computational linguistics who are looking for insights they can acquire from current theoretical developments in dynamic semantics for natural language and subsequently apply in their NLP applications. We do not pretend that what we have to say bridges the gap between theory and practice in dynamic semantics. Our ideal reader is benevolent enough to bear with us through a formal presentation of systems and proposals, patient enough to get thoroughly acquainted with the material, and finally brave enough to jump.

2.1 Semantic Tools

2.1.1 Dynamic horizons

As the scope of dynamic semantics is much wider than computational linguistic applications only, it was inevitable that this got reflected in the presentation. Many of the motivations for a linguist to ‘go dynamic’ are shared by researchers in the fields of cognitive and computer science. In fact, many dynamic formalisms originate from semantic theories of programming languages and computational processes.

Generally speaking, the aim of dynamic semantics is to find appropriate mathematical logical means for reasoning about change. Of course, different areas and topics of change give rise
to different dynamic theories. Also, the variety of formal philosophical perspectives on the model-theoretic interpretation of the same class of changes leads to substantial divergence.

The result is a wide spectrum of different perspectives in dynamic semantics, which inevitably makes its way into an overview like this. The various theories do not yet converge, and still we think of convergence as a kind of holy grail of general research in semantics of natural language. By taking a point of view which is elevated enough, we think we can still manage to convey that there is unity underlying the seeming chaos. Our reason for not sticking to linguistic applications proper is to get at such a proper vantage point. For a clear general picture we have decided to introduce our reader to some important ‘dynamic’ developments in cognitive and computer science. We are aware of the danger of over-exposure here, and therefore we have arranged the document in such a way that individual subsections can be skipped safely (all within reason, of course). This has resulted in a somewhat staccato style of presentation. The reader who is primarily interested in linguistic applications is advised to turn to the deliverable 9, where we commit ourselves to our linguistic tasks by giving a more down to earth presentation of some linguistic implications of dynamic semantics.

2.1.2 Dynamic Parameters

2.1.2.1 Score Keeping in Language Games

‘Score Keeping in a Language Game’ was the metaphor proposed by Lewis [Lewis, 1979] for the process of context updating which takes place during the activity of communication by means of natural language. Aspects of the context that get updated are the continually changing answers to questions like: ‘Who is the current speaker?’, ‘What is the current topic of conversation?’, ‘What are the currently most salient referents for personal pronouns and demonstratives?’, ‘What is the current time point of speech in the narrative?’, ‘What is the current temporal reference point in the narrative?’, and so on.

The changes in context that take place can be viewed as updates of a score board of dynamically changing information. In fact, the different dimensions for score keeping that Lewis distinguishes (and the list can be extended almost at libitum) are different dynamic parameters that can change. Dynamic logics and update logics have been devised to keep track of single parameters or parameter lists.

Dynamic predicate logic (DPL), proposed by Groenendijk and Stokhof [Groenendijk and Stokhof, 1991a], keeps track of what the context says about current antecedents for anaphoric pronouns. Processing an indefinite description results in the storage of a new value in a register, and that register can subsequently be accessed to make an anaphoric reference to the indefinite.

Eliminative update logics (Heim [Heim, 1982], Landman [Landman, 1986], Veltman [Veltman, 1991]) can be used to keep track of the changing state of information of a hearer of a piece of declarative natural language text. The account is eliminative because every addition to
the discourse ‘eliminates’ epistemic possibilities from a space of possible worlds or possible situations.

One could also say that a context-parameter changes when a natural language imperative is heard, understood and obeyed. If one adopts the simplifying assumption that imperatives are obeyed, one might define the meaning of natural language imperatives as the change that takes place in the world when the command is carried out. A command like ‘close the door’ would then be interpreted as the (minimal) change in the current state of the world that results in a closed door, and so on: a change in yet another context parameter. More realistically, imperatives are interpreted as obligations to change the world which the speaker puts on the addressee by performing the speech act of the imperative. On this view, what changes are chores of obligations of discourse participants. But there is no harm in trying to understand the simplified semantics, too.

Eventually, in a dynamic set-up of a semantics for natural language an account is due of the way in which several parameters of context change at the same time and interact with each other. It is often convenient, however, to first concentrate on the changing parameters one by one, as it can be confusing to keep track of too many aspects of context change at once.

It is also possible to try to identify parameters of context change while starting out from the perspective of standard first-order logic. This leads to the theme of ‘Tarskian variations’ (Section 2.1.2.3).

2.1.2.2 States versus Transitions

If one adopts a dynamic perspective on context change, then one takes the linguistic items to be actions or procedures \( \pi \) which effect transition from one state to another:

\[
s \xrightarrow{\pi} s'.
\]

What the states look like depends of course on the nature of the context parameters which are taken into account.

This perspective is very similar to the perspective of computer science, where an imperative program \( \pi \) is viewed as a transition from an input state to an output state of a machine. Here the states are viewed as memory states: allocations of registers (memory stores) to a list of variable names, and of values to the list of registers that constitutes the computer memory.

In computer science, it is customary to link the transition level description of what a program does to state level descriptions, by using assertions which states the preconditions of a program, given a particular output state. Precondition reasoning constitutes in effect a move from transition-level talk to state-level talk. E.g., \( \langle \pi \rangle \phi \) will be true in a given state \( s \) just in case there is a state \( s' \) with \( s \xrightarrow{\pi} s' \) such that \( \phi \) is true in \( s' \). A converse move is made when checking whether a certain condition \( \phi \) is the case in state \( s \) is viewed as a transition from \( s \) to \( s' \):

\[
s \xrightarrow{\phi?} s.'\]
This constitutes in effect a switch from state level talk to transition level talk.

In logic, *propositions* are used to talk about states and *procedures* to talk about transitions. Switches from propositions to procedures (e.g., from \( \phi \) to \( \phi' \)) and from procedures to propositions (e.g., from \( \pi \) to \( (\pi)\top \), i.e., from the procedure \( \pi \) to the statement that \( \pi \) succeeds for the current input state) are a common ingredient to various brands of dynamic logic.

In the application to natural language semantics, the connection between transition level talk and state level talk serves to incorporate the traditional key notion of natural language semantics, *truth in a model or situation*, into the new dynamic notions.

### 2.1.2.3 Tarskian Variations

The semantic parameters involved in the dynamics of standard first-order logic are made explicit in the following scheme:

\[
D, I, A \xrightarrow{x} D', I', A',
\]

where \( D \) stands for the domain of discourse, \( I \) for the interpretation function and \( A \) for the variable assignment. (A survey of the general dynamic setting of Tarskian variations is given in Van Benthem and Cepparello [Benthem and Cepparello, March 1994].)

In fact, ordinary Tarskian semantics already incorporates a bit of dynamics over assignment by means of quantifiers:

\[
D, I, A \xrightarrow{\exists x} D, I, A_x.
\]

The program \( x := ? \) should be read as "give \( x \) an arbitrary value". So, the assignment \( A_x \) is an assignment which is a ‘modulo-\( x \)-copy’ of the ‘input’-assignment \( A: A_x(y) = A(y) \) for all variables \( y \neq x \). The existential quantifier can then be read as a possible successful execution of this primitive program: \( \exists x \phi \) means that it is ‘possible to put \( x \) into the property \( \phi \)’. In the format of the previous section this would be written as \( \langle x := ? \rangle \phi \). Dually, the universal quantifier reflects ‘success guaranteed’ of this program. This is also written as \([x := ?]\phi\).

In Groenendijk and Stokhof’s dynamic predicate logic (DPL) this simple dynamics is employed for interpreting introduction of indefinites and anaphoric description. From their point of view, an existential quantified proposition does not only look for possible success of the random assignment program, but indeed performs the program. A continuation of the discourse starts after this execution. In other words, \( \exists x \phi \) means ‘put \( x \) into \( \phi \) if possible, and go on’. In the format of the previous section, the DPL-program \( \exists x \phi \) refers to \( x := ?; \phi' \), where \( : \) is the composition of the programs. The anaphoric interpretation is then brought about by the actual performance of the execution of \( x := ? \), which links later occurrences of \( x \) to this actual assignment.
Kamp’s discourse representation theory (DRT) and Heim’s file change semantics (FSC) employ dynamics over partial finite variable assignment for interpretation of introduction of ‘reference markers’.

\[
\begin{array}{c}
D, I, A \\
\hline
x
\end{array} \\
\rightarrow \\
\begin{array}{c}
D, I, A^x.
\end{array}
\]

Here \(A^x\) refers to a partial assignment which has a domain \(\text{Dom}(A) \cup \{x\}\) and for all \(y \in \text{Dom}(A): A(y) = A^x(y)\). When a formula \(\phi\) is put into the lower empty discourse box, we refer to the same relation but then restricted to \(\phi\)-outputs. The relation to the DPL-formula \(\exists x \phi\) is pretty close. The latter means that ‘\(x\) is changed into a \(\phi\)-er’ while the former interpretation means that ‘\(x\) is given a \(\phi\)-value’.

The simple-minded interpretation of imperatives by means of changes in the world that was mentioned above fits into this scheme too, as it corresponds to the process of changing the interpretation function \(I\).

But it is not always possible to study Tarskian variations in isolation. To mention an example, intuitionistic predicate logic employs dynamics over extensions of domains and interpretation functions at the same time: \(D' \supseteq D\) and \(I' \supseteq I\), while it uses the ordinary \(=\)-relation over assignments to interpret quantifiers. A universal quantified formula combines all these. Models of quantified intuitionistic logic are defined over a so-called information structure \(\langle S, \subseteq \rangle\) where \(S\) is a non-empty set of information states and \(\subseteq\) a reflexive transitive relation, or pre-order, over this set. A model of first-order intuitionistic logic assigns to every information state \(s\) a domain \(D_s\) and an interpretation function \(I_s\), in such a way that these parameters grow with the pre-order:

\[
D_s \subseteq D_t, \quad \text{and} \\
I_s(P) \subseteq I_t(P) \quad \text{for all } s, t \text{ with } s \subseteq t \text{ and predicates } P.
\]

This increase of these semantic parameters is meant to model construction of individuals and their properties ‘on line’, so to speak. A universal quantified proposition \(\forall x \phi\) says that the property \(\phi\) holds of any individual which may be constructed ‘later on’. The variable assignments range therefore over the collection of all ‘conceivable’ individuals: \(VAR \rightarrow \bigcup_{s \in S} D_s\). The universal quantifier is subsequently interpreted as a universal statement over the relation

\[
D_s, I_s, A \rightarrow D_t, I_t, A^x,
\]

\footnote{Of course, there’s more to DRT and FCS then its dynamic first order model-theory. Many linguists affiliated with DRT stress the importance of its representational level, i.e. the closer resemblance of DRT’s syntax to NI than e.g. classical logic. We refer the reader to the FraCaS-document on DRT [van Genabith and Kamp, 1994] for such discussions which are outside the scope of the general dynamic picture of this document.}

\footnote{This relation is also abbreviated by \(A \leq_s A'\). For \(\leq_s, \ldots, \leq_s\) we also write \(\leq_{s_1, \ldots, s_n}\).}

\footnote{See also the later subsections on modal settings for dynamic logic in section 2.1.3.}
with $s \subseteq t$, and $A_x$ as defined for the program $x := ?$ above.

The styles of intuitionistic model theory and of other constructive logics present a very general dynamic perspective. Later on we will try to make plausible that this style of dynamic reasoning provides a format for unifying different dynamic semantic theories.

### 2.1.2.4 Changing Assignments

In uses of dynamic logic to account for anaphoric linking, often the simplifying assumption is made that the piece of discourse to be analysed has a fixed anaphoric reading, and that the anaphors are linked to their antecedents by means of subscripts and superscripts. Barwise [Barwise, 1987b] assumes that antecedents have superscripts and anaphoric pronouns subscripts.

1. *A man$^1$ walked in. He$_1$ smiled.*

Natural language sentences as encountered in real life (in newspapers, say) do not contain indices, of course. If such indices are assumed by some theory of natural language semantics, this means that the theory implicitly postulates the existence of an ‘indexing module’ that presumably works at the level of surface syntax and which fixes a ‘possible anaphoric reading’ of a text. Thus, we start with

2. *A man walked in and greeted the publican. He smiled.*

and get either

3. *A man$^1$ walked in and greeted the publican$^2$. He$_1$ smiled.*

or

4. *A man$^1$ walked in and greeted the publican$^2$. He$_2$ smiled.*

The first indexing construct an anaphoric arrow between the pronominal subject of the second sentence and the antecedent *a man*, while the second links the pronoun to the antecedent *the publican*.

In this example, both indexings are possible, but in general there are rather clear syntactic/semantic constraints on possible indexings. The indexing of the following example, for instance, is ruled out:

5. *No customer$^1$ had greeted the publican$^2$. He$_1$ smiled.*
Theories of dynamic semantics will have to give an account of the constraints involved. From the point of view of native speakers of the language under considerations, the fact that certain anaphoric links are ruled out is a matter of linguistic intuition. Theories about anaphoric linking should try to give an account of the constraints in terms of predictions made by the theory. Theories like DRT and DPL do in fact have predictions to offer about constraints on anaphora, but they have virtually nothing to say about the preference ordering on the set of possible indexings.

John hates a man who hates him and another man who does not.

Figure 2.1: Anaphoric links are arrows.

Talk about anaphoric indexings should not make one forget that the indices are just a convenient means to represent anaphoric arrows running from anaphoric elements to their antecedents (Figure 2.1). The anaphoric reading indicated by the arrows in the picture can be represented with indices as follows:

6 John\textsuperscript{1} hates a man\textsuperscript{2} who hates him\textsubscript{1} and another\textsubscript{2} man\textsuperscript{3} who does not.

If one chooses antecedent indices unwisely, previous antecedents can be cut off and become unavailable for further anaphoric linking:

7 A man\textsuperscript{1} walked in and greeted the publican\textsuperscript{1}.

Now a link to the first man by means of an anaphoric index in subsequent discourse becomes impossible.

The following indexing constraint ensures that this blocking off of anaphoric potential does not take place:

An index $i$ can occur as an antecedent index in a text at position $P$ only if $i$ does not occur as an index to the left of $P$.

In Section 2.2 this problem will show up again in connection with the syntax-semantics interface of natural language systems, and some pointers will be given concerning how it may be tackled.
2.1.2.5 Changing States of Information

Computer programs can be said to change states of information, where a state of information is taken to be a memory state of a machine. Abstracting from the nature of the states involved, we can specify the change that an atomic program $e$ effects by means of a two place transition relation $R_e$ on a set of states. This perspective gives rise to the study of so-called transition systems. The most general style of reasoning about programs and transition system is found in propositional dynamic logics (Pratt [Pratt, 1976] [Pratt, 1980], Harel [Harel, 1984]) and in algebras of processes (Hennessy [Hennessy, 1988]). Processes and transition systems are studied from the perspective of modal logic in Stirling [Stirling, 1987] and Van Benthem and Bergstra [Benthem and Bergstra, 1993].

Dynamic semantics can be put to use to stipulate relational denotations for propositions. In this perspective, a state of information is a set of possible worlds, and a program updates a state of information by removing the worlds incompatible with the new information. Thus, the semantics of language is defined in terms of its potential to change the state of information of a hearer who is exposed to it. This perspective is adopted for pragmatics in Stalnaker [Stalnaker, 1979], for formal semantics in Heim [Heim, 1982] and Landman [Landman, 1986] (and many others), and for common-sense reasoning in Veltman [Veltman, 1991]. The eliminative approach has the disadvantage that retraction and revision of information cannot be accommodated naturally.

To handle retraction and revision one has to be able to move back and forth along the dimension of information ordering, for one now has to be able to give up beliefs as well as acquire new information. Retracting $\phi$ from one's state of information boils down to: going back to a state of less information where $\phi$ does not hold anymore, by giving up some of one's beliefs. Of course, one should make sure not to give up more than necessary, and of course, one might be faced with choices about which beliefs to give up. If one retracts $p \land q$ in a state where one believes both $p$ and $q$, one has to decide whether to give up give up $p$ or $q$ (if one gives up both, the revision will probably not be minimal). Revising by $\phi$ can be viewed as a combination of retraction and expansion. In case $\phi$ is consistent with one's current state of belief, revision by $\phi$ boils down to expansion with $\phi$: one moves to a state of more information where $\phi$ holds. In case $\phi$ is inconsistent with one's current state of belief, one first retracts $\neg \phi$ by moving back to a state of less information where $\neg \phi$ does not hold anymore (i.e., to a state which is consistent with $\phi$), and then expands with $\phi$.

In Gärdenfors style belief revision [Gärdenfors, 1988], a belief set is simply a deductively closed consistent set of sentences. Revision is defined by means of an explicit order of epistemic entrenchment (Gärdenfors and Makinson [Gärdenfors and Makinson, 1988]). When a new sentence is added to the belief set, possible inconsistencies should be removed by first giving up beliefs with the lowest level of entrenchment. This kind of approach is known as the coherence approach. The axiomatic style of belief revision will be discussed in Section 2.1.3.6. Precursors of this approach are logics of theory change where belief expansion, retraction and revision are modelled by means of manipulations of theories (deductively closed sets of sentences), without any further ordering on such sets.
Modal or ‘Kripkean’ approaches to belief revision are presented in Van Bentham [Bentham, 1989], Fuhrmann [Fuhrmann, 1991], Van Bentham [Benthem, 1991b] and De Rijke [Rijke, 1992]. The modal approach employs Kripke models with a relation $\sqsubseteq$ of increase of information. Changes of belief are now interpreted as leaps inside a model of possible worlds. Retraction gets modelled by moving back along the $\sqsubseteq$ ordering, expansion by moving in the forward direction. As before, revision is rendered as a combination of retraction and expansion.

A constructive approach to belief revision is proposed in Pearce and Rautenberg [Pearce and Rautenberg, 1991]. In Jaspars [Jaspars, 1994] a combination of the modal and constructive approach is proposed, and various calculi for modal/constructive belief revision are defined. In fact, this constructive dynamic approach is quite closely related to to the ‘possible world dynamics’ based on the $\sqsubseteq$ information ordering, the only difference being that instead of classical worlds partial worlds are employed as the primitive carriers of information. The flow of information is then a matter of extending and reducing the content of such partial states.

The Truth Maintenance approach to belief revision proposed in Doyle [Doyle, 1979] differs from Gärdenfors style belief revision systems in the fact that maintenance systems also keep track of the justifications of beliefs. This additional information is put to use to retract beliefs in a sensible way. This style is known as the foundation approach, and its relation to coherence style systems is spelled out in Doyle [Doyle, 1992].

A possible formal combination of these two different perspectives is given by theories of belief base revision [Hansson, 1991] [Nebel, 1992]. In such theories a minimal restructuring of belief sets is proposed by means of selecting a finite base of axioms of the beliefs. Contraction and revision are then defined on the basis of this base selection. This set contains the ground arguments of the beliefs an agent has, and therefore, rearrangement of this belief base is the core engine of belief dynamics as contraction and revision. This kind of base dynamics has also been proposed in different theories of conditionals, and is known in linguistics as premise semantics [Veltman, 1976] [Kratzer, 1979] and [Ginsberg, 1986].

A logical framework which also treats justifications as first-class citizens is so-called constructive type theory (CTT). Although CTT is not very well known in this connection, this framework has clear advantages for belief representation. CTT is meant to represent ‘mathematical discourse’, that is, it provides an inference system for checking mathematical proofs (De Bruyn [Bruyn, 1980]). For this purpose, it keeps track of the mathematician along the way. Ranta [Ranta, To appear] has advocated the use of this mathematically well-defined formalism for linguistic representation. A simple encoding of Kamp’s discourse representation theory in Huet and Coquand’s calculus of constructions is given in Ahn and Kolb [Ahn and Kolb, 1990].

An important facility of CTT for general epistemic reasons is its registration of proofs-as-objects. This explicit representation provides reasoning about justification such as in maintenance systems. Propaganda for CTT as an instrument for general epistemic use is made in Ahn [Ahn, 1992], Borghuis [Borghuis, 1992] and Van Benthem [Benthem, 1994]. The additional argumentation structure provided by type theory has the advantage over semantics in terms of possible worlds that it allows one to keep track of justifications as well as contents.
We must add immediately that comparing CTT with Kripke semantics is perhaps not quite fair. Possible worlds are meant as semantic representation, while CTT is purely inferential. From our point of view, the two are in fact good neighbours.

We think that CTT’s can be very useful for computational semantics, not only for the fundamental reasons given above, but also for pragmatic reasons. In the field of CTT a lot of research has been invested in computational issues, e.g. automated proof-checkers and theorem-provers have been developed.\(^4\)

### 2.1.2.6 Constructivism and Information Growth

Constructive logic arose from subjectivist philosophies on the foundations of mathematics. Instead of taking mathematics to be about some bivalent external ‘Platonic’ world, constructivists take mathematics as a purely internal human creation: it’s all in the mind of the mathematician. In this perspective, the notion of truth of a proposition \(\phi\) is defined as the current presence of a construction which demonstrates \(\phi\). This subjectivistic conception of truth which includes a temporal dimension, as the mathematician may discover more proofs in the course of time, makes the underlying constructive logics highly intensional and dynamic.

Probably the most well-known constructivistic philosophy of mathematical reasoning is Brouwer’s *intuitionism*. It is purely inspired on the notion of *proof*. Heyting’s formalization of the reasoning of Brouwer’s creative subject is called intuitionistic logic (Heyting [Heyting, 1956]). Kripke gave it a clear intensional truth-conditional semantics by means of possible worlds models (Fitting [Fitting, 1969]). The structure of these models reappears in many information oriented approaches to formal reasoning. It consists of a non-empty set of states of information and a temporal order over these states.

Certain connectives have an intensional reading in intuitionistic logic. This becomes clear from the so-called Brouwer-Heyting-Kolmogorov (BHK) interpretation (see e.g. Troelstra and Van Dalen [Troelstra and van Dalen, 1988]). In the first-order version of intuitionistic logic, negation, implication and universal quantification are intensional. The meaning of the negation of a proposition \(\phi\), for example, is defined as the current availability of a method or function to transform any hypothetical proof of \(\phi\) into a proof of \(\bot\) (the absurd proposition).

So, if \(s, s'\) are states and \(\sqsubseteq\) the temporal order, then

\[
s \models \neg \phi \iff s' \not\models \phi \text{ for all } s' \sqsubseteq s.
\]

The intensionality of the conceivability of the proof of \(\phi\) is caught by the temporal order \(\sqsubseteq\). Implication is defined in a similar way. According to the BHK interpretation, the proof of an implication is a function which maps any hypothetical proof of the antecedent onto a proof of the conclusion:

\[
s \models \phi \rightarrow \psi \iff [s' \models \phi \Rightarrow s' \models \psi] \text{ for all } s' \sqsubseteq s.
\]

\(^4\)Compare the extensive research on interactive CTT-theorem-provers in the ESPRIT project ‘TYPES’.
Nelson [Nelson, 1949] defined an interesting extension of intuitionistic logic. Many objections have been raised against the intensional conception of negative information in intuitionistic logic. In many constructivistically acceptable proofs, extensional negative information is employed (Lakatos [Lakatos, 1976]). Nelson used the notion of refutation as a symmetric concept of the only BHK-construction 'proof'. A slight modification of the possible worlds semantics is needed to accommodate this 'negative' construction (see e.g. Gurevich [Gurevich, 1977]). A Kripke style semantics for Nelson's first-order logic [Nelson, 1959] is given in Akama [Akama, 1988]. Allowing negative constructions boils down to enriching states with an additional notion of 'local falsity'.

The Kripke-style reformulation of intuitionistic and constructive logic represents a model-theoretic perspective on information flow. It is fair to say that so far this perspective has not exercised much influence in linguistics. One example of a connection is Veltman's data semantics [Veltman, 1985] (see also Landman [Landman, 1986]). In data semantics Nelson style information models are used to interpret natural language conditionals. In this analysis, truth of conditionals is handled in the same way as intuitionistic implication. Falsity of a conditional is taken to mean that we can extend the current information state with both the truth of the antecedent and the falsity of the conclusion. (Intuitively, this analysis says that a situation which denies the conditional is 'conceivable'.)

In models for data semantics, the information structure $\mathbb{I}$ should not be taken as a temporal order but as an information extension order, just like in dynamic modal logic (DML). Jaspars [Jaspars, 1994] contains a proposal to extend data logic with connectives for reasoning about retraction of information.

2.1.3 Existing Systems

In this section we briefly survey some existing update and dynamic logics which have been applied in natural language semantics.

2.1.3.1 Update Logic

Let us first look at information updating in a very simple system, basic Update Logic, due to Veltman [Veltman, 1991]. In this logic, current information is represented as the set of propositional valuations ('worlds') that agree with what I know or believe. In the following language description, $p$ ranges over a set $P$ of atomic propositions.

$$\text{UL } \pi ::= p | \pi \cdot \pi | \pi \cup \pi | \neg \pi | M \pi.$$  

The semantics for Update Logic is given in terms of updates of context sets, where a context set $I$ is a subset of the set $W$ of all propositional valuations for $P$. We write the result of
updating context set $I$ with the information $\pi$ as $I[\pi]$.

\[
\begin{align*}
I[p] &= \{ w \in I \mid w \models p \} \\
I[\pi_1; \pi_2] &= I[\pi_1][\pi_2] \\
I[\pi_1 \cup \pi_2] &= I[\pi_1] \cup I[\pi_2] \\
I[\neg \pi] &= I - I[\pi] \\
I[M \pi] &= \begin{cases} I & \text{if } I[\pi] \neq \emptyset \\ \emptyset & \text{otherwise.} \end{cases}
\end{align*}
\]

The state change effected by $\pi$ in state $I$ can be pictured as $I \xrightarrow{\pi} I[\pi]$. A simple induction argument shows that for all $I$ and all $\pi$: $I[\pi] \subseteq I$ (eliminativity of Update Logic). Also, the transition system for information updating is deterministic: for a given $I$, $\pi$ there is always precisely one $J$ with $I \xrightarrow{\pi} J$, to wit $J = I[\pi]$. This determinism is built into the notation.

What makes Update Logic interesting is the ‘consistency’ operator $M$ (for: ‘maybe’). The following examples illustrate that $M$ causes non-commutativity of sequential updating:

\[
\{pq, p\bar{q}, \bar{p}q, \bar{p}\bar{q}\} \xrightarrow{M_p} \{\bar{p}q, \bar{p}\bar{q}\} \xrightarrow{M_p \bar{p}} \emptyset.
\]

\[
\{pq, p\bar{q}, \bar{p}q, \bar{p}\bar{q}\} \xrightarrow{M_p} \{pq, p\bar{q}, \bar{p}q, \bar{p}\bar{q}\} \xrightarrow{M_p \bar{p}} \{\bar{p}q, \bar{p}\bar{q}\}.
\]

A notion of consequence for UL is the following: $\pi$ logically follows from $\pi_1, \ldots, \pi_n$ (in that order), notation $\pi_1; \ldots; \pi_n \models \pi$ iff for all information states $I$, $I[\pi_1; \ldots; \pi_n][\pi] = I[\pi_1; \ldots; \pi_n]$. In other words, after updating with $\pi_1, \ldots, \pi_n$, in that order, an update with $\pi$ does not change the information state any further. It is clear that this consequence relation is not monotonic: adding extra premises to the right can spoil inferences. An example is: $Mp \models Mp$ versus $Mp; \neg p \not\models Mp$.

One slightly awkward feature of the UL system that we have presented is that embedded occurrences of $M$ may behave rather strangely. Define $\pi_1 \cap \pi_2$ as $\neg(\neg \pi_1 \cup \neg \pi_2)$. This stipulation boils down to the following semantic clause for $\cap$:

\[
I[\pi_1 \cap \pi_2] = I[\pi_1] \cap I[\pi_2].
\]

For an update like $p \cap q$ this has the required effect of throwing away all but the worlds which are both $p$ and $q$ worlds. But now consider $p \cap M \neg p$. This update has the strange property that it is not idempotent, in the sense that we do not have $I[p \cap M \neg p] = I[p \cap M \neg p][p \cap M \neg p]$, and therefore:

\[
p \cap M \neg p \not\models p \cap M \neg p.
\]

One may restore idempotency of the consequence relation by restricting the language in such a way that $M$ can only be out-scoped by $;$ (this is how Veltman solves the problem). A logical question here is: what is the largest fragment of the language of UL for which the update consequence relation is idempotent?

Another possible constraint on updates is that $J \subseteq I$, $J \neq \emptyset$ and $I[\pi] = J$ together imply that $I[\pi] \neq \emptyset$. This is different from idempotency, and indeed $p \cap M \neg p$ satisfies it trivially;
as the only $J$ with $J[p \cap M \neg p] = J$ happens to be $\emptyset$. But $\neg M p$ does not satisfy it, as:

$$\{\bar{p}\} \xrightarrow{\neg M p} \{\bar{p}\}$$

$$\{p, \bar{p}\} \xrightarrow{\neg M p} \emptyset.$$  

Again there is a logical question: what is the largest fragment of the UL language satisfying this principle?

### 2.1.3.2 An Assertion Logic for UL

It is quite easy to build an assertion logic for Update Logic, linking the dynamic perspective on information updating to a static perspective (see Van Benthem [Benthem, 1989; Benthem, 1991a] and Van Eijck and De Vries [Eijck and de Vries, 1993]).

**AUL** $\phi ::= p \mid (\phi \land \phi) \mid \neg \phi \mid \Diamond \phi \mid \text{NS}(\phi, \pi)$.  

Here $p$ ranges over the set $P$ of propositional variables for UL, and $\pi$ over the set of UL procedures. We assume the usual abbreviations: $\phi \lor \psi ::= \neg (\neg \phi \land \neg \psi)$, $\phi \rightarrow \psi ::= \neg (\phi \land \neg \psi)$, $\phi \leftrightarrow \psi ::= (\phi \rightarrow \psi) \land (\psi \rightarrow \phi)$, $\bot ::= p \land \neg p$, $\top ::= \neg \bot$, $\square \phi ::= \neg \Diamond \neg \phi$.

The NS assertion ties the knot between modal propositional logic and Update Logic. We define the semantics of AUL as a function $||\phi||_I$ that gives the interpretation of $\phi$ in state $I$ (the reference set $I$ is necessary to make sense of the modality).

$$||p||_I = \{w \in I \mid w \models p\}$$

$$||\neg \phi||_I = I - ||\phi||_I$$

$$||\phi \land \psi||_I = ||\phi||_I \cap ||\psi||_I$$

$$||\Diamond \phi||_I = \begin{cases} I & \text{if } ||\phi||_I \neq \emptyset \\ \emptyset & \text{otherwise} \end{cases}$$

$$||\text{NS}(\phi, \pi)||_I = ||\phi||_{I[\pi]}.$$

In particular, it follows from this definition that $||\text{NS}(\top, \pi)||_I = I[\pi]$. NS is short for 'next state': $\text{NS}(\phi, \pi)$ is the formula characterizing the next state that one gets by updating a $\phi$ state with information $\pi$.

Here are the axioms schemes linking UL and AUL:

**A 2.1.3.1** $\text{NS}(\phi, p) \rightarrow (\phi \land p)$

**A 2.1.3.2** $\text{NS}(\phi, \pi_1; \pi_2) \rightarrow (\text{NS}(\phi, \pi_1), \pi_2)$.

**A 2.1.3.3** $\text{NS}(\phi, \pi_1 \cup \pi_2) \rightarrow (\text{NS}(\phi, \pi_1) \lor \text{NS}(\phi, \pi_2))$.  

92
A 2.1.3.4 $\text{NS}(\phi, \neg \pi) \leftrightarrow (\phi \land \neg \text{NS}(\phi, \pi))$.

A 2.1.3.5 $\text{NS}(\phi, M \pi) \leftrightarrow (\phi \land \Diamond \text{NS}(\phi, \pi))$.

It is left to the reader to check that these axiom schemes are sound, in the sense that for every scheme $\phi \rightarrow \psi$ and for every information state $I$ it holds that $\|\phi\|_I = \|\psi\|_I$.

Here are two examples of applications of these schemes:

$$\text{NS}(\top, Mp, \neg p)$$

$$\ns$$

For given $I$, the clauses for $\|\| \|$ now give:

$$\|\Diamond p \land \neg p\|_I = \begin{cases} I \setminus \{w \in I \mid w \models p\} & \text{if } \{w \in I \mid w \models p\} \neq \emptyset \\ \emptyset & \text{otherwise.} \end{cases}$$

$$\text{NS}(\top, \neg p; Mp)$$

For every $I$, $\|\bot\|_I = \emptyset$.

Note that in this analysis, $M$ gets related to the $S5$ modality $\Diamond$ ($S5$ is the modality with the universal accessibility relation).

2.1.3.3 Dynamic Predicate Logic

Dynamic predicate logic (DPL) is a dynamic variant of first order predicate logic that has been proposed as a medium for natural language representation because of its dynamic way of handling variable binding (Groenendijk and Stokhof [Groenendijk and Stokhof, 1991a]). Basically, DPL is the result of replacing existential quantification over a variable $x$ by random assignment to variable $x$, and conjunction of formulas by sequential composition. A man
walked in. He sat down can be translated as \( x := ?; M x; W x; S x \). This avoids the problem of variable binding that occurs if existential quantifiers and ordinary conjunction are used: \( \exists x (M x \land W x) \land S x \). Apart from these changes, DPL procedures are like first order formulas, which means that they can be negated. To a computer scientist the concept of the negation of a procedure \( \pi \) may seem strange, but in fact, \( \neg \pi \) simply expresses a test: \( \neg \pi \) succeeds in precisely those states from which there are no \( \pi \) transitions.

Let \( C \) be a set of constants, \( V \) a set of variables, and assume \( C \cap V = \emptyset, c \in C, v \in V \).

**DPL terms** \( t := c \mid v \).

Assume a set of relation symbols \( R \) with arities.

**DPL procedures** \( \pi ::= t \doteq t \mid Rt \cdots t \mid (\pi; \pi) \mid (\neg \pi) \mid \nu v \mid \nu v : \pi \).

\( \nu v \) is our notation for \( v := ? \). An indefinite noun phrase like *a man* will be represented in DPL as a sequential composition of \( \eta x \) and \( M x \). We use \( \nu v : \pi \) for definite assignment, a procedure to interpret definite descriptions (see below). While *a man* can be decomposed with ‘;’, such a decomposition is impossible for \( \nu x : K x \) (the DPL translation of the definite noun phrase *the king*), as the reader can check below.

DPL procedures are interpreted in ordinary first order models, with assignments (mappings from the set of variables into the universe of the model) functioning as states. Let \( M = \langle U, I \rangle \) be a first order model, and let \( S = U^V \) (the set of assignments for the model). We use \( s(v|d) \) for the assignment which is like \( s \) except for the possible difference that \( v \) is mapped to \( d \).

The semantic clauses are dynamic, i.e., they define a two-place relation on the set of states:

1. \( M, s, s' \models Rt_1 \cdots t_n \) iff \( s = s' \) and \( M \models s, M \models t_1 \cdots t_n \).
2. \( M, s, s' \models t_1 \doteq t_2 \) iff \( s = s' \) and \( M \models t_1 = t_2 \).
3. \( M, s, s' \models \pi_1; \pi_2 \) iff there is an \( s'' \) with \( M, s, s'' \models \pi_1 \) and \( M, s'', s' \models \pi_2 \).
4. \( M, s, s' \models \neg \pi \) iff \( s = s' \) and there is no \( s'' \) with \( M, s, s'' \models \pi \).
5. \( M, s, s' \models \nu v \) iff there is some \( d \in U \) with \( s' = s(v|d) \).
6. \( M, s, s' \models \nu v : \pi \) iff
   - there is a \( d \in U \) for which \( M, s(v|d), s' \models \pi \),
   - there is a unique \( d \in U \) for which \( M, s(v|d), s'' \models \pi \) for some \( s'' \).

Note that atomic procedures \( Rt_1 \cdots t_n \) and \( t_1 \doteq t_2 \) are interpreted as tests. The procedures \( \nu v \) and \( \nu v : \pi \) do change assignments. Of these, \( \nu v \) is indeterministic (provided the model has size \( \geq 2 \)). The procedure \( \nu v : \pi \) is deterministic iff \( \pi \) is deterministic.
A procedure for so-called ‘dynamic implication’ is defined as follows: \( \pi_1 \Rightarrow \pi_2 \) abbreviates \( \neg(\pi_1; \neg\pi_2) \). It follows from the semantic clauses above that its semantics is given by:

- \( M, s, s' \models \pi_1 \Rightarrow \pi_2 \) iff
  - \( s = s' \), and
  - for all \( s'' \) with \( M, s, s'' \models \pi_1 \) there is an \( s''' \) with \( M, s'', s''' \models \pi_2 \).

Example sentences of DPL which have been proposed for the analysis of intensional anaphoric links and for donkey pronouns are the following:

\[
\pi \equiv Fx; \eta y; D y; O x y; B x y.
\]

The first of these is the DPL translation of \textit{Some farmer owns a donkey. He beats it.}, while the second translates \textit{If a farmer owns a donkey, he beats it}. In the next section we will give a further analysis of DPL formulas like these.

### 2.1.3.4 An Assertion Logic for DPL

An assertion logic of DPL now relates the DPL procedures to first order logic. A suitable syntax for this logic is (terms and basic relations are the same as for the DPL language under consideration; \( \pi \) ranges over DPL procedures):

\[
\text{QDL} \quad \phi := t = t \mid R t \cdots \mid (\phi \land \phi) \mid (\neg \phi) \mid \exists v \phi \mid \langle \pi \rangle \phi
\]

We use the customary abbreviations for the boolean connectives, plus: \( \forall v \phi := \neg \exists v \neg \phi \), \( \exists v \phi := \exists v \forall w (w = v \rightarrow \phi[w/v]) \), \( \langle \pi \rangle \phi := \neg \langle \pi \rangle \neg \phi \), \( x \neq y := \neg (x = y) \), \( \bot := \neg \top \).

The semantic clauses for QDL are as for first order logic, with the following addition for the DPL modality:

- \( M \models_s \langle \pi \rangle \phi \) iff there is a state \( s' \) with \( M, s, s' \models \pi \) and \( M \models_{s'} \phi \).

Axiom schemes relating DPL to this assertion logic take the following shape:

A 2.1.3.6 \( \langle R t_1 \cdots t_n \rangle \phi \equiv (R t_1 \cdots t_n \land \phi) \).

A 2.1.3.7 \( \langle t_1 = t_2 \rangle \phi \equiv (t_1 = t_2 \land \phi) \).
A 2.1.3.8 \( \langle \pi_1; \pi_2 \rangle \phi \rightarrow \langle \pi_1 \rangle (\pi_2) \phi \).

A 2.1.3.9 \( \langle \neg \pi \rangle \phi \rightarrow (\neg \pi) \bot \wedge \phi \).

A 2.1.3.10 \( \langle \eta v \rangle \phi \rightarrow \exists v \phi \).

A 2.1.3.11 \( \langle \nu : \pi \rangle \phi \rightarrow (\exists v (\pi) \top \wedge \exists v (\pi) \phi \).

This axiom system for DPL is analysed in Van Eijck [Eijck, 1994]. In fact, the schemes can be used to relate DPL procedures to first order formulas, by computing the success condition \( (\pi) \top \) of a DPL procedure \( \pi \). The reader is invited to check that the following scheme for \( \Rightarrow \) is derivable from the schemes above:

\[ \langle \pi_1 \Rightarrow \pi_2 \rangle \phi \rightarrow ([\pi_1] (\pi_2) \top \wedge \phi) \]

For the following example, it is also useful to work out some duals of the schemes. Here is a calculation of the success condition of the DPL translation of *If a farmer owns a donkey, he beats it*:

\[ \langle (\eta x; Fx; \eta y; Dy; Oxy) \Rightarrow Bxy \rangle \top \]
\[ \rightarrow [\eta x; Fx; \eta y; Dy; Oxy] (Bxy) \top \]
\[ \rightarrow [\eta x][Fx][\eta y][Dy][Oxy] (Bxy) \top \]
\[ \rightarrow \forall x (Fx \rightarrow \forall y (Dy \rightarrow (Oxy \rightarrow Bxy))). \]

This is indeed a first order rendering of the meaning of the example sentence. Modulo a compositional definition of the DPL meaning representation for the example (which can easily be given using standard techniques; see Section 2.2), this illustrates how the assertion logic relates the dynamic meaning of natural language sentences to their static meaning.

The assertion logic for DPL also illustrates that the treatment of definite descriptions is just a dynamic version of Russell’s well known description theory [Russell, 1905]. Here is a computation of the success condition for the DPL translation of *Some farmer beats his donkey*:

\[ \langle \eta x; Fx; \eta y: (Dy; Oxy); Bxy \rangle \top \]
\[ \rightarrow \langle \eta x; Fx; \eta y: (Dy; Oxy); Bxy \rangle \top \]
\[ \rightarrow \langle \eta x \rangle (Fx)(\eta y: (Dy; Oxy))(Bxy) \top \]
\[ \rightarrow \exists x (Fx \wedge \exists y (Dy \wedge Oxy) \wedge \exists y (Dy \wedge Oxy \wedge Bxy)). \]

2.1.3.5 Dynamic Versions of Montague Grammar

A Montagovian version of DPL called ‘Dynamic Montague Grammar’ was proposed in [Groenendijk and Stokhof, 1990]. The peculiarity of this system is that, although intensional terminology is used (the underlying logic of the natural language fragment is called ‘Dynamic Intensional Logic’), the system does not cover the intensional phenomena that are the stock in trade of traditional Montague grammar.
Rather, the worlds in DMG are nothing but assignments of values to variables (discourse markers). Thus, Montagovian cups and caps are now used for essentially different purposes than in traditional Montague grammar, and the ‘intension’ of a discourse marker is the set of states differing only in the value of the relevant variable register.

A slightly different set-up can be found in [Muskens, 1991], where the notion of intension is similarly overloaded, but where Gallin’s [Gallin, 1975] Ty2 rather than Montague’s IL [Montague, 1973] serves as the point of departure.

A set-up where a typed logic is built up from basic types $T$ (for transition) and $e$ (for entity), as proposed in Van Eijck and Kamp [Eijck and Kamp, 1994], avoids this overloading of the notion of intensionality. What is done here is that truth values are replaced by state transitions as basic building blocks of the system. An intensional system would add a basic type $s$ for indices or possible worlds.

What is Montagovian about systems like this is the compositional organisation of the syntax-semantics interface. In Section 2.2 we give a sketch of a system along these lines.

2.1.3.6 Belief Revision Systems

In early theories of belief revision ([Alchourron et al., 1985]) one represents beliefs as sets of sentences in a suitable logical language, and one studies ways of dealing with such sets. Three ways of dealing with a belief set $T$ in the light of changing insight in what the world is like are: (i) expansion of $T$, (ii) contraction of $T$, and (iii) revision of $T$. Of these, expansion is the simplest, of course. Expansion happens in case we learn a new fact which does in no way conflict with our current belief set $T$.

Following Gärdenfors, we assume that belief sets are closed under logical deduction, i.e. that for any belief system $T$ we have $T = \overline{T}$, where $\overline{T}$ gives the deductive closure of $T$. In other words, we assume that belief sets are theories. We would also like belief sets to be consistent, of course. A deductively closed set is consistent if there is at least one formula of the language that is not a member of it. In other words, there is only one inconsistent theory, namely the set of all formulas of the language.

Expansion of $T$ with $\phi$ (notation $T + \phi$) is the operation which maps $T$ to $\overline{T} \cup \{\phi\}$. Of course, in case $\phi$ is inconsistent with $T$, $T + \phi$ will be the inconsistent theory (all formulas of the language). The question now becomes: what should one do in such a case? Here the second way of dealing with belief comes in: in case expansion of $T$ with $\phi$ leads to inconsistency, we first have to contract $T$ before we can add $\phi$.

The operation of contracting $T$ by $\phi$ (notation $T - \phi$) consists of pruning $T$ in such a way that $\phi$ does not follow from it anymore, while the result is again a deductively closed theory. Let $T \perp \phi$ be the set of all maximal subsets of $T$ that fail to imply $\phi$. Assume $T$ is consistent and deductively closed. Then the members of $T \perp \phi$ will again be consistent and deductively closed. (If a member of $T \perp \phi$ were not deductively closed, it would not be maximal. Also,
members of \( T \perp \phi \) are consistent by definition, for they do not contain \( \phi \).

Now there are three possibilities. If \( \neg \phi \) does follow from \( T \), \( T \perp \phi = \{ T \} \). In this case \( \phi \) was not in \( T \), so we don’t have to do anything; we can take \( T - \phi = T \).

If \( \phi \in \emptyset \), i.e., if \( \phi \) is true by virtue of logic alone, then \( T \perp \phi = \emptyset \). In this case, contracting by \( \phi \) is impossible.

Finally, there is the case where \( T \perp \phi \) is neither empty nor a singleton. Now we are faced with the problem of pruning \( T \) in a reasonable way. One possibility is to set \( T - \phi \) equal to the intersection of \( \bigcap (T \perp \phi) \). Call this set \( T \sim \phi \). This option is called full meet contraction. It is easy to see that \( T \sim \phi \) is consistent and deductively closed. (for the intersection of deductively closed sets is again deductively closed). The trouble is that this way of pruning theories is perhaps too radical: it is easy to show that \( T \sim \phi \) equals \( T \cap \{ \neg \phi \} \), in other words the only propositions that are left after contraction by \( \phi \) are the members of \( T \) that are already consequences of \( \neg \phi \).

Another possibility is to replace full meet contraction by partial meet contraction, letting a choice function \( \gamma \) pick out the most important sets in \( T \perp \phi \) and taking the intersection of those: \( T - \phi = \bigcap \gamma (T \perp \phi) \). In the literature on belief revision systems one finds many proposals for definition of such selection functions.

It is not difficult to see that once one has operations for belief expansion and belief contraction, belief revision can be defined on terms of these, by defining revision with \( \phi \) as first contracting with \( \neg \phi \) and then expanding with \( \phi \), in other words by putting \( T + \phi \) (the result of revising \( T \) with \( \phi \)) equal to \( (T - \neg \phi) + \phi \). This definition is known as the Levi identity.

Investigations in belief revision formalisms are most often judged on the basis of the so-called Alchourrón–Gardenfors-Makinson (AGM) postulates [Alchourrón et al., 1985]. These principles are meant as a kind of decency axioms which ought to be verified by any definition of contraction and revision postulates. The following table presents these sixteen postulates.

In this table \( K \) stands for an arbitrary theory, while \( K \) represents the collection of all theories. \( K \perp A \) is the theory which evolves from contracting \( A \) from \( K \) and \( K* A \) is the theory that is the result from revising \( K \) by the proposition \( A \).

<table>
<thead>
<tr>
<th>Contraction</th>
<th>Revision</th>
</tr>
</thead>
<tbody>
<tr>
<td>K-1 ( \neg A \in K )</td>
<td>K<em>1 ( K</em>A \in K )</td>
</tr>
<tr>
<td>K-2 ( \neg A \subseteq K )</td>
<td>K<em>2 ( A \in K</em>A )</td>
</tr>
<tr>
<td>K-3 ( A \not\in K \implies \neg A = K )</td>
<td>K<em>3 ( K</em>A \subseteq K+A )</td>
</tr>
<tr>
<td>K-4 ( \neg A \in A \implies \neg A \subseteq \neg K )</td>
<td>K<em>4 ( \neg A \notin K \implies K+A \subseteq K</em>A )</td>
</tr>
<tr>
<td>K-5 ( K \subseteq (K^*)+A )</td>
<td>K*5 ( K^A = K \iff A \iff \neg A )</td>
</tr>
<tr>
<td>K-6 ( A \leftrightarrow B \implies K^+-A = K^+B )</td>
<td>K*6 ( \iff A \leftrightarrow B \implies K^*A = K^*B )</td>
</tr>
<tr>
<td>K-7 ( K^+-A \cap K^-B \subseteq K^+(A \land B) )</td>
<td>K*7 ( K^+(A \land B) \subseteq K^+(A \land B) )</td>
</tr>
<tr>
<td>K-8 ( A \not\in K^-(A \land B) \implies K^-(A \land B) \subseteq K^A )</td>
<td>K*8 ( \neg B \not\in K^*A \implies (K^*A)+B \subseteq K^+(A \land B) )</td>
</tr>
</tbody>
</table>
In alternative, structurally subtler, definitions of contraction and revision, the notion of epistemic entrenchment has been employed. This entrenchment represents the preservation quality of beliefs. It is implemented over the theory representation above by means of a pre-order over the sentences in such a theory. Revision can then be defined by replacing as few as possible inconsistencies with the lowest entrenchment.

In [Gärdenfors, 1988] one finds constraints on this epistemic entrenchment which compel the revision function to meet the AGM-postulates.

### 2.1.3.7 Dynamic Modal Logics

Dynamic modal logics, as introduced by Van Benthem [Benthem, 1989] [Benthem, 1991b] and further explored by De Rijke [Rijke, 1992] [Rijke, 1993], are dynamic logics in the style of Pratt’s propositional dynamic logic [Pratt, 1980] and quantified dynamic logic [Pratt, 1976] where the atomic actions are taken to be expansions and reductions of information states. These actions are interpreted by means of an information structure of the form \(\langle S, \subseteq \rangle\), where \(S\) is a non-void collection of information states and \(\subseteq\) is a relation along which the information flows. Most often, this relation models the direction along which information grows, and is therefore taken to be a pre-order.\(^5\)

Such very general information frames can now be used to develop a modal setting of dynamic semantics by assigning a model-theoretic structure and a truth-conditional semantics to the states [Benthem, 1989] [Benthem, 1991b].\(^5\) If such an assignment has been defined – call it \(\sigma\) – we can specify a normal static interpretation and a dynamic interpretation of a proposition \(\phi\) with respect to an information structure \(\langle S, \subseteq \rangle\). Supplying the assignment \(\sigma : L \rightarrow \wp(S)\), where \(L\) is some given language, to the information structure is the only addition which we need to dress up \(L\) dynamically. The triple \(M = \langle S, \subseteq, \sigma \rangle\) will be called an \(L\)-information model. The static and dynamic interpretation with respect to \(M\) of a proposition \(\phi \in L\) are then given by the following simple definitions.

\[
\begin{align*}
\llbracket \phi \rrbracket_{st} & = \sigma(\phi) & \text{static meaning of } \phi \text{ (in } M) \\
\llbracket \phi \rrbracket_{dy} & = \{ \langle s, t \rangle \in S^2 \mid s \subseteq t \land t \models \phi \} & \text{dynamic meaning of } \phi \text{ (in } M) \\
\end{align*}
\]

Referring to \(M\) by some additional index for the meaning functions would have been more accurate. We leave it for reasons of readability.

Instead of \(s \in \llbracket \phi \rrbracket_{st}\) and we also write \(s \models \phi\).

In the following picture an information structure \(\langle S, \subseteq \rangle\) is depicted. The static denotation

\(^{5}\)A reflexive transitive relation.

\(^{6}\)Similar ideas have been presented in Fuhrmann’s modal update logic [Fuhrmann, 1991]. He uses the technique which is known in modal logic as general frames. Van Benthem’s information structures are somewhat more concrete, but these models are harder to axiomatize (see De Rijke’s dissertation [Rijke, 1993]).
of $\phi$, $\sigma(\phi)$, is the grey colored area. Clearly, $s \not\models \phi$, but $\langle s, t \rangle \in [\phi]_{dy}$. Informally speaking, the pairs in $[\phi]_{dy}$ model the input-output relation of the action of expanding a state with the information $\phi$.

In this setting we could speak of context sensitive interpretations. The interpretation of $\phi$ with respect to the ‘context’ $s$ is the set of states which are $\phi$-extensions of $s$. Formally, $[\phi]_s = \{ t \in S \mid \langle s, t \rangle \in \langle \phi \rangle_{dy} \}$. In the third figure of the picture above, this interpretation is represented by the intersection of the dash-lined triangle of $s$-extension and the grey area: $\sigma(\phi)$.

The clear difference of dynamic modal logical format with respect to other dynamic theories, which is advantageous from a FraCaS point of view, is that it has no specific application in mind and is therefore so general in nature that many dynamic semantic theories can be comprehended formally by substitution of specific semantic constants to the limited set of parameters of the information models of the dynamic modal setting of above. In section 2.1-4 we will shortly discuss some examples. For a further exploration of the dynamic field we wish to refer the reader to the forthcoming report [Jaspers et al., 1994]. Also De Rijke [Rijke, 1994] discusses dynamic modal logic in relation to some dynamic semantic theories.

Of course, additional definitions are required to enhance the dynamic capacity in such a way that accommodation of dynamic semantic theories in this dynamic modal setting. The most obvious additions are retraction relations and minimizations of interpretations. The former are needed to interpret loss of information, which is needed to capture belief contraction as in the Gärdenfors’ style systems of belief dynamics. In current dynamic systems for NL-semantics such ‘downward’ reasoning is sometimes needed to define ‘refreshment’ of variables. Such a negative dynamic meaning of a proposition $\phi$ is acquired in the following way:

$$[\phi]_d^{-} = \{ \langle s, t \rangle \in S^2 \mid t \subseteq s \mid t \not\in [\phi] \}.$$ 

In the picture above the pair $\langle t, s \rangle$ is a member of the negative dynamic denotation of $\phi$. Analogous to the context sensitive reading $[\phi]_s$ we can define the negative context sensitive interpretation of $\phi$: $[\phi]_s^{-} := \{ t \in S \mid \langle s, t \rangle \in [\phi]_d^{-} \}$.

Minimal interpretations are needed to interpret propositions as updates, which is most often defined as a minimal expansion. The minimal variant of retraction, also known as ‘downdates’, are needed to find satisfactory denotations for operations as contraction and revision in belief dynamic systems, but also for the above mentioned variable dynamics. The following table presents the minimal definitions in a formal manner.

$$[\phi]_s^\prime = \{ s \in [\phi] | \forall t \in [\phi] : t \subseteq s \Rightarrow s \not\subseteq t \} \quad \text{the minimal stating meaning of } \phi$$
$$[\phi]_dy = \{ \langle s, t \rangle \in [\phi] | \forall u : u \subseteq t \Rightarrow t \not\subseteq u \} \quad \text{the minimal dynamic meaning of } \phi$$
$$[\phi]_dy^\prime = \{ \langle s, t \rangle \in [\phi] | \forall u : u \subseteq t \Rightarrow u \not\subseteq t \} \quad \text{the minimal negative dynamic meaning of } \phi$$

The two lower figures in the picture above displays $\langle s, u \rangle$ and $\langle s, v \rangle$ as members of the minimal dynamic interpretation of $\phi$. The pair $\langle t, w \rangle$ is an element of the minimal negative dynamic denotation of $\phi$. 

100
Explicit dynamic modal logical means is supplied by extending the basic language \( L \) with modal up- and down-operators that enable reasoning over the relations of dynamic interpretation. For example:

\[
[[\phi]]_u \psi = \{ s \in S \mid [\phi]_s \subseteq [\psi]_s \} \\
[[\phi]]_d \psi = \{ s \in S \mid [\phi]_s \cap [\psi]_s \neq \emptyset \}
\]

Informally, \([\phi]_u \psi\) is true in a state \( s \) iff extending it with \( \phi \) leads necessarily to a \( \psi \)-state. The second is the dual modal formula: replace ‘necessarily’ by ‘possibly’ in the last sentence. In a similar way we can define down-operators, \([\phi]_d \) and \((\phi)_d \) over the relation \([\phi]_d \). Minimal variants of these dynamic operators, \([\phi]_u^\mu, (\phi)_u^\mu, [\phi]_d^\mu, (\phi)_d^\mu \) are interpreted in the same way over the relations \([\phi]_d^\mu \) and \([\phi]_d^\mu \). Explicit reference to minimal statics is enabled by the \( \mu \)-connective: \( [\mu(\phi)]_u := [\phi]_d^{\mu}[\phi]. \) Informally, \( \mu(\phi) \) holds if the current information state is a \( \phi \)-state which cannot be reduced without losing \( \phi \).

Van Benthem and De Rijke use many additional constructions over relations in order to give the logics more expressive capacity. In analogy with Pratt’s logics, they use union and composition of relations. Furthermore, complement and intersection have been employed there. The basic language, called \( L \) above, is just the ordinary propositional language as in ordinary modal logic, with the standard decomposition. The discrepancy of DML’s relational wealth and its relatively poor basic static input is due to the focus on the ‘new’ relational part of DML. In many dynamic semantic theories, especially those in NL-semantics, the additional relational constructions are not used. The dynamic modal differences between dynamic semantic theories, especially those in NL-semantics, usually boils down to variation of the basic static input (\( L \)) and their corresponding ‘state-semantics’ (see [Jaspars et al., 1994] and section 2.1.4 below).

### 2.1.3.8 Information states and argument structure

In the dynamic up- and down-reasoning formalisms we did not consider the means by which information is supported or rejected. In the denotational model-theoretic tradition these structures are most often abstracted away by means of truth-value assignments.

In so-called reason or truth maintenance systems justifications or arguments are treated as the carriers of information and therefore as first-class citizens [Doyle, 1979]. Retraction or revision of information is seen as a consequence of canceling or replacement of their underlying justification. Once all justifications of a proposition are removed the reasoning agent has to give up his belief in this proposition. This more procedural kind of approach to belief dynamics is known as foundational.

In Gärdenfors [Gärdenfors, 1990] and Harman [Harman, 1986] it has been argued that the

\[
\sigma(\neg \phi) = S \setminus \sigma(\phi) \quad \text{and} \quad \sigma(\phi \wedge \psi) = \sigma(\phi) \cap \sigma(\psi). \]

De Rijke uses \( \sigma'(\phi) = \{ t \in T \mid \forall s \in S : s \subseteq t \Rightarrow t \in \sigma(\phi) \} \) as if \( \subseteq \) were an epistemic accessibility relation. The advantage of \( \sigma' \) is that knowledge or information grows with the information order indeed: \( s \subseteq t \wedge \sigma(\phi) \Rightarrow t \in \sigma'(\phi) \).

---

\(^7\)For primitive proposition \( p \), \( \sigma(p) \) is chosen arbitrarily, \( \sigma(\neg \phi) = S \setminus \sigma(\phi) \) and \( \sigma(\phi \wedge \psi) = \sigma(\phi) \cap \sigma(\psi) \). De Rijke uses \( \sigma'(\phi) = \{ s \in S \mid \forall t \in S : s \subseteq t \Rightarrow t \in \sigma(\phi) \} \) as if \( \subseteq \) were an epistemic accessibility relation. The advantage of \( \sigma' \) is that knowledge or information grows with the information order indeed: \( s \subseteq t \wedge \sigma(\phi) \Rightarrow t \in \sigma'(\phi) \).
foundations and coherence approach are incompatible. They prefer the latter over the former style of belief dynamics for reasons of psychological realism and complexity. Gärdenfors also claims that argumentation structure can be described in coherence theories of belief revision using the notion of epistemic entrenchment (see subsection 2.1.3.6).

Doyle [Doyle, 1992] is more optimistic about the compatibility of these different perspectives. He refutes the psychological and economic criticisms of [Gärdenfors, 1990] and [Harman, 1986] against reason maintenance theories, and maintains that recent developments of both sides show that the theories are converging more and more.

From our point of view, the difference of foundations and coherence Status: R approaches is rather a matter of backgrounds and their related traditions. The coherence approach originates from formal philosophy and logic, while the truth maintenance formalisms are historically rooted in artificial intelligence and applied computer science. This explains to a large extent why the former relies on a denotational truth-conditional model-theoretic approach, while the latter is more interested in procedural representation. We think that the earlier mentioned constructive type theories may turn out to be a point of convergence for these different styles of belief dynamics.

2.1.4 Towards a Common Framework for Dynamic Semantics

In the previous section we have seen different dynamic semantic theories which diverge both in logical style and their intended applications. Roughly speaking, the three main directions of application are computer science, formal cognitive science and natural language semantics. Of course, within those fields divergence of different theories also emerge from further specific applications. From the viewpoint of the FraCaS-project, this wide spectrum of dynamic semantics requires a step towards a more unified format among dynamic theories themselves, before we can start thinking about unification with other formal theories of NL-semantics.

2.1.4.1 Modal re-styling of dynamic semantic theories

As promised in subsection 2.1.3.7 we will illustrate in this section a small series of examples of embeddings of different dynamic theories into the format of dynamic modal logic. We have chosen for three different theories: intuitionistic logic as a component of constructive mathematical reasoning, Veltman’s semantics for the simple might-language as a representative of update logics, and DRT as a dynamic logical exponent of discourse logics. They have been chosen, because they all have shown their importance for NL-semantics. In fact, this embedding boils down to a specification of the basic static language input $L$, its state-semantics, i.e. a definition of information states ($S$) and a $L$-specification over these states ($\sigma : L \rightarrow \wp S$), and an implementation of the notion of information growth over the information states ($\Box$).
Intuitionistic logic  Intuitionistic propositional logic can be seen as one of the most simple logics over such information structures. The basic static language ($L$) consists of a set of propositional variables $I_P$ and falsum $\bot$, and is closed under disjunction and conjunction, i.e. the smallest superset of $I_P \cup \{\bot\}$ such that for all $\phi, \psi \in L$ also $\phi \lor \psi \in L$ and $\phi \land \psi \in L$. A static specification $\sigma$ of $L$ over an info structure $\langle S, \sqsupset \rangle$ is taken to be strict, monotonic and compositional. The last property means that the $\sigma$-values of complex $L$-propositions are completely determined by the $\sigma$-values of the atoms $I_P \cup \{\bot\}$. As in classical logic, disjunctions decompose as unions and conjunction as intersections:

$$\sigma(\phi \lor \psi) = \sigma(\phi) \lor \sigma(\psi)$$

$$\sigma(\phi \land \psi) = \sigma(\phi) \land \sigma(\psi)$$

Strictness simply says that $\sigma(\bot) = \emptyset$ and monotonicity means that the $\sigma$-values of propositional variables are closed under information growth ($\subseteq$):

$$s \in \sigma(p) \& s \subseteq t \implies t \in \sigma(p) \text{ for all } p \in I_P.$$ Intuitionistic propositional logic is then the extension of $L$ with the operators $[\phi]_w$. The intuitionistic implication $\phi \rightarrow_i \psi$ coincides with $[\phi]_w \psi$ and the intuitionistic negation $\neg_i \phi$ is simply $[\phi]_w \bot$.

Update semantics  For implementation of Veltman’s update semantics for the $might$-language, we need one information structure. The collection of information states $S$ is the collection of all sets of classical valuations, that is $S(I_P \rightarrow \{0, 1\})$ with $I_P$ the set of propositional variables. The information order is the ‘eliminative’ superset order (information grows by elimination of alternative valuations). The basic language $L$ consists of the ordinary propositional language $L_0$ with an additional $might$-operator which may only occur in front of propositions.

$$L_0 \quad \phi ::= p \mid \neg \phi \mid \phi \land \phi$$

$$L ::= L_0 \cup \{ \text{might } \phi \mid \phi \in L_0 \}$$

The semantic specification $\sigma : L \rightarrow \wp S$ is completely determined by the valuations.

$$\sigma(\phi) = \{ I \in S \mid \forall V \in I : V(\phi) = 1 \}$$

$$\sigma(\text{might } \phi) = \{ I \in S \mid \exists V \in I : V(\phi) = 1 \} \text{ for all } \phi \in L_0.$$ Over the only model $\langle S, \sqsupset, \sigma \rangle$, the dynamic definitions of Veltman are recaptured by means of the minimal update definitions in the dynamic modal logical style. In other words, a static truth-conditional semantics for the $might$-language gives rise to an equivalent dynamic modal definition on the basis of the structure of eliminative dynamics.
In this definition \( I[\phi] \) is the set of valuations \( \{ V \in I \mid V(\phi) = 1 \} \).

In this simple logic no dynamic up- and down-operators are defined. The only notion which employs dynamics are Veltman's definitions of dynamic entailment. For the most general one, we only need the \([\phi]^u_d\)-operators:

\[ \phi_1 \cdots \phi_n \Rightarrow \psi := [\phi_1]^u_d \cdots [\phi_n]^u_d \psi. \]

In other words, after updating with \( \phi_1 \cdots \phi_n \) consecutively we end up in a \( \psi \)-state. The stricter notion of dynamic entailment, which relates this updating procedure to the initial state (the complete set \( \mathcal{P} \rightarrow \{0, 1\} \)), requires the use of minimal static meanings:

\[ \phi_1 \cdots \phi_n \Rightarrow' \psi := \mu(p \lor \neg p) \land [\phi_1]^u_d \cdots [\phi_n]^u_d \psi. \]

Note that updates for this language are indeed functional (deterministic), and that their interpretation coincide with the dynamic interpretation of the corresponding sublanguage of UL in subsection 2.1.3.1. Extending Veltman's language like the update language of subsection 2.1.3.1 amounts to re-interpretation of the troublesome cases found there.

\[ [p \land \text{might } \neg p]^u_d = \{ \langle I, \emptyset \rangle \mid I \in S \} \]

Other dynamic modal re-interpretation of troublesome updates reappear non-deterministic (relational). An example is \( \neg\text{might } p \lor q \):

\[ \langle \{\neg q, \neg p\}, I \rangle \in [\neg\text{might } p \lor q]^u_d \iff I = \{\neg q\} \text{ or } I = \{\neg p\}. \]

Also minimal retractions would appear relational:

\[ [p]^d_{\neg} := \{ \langle I, I \rangle \mid \exists V \in I : V(p) = 0 \} \cup \{ \langle I, I \uplus \{V\} \rangle \mid I \in [p]_{st}, V(p) = 0 \}. \]

**Discourse Representation Theory** Assigning the semantic variations of first-order logic to the information states accommodates adaptation of dynamic semantic theories such as DRT in a dynamic modal logical setting. In these semantic theories, only variable assignments are taken to be changeable, while the domain of individuals \( D \) and the interpretation function \( I \) are fixed over the universe of information states.
$SD,I = \{ h \mid h : \text{VAR} \sim D \}.^8$

The dynamic structure is then the extension relation between partial assignments:

\[
g \subseteq h \iff \forall y \in \text{Dom}(g) : h(y) = g(y).^9
\]

The static input language $L$ is the set of atoms $P(t_1, \ldots, t_n)$ with $P$ an $n$-ary predicate and $t_1 \ldots t_n$ terms, and, the semantic specification $\sigma : L \rightarrow \wp SD,I$ is defined in the classical fashion:

\[
\sigma(P(t_1 \ldots t_n)) = \{ h \in SD,I \mid \langle h_{D,I}(t_1) \ldots h_{D,I}(t_n) \rangle \in I(P) \}\]

with

\[
h_{D,I}(t_i) = \begin{cases} 
  h(t_i) & \text{if } t_i \text{ is a variable,} \\
  I(f)(h_{D,I}(u_1 \ldots u_m)) & \text{if } t_i = f(u_1 \ldots u_m)
\end{cases}
\]

An introduction of an indefinite $\phi(x_1 \ldots x_n)$, where $x_1 \ldots x_n$ are the variables occurring in $\phi$ is then interpreted dynamically, i.e. $[\phi]^\mu_{dy}$. This relation precisely describes the increase of the variable domain with $x_1 \ldots x_n$ (see also subsection 2.1.2.3) such that $\phi$ becomes true:

\[
[\phi]^\mu_{dy} = \{ \langle g, h \rangle \mid g, h \in SD,I \& g \leq x_1 \ldots x_n \& \& D, I, h \models \phi \}.
\]

‘Downward’-information can be relevant here as well. The following operation refreshes the variables $x_1 \ldots x_n$. It might be the case that we need free registers for interpretation of indefinites. A ‘tautological downdate about $x_1 \ldots x_n’$ beforehand would take care of this variable cleaning. The following composition would be satisfactory for this purpose.

\[
[\neg \phi \lor \phi]^-_{dy} \circ [\phi]^\mu_{dy}
\]

In DPL, as we saw earlier, dynamics comes with assignment switches rather than assignment growth. A revision-like definition like the one above comes close to an in-between definition. Application to total assignments would indeed yield an equivalent of the variable assignment dynamics of DPL.\(^12\)

The aim of this short section was to give a brief technical exposition of how different theories of dynamic semantics can be incorporated in dynamic modal logic by filling in more specific

\(^8\)For interpreting FCS in this fashion take $\wp SD,I$ as the information states.

\(^9\)For FCS: $G \subseteq H \iff \forall h \in H \exists y \in G : g \subseteq h$.

\(^10\)Most often it is practical to use ordinary conjunctions in $L$ as well.

\(^11\)Take the same language for FCS, and $\sigma_{FC,S}(P(t_1 \ldots t_n)) = \{ H \subseteq SD,I \mid \forall h \in H : \langle h_{D,I}(t_1) \ldots h_{D,I}(t_n) \rangle \in I(P) \}$.

\(^12\)For a discussion on switches and information growth, the reader may consult Groenendijk and Stokhof’s [Groenendijk and Stokhof, 1991b] for a comparison between their DPL and Veltman’s Update Semantics.
values to the parameters of this general semantic format \((L, S, \subseteq \text{ and } \sigma : L \rightarrow \wp(S))\). Because of the general nature of dynamic modal logic, we have restricted ourselves to its technique. A possible application which we would like to mention, however, are conditionals. Most semantic analyses of this NL-phenomenon relate to model-theoretic interpretations of expanding and reducing information states.

### 2.1.4.2 Conditionals

The traditional starting point for an account of the meaning of conditionals is Ramsey’s rule, which says the following:

The conditional ‘\(\phi \Rightarrow \psi\)’ is true whenever any consistent expansion of a ‘current stock of beliefs’ with the antecedent \(\phi\) leads to a new belief state which contains the consequence \(\psi\).

See e.g. Stalnaker [Stalnaker, 1968]. Thus, according to Ramsey’s rule, a dynamic context-sensitive semantics is required for implementing a proper evaluation of conditionals.

In theories of conditionals two kinds are distinguished. So-called indicative conditionals, where the antecedent is supposed to be consistent with respect to the current information state, can be analysed as pure expansive or update conditionals. Conditionals of the other type, whose antecedent is taken to be inconsistent with the current information state, are called counterfactuals.

In a dynamic modal logical formalism this distinction can be given a formal description once we have given proper definitions of updating and revision. What we need here are the up- and downdate operators:

\[
\phi \Rightarrow_{\text{ind}} \psi := \{T\}_u \phi \land [\phi]^u \psi, \text{ and}
\]
\[
\phi \Rightarrow_{\text{cf}} \psi := [\phi]^u \bot \land [\neg \phi]^u \land [\phi]^u \psi.
\]

In the latter case we retract \(\neg \phi\) and expand with \(\phi\) in a minimal way. This is the dynamic modal logical interpretation of ‘revision with \(\phi\)’.

For a Gärdenfors’ style analysis of conditionals the reader is advised to consult Morreau’s dissertation [Morreau, 1992].

By way of conclusion let us say something about the relation of the present perspective on semantics to other approaches, as we see it.

Dynamic predicate logic and dynamic Montague grammar were devised as compositional alternatives for discourse representation theory, so it is not surprising that there are close links to DRT. This relation has been outlined formally in different sections above.
If one defines truth with respect to partial models, there is also a link with (the simplest possible version of) Situation Semantics. For example, the semantic setting of Nelson's logic and Veltman's data semantics, which are in fact dynamic variations of partial logic, could be used as a framework for a dynamic version of situation theory. A similar link is possible to monotonic semantics, which captures 'underspecification' by means of partial model-theory. An example of a treatment of 'underspecification by ambiguity' along these lines is proposed in [van Deemter, 1990].

Dynamic semantics has been linked up to property theory in Chierchia and Turner [Chierchia and Turner, 1988]. The fine-grained semantics of property theory might be very useful to establish a more procedural dynamic theory that might replace the denotational dynamics of theories like DRT and DPL. Such a move towards procedural semantics would be especially welcome for dynamic interpretations of intensional phenomena in NL (the treatment of the attitudes).

We take it that the FraCaS project provides dynamic semantics with the challenge to define a general dynamic framework where different theories can be combined and compared. In fact, the general modal style of modelling information flow which has been discussed in sections 2.1.3.7 and 2.1.4 seems a very useful tool of unification here, as it provides us with a framework to detect and discuss formal differences between semantic theories.

## 2.2 Syntax-semantics Interface

If one wants to build a syntax/semantics interface which translates natural language sentences and texts into formulas of a dynamic representation language such as DPL, the basic approach to compositionality by means of constructing typed lambda expressions for components of sentences and reducing those with lambda conversion applies without much further ado (see e.g. Muskens [Muskens, 1991]).

In constructing a dynamic version of a Montague style compositional system of natural language interpretation, some choices have to be faced. We will opt for a logic with $e$ (entity) and $T$ (transition) as basic types, with a categorial grammar with basic categories $E$ and $S$ to match these.

To illustrate the process of constructing meaning representations for natural language fragments, we will define a sentence grammar for a toy fragment, with a matching compositional semantics.

### 2.2.1 Syntactic Component

Basic categories are $S$ (without features) and $E$, with features for case, antecedent index $i$, anaphoric index $j$. We assume the following category abbreviations:
To see how this grammar works, note that the following is a possible sentence structure according to the category definitions:

\[ S \mid NP(\cdot,i,j) \mid DET(i,j) \mid CN \mid VP(Tensed) \mid AUX \mid VP(Inf) \]

### 2.2.2 Compositional Extensional Semantics

For the semantics, we assume we have basic types \( e \) (for entities) and \( T \) (for state transitions). Higher types are built up as follows (this is the usual definition):

\[
\text{type} ::= e \mid T \mid (\text{type} \ \text{type})
\]

It is convenient to abbreviate type \((e, \ldots, (e, T), \ldots)\) as \((e^n, T)\).

The reason for taking state transitions rather than truth values as basic is that we want to create the leeway to treat certain variables of type ‘entity’ as special, in the sense that a change in their value effects a transition from one state to another.

The approach taken here is taken from Van Eijck and Kamp [Eijck and Kamp, 1994]. It differs from Muskens’ [Muskens, 1991] approach in the fact that Muskens takes states as basic and considers markers (store names) as functions from states to states, whereas we take transitions from states to states and the markers (store names) that go with them as basic and consider states as a derived notion. The advantages of our approach, as we see it, are that our general set-up is simpler, and that the notion of state (or index, or world) remains available for treating intensional phenomena in the spirit of traditional Montague grammar.

In our semantic representation language, we first populate the sets of expressions of types \( e \) and \( T \) with basic expressions of the type, and then add constants, variables and expressions formed by application and abstraction. We assume that \( x \) ranges over \( X \) (we call this the set of markers) and \( v_e \) over \( V_e \) (the set of individual variables), where \( X \cap V_e = \emptyset \).

**basic expressions of type \( e \)**

\[
b_e ::= x
\]

**basic expressions of type \( T \)**

\[
b_T ::= E_e \mid E_e \mid E_{(e^n,T)}(E_e \cdots E_e) \mid (E_T; E_T) \mid (\neg E_T) \mid \eta x \mid \eta x : E_T
\]
basic expressions of type $A \notin \{e, T\}$ $b_A ::= \Lambda$

expressions of type $A$ $E_A ::= b_A | e_A | v_A | (E_{(B,A)}, E_B)$

expressions of type $(A,B)$ $E_{(A,B)} ::= \lambda v_A.E_B$.

Note that the basic expressions of type $e$ are the DPL variables (henceforth called store names), and the basic expressions of type $T$ the DPL procedures.

Lambda conversion and reduction work as usual. Note that as markers (store names) are not used as variables in forming lambda abstracts, there is no problem with $\alpha$ conversion.

In Section 2.1-2.4 we mentioned a problem with marker clashes and anaphoric linking. The compositional approach to DRT presented in Van Eijck and Kamp [Eijck and Kamp, 1994] solves this problem by employing an operation $\bullet$ for unreduced merging of representation structures, together with merge reduction instructions that effect marker renamings to avoid possible clashes. Another possible solution is to allow stores to contain sequences of values, so that a new assignment to an old store is not a destructive operation anymore, as it just pushes a new value on the stack. See Vermeulen [Vermeulen, 1993] for details. Still another way to go is to interpret every assignment as a declaration of a local variable followed by the action of storing a value at the indicated location. Again, under this regime assignment is not a destructive action anymore. See Vermeulen [Vermeulen, December 1991] and Van Eijck and Francez [Eijck and Francez, to appear] for details. In the sequel, we will simply ignore the problem.

Figure 2.2 specifies the lexicon of our fragment. Note that $x_i, x_j$ range over markers (of type $e$), $v, u$ are used for variables of type $e$, variables $p, q$ range over type $T$, variables $P, Q$ range over type $((e, T), T)$, and variables $P$ range over type $((e, T), T)$.

The category table in the lexicon makes clear that example sentence 9 has the structure specified in 8.

9 The man who smiles does not hate John.

For convenience, we have assumed that the connective `.' serves as a discourse constructor. Example 10 gives a text which is in the fragment.

10 The man who smiles does not hate John. He respects John.

The composition of representation structures for these example sentences is a matter of routine (see Gamut [Gamut, 1991] for a didactic account of the general procedure).

We conclude with some brief remarks on the treatment of proper names and definite descriptions. Proper names do have anaphoric indices, and they are anaphorically linked to an
<table>
<thead>
<tr>
<th>expression</th>
<th>category</th>
<th>translates to</th>
<th>type</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>DET(i,*)</td>
<td>$\lambda P \lambda Q(\eta x; P(x_i); Q(x_i))$</td>
<td>$(e,T),(e,T,T)$</td>
</tr>
<tr>
<td>every</td>
<td>DET(i,*)</td>
<td>$\lambda P \lambda Q(\eta x; P(x_i)) \Rightarrow Q(x_i)$</td>
<td>$(e,T),(e,T,T)$</td>
</tr>
<tr>
<td>no</td>
<td>DET(i,*)</td>
<td>$\lambda P \lambda Q(\eta x; P(x_i); Q(x_i))$</td>
<td>$(e,T),(e,T,T)$</td>
</tr>
<tr>
<td>another</td>
<td>DET(i,j)</td>
<td>$\lambda P \lambda Q(\eta x; x_i \neq x_j; P(x_j); Q(x_i))$</td>
<td>$(e,T),(e,T,T)$</td>
</tr>
<tr>
<td>the</td>
<td>DET(i,j)</td>
<td>$\lambda P \lambda Q(i x_i : (x_i \neq x_j; P(x_j); Q(x_i))$</td>
<td>$(e,T),(e,T,T)$</td>
</tr>
<tr>
<td>his</td>
<td>DET(i,j)</td>
<td>$\lambda P \lambda Q(\eta x; : P(x_i); Q(x_i))$</td>
<td>$(e,T),(e,T,T)$</td>
</tr>
<tr>
<td>John</td>
<td>NP(<em>,</em>,i)</td>
<td>$\lambda P(j = x_i; P(x_i))$</td>
<td>$(e,T),(e,T,T)$</td>
</tr>
<tr>
<td>who</td>
<td>REL</td>
<td>$\lambda P \lambda Q \lambda v(Q(v); P(v))$</td>
<td>$(e,T),(e,T,T)$</td>
</tr>
<tr>
<td>he</td>
<td>NP(nom,*,i)</td>
<td>$\lambda P(P(x,i))$</td>
<td>$(e,T),(e,T)$</td>
</tr>
<tr>
<td>him</td>
<td>NP(acq,*,i)</td>
<td>$\lambda P(P(x,i))$</td>
<td>$(e,T),(e,T)$</td>
</tr>
<tr>
<td>man</td>
<td>CN</td>
<td>$\lambda v(man(v))$</td>
<td>$(e,T)$</td>
</tr>
<tr>
<td>boy</td>
<td>CN</td>
<td>$\lambda v(boy(v))$</td>
<td>$(e,T)$</td>
</tr>
<tr>
<td>smiles</td>
<td>VP(Tensed)</td>
<td>$\lambda v(smile(v))$</td>
<td>$(e,T)$</td>
</tr>
<tr>
<td>smile</td>
<td>VP(Inf)</td>
<td>$\lambda v(smile(v))$</td>
<td>$(e,T)$</td>
</tr>
<tr>
<td>has</td>
<td>TV(Tensed)</td>
<td>$\lambda P \lambda v(\lambda P \lambda v(poss(u,v)))$</td>
<td>$((e,T),(e,T),(e,T))$</td>
</tr>
<tr>
<td>have</td>
<td>TV(Inf)</td>
<td>$\lambda P \lambda v(\lambda P \lambda v(poss(u,v)))$</td>
<td>$((e,T),(e,T),(e,T))$</td>
</tr>
<tr>
<td>hates</td>
<td>TV(Tensed)</td>
<td>$\lambda P \lambda v(\lambda P \lambda v(hate(u,v)))$</td>
<td>$((e,T),(e,T),(e,T))$</td>
</tr>
<tr>
<td>hate</td>
<td>TV(Inf)</td>
<td>$\lambda P \lambda v(\lambda P \lambda v(hate(u,v)))$</td>
<td>$((e,T),(e,T),(e,T))$</td>
</tr>
<tr>
<td>does not</td>
<td>AUX</td>
<td>$\lambda P \lambda v(\neg P(v))$</td>
<td>$((e,T),(e,T),(e,T))$</td>
</tr>
<tr>
<td>if</td>
<td>(S/S)/S</td>
<td>$\lambda P \lambda q(p \Rightarrow q)$</td>
<td>$(T,T,T)$</td>
</tr>
<tr>
<td>.</td>
<td>S(TXT/S)</td>
<td>$\lambda P \lambda q(p; q)$</td>
<td>$(T,T,T)$</td>
</tr>
<tr>
<td>.</td>
<td>TXT(\langle\langle\txt\rangle\rangle)</td>
<td>$\lambda P \lambda q(p; q)$</td>
<td>$(T,T,T)$</td>
</tr>
</tbody>
</table>

Figure 2.2: Lexical component of a toy fragment for English.

antecedent. Of course, proper names will not always have an antecedent in the preceding discourse. But we may assume that ‘preset markers’ or ‘anchored markers’ are available that link a proper name to an antecedent outside the discourse. To link up a pronoun to a proper name, just use the same index.

Definite descriptions have both antecedent and anaphoric indices. This reflects the fact that they are often used anaphorically, as in the following example.

11 A man walked in. The guy asked for the manager.

Here the guy is anaphorically linked to a man, while the definite can in turn serve as antecedent for pronouns in subsequent discourse. Note that the uniqueness condition imposed by the $i$ operator is trivially obeyed as precisely one individual will be identical to the value of the antecedent store.

The next lexical entry for defines, with just an antecedent index, is the entry for ‘absolute’ uses of definite descriptions (see 12).

12 The queen of the Netherlands walked in.

The only change in the lexical entry for the with respect to the anaphoric sense is that the
link to an antecedent store is left out. The effect is that the definiteness operator now imposes a genuine uniqueness condition, as it should for such cases.

Finally, note that nothing rules out ‘wrong anaphoric indices’, as in the following example.

13 Every man\(^1\) has a donkey\(^2\). He\(_1\) beats it\(_2\).

This gets the following translation:

14 \((\eta x_1; \text{man } x_1 \Rightarrow (\eta x_2; \text{donkey } x_2; \text{poss } (x_1, x_2))); \text{beat } (x_1, x_2)\).

A check with the semantic clauses of DPL given in Section 2.1.3.3 shows that the values of \(x_1\) and \(x_2\) in \text{beat } (x_1, x_2)\) will not be influenced by what happens in the translation of the first sentence. This is because \(\Rightarrow\) functions as a test, and does not change the input assignment function.

Also, nothing rules out indexings where the same variable index is used again to introduce a new antecedents. Such new introductions result in destructive assignments, which effectively block off anaphoric links to previously introduced referents.

2.2.3 Compositional Intensional Semantics

We will now briefly discuss how the approach can be extended to incorporate intensional phenomena. This extension relates to the previous fragment just like intensional versions of Montague grammar formulated in \(\text{Ty2}\) (see Gallin [Gallin, 1975]) relate to extensional versions of Montague grammar.

The basic stuff of our semantics now consists of transitions (type \(T\)), indices, worlds or situations (type \(s\)) and entities (type \(e\)). This gives intensional propositions (type \((s, T)\)), intensional properties (type \((s, (e, T))\)), and so on. An intensional proposition is a function which for every world gives a transition relation (based on changing values of the registers \(x_i\) and evaluating the result \textit{in that world}). It should be noted here that in the underlying model we assume one domain of entities \(D_e\). Also, there is just one domain of transitions \(D_T\), namely the set of all two-place relations on \(D_e^X\), where \(X\) is the set of all markers. The extra ingredient we add to this is a set of worlds \(D_s\).

In the set-up of our intensional example fragment, expressions of type \((e, T)\) (extensional property denoting expressions) are replaced everywhere by expressions of type \((s, (e, T))\) (intensional property denoting expressions), and so on. For instance, the translation of an intransitive verb like \textit{walk} becomes \(\lambda a \lambda v \text{walk } (a)(v)\), the translation of a common noun \textit{boy} \(\lambda a \lambda v \text{boy } (a)(v)\).

Here we assume that variable \(a\) ranges over worlds or situations, so the translation for \textit{walks} picks in every world or situation the walkers in that world or situation, namely in world \(w\).
the set denoted by \( \lambda w \text{walk}(w)(v) \). Similarly, the translation for boy picks in every world the boys in that world.

<table>
<thead>
<tr>
<th>expression</th>
<th>category</th>
<th>translates to</th>
<th>type</th>
</tr>
</thead>
<tbody>
<tr>
<td>( a )</td>
<td>DET(i,*)</td>
<td>( \lambda P \lambda \eta \eta \xi_1; P(a)(\eta_1); Q(\eta_1)(\xi_1) )</td>
<td>((s,(e,T)),((s,(e,T)),T)))</td>
</tr>
<tr>
<td>every ( a )</td>
<td>DET(i,*)</td>
<td>( \lambda P \lambda \eta \eta \xi_1; P(\eta_1)(\eta_1); Q(\eta_1)(\eta_1) )</td>
<td>((s,(e,T)),((s,(e,T)),T)))</td>
</tr>
<tr>
<td>no ( a )</td>
<td>DET(i,*)</td>
<td>( \lambda P \lambda \eta \eta \xi_1; x_1 \neq x_1; P(\eta_1)(\eta_1); Q(\eta_1)(\eta_1) )</td>
<td>((s,(e,T)),((s,(e,T)),T)))</td>
</tr>
<tr>
<td>another ( a )</td>
<td>DET(i,*)</td>
<td>( \lambda P \lambda \eta \eta \xi_1; x_1 \neq x_1; P(\eta_1)(\eta_1); Q(\eta_1)(\eta_1) )</td>
<td>((s,(e,T)),((s,(e,T)),T)))</td>
</tr>
<tr>
<td>the ( a )</td>
<td>DET(i,*)</td>
<td>( \lambda P \lambda \eta \eta \xi_1; x_1 = x_1; P(\eta_1)(\eta_1); Q(\eta_1)(\eta_1) )</td>
<td>((s,(e,T)),((s,(e,T)),T)))</td>
</tr>
<tr>
<td>the ( a )</td>
<td>DET(i,*)</td>
<td>( \lambda P \lambda \eta \eta \xi_1; P(\eta_1)(\eta_1); Q(\eta_1)(\eta_1) )</td>
<td>((s,(e,T)),((s,(e,T)),T)))</td>
</tr>
<tr>
<td>his ( a )</td>
<td>DET(i,*)</td>
<td>( \lambda P \lambda \eta \eta \xi_1; P(\eta_1)(\eta_1); Q(\eta_1)(\eta_1) )</td>
<td>((s,(e,T)),((s,(e,T)),T)))</td>
</tr>
<tr>
<td>John ( a )</td>
<td>NP(<em>,i,</em>)</td>
<td>( \lambda P(j = \eta_1; P(\eta_1)(\eta_1)) )</td>
<td>((s,(e,T)),T))</td>
</tr>
<tr>
<td>who ( a )</td>
<td>REL</td>
<td>( \lambda P \lambda \alpha \lambda \nu(Q(\alpha)(\nu); P(\alpha)(\nu)) )</td>
<td>((s,(e,T)),((s,(e,T)),(s,(e,T))))</td>
</tr>
<tr>
<td>he ( a )</td>
<td>NP(nom,* ,i)</td>
<td>( \lambda P(P(\alpha)(\nu)) )</td>
<td>((s,(e,T)),T))</td>
</tr>
<tr>
<td>him ( a )</td>
<td>NP(acc,* ,i)</td>
<td>( \lambda P(P(\alpha)(\nu)) )</td>
<td>((s,(e,T)),T))</td>
</tr>
<tr>
<td>man ( a )</td>
<td>CN</td>
<td>( \lambda \alpha \lambda \nu \lambda (\eta_1)(\nu) )</td>
<td>((s,(e,T)),T))</td>
</tr>
<tr>
<td>boy ( a )</td>
<td>CN</td>
<td>( \lambda \alpha \lambda \nu \lambda (\eta_1)(\nu) )</td>
<td>((s,(e,T)),T))</td>
</tr>
<tr>
<td>smiles ( a )</td>
<td>VP(Tensed)</td>
<td>( \lambda \alpha \lambda \nu \lambda \nu \lambda (\eta_1)(\nu) )</td>
<td>((s,(e,T)),T))</td>
</tr>
<tr>
<td>smile ( a )</td>
<td>VP(Inf)</td>
<td>( \lambda \alpha \lambda \nu \lambda (\eta_1)(\nu) )</td>
<td>((s,(e,T)),T))</td>
</tr>
<tr>
<td>has ( a )</td>
<td>TV(Tensed)</td>
<td>( \lambda P \lambda \alpha \lambda \nu \lambda (\eta_1)(\nu)(\eta_1)(\nu) )</td>
<td>((s,(e,T)),T),((s,(e,T)),T))</td>
</tr>
<tr>
<td>have ( a )</td>
<td>TV(Inf)</td>
<td>( \lambda P \lambda \alpha \lambda \nu \lambda (\eta_1)(\nu)(\eta_1)(\nu) )</td>
<td>((s,(e,T)),T),((s,(e,T)),T))</td>
</tr>
<tr>
<td>hates ( a )</td>
<td>TV(Tensed)</td>
<td>( \lambda P \lambda \alpha \lambda \nu \lambda (\eta_1)(\nu)(\eta_1)(\nu) )</td>
<td>((s,(e,T)),T),((s,(e,T)),T))</td>
</tr>
<tr>
<td>hate ( a )</td>
<td>TV(Inf)</td>
<td>( \lambda P \lambda \alpha \lambda \nu \lambda (\eta_1)(\nu)(\eta_1)(\nu) )</td>
<td>((s,(e,T)),T),((s,(e,T)),T))</td>
</tr>
<tr>
<td>seeks ( a )</td>
<td>TV(Tensed)</td>
<td>( \lambda P \lambda \alpha \lambda \nu \lambda (\eta_1)(\nu)(\eta_1)(\nu) )</td>
<td>((s,(e,T)),T),((s,(e,T)),T))</td>
</tr>
<tr>
<td>seek ( a )</td>
<td>TV(Inf)</td>
<td>( \lambda P \lambda \alpha \lambda \nu \lambda (\eta_1)(\nu)(\eta_1)(\nu) )</td>
<td>((s,(e,T)),T),((s,(e,T)),T))</td>
</tr>
<tr>
<td>does not ( a )</td>
<td>AUX</td>
<td>( \lambda P \lambda \alpha \lambda \nu \lambda (\eta_1)(\nu) )</td>
<td>((s,(e,T)),((s,(e,T)),T)))</td>
</tr>
<tr>
<td>if ( a )</td>
<td>(S/S)/S</td>
<td>( \lambda P \lambda \alpha \lambda \nu \lambda (\eta_1)(\nu) )</td>
<td>((T,T),T))</td>
</tr>
<tr>
<td>.</td>
<td>S(TXT/S)</td>
<td>( \lambda P \lambda \alpha \lambda \nu \lambda (\eta_1)(\nu) )</td>
<td>((T,T),T))</td>
</tr>
<tr>
<td>.</td>
<td>TXT/(TXT/S)</td>
<td>( \lambda P \lambda \alpha \lambda \nu \lambda (\eta_1)(\nu) )</td>
<td>((T,T),T))</td>
</tr>
</tbody>
</table>

**Figure 2.3**: Lexical component of an intensional fragment for English.

Of course, in most cases these intensions do not really perform any work, and they disappear again after lambda reduction. The translation of a *man*\(^1\) *smiles* now becomes, after reduction to normal form:

15. \( \eta x_1; \text{man}(a)(x_1); \text{smile}(a)(x_1) \).

This expression contains a free index variable \( a \), which we assume denotes evaluation at the current world. But the intension of this expression might come in handy as well, e.g., as a complement to believe, in *John believes that a man smiles*. The translation of the complement should then be:

16. \( \lambda a(\eta x_1; \text{man}(a)(x_1); \text{smile}(a)(x_1)) \).

This denotes a map from worlds \( w \) to transition relations \( R_w \) which is such that \((f,g) \in R_w\) iff \( f \) and \( g \) differ only in the value assigned to \( x_1 \), and \( g(x_1) \) is a smiling man in \( w \).

Intensions also play a real rôle in the translation of the next sentence.
A boy\(^1\) seeks a girl\(^2\).

After reduction to normal form, the translation becomes:

\[ \eta x_1; \text{boy}(a)(x_1); \text{seek}(a)(\lambda a \lambda Q(\eta x_2; \text{girl}(a)(x_2); Q(a)(x_2)))(x_1). \]

This is true at the world denoted by \(a\) if \(x_1\) can be mapped to a boy in that world with the property that he stands in relation indicated by seek to the ‘concept’ of a girl. This is all completely Montagovian, except for the fact that the values of \(x_1\) and \(x_2\) remain available for subsequent use.

A boy\(^1\) seeks a girl\(^2\). She\(_2\) should be pretty.

This is potentially useful for an account of so-called ‘modal subordination’, as in example 19 (although it should be noted that the present set-up does not provide all the tools for handling such cases).
Chapter 3

Situation Semantics

Work on situation semantics such as [Barwise and Perry, 1983; Barwise, 1989b; Devlin, 1991] has tended to emphasize the philosophical motivations that led to the adoption of situation theoretic tools—the ‘relational’ theory of meaning, the primacy of information in communication, the ‘realist’ hypothesis—over the use of such tools to provide a semantic analysis of natural language constructions. There has also been a lack of a clear compact mathematical foundation for the complete theory although there have been a number of detailed foundational studies relating to fragments of the theory. Thus, although situation theory has been employed to provide analyses of the semantics of Naked Infinitive (NI) perception statements [Barwise, 1981], of definite descriptions and indexical expressions [Barwise and Perry, 1983], of quantified NPs [Gawron and Peters, 1990; Cooper, 1993], of tense [Cooper, 1985; Glasbey, 1994] and of questions [Ginzburg, 1993], and indeed a fairly comprehensive fragment is discussed in [Gawron and Peters, 1990], it is often difficult to understand the motivations behind situation-theoretic analyses, and to compare the predictions of these analyses with those of analyses formulated in terms of other semantic frameworks.

The aim of this chapter is to provide an introduction to situation semantics that may serve as the basis for a comparison with the other semantic approaches analyzed in the FRACAS project. On the one hand, we provide an introduction to recent work concerned with spelling out the mathematical details of situation theory in which the notions of infon algebras and of parametric universe with abstraction are used, and we motivate the adoption of these tools. On the other hand, we put together the most recent situation-semantic treatments of the semantic phenomena in the D2 fragment.

We omit almost entirely a discussion of the philosophical motivations, which can be found in texts such as [Barwise and Perry, 1983; Barwise, 1989b; Devlin, 1991].
3.1 Semantic Tools

3.1.1 Introduction: Syntax and Semantics of a Situation-Theoretic Language

A typical problem with situation semantics is that the theory is presented in such a way as to make it necessary for a semanticist to adopt wholesale a large number of new tools. We have therefore attempted to isolate the main ideas of situation semantics and present them in progressive stages, in a similar fashion to the presentation of Montague semantics in [Dowty et al., 1981], motivating these ingredients in purely semantical terms:

- the shift from a semantics based on truth conditions to one based on the properties of information exchange, and the resulting ‘algebraic’ approach to semantic interpretation;
- the central role given to the idea that the information being exchanged is primarily partial, i.e., is information about situations, and the consequent adoption of ‘Austinian’ propositions as the content of declarative statements;
- the attention paid to context dependency phenomena in language such as the interpretation of indexicals or anaphoric expressions and domain restriction, and the resulting claim that the linguistic objects to be interpreted are utterances in discourse situations rather than sentences or discourses regarded as abstract objects.

A lot of work in situation semantics is only semi-formal [Barwise and Perry, 1983; Gawron and Peters, 1990; Devlin, 1991]. To some extent this situation has been ameliorated by work on situation theory by Barwise [1987a; 1989b], Barwise and Etchemendy [1987; 1990], Aczel [Aczel, 1990] and Westerstahl [1990] among others. The version of situation theory adopted here is based on [Barwise and Cooper, 1993], the most comprehensive attempt to date to provide a compact treatment of situation theory. In that paper, the language we will use in the rest of the document to provide a semantic analysis of the D2 fragment, called EKN, is also introduced.

3.1.2 An Information-Based, Algebraic Approach to Semantics

3.1.2.1 From Truth to Information

In situation semantics a good deal of emphasis is placed on developing a theory that accounts for the information conveyed by utterances, rather than on the truth conditions of sentences or discourses. This means that the semantic analysis of an utterance should be able to say something about the information that an agent intends to convey by means of that utterance and the information that another agent might extract from the utterance (they may, of course,

\footnote{For Extended Kamp Notation}
not be the same), rather than in terms of the relation between the sentence that is uttered and the state of the world. This aim is, of course, more ambitious than what is elsewhere considered to be “formal semantics”. It is putting communication and information exchange at the beginning and this sometimes leads people to say that situation semantics is more of a pragmatic theory than a semantic theory. However, the kind of tools that you need to deal with information and information exchange are also the ones needed to deal with some classic semantic problems. The emphasis here is on an information structure which is more fine-grained than that provided by the classical view of propositions as sets of possible worlds. A central problem has to do with logically equivalent propositions that intuitively convey different information. A class of sentences that have traditionally proved difficult are propositional attitude reports of the form in (1) (from the D2 fragment):

(1) a. Smith believed that ITEI had won a contract.

Among the problems raised by this kind of sentence are the problem of logical omniscience and the problem of substitution of equals \[Cresswell, 1987; Levesque, 1990; Fagin and Halpern, 1988\]. The solutions advocated for this problem (among others, \[Cresswell, 1987; Haas, 1986; Kamp, 1990; Fagin and Halpern, 1988\]) are all based on the intuition that propositions (at least when in complement position of propositional attitude verbs) are more ‘structured’ than one would expect from modeling them as, say, sets of possible worlds. Cresswell, for example, proposes in \[Cresswell, 1987\] that

\[\text{... the meanings of propositional attitude words like ‘believes’ are sensitive not solely to the proposition expressed by a whole sentence but also to the meanings of its separate parts. Sentence (3) has a much more complex structure that (2), and so the meanings of its parts fit together in a much more complex manner.}\]

(2) Robin will win.

(3) Everyone who does not compete, or loses, will have done something that Robin will have not done.

The problem of the kind of objects to be used as the denotation of sentences surfaces again when we try to assign an interpretation to inconsistent discourses. A statement may be inherently inconsistent, either on logical grounds—for example, (4):

(4) That was a lie and it was not a lie.

—or because of lexical meaning, as in John is a married bachelor. Sentences such as (4) can be used to achieve a rhetorical effect but often (and much more commonly) discourses are inconsistent because of the limits of human memory and reasoning ability. And yet human beings are still able to extract information from such discourses. For example, people are often inconsistent in planning situations: e.g., an agent can propose to send the boxcar from Dansville to Avon at 2pm and load it with oranges at 3pm even after being told that getting from Dansville to Avon takes 3 hours. If the denotation of a discourse is identified with the
set of possible worlds in which all of the statements are true, all inconsistent discourses will
denote the empty set, the necessarily false proposition from which everything follows.

In situation semantics, a finer-grained classification of information is achieved by adopting a
structured universe of situation theoretic objects, in which ‘units of information,’ or infons,
are objects in their own right, as the domain out of which denotations are assigned. In
classical Tarskian model theory, predicates like sell denote relations among the individuals
in the universe which are construed as sets of ordered tuples, in this case triples representing
who sells what to whom. Because of the lexical semantic relationship between buy and sell
the denotations of these two words will involve exactly the same sets of individuals but with
the buyer and the seller in different argument positions so that a buys b from c iff c sells b
to a. This has the consequence that in terms of the classical possible worlds approach a buys
b from c represents exactly the same proposition as c sells b to a. This means that it would
be difficult to make sense of the following discourse on such a theory:

There were records showing that Electron plc had sold the chips to Itel but there
were no records showing that Itel had bought them from Electron plc. Somebody
had managed to divert them to a foreign company.

In situation theory, such predicates correspond to distinct objects in the situation theo-
retic universe, and these objects are used to construct distinct infons \((\text{sell}, j, m, a)\)\(^2\) and
\((\text{buy}, m, j, a)\). These infons are distinct structured objects with their respective distinct rela-
tions buy and sell, although there are various kinds of equivalences that could be required
of these infons, e.g. that if one of them is a fact (i.e. actually obtains in the world) then the
other must be a fact. However, the fact that two things are equivalent in terms of truth does
not, in a structured universe, require them to be the same thing.

Infons constructed from a relation and arguments in this way are known as basic infons,
which may be either positive or negative. An infon lattice is then constructed, closing the
set of infons under meet and join operations that provide an interpretation for conjunction
and disjunction, respectively, thus replacing the Boolean interpretation of connectives, where
conjunction denotes set intersection, disjunction denotes union, and negation denotes com-
plementation.

As we will discuss shortly, in situation theory infons are ways of classifying situations, and the
content of a declarative statement—an (Austinian) proposition—can be regarded as the claim
that a certain infon describes a situation.\(^3\) An Austinian proposition is true just in case the
situation is of the type represented by the infon. There is an algebra of propositions as well
as the infon algebra. Propositions are closed under conjunction, disjunction and negation.
For the moment, we will concentrate on the infon structure and consider the consequences for
semantics of a structured approach to semantic objects. As we will see below, this algebraic
approach provides a way to achieve the separation between the meaning of (2) and the meaning
of (3) advocated by Cresswell.

\(^2\)We will introduce a different notation below.

\(^3\)More on the relation between the classical notion of proposition and the notion of Austinian proposition
below.
We will conclude by observing that lattice-theoretical notions are not a novelty anymore in semantics; most recent work on event structures and plurals adopts notions of this sort, and property theory adopts a similar approach to propositional objects as well in a way that is quite parallel to the situation theoretic approach.

3.1.2.2 Infons

We begin the description of a situation theoretic universe with the basic units of information. Basic infons are built up from relations and assignments of objects to their roles. The two facts that we intend to capture at this point are (i) that relations have roles, (ii) that these roles are ‘filled’ by arguments, and (iii) that there are appropriateness restrictions on the objects that can be assigned to the roles of a relation.\(^4\)

The structures we will use will be of the form

\[<A, \text{Sorts}, \text{Rlns}>\]

where:

1. \text{Sorts} is a set of unary relations on (subsets of) \(A\), the set of \text{sorts} introduced in the structure
2. \text{Rlns} is a set of relations on \(A\)
3. there is a set \(X \in \text{Sorts}\) (which we will always refer to as \text{Obj}) such that \(\forall Y \in \text{Sorts}, Y \subseteq X\)

These structures can be regarded as relational structures of the standard kind, i.e. \(<A, \text{Sorts} \cup \text{Rlns}>\). We have separated out the sorts from the rest of the relations because they play a special role in the theory.

3.1.3 Predicate Structures

The relations which are used to construct infons are one kind of predicate found in situation theory.\(^5\) We will therefore start by saying in general what kinds of objects predicates are and what kinds of operations they may undergo. A \textit{predicate structure} tells us what predicates

\(^4\)The appropriateness restrictions are not necessarily used for selectional restrictions (although that would be possible), but are necessary to specify the sorts of objects that can be assigned to the roles of relations in order to avoid paradoxes. Westerstål [1990] observes that these appropriateness restrictions play a very important role in a typeless theory such as situation theory.

\(^5\)The other kind of predicate is called a type and this will be discussed below.
there are, what roles they have and which assignments of objects are appropriate to them. There is nothing particularly situation theoretic about the notion of a predicate structure. It is a notion that could be applied to an algebraic semantics for predicate calculus or the typed lambda-calculus, for example.

A *predicate structure*

\[
< A, \{ \text{Pred}, \text{Asst}, \text{Obj} \}, \{ \text{Roles}, \Sigma, \text{appr} \} >
\]

is such that:

1. \( \text{Roles} \subseteq A \)
2. \( \text{Asst} \), the set of assignments, is the set of partial functions \( f: \text{Roles} \rightarrow \text{Obj} \)
3. \( \Sigma: \text{Pred} \rightarrow \text{Pow}(\text{Roles}) \) is a function which assigns a set of roles to each predicate in \( \text{Pred} \). \( \Sigma \) is called the *role assignment function*.
4. \( \text{appr} \), the appropriateness relation, is a relation between \( \text{Pred} \) and \( \text{Asst} \) such that for any \( r \in \text{Pred} \) \( \text{appr}(r, f) \) implies \( f \) is a function whose domain includes \( \Sigma(r) \), i.e. \( \text{appr} \) relates a relation to the appropriate assignments to its roles.

**Remark:** it is possible for an appropriate assignment to a relation to assign the relation itself to one of its roles.

**Remark:** the set of roles may overlap with the set of objects.

A *predication operation* for a predicate structure \( < A, \{ \text{Pred}, \text{Asst}, \text{Obj} \}, \{ \text{Roles}, \Sigma, \text{appr} \} > \) is a partial binary operation \( * \) on \( A \) which is defined for any \( r \in \text{Pred} \) and \( f \in \text{Asst} \) such that \( \text{appr}(r, f) \) and is undefined otherwise.

### 3.1.4 Infon Structures

The elementary units of information, basic infons, are constructed from relations, which are required to be predicates in the sense defined above and appropriate assignments. The operations which construct basic infons are predication operations. There are two: one which constructs positive infons and another which constructs negative infons. A *basic infon structure*

\[
< A, \{ \text{BIInfon} \cup \text{Rel Sorts} \{ \ll \cdot \gg^+, \ll \cdot \gg^- \} \cup \text{Rel Rln} >
\]
is a structure such that:

1. \(< A, Rel_{Sorts}, Rel_{Rln} >\) is a predicate structure. We call the predicates of this structure the \textit{relations} of the basic infon structure.

2. \(\ll . \gg^+, \ll . \gg^-\) are predication operations for \(< A, Rel_{Sorts}, Rel_{Rln} >\). These are referred to as positive and negative infon operations, respectively.

3. \(B\text{Infon}\), the set of basic infons, is the union of the ranges of \(\ll . \gg^+\) and \(\ll . \gg^-\).

4. If \(\ll r, f \gg^+ = \ll r', f' \gg^+\), then \(r = r'\) and \(f = f'\) and similarly for \(\ll r, f \gg^-\) (structure)

\(\ll r, f \gg^+\) and \(\ll r, f \gg^-\) are called \textit{duals}. The dual of an infon \(\sigma\) is written \(\overline{\sigma}\).

\textbf{Remark:} This allows infons to be non-well-founded e.g., \(\sigma = \ll r, \sigma \gg^+\)

### 3.1.5 Infon Algebras

Conjoining and disjoining basic infons result in new infons. An \textit{infon algebra} is a structure built out of basic infons by conjunction and disjunction operations. More formally, an infon algebra

\[ < A, \{\text{Infon}\} \cup B\text{Infons}_{Sorts}, \{\land, \lor\} \cup B\text{Infons}_{Rln} > \]

is such that

1. \(< A, B\text{Infons}_{Sorts}, B\text{Infons}_{Rln} >\) is a basic infon structure, whose set of basic infons is \(B\text{Infon}\).

2. \(\land\) and \(\lor\) are binary operations on \(\text{Infons}\), the set of infons, which is the closure of \(B\text{Infon}\) under these operations.

\textbf{Remark:} Non-well-foundedness is not introduced in complex infons although it is in basic infons. This is as in The Liar.

It may appear that infons look very like pieces of language given that we can have negation, conjunction and disjunction. However, there is a crucial difference between the syntax of languages and objects resulting from operations in an algebraic approach. In a language the sentences \(p \land q\) and \(q \land p\) are simply not syntactically identical for all they may be logically
equivalent or denote the same proposition, depending on the semantics you provide for the language. An algebraic approach gives us a middle ground between syntax and the traditional model theory in terms of sets and truth values. It allows us some of the kind of structure that we have in syntax but also allows us to express equalities between the results of some of the operations applied to certain arguments. What equalities we get depends on the type of lattice that we assume. Typical constraints one might want to impose on the meet and join operations on the lattice are the following:

1. For any \( \sigma, \tau \), \( \sigma \lor \tau = \tau \lor \sigma \) and \( \sigma \land \tau = \tau \land \sigma \) (commutativity)
2. For any \( \sigma, \tau, \rho \), \( \sigma \lor (\tau \lor \rho) = (\sigma \lor \tau) \lor \rho \) and \( \sigma \land (\tau \land \rho) = (\sigma \land \tau) \land \rho \) (associativity)
3. For any \( \sigma \), \( \sigma \lor \sigma = \sigma \) and \( \sigma \land \sigma = \sigma \) (idempotency)

We may well in addition want to include axioms corresponding to distributive laws and de Morgan's laws as well. The various options correspond to various kinds of lattices with basic infons as their atoms. [Check] For example, ...

The point is that we have much more control over than in a traditional set-theoretic approach, with the results that certain problems resulting from treating logical equivalence as identity are avoided. For example, (5a) is not required with respect to the algebra even if we assume all the axioms above. Or, assuming that we introduce a constant 1 that represents the state of 'minimal information,' (5b) and (5c) are not required, either:

\[
\begin{align*}
(5) \quad & a. \quad \sigma = \sigma \land [\tau \lor \neg \tau] \\
& b. \quad \tau \lor \neg \tau = 1 \\
& c. \quad \tau \land 1 = \tau
\end{align*}
\]

3.1.5.1 The Language of Infons

The EKN notation for situation theory discussed in [Barwise and Cooper, 1993] differs from the notation used in [Barwise and Perry, 1983] or [Devlin, 1991], and is instead inspired by the notation used in Discourse Representation Theory [Kamp and Reyle, 1993]. We begin with the rules of EKN, that deal with the formation of infons. See [Barwise and Cooper, 1993] for a complete specification of the notation. The infon corresponding to the fact that John sells an object \( a \) to Mary is represented by a Drt-style 'box' as in (6) rather than in the format (7) used previously in the situation theoretic literature.

\[
(6) \quad \text{\textsc{sell}}(j,m,a)
\]

\[
(7) \quad \langle \langle \text{\textsc{sell}}, j, m, a \rangle \rangle
\]
The negative infon representing the information that John didn’t sell a to Mary is represented in EKN notation as in (8), instead than by means of a polarity component, as in (9):

\[(8) \quad \neg \text{SELL}(j,m,a)\]

\[(9) \quad \langle \langle \text{SELL},j,m,a;0 \rangle \rangle\]

The conjunction of two infons is represented in EKN as in (10); a conjunction of two infons, in other words, is represented as a box containing both infons, as in DRT. Infon disjunction is represented as in (11).^6

\[(10) \quad \begin{array}{c} \sigma \\ \tau \end{array}\]

\[(11) \quad \sigma \lor \tau\]

The expressions in (6)-(11) denote infons in an infon algebra as defined above. More precisely, their denotation are defined as follows:

1. Basic infon terms

   **Syntax** If \( \Box \) is a relation term and \( A \) is an assignment term then

   \[ \Box (A) \]

   and

   \[ \neg \Box (A) \]

   are infon terms \( \alpha \) and \( \overline{\alpha} \).

   **Semantics**

   \[ [\alpha] = \langle \langle B \rangle, \llbracket A \rrbracket \rangle^+ \]

   \[ [\overline{\alpha}] = \langle \langle B \rangle, \llbracket A \rrbracket \rangle^- \]

---

^6 Here, as in the rest of the paper, the Greek letters \( \sigma \) and \( \tau \) are reserved for infons.
2. Infon conjunction

**Syntax** If \( B_1, \ldots, B_n \) are infon terms then

\[
\begin{array}{c}
B_1 \\
\vdots \\
B_n
\end{array}
\]

is an infon term \( \alpha \).

**Semantics**

\( [\alpha] = [B_1] \land \ldots \land [B_n] \)

That is, this denotes the conjunction of the infons \( [B_1], \ldots, [B_n] \).

3. Infon disjunction

**Syntax** If \( B_1, \ldots, B_n \) are infon terms, then

\[
B_1 \lor \ldots \lor B_n
\]

is an infon term \( \alpha \).

**Semantics**

\( [\alpha] = [B_1] \lor \ldots \lor [B_n] \)

That is, this denotes the disjunction of the infons \( [B_1], \ldots, [B_n] \).

Note that infon terms are terms of the language which denote objects in the situation theoretic universe, namely infons. Even though they look like expressions of predicate logic they are not the kind of things that are true or false. Rather they are the kind of thing that can be predicated of a situation to create a proposition that is true or false.

### 3.1.6 Situations

Situation-like entities are used in much current work in the semantic literature. Most notable perhaps is the work on tense and aspect where objects like event and states are introduced into the semantic universe [Bach, 1981; Bach, 1986; Kamp and Rohrer, 1983; Hinrichs, 1986; Krifka, 1988; Krifka, 1989; Glasbey, 1994]. A number of these theories propose that situations or events are 'parts of the world' that enter into a a partial order which is a part-of relation. Bach [1981] introduced the term 'eventuality' as a cover term for all such objects. These objects are closely related to the situation theoretic notion of situation.
In situation theory, a situation is a part of the world that provides certain information, or, in the terminology of the theory, supports certain insons. A situation need not 'provide an answer' on all issues, and indeed, the main motivation for situations discussed in [Barwise, 1981] was the need to formalize the partiality assumptions built into natural language. The problem addressed by Barwise in the paper was the semantics of perception verbs such as see or hear that, he argued, capture a simpler relation between an agent’s mental state and the world that propositional attitude verbs such as believe or hope, and therefore should be looked at first.

Why should the semantics of perception complements involve something like a partial situation or event rather than a proposition viewed as a set of possible worlds? The motivation has to do with logical or necessary equivalence. Consider (12)

(12) The share prices for Intel either changed or held steady

(12) is necessarily true, given that presuppositions about the existence of Intel as a company trading on the stock market are met. The share prices either change or they don’t so the sentence is essentially of the tautological form $p \land \neg p$. Now note that (13), uttered about Harry when he is sitting looking at a monitor reporting share price fluctuations is contingent just like (14). Both examples might be appropriate answers to the question Why did Harry go pale?

(13) Harry saw the share prices of Intel either change or hold steady (but I don’t know which).

(14) Harry saw the chief executive flinch

Now notice that (15) and (16) are equivalent in the sense that they are true in the same possible worlds (given that the presuppositions are met). That is, on the possible worlds view they represent the same proposition.

(15) The chief executive flinched
(16) The chief executive flinched and the share prices of Intel either changed or held steady

The conjunction of (15) with a tautology in (16) does not make any difference to its truth conditions. Nevertheless we have somehow the intuition that there is more information in (16). This fact is confirmed when we consider corresponding perception sentences.

(17) Harry saw the chief executive flinch
(18) Harry saw the chief executive flinch and the share prices of Intel change or hold steady

It seems clear that (18) requires Harry to have seen more of the world (e.g. an event on a monitor) that is required by (17). Perhaps more natural examples are constructed when perception
complements are conjoined with sentences that necessarily follow from them. Consider the sentences in (19) and (20).

(19) The secretary typed in the letter
(20) The secretary typed in the letter and his fingers touched the keyboard

Given the lexical semantics of typing and the satisfaction of certain presuppositions about gender and the keyboard being referred to, it follows necessarily from the secretary’s typing the letter that his fingers touch the keyboard. Thus in terms of possible worlds (20) does not cut down on the set of worlds and (19) and (20) represent the same proposition. Yet still we have the intuition that (20) expresses more information than (19). This is confirmed when we embed these sentences as perception complements as in (21) and (22).

(21) Harry saw the secretary type in the letter
(22) Harry saw the secretary type in the letter and his fingers touch the keyboard

(22) could be false while (21) is true. For example, Harry may have been standing behind the secretary or watching the computer screen as the secretary typed the letter.

Building on Dretske’s [1981] distinction between ‘epistemically neutral’ see (as in John saw Bill climb the fence) versus ‘epistemically positive’ see that (as in John saw that Bill climbed the fence), Barwise [1981] argued that the former are relations between an agent and a ‘scene,’ that, according to Barwise, is a visually perceived situation; situations, in turns, are “Any part of the way the world happens to be.” ([Barwise, 1981], p.27) That is, a perception statement of the form “a sees Φ” is interpreted as asserting that a sees a scene s that supports the truth of Φ. The semantics of epistemically neutral perception statements proposed by Barwise can be rephrased in terms of situations and infons as follows:

(23) A perception statement of the form “a sees Φ” is interpreted as asserting that a sees a situation s, and that s supports the infon Φ.

Another aspect of natural language semantics in which partiality assumptions are made is domain restriction in quantified expressions. Whenever a speaker utters a sentence like Everybody is asleep, she implicitly requires her listeners to ‘adjust’ the domain so that only the ‘relevant’ individuals are quantified over. The proposal that this adjustment is a case of selecting the appropriate situation, sketched in S&A, was developed by Gawron and Peters in [1990] and by Cooper in [1993]. We discuss the issue below when talking about quantification.

In order to support this kind of analysis we need to introduce the notion of situation structure into situation theory. A situation structure is based on an infon algebra and adds to it a set of situations and a supports relation which holds between situations and infons.

A situation structure
\(< A, \{ \text{Sit} \} \cup \text{Infon Sorts} \cup \{ \models \} \cup \text{Infon Rln} > \)

is such that

1. \(\text{Sit}\) is a set/collection, the situations

2. \(< A, \text{Infon Sorts}, \text{Infon Rln} >\) is an infon algebra, whose set of infons is \(\text{Infons}\) and whose set of basic infons is \(\text{BInfons}\)

3. \(\models\) is a relation between \(\text{Sit}\) and \(\text{Infons}\), the supports relation

One situation can be part of another. We will introduce here a simple extensional notion of part-of in terms of sets of infons supported by a situation. For some discussion of another alternative see \([1989a]\).

\[ s \preceq s' \text{ iff } \{ \sigma | s \models \sigma \} \subseteq \{ \sigma | s' \models \sigma \}. \]

This means that any infon is persistent under \(\preceq\) because if it can be used to classify a situation \(s\) then it can be used to classify all situations of which \(s\) is a part, i.e. if \(s \models \sigma\) and \(s \preceq s'\) then \(s' \models \sigma\).

There are a number of additional axioms which are standardly assumed to hold of situation structures.

**Optional axioms**

1. if \(s, s' \in \text{Sit}\) and \(\{ \sigma | s \models \sigma \} = \{ \sigma | s' \models \sigma \}\) then \(s = s'\) (extensionality)

2. \(s \models \sigma \land \tau \text{ iff } s \models \sigma\) and \(s \models \tau\) (conjunction)

3. \(s \models \sigma \lor \tau \text{ iff } s \models \sigma\) or \(s \models \tau\) (disjunction)

4. If \(\sigma \in \text{BInfons}\) then \(s \models \sigma\) implies \(s \not\models \overline{\sigma}\) (consistency)

5. If \(s_1, s_2\) are situations then there is some situation \(s_3\) such that \(\{ \sigma | s_1 \models \sigma \} \cup \{ \sigma | s_2 \models \sigma \} \subseteq \{ \sigma | s_3 \models \sigma \}\) (directedness)

Axioms 4 and 5 together mean that no two situations can be inconsistent and therefore rule out the existence of possible non-actual situations. If you want possible situations then you should probably relativize directedness to possible worlds and this might give you the essential characteristics of a Kratzer-style situation theory [Kratzer, 1989].
3.1.7 Truth and Propositions

3.1.7.1 From Situations and Infons to Propositions

We have now introduced situations and infons as types of situations: the next step is to introduce the class of propositions called Austinian propositions by Barwise and Etchemendy in *The Liar* [Barwise and Etchemendy, 1987], which are true just in case a situation is of the type corresponding to an infon.

Austinian propositions are traditionally represented in situation semantics by means of the linear notation in (24a). We will reserve the notation |= for the supports relation introduced in situation structures. To represent the proposition as an object in the universe, we will use for the proposition that s supports σ the EKN notation shown in (24b), or the linear notation (s;σ), using ‘:’ to represent ‘of type’.

\[
\begin{align*}
\text{a.} & \quad (s \models \sigma) \\
\text{b.} & \quad \sigma
\end{align*}
\]

This kind of proposition enters into play when describing the meaning of simple declarative statements. These statements are normally regarded as describing a situation: for example, if the circumstances of an utterance of *Smith hired Jones* specify s as the described situation we might say that its content is

\[
\begin{array}{c}
s \\
hired(\text{Smith,Jones})
\end{array}
\]

This way of associating situations with statements is similar in many respects to what proposed by Davidson [Davidson, 1967] and in the neo-Davidsonian proposals inspired by his proposal (e.g., [Parsons, 1990]), in which each predicate is taken to have an event argument.

Austinian propositions are not the only kind of proposition in situation theory. More in general, a proposition is a predication that some arguments are of a type; Austinian propositions are the special case where the type is an infon and the argument is a situation. Non-Austinian propositions are called Russellian propositions in [Barwise and Etchemendy, 1987], where the philosophical motivations for the distinction are discussed. Russellian propositions, it is suggested by Barwise and Etchemendy, are the content of statements which do not describe a particular situation, such as mathematical statements like \(2+2=4\). The distinction between Russellian propositions and Austinian propositions could also reflect the distinction between

---

\(^7\)Following Austin’s idea that propositions are claims about part of the world.
categorical statements (such as generics) and thetic statements, which appear to involve an event argument.

3.1.7.2 Propositions

**Basic proposition structures** Basic propositions are constructed from types and appropriate assignments to those types. A basic proposition structure \(<A, \{BProp\} \cup Type_Sorts,\ {\text{True}},(\_\_)) \cup Type_{Rln}>\) is such that

1. \(<A, Type_Sorts, Type_{Rln}>\) is a predicate structure, whose predicates are \(Types\), the set of types, and whose appropriateness relation is \(appr_{Types}\).
2. \((\_\_))\) is a predication operation for \(<A, Type_Sorts, Type_{Rln}>\)
3. \(BProp\) is the range of \((\_\_))\), the set of basic propositions.
4. \(True\) is a subset of \(BProp\), the true propositions
   Optional axiom
5. If \((f : T) = (f' : T')\), then \(f = f'\) and \(T = T'\) (structure)

**Austinian basic proposition structures** Among the basic propositions are Austinian propositions whose types are infons which are assigned a single situation. An Austinian proposition \((s : \sigma)\) is true iff \(s \models \sigma\). An Austinian basic proposition structure \(<A, BProp_Sorts \cup Sit_Sorts, BProp_{Rln} \cup Sit_{Rln}>>\) where

1. \(<A, BProp_Sorts, BProp_{Rln}>>\) is a basic proposition structure, whose types are \(Type\) and whose true propositions are \(True\).
2. \(<A, Sit_Sorts, Sit_{Rln}>>\) is a situation structure, whose infons are \(Infon\).
3. \(Infon \subseteq Type\)
4. if \(\sigma \in Infon\), then \(\Sigma(\sigma)\) is a singleton (i.e. \(\sigma\) is a unary type)
5. if \(\sigma \in Infon\) and \(\rho \in \Sigma(\sigma)\) then \(appr_{Type}(\sigma, f)\) iff \(f(\rho(\sigma)) \in Sit\) (i.e. all and only situations are appropriate to \(\sigma\))
6. if \(\sigma \in Infon\) and \(s \in Sit\), then \((s : \sigma) \in True\) iff \(s \models \sigma\)
**Proposition algebras** Propositions can be conjoined, disjoined and negated and their logic is classical.\(^5\) An (Austinian) proposition algebra \(\langle A, \{\text{Prop}\} \cup \text{BProp}_{\text{Sorts}}, \text{BProp}_{\text{Rel}} \cup \{\{\text{True}\}, \land, \lor, \neg\} \rangle\) is such that

1. \(\langle A, \text{BProp}_{\text{Sorts}}, \text{BProp}_{\text{Rel}} \rangle\) is a basic (Austinian) proposition structure, whose propositions are \(\text{BProp}\), whose true propositions are \(\text{BTrue}\).

2. \(\land\) and \(\lor\) are binary operations and \(\neg\) a unary operation on \(\text{Prop}\), the set of propositions, which is the closure of \(\text{BProp}\) under these three operations.

3. \(\text{BTrue} \subseteq \text{True}\)

4. \(p \land q \in \text{True} \iff p \in \text{True} \text{ and } q \in \text{True}\)

5. \(p \lor q \in \text{True} \iff p \in \text{True} \text{ or } q \in \text{True}\)

6. \(\neg p \in \text{True} \iff p \notin \text{True}\)

**Optional axioms**

7. For any \(p, p \land p = p \lor p = p\) (idempotency)

8. For any \(p, q, p \lor q = q \lor p\) and \(p \land q = q \land p\) (commutativity)

9. For any \(p, q, r, p \lor (q \lor r) = (p \lor q) \lor r\) and \(p \land (q \land r) = (p \land q) \land r\) (associativity)

### 3.1.7.3 Entailment

Barwise and Cooper [1993] propose the following treatment for entailment. They assume that there is a binary type \(\Rightarrow\) which holds of a pair \([p, q]\) of non-parametric propositions just in case \(p\) logically entails \(q\). The following axiom is imposed on \(\Rightarrow\):

\((\text{Soundness})\) If \((p \Rightarrow q)\) is true, and \(p\) is true, then so is \(q\).

---

\(^5\)Why do we need both infon algebras and proposition algebras? Once we have introduced abstraction, we might consider the option of defining infon conjunction and disjunction in terms of proposition conjunction and disjunction. For example, we could set

\[\sigma = \lambda[s]\]

\[\sigma \land \tau = \lambda[s]\]

However, such identities are not generally assumed in situation theory.

129
Similarly, Barwise and Cooper introduce a binary type \( \equiv \), that holds of a pair \([p, q]\) of propositions if they are logically equivalent.

If we assume the conjunction and disjunction axioms for infons, then the following equivalences will follow for propositions:

1. \((s : \sigma) \land (s : \tau) \in True\) iff \((s : \sigma \land \tau) \in True\)
2. \((s : \sigma) \lor (s : \tau) \in True\) iff \((s : \sigma \lor \tau) \in True\)

It would be natural to include these equivalences in the entailment relation together with the standard tautologies of propositional logic as these follow from our characterization of truth in proposition algebras.

### 3.1.7.4 The Language of Propositions

We will represent basic propositions using either the EKN notation or the linear notation \((s : \sigma)\). Just as in the case of infons, more complex propositions can be obtained by conjoining and disjoining basic propositions like the one in (25). For any two propositions \(p\) and \(q\) we have their conjunction and disjunction, represented below in EKN notation and in standard linear notation.

\[
\begin{align*}
(26) & \quad a. \\
 & \quad \begin{array}{c}
p \\
q
\end{array} \\
 & \quad b. \quad p \land q
\end{align*}
\]

\[
\begin{align*}
(27) & \quad a. \\
 & \quad \begin{array}{c}
p \lor q
\end{array} \\
 & \quad b. \quad p \lor q
\end{align*}
\]

The language of EKN also allows for negation of basic and non-basic propositions:

\[
\begin{align*}
(28) & \quad a. \\
 & \quad \begin{array}{c}
\neg
\end{array} \\
 & \quad \begin{array}{c}
\sigma
\end{array} \\
 & \quad b. \quad (s \not\models \sigma)
\end{align*}
\]
The form of proposition just discussed, composed by a situation and an infon, is just the simpler form of proposition. As we will see below, infons are a particular case of type, namely, a situation type; the claim that situation $s$ supports infon $\sigma$ can thus be reformulated as a claim that $s$ is of type $\sigma$. We use the following notation to indicate that:

$$s:\sigma$$

More in general, we will say that a proposition is the claim that object $x$ is of type $\tau$. When $x$ is a situation and $\tau$ a situation type, we will have an Austinian proposition; otherwise, a Russellian proposition.

Here is a more formal account of the syntax and semantics of EKN proposition terms taken from [Barwise and Cooper, 1993].

**Proposition terms**

1. Basic proposition terms

   (a) **Syntax** If $s$ is an situation term, and $B$ is an infon term, then

   $$s : B$$
is a proposition term $\alpha$.

*Semantics*

$$[\alpha] = ([s] \models [B])$$

i.e. the proposition that $[s]$ supports $[B]$.

(b) *Syntax* If $T$ is a type term and $A$ is an assignment term then

$$\begin{array}{c}
A \\
\hline
T
\end{array}$$

is a proposition term $\alpha$.

*Semantics*

$$[\alpha] = ([A] : [T])$$

i.e. the proposition that $[A]$ is of type $[T]$.

2. Propositional conjunction

*Syntax* If $P_1, \ldots, P_n$ are proposition terms, then

$$\begin{array}{c}
P_1 \\
\vdots \\
P_n
\end{array}$$

is a proposition term $\alpha$.

*Semantics*

$$[\alpha] = [P_1] \land \ldots \land [P_n]$$

i.e., the classical conjunction of $[P_1], \ldots, [P_n]$.

3. Propositional disjunction

*Syntax* If $P_1, \ldots, P_n$ are proposition terms, then

$$P_1 \lor \ldots \lor P_n$$

is a proposition term $\alpha$.

*Semantics*

$$[\alpha] = [P_1] \lor \ldots \lor [P_n]$$

i.e., the classical disjunction of $[P_1], \ldots, [P_n]$.

4. Propositional negation
Syntax  If \( P \) is a proposition term, then
\[
\neg P
\]
is a proposition term \( \alpha \).

Semantics
\[
[\alpha] = \neg[P]
\]
i.e., the classical negation of \([P]\).

### 3.1.8 Parameters and Quantification

#### 3.1.8.1 Parameters

Parameters\(^9\) are objects of the situation theoretic universe with a variable-like status, i.e.,
that may occur in infons as place-holders for “real” objects. Parameters can take part in the
constructive operations we have introduced together with other objects in the universe and
result in parametric objects. For example, if \( X \) and \( Y \) are parameters we and take a non-
parametric relation \( r \) and apply the positive infon forming operation to \( r \) and an assignment
of \( X \) and \( Y \) to \( r \)'s roles resulting in the parametric infon that can be represented informally as
\( \langle\langle r, X, Y \rangle\rangle \). This is an object with parameters \( X \) and \( Y \). To take another example, we might
represent the proposition that Anna likes Maria and Maria seems to like Anna as
\( p(a, m) \).

We could then represent the parametric proposition “\( X \) likes \( Y \) and \( Y \) seems to like \( X \)” as
\( p(X, Y) \).

The ‘place-holding’ role of parameters is represented in the theory by defining the notion
of anchor which will assign appropriate situation theoretic object to parameters. (Anchors
are thought of as a function in the set theoretic sense and are exactly parallel to variable
assignments except that their domains are objects in the universe (parameters) rather than
expressions of a language (variables).)

There are two reasons for introducing parameters into the universe. Firstly, and this was
the main motivation in early situation theory and situation semantics, one needs to be able
to deal with the kind of context dependence expressed in natural language by indexicals of
various kinds (e.g. deictic pronouns, demonstratives, some uses of tensed verbs, and also
proper names according to the situation semantics analysis). Situation semantics has been
concerned with accounting for this kind of indexicality without introducing formal indexing
mechanisms in the syntax or an intermediate syntactic level of logical form or discourse
representation. Hence the “variability” had to be moved into the semantic domain in the
form of parameters.

\(^9\)Parameters were called indeterminates in *Situations and Attitudes*. 

133
The second reason for introducing parameters is related to the working out of the form of abstraction suggested by Aczel and Lunnon [Aczel and Lunnon, 1991]. In order to represent the information carried by quantified statements, as well as to provide a compositional account of interpretation, it is essential to have a notion of abstraction. Aczel and Lunnon develop a theory of abstracts as objects in a universe, in which abstracts are constructed out of parametric objects. This corresponds to the fact that in the more familiar syntactic view of abstraction λ-expressions are constructed from expressions with free variables. Thus, for example, if we have the proposition \( p(X, Y) \) we can abstract over its parameters to form the proposition abstract

\[
\begin{array}{c}
r_1 \rightarrow X, r_2 \rightarrow Y \\
p(X, Y)
\end{array}
\]

(29)

This is an abstract that has two roles indexed by \( r_1 \) and \( r_2 \). If you want intuitive characterizations of the roles try “liking and seeming to be liked by” for \( r_1 \) and “being liked and seeming to like” for \( r_2 \). We are going to discuss abstracts below.

Various alternative proposals for the treatment of parameters in situation theory have been made and technical tools for them have been developed by Aczel, Westerstahl, Lunnon, Crimmins and others [Aczel, 1990; Westerstahl, 1990; Aczel and Lunnon, 1991].

The proposal below is a version of Aczel and Lunnon’s proposal for parametric universes and abstraction. Abstraction is treated as a *binding operation*: it takes an indexed sets of parameters and a parametric object containing those parameters as arguments and returns an object in which those parameters have been bound.

### 3.1.8.2 Parametric Infon Algebras

**Parametric structures** A parametric structure is like those we have been discussing so far, except that it contains infinitely many parameters of each sort. More formally, a *parametric structure* based on the structure \( < A, \text{Sorts}, \text{Rlns} > \) is a structure

\[
< A, \text{Sorts}, \{ \text{Par}_X \}_{X \in \text{Sorts} \cup \text{Rlns}} >
\]

where

1. for each \( X \in \text{Sorts} \), \( \text{Par}_X \subseteq X \), the *parameters of sort* \( X \), where \( \text{Par}_X \) is a countably infinite set.
2. for any \( X \in \text{Sorts} \), \( \text{Par}_X \) is disjoint from any sort other than \( X \).
**Anchors** Parameters are assigned a value by an anchor. An anchor is a function which maps parameters to objects of the appropriate sort. An anchor may map a parameter to a parameter.

If $S = < A, Sorts, Rlns >$ is a parametric structure then an $S$-anchor $h$ is a partial function $h : \bigcup_{X \in Sorts} \{ Par_X \} \to \bigcup_{X \in Sorts} \{ X \}$ such that for any parameter $x \in Par_X$, $h(x) \in X$ if $h(x)$ is defined (i.e., anchors assign parameters of a given sort to objects of that sort, including parameters).

**Parametric objects** The presence of a parameter in an object makes it into a parametric object. Parameters themselves are parametric objects whose parameters are themselves; when a parametric object $o$ is constructed from other parametric objects $o_1, \ldots, o_n$ the parameters of $o$ are the union of the sets of parameters of $o_1, \ldots, o_n$, except in the special case of the binding operation $\lambda$. If $S = < A, Sorts, Rlns >$ is a parametric structure, we can define a function $\text{par}$ based on $S$ which for each object of each sort returns the set of its parameters:

1. if $x$ is a parameter (i.e., $x \in Par_X$ for some sort $X$), then $\text{par}(x) = \{ x \}$
2. if $o = o_1 \ast o_2$ for some non-binding operator $\ast \in Rln$, then $\text{par}(o) = \text{par}(o_1) \cup \text{par}(o_2)$
3. if $X$ is a set then $\text{par}(X) = \bigcup_{x \in X} \text{par}(x)$. In particular, $\text{par}(X \cup Y) = \text{par}(X) \cup \text{par}(Y)$ and $\text{par}(< X_1, \ldots, X_n >) = \text{par}(X_1) \cup \ldots \cup \text{par}(X_n)$

The definition of $\text{par}$ will need to be extended for parametric structures which include other kinds of objects than those covered here. We shall do this on a case by case basis as we introduce the new kinds of objects.

**Substitution** A substitution operation produces a new object in the situation theoretic universe by substituting objects supplied by an anchor for parameters in a parametric object. If $S = < A, Sorts, Rlns >$ is a parametric structure, then $\text{Sub}$ is a $S$-substitution operation if for any anchor $h$

1. if $X$ is a parameter $\text{Sub}(h, X) = h(X)$ if $X \in \text{dom}(h)$ and $\text{Sub}(h, X) = X$ if $X \notin \text{dom}(h)$
2. $\text{Sub}(h, b) = b$ if $h$ is the empty function
3. $\text{Sub}(h, b) = \text{Sub}(h_0, b)$ where $h_0$ is the restriction of $h$ to $\text{par}(b)$
4. $\text{Sub}(h, b_1 \ast b_2) = \text{Sub}(h, b_1) \ast \text{Sub}(h, b_2)$, if $\ast$ is a non-binding operation. $\text{Sub}(h, b_1 \ast b_2)$ is not defined if $\text{Sub}(h, b_1) \ast \text{Sub}(h, b_2)$ is not defined.
5. $\text{par}(\text{Sub}(h, b)) = (\text{par}(b) - \text{dom}(h)) \cup \bigcup_{x \in \text{dom}(h)} \{ \text{par}(h(x)) \}$

**Building a parametric universe** We will define parametric versions of all the structures that contributed to non-parametric universes.
**Parametric predicate structure**  Assignments include assignments of roles to parametric objects (including parameters). A parametric object is appropriate to a predicate if there is some way of anchoring its parameters to non-parametric objects which is appropriate. A *parametric predicate structure* is a parametric structure $S$ based on a predicate structure \( \{A, \{Pred, Asst, Obj\}, \{Roles, \Sigma, appr\}\} \) where

\[\text{appr} \text{ is a relation between } Pred \text{ and } Asst \text{ such that for any } r \in Pred, appr(r, f) \text{ implies } \]
\[f \text{ is a function whose domain is } \Sigma(r) \text{ and } appr(r, f) \iff \text{ there is some } S\text{-anchor } h \text{ such that }\]
\[par(Sub(h, r)) = par(Sub(h, f)) = \emptyset \text{ and } appr(Sub(h, r), Sub(h, f))\]

**Parametric basic infon structures**  The infon forming operations create basic infons using assignments of any parametric objects that are appropriate to the relations, using the notion of appropriateness in parametric predicate assignment signatures. A *parametric basic infon structure* is a parametric structure based on a basic infon structure such that

- it obeys *1

*2 \( \ll \cdot \gg^+, \ll \cdot \gg^- \) are non-binding operations

**Parametric infon algebras**  Parametric infons can be conjoined and disjoined just as non-parametric infons are; the parameters of the conjunction (disjunction) are the union of the sets of parameters of the conjuncts (disjuncts), i.e., the conjunction and disjunction operations are non-binding.

A *parametric infon algebra* is a parametric structure based on an infon algebra such that

- it obeys *1–*2

*3 \( \land \) and \( \lor \) are non-binding operations

**Parametric situation structures**  The supports relation holds only between situations (not situation parameters) and non-parametric infons. Apart from situation parameters there are no parametric situations.

A *parametric situation structure* is a parametric structure based on a situation structure such that

- it obeys *1–*3

*4 \( s \models \sigma \) implies \( \text{par}(s) = \text{par}(\sigma) = \emptyset \)
**Parametric basic proposition structures** The proposition forming operation creates propositions using assignments of any parametric objects that are appropriate to the types, using the notion of appropriateness in parametric predicate assignment signatures.

A *parametric basic proposition structure* is a parametric structure based on a basic proposition structure such that

\[ \text{it obeys } *1-*4 \]

\[ \text{*5 } p \in \text{True implies } \text{par}(p) = \emptyset \]

**Parametric Austinian basic proposition structures** There are parametric Austinian propositions but only non-parametric ones can be true.

An *parametric Austinian basic proposition structure* is a parametric structure based on an Austinian basic proposition structure such that it obeys *1-*5.

**Parametric proposition algebras** Conjunction, disjunction and negation are extended to parametric propositions. These are non-binding operations, i.e. the parameters of a complex proposition is the union of the sets of parameters of the propositions from which it is formed.

A parametric (*Austinian*) *proposition algebra* is a parametric structure based on an (*Austinian*) proposition algebra such that it obeys *1-*5.

**Parametric universes** A parametric situation theoretic universe is like a non-parametric one except that it is built out of the corresponding parametric structures and parametric objects are included among the elements of the set theoretic universe.

A *parametric situation theoretic universe* is a parametric structure based on a situation theoretic universe such that it obeys *1-*5.

### 3.1.8.3 Abstraction

Once we have parametric structures, we can introduce abstracts as follows. A new operation is defined: *abstraction*. New objects in the universe called *abstracts* are obtained by (simultaneous) abstraction over one or more parameters in an object. Intuitively abstracts might be thought of as objects with holes in them, their roles, which can have objects assigned to their roles in order to fill the holes and make an object which is not an abstract.\(^{10}\) We will introduce *assignments* to the roles, conceived of as functions in the set theoretic sense. An

\(^{10}\)Or “less of an abstract” since we can also form abstracts out of parametric abstracts.
appropriate assignment for (15) is one that assigns objects to all of the role indices of the abstract (and possibly to other objects as well).

\[
\begin{bmatrix}
  r_1 \rightarrow a \\
  r_2 \rightarrow c
\end{bmatrix}
\]

The structures we are going to introduce next, lambda structures, are include both abstraction and application operations. Application is an operation which takes abstracts and assignments into objects. Applying abstracts to assignments ‘fills in’ the holes represented by parameters with the objects assigned to the role indices by the assignment. The operation which application uses is known as substitution. We represent application as in (2)

\[
\begin{bmatrix}
  r_1 \rightarrow a \\
  r_2 \rightarrow c
\end{bmatrix}
\]

(2) represents the same proposition as (3).

\[
p(a, c)
\]

(3) is the result of removing the tab from the top of (29) and replacing the parameters X and Y with a and c as indicated by the assignment. This operation is known as β-conversion.

**Lambda Structures** More formally, abstracts are formed from indexed sets of parameters and parametric objects. The parameters in the indexed set are bound within the abstract. Substitution of bound parameters results in the same abstract (α-equivalence). A lambda structure

\[
\langle A, Abstrs \cup Sorts, \{\lambda, apply\} \cup Rlms \rangle
\]

is such that
1. \(<A, \text{Sorts}, \text{Rlns}\)> is a parametric structure with objects \(\text{Obj}\)

2. \(\lambda\) is a binary operation whose first argument is a one-one function \(f\) whose domain is included in the non-parametric objects \(\{x \in \text{Obj} \mid \text{par}(x) = \emptyset\}\) and whose range is included in the set of parameters \(\bigcup_{X \in \text{Sorts}} \{\text{Par}_X\}\) and whose second argument is an object (i.e. member of \(\text{Obj}\)) such that \(\text{ran}(f) \subseteq \text{par}(o)\)

3. \(\text{par}(\lambda(f, o)) = \text{par}(o) - \text{par}(f)\) (\(\lambda\) is a binding operation)

4. \(\text{Sub}(h, \lambda(f, o)) = \lambda(f', \text{Sub}(h', o'))\) where \(h'\) is the restriction of \(h\) to the parameters of \(\lambda(f, o)\) (i.e. no substitution for bound parameters) and \(f'\) and \(o'\) are the result of relettering parameters in the range of \(f\) (i.e. bound in \(\lambda(f, o)\)) so that no parameters in the range of \(f'\) are parameters of \(h'\) (i.e. no (free) parameters of \(h'\) are captured during the substitution).

5. If \(h\) is a one-one function from \(\text{par}(f)\) to parameters such that for any parameter \(x, h(x)\) is a parameter of the same sort as \(x\) and \(\text{ran}(h) \cap \text{par}(\lambda(f, o)) = \emptyset\) (i.e., no new parameters are getting bound) then
   \[\lambda(f, o) = \lambda(\text{Sub}(h, f), \text{Sub}(h, o))\]
   \((\alpha\text{-equivalence})\)

6. If \(X \in \text{Sorts} \cup \text{Abstr}\), then there is a set \(X_{\text{abstr}} \subseteq \text{Abstr}\) which is the set of abstracts \(\lambda(f, o)\) such that \(o \in X\). No other set is included in \(\text{Abstr}\).

7. \(\text{apply}\) is a partial operation defined on abstracts \(\lambda(f, o)\) and parametric assignments \(g\) such that \(\text{dom}(f) \subseteq \text{dom}(g)\) and \(\text{apply}(\lambda(f, o), g) = \text{Sub}(h, o)\) (if defined) where \(h\) is defined by
   \[(a) \ \forall x \in \text{dom}(f) \ h(f(x)) = g(x)\]
   \[(b) \ h\text{ is undefined otherwise}\]

Remark If \(\lambda(f, o) \in X_{\text{abstr}}\) and \(\text{apply}(\lambda(f, o), g)\) is defined then \(\text{apply}(\lambda(f, o), g) \in X\) [I think this follows.]

Remark \(\text{apply}(\lambda(f, o), f) = o\)

Remark \(\text{apply}\) behaves as a non-binding operation, i.e. \(\text{Sub}(h, \text{apply}(\lambda(f, o), g)) = \text{apply}(\text{Sub}(h, \lambda(f, o)), \text{Sub}(h, g))\).

3.1.8.4 The language of abstracts

1. Term abstracts
Syntax

If $B$ is a box of some sort $\nu$, $X_1, \ldots, X_n$ are parameter symbols, and $r_1, \ldots, r_n$ are constants, then the following box $\alpha$ is a $\nu$-abstract box:

$$
\begin{array}{c}
r_1 \rightarrow X_1, \ldots, r_n \rightarrow X_n \\
B
\end{array}
$$

Semantics

$$[\alpha] = \lambda F[B]$$

where $F$ is the indexed family of parameters, where $[X_i]$ is indexed by $[r_i]$, i.e., $F([r_i]) = [X_i]$ for each $i = 1, \ldots, n$.

That is, this term denotes the object which results from abstracting the parameters $[X_1], \ldots, [X_n]$ from $[B]$ using $[r_1], \ldots, [r_n]$ as role indices, by means of abstraction in the sense of $[AL]$. This will only be defined if $[r_i] \rightarrow [X_i]$ determines a one-to-one function $F$. If $[B]$ is a restricted object, then the abstract will automatically have appropriateness conditions corresponding to the restrictions on $[B]$. These appropriateness conditions restrict the assignments to which the abstract can be applied.

Notation:

- When $[r_1], \ldots, [r_n]$ are the natural numbers $1, \ldots, n$ we can write $r_1 \rightarrow X_1, \ldots, r_n \rightarrow X_n$ as $X_1, \ldots, X_n$, following standard practice in logical notation by using the order to encode the numbers.

- If case $B$ is a proposition box, then the new term is also a type box, and can be written

$$
\begin{array}{c}
r_1 \rightarrow V_1, \ldots, r_n \rightarrow V_n \\
B
\end{array}
$$

Similarly, if $B$ is an infon box, then the new term is also a relation box, and can be written in the same way.

2. Application of abstracts

Syntax

If $B$ is a $\nu$-abstract term and $A$ is an assignment term then the following is a $\nu$ term $\alpha$:

$$
\begin{array}{c}
B A
\end{array}
$$

Semantics

$$[\alpha] = Apply(\lambda F[B], [A])$$

That is, this term denotes the result of applying (in the sense of $[BC]$) the abstract $[B]$ to the assignment $[A]$ providing the domain of $[A]$ is the same as the role indices of $[B]$. Intuitively, this results in the substitution of the objects in the range of $[A]$ in $[B]$. 

140
Properties of Abstracts  One important property of abstracts is that it does not matter which parameter symbols you use to represent the roles – the particular parameters have been abstracted away from. Thus (29) represents the same object as (4)

$$\begin{align*}
\text{(4)} & \quad r_1 \rightarrow Z, r_2 \rightarrow W \\
p(Z, W)
\end{align*}$$

In addition, the order in which the parameter symbols together with their role indices are written down does not make any difference. Thus (29) is identical with (5).

$$\begin{align*}
\text{(5)} & \quad r_2 \rightarrow Y, r_2 \rightarrow X \\
p(X, Y)
\end{align*}$$

3.1.8.5 Quantification

In Generalized Quantifiers theory, quantifiers are relations between sets [Barwise and Cooper, 1981; Gärdenfors, 1987]. As each abstract can be associated with a set, we can reformulate this proposal in the framework introduced here by treating generalized quantifiers as relations between types.

A sentence such as *Every Scotsman heard about Bannockburn* can be formalized as follows (again, translating the proper name *Bannockburn* as a constant):

$$\begin{align*}
\text{(30)} & \quad \text{every, } (i \rightarrow X) \\
& \quad \begin{align*}
\text{scotsman}(X) & \quad \text{heard-about}(X, b)
\end{align*}
\end{align*}$$
3.1.9 Context-Dependency

3.1.9.1 Utterance Situations and the Relational Theory of Meaning

The interpretation of many lexical items is context-dependent. A clear example of context-dependent lexical items are pronouns: the value assigned to a pronoun like *he* when uttered as part of statement Φ depends entirely on the context in which Φ is uttered. But pronouns are not the only case of context-dependent lexical item: indexical expressions, definite descriptions, proper names are all context-dependent in greater or lesser measure. Quantifiers have a context-dependent aspect as well, as shown by the phenomenon of domain restriction, discussed above [Cooper, 1993; Poesio, 1994a]. Nor are NPs the only category including context-dependent elements: Partee has convincingly argued that tense has anaphoric properties [Partee, 1973], and Kratzer has proposed that the interpretation of modals, as well, depend on the contextual selection of a modal base [Kratzer, 1977]. Barwise and Perry argue in [1983] that this efficiency of language is one of the fundamental properties of communication, and it should play an important role in the development of a semantic theory.

What’s more, statements do not simply exploit a context to tell us something about a certain situation; they can also tell us something about the context of the statement. The whole point of statements like *It’s me* is to communicate information about the speaker of the utterance. This phenomenon of reverse information, to use once again Barwise and Perry’s term, also seems to be a central aspect of communication.

These observations are consistent with the ‘relational’ theory of meaning proposed by Barwise and Perry. According to them, ‘meaning’ is a relation between situations: thus, we can say that *Smoke means fire* because we are attuned to a regularity in our environment according to which for every situation in which there is smoke, there is a situation in which there is fire. Barwise and Perry propose that the meaning of a statement, as well, is a relation between two situations: an utterance situation—the context in which that statement is made, that is treated as a situation in its own right—and a described situation—the situation the statement is ‘about’. This means that the statement *John left* imposes constraints both on the situation the statement is about—the statement makes a true claim only if the person called ‘John’ left—and on the situation in which that statement is uttered—namely, whoever makes the statement must know someone called ‘John’.

3.1.9.2 Presuppositions as Restrictions

The possibility for a sentence to impose constraints on utterance situations is typically formulated by having lexical items contribute parameters to be anchored by context. This way of formalizing context-dependency was spelled out in most detail by Gawron and Peters [1990]. They propose, for example, that the lexical rule NP → *John* is associated with the following semantic rule:
This rules reads as follows: the utterance of the proper name John in an utterance situation c means the parameter X if and only if the utterance situation c supports the information

\( \langle \text{REFREL}, np, X \langle \text{NAMED}, X, "John" \rangle \rangle \)

namely, the information that that NP is used to refer to an object in c characterized by the property of being named “John”.

Gawron and Peters use in (31) the restricted parameter \( X \langle \text{NAMED}, X, "John" \rangle \) to specify the built-in restrictions on what the parameter can be anchored to. Restricted parameters have proved difficult to formalize, however; for this reason, Plotkin [1987] proposed to adopt instead a restriction operation \( \mathcal{R} \). Any object may be restricted by a proposition. If the proposition is true then the restriction is identical with the unrestricted object. If the proposition is false, the restriction is undefined. We will use restrictions to represent presuppositions.

As an example of the use of restrictions, let us consider the semantics of proper names. As proposed by Gawron and Peters, we assume that proper names are context dependent elements, that contribute to a sentence’s meaning a parameter and a restriction. The proper name Smith, for example, introduces a parameter, say \( X_1 \), together with a restriction to the effect that the referent of the parameter be an individual called “Smith”. The information \( \text{named}(X_1, "Smith") \) must be supported by a contextually introduce resource situation—a situation introduced as part of the discourse. The translation of the sentence Smith hired Jones in EKN notation is shown below.

(32) a. Smith hired Jones

b. \( \text{hired}(X_1, X_2) \)

\( \text{named}(X_1, "Smith") \)

\( \text{named}(X_2, "Jones") \)

3.1.9.3 Restriction algebras

An (Austinian) restriction algebra \( < A, \text{Prop}, \text{Sorts}, \{ \} \cup \text{Prop Rln} > \) is such that
1. \(<\text{Prop}_{\text{Sorts}}, \text{Prop}_{\text{Aux}}, \text{Prop}_{\text{Rlh}}\> < A, \text{Prop}_{\text{Sorts}} \cup \text{Prop}_{\text{Rlh}}\) is an (Austinian) proposition algebra whose objects are \(\text{Obj}\) and whose true propositions are \(\text{True}\).

2. \(\circ\) is a binary operation on \(\text{Obj}\) and \(\text{True}\) such that for any \(o \in \text{Obj}\) and \(p \in \text{True}\),
\[
o \circ p = o
\]

3. \(o \circ (p \circ p') = (o \circ p) \circ p' = o \circ p \land p'
\]

**Remark** If \(p\) and \(p'\) are non-parametric then this follows from the first two axioms:
\[
o \circ (p \circ p') = (o \circ p) \circ p' = o \circ p \land p'
\]

### 3.2 Syntax-semantics Interface

#### 3.2.1 A Sample Grammar Illustrating the syntax-semantics interface

To give a preliminary idea of the approach to grammar writing taken in this document, we will begin by specifying a small grammar, that can be used to analyze the sentence *Smith hired Jones*. We will then expand this grammar and refine some of the rules.

In Situation Theory, the centrality of utterances is reflected by the fact that a grammar rule specifies the meaning of an utterance \([X \circ a]\), where \(X\) is any lexical or phrasal category, and \(a\) is an instance of that category. Each utterance constituent is also an utterance: for example, one of our rules will specify the meaning of the utterance \([\text{NP}\; \text{Smith}]\). But perhaps the most striking feature of the grammars presented in [Barwise and Perry, 1983; Gawron and Peters, 1990; Devlin, 1991] is that the rules which compose them do not simply assign a single meaning to each utterance; they additionally specify constraints on the situation in which the utterance occurs, thus implementing the hypothesis that meaning is relational. For example, Gawron and Peters specify the meaning of proper names as follows ([Gawron and Peters, 1990], p.170):

\[
(33) \quad \text{NP \rightarrow John}
\]
\[
c[\text{NP}] \circ \text{DO, RES} \text{ iff } (C \models \langle \text{REFREL, NP, DO, RES} \models \langle \text{NAMED, DO, "John"} \rangle \rangle)
\]

Several aspects of this rule are unusual. First of all, it specifies the meaning of a use of the NP *John* as a relation between a discourse situation \(C\) (for \(\text{Circumstances}\)) and the two parameters \(\text{DO}\) and \(\text{RES}\) (for \(\text{Described Object}\) and \(\text{Resource Situation}\), respectively) instead of as a function from the use to a single object. Secondly, it imposes a constraint on \(C\), namely, that a discourse situation in which the NP *John* is used must be one in which that utterance is used to refer to an object in the discourse situation restricted by the requirement that it refers to an object named *John*. 

144
In the grammar we are going to present, the intuitions that meaning is a relation between objects, and that uses of natural language expressions impose constraints on the utterance situation, are captured in a more traditional way, in the sense that we use a function \([u] \rightarrow \text{denotation}\) from utterances to denotations of expressions to express the meaning of utterances. The context-dependency of such meanings is captured by letting the meaning of an utterance \(u\) be an abstract whose values have to be supplied by context. For example, the meaning of an utterance of the sentence *Smith hired Jones*, which involved uttering the proper name *Smith* (let us call this \(u_1\)), the verb *hired* (let us call this \(u_2\)), and the proper name *Jones* (utterance \(u_3\)), is shown in (34b). This meaning is a type obtained by simultaneous abstraction over the parameter \(S\) which stands for the described situation of utterance \(u_2\), the parameters \(X_1, R_1, X_2\) and \(R_2\) that stand for the referents and the resource situations of the proper names *Smith* and *Jones*, and the parameter \(N\), that stands for the utterance time.

(34)  
\[
\begin{align*}
\text{a.} & \quad \text{Smith hired Jones} \\
\text{b.} & \quad \exists^* \quad \text{hire}(X_1, X_2, T)
\end{align*}
\]

Note how the abstraction is used to encode the relational theory of meaning, and how the restriction are used to encode some of the constraints proposed by Gawron and Peters. The constraint on \(C\) that it must be one in which the speaker refers to an object named *Smith
becomes a restriction on the parameter $DS$. Gawron and Peters’ assumption that a rule of the form $[XP \ ZP \ YP]$ included a constraint of the form $(C^{XP} \geq C^{ZP} \land C^{YP})$ (the circumstances of the use of $XP$ must provide all values required by $ZP$ and $YP$) will be implemented by requiring that the meaning of $XP$ is an abstraction whose domain includes all indices in the domain of the meanings for $ZP$ and $YP$.

The content of an utterance is obtained by fixing the values of the parameters that represent its context-dependent aspects. Thus, for example, the content of $Smith \ hired \ Jones$ is obtained by fixing the values of the parameters $DS$, $N$, $S$, …. The content of an assertional utterance $u$ is a proposition, obtained by applying the $\exists^*$ operation to the content $\Phi$ of the utterance $u'$ of type $S$ that is the only constituent of $u$. If $\Phi$ is already a proposition, $\exists^* \Phi$ is the same proposition; if $\Phi$ is a type with parameters $X_1 \ldots X_n$, $\exists^* \Phi$ is the proposition that is true if there is an assignment $a$ of values to $X_1 \ldots X_n$ such that $\Phi[a]$ is a true proposition. The content of other utterances depends on the type of utterance.

Here follows our initial grammar:

**LEX-PN** If $u$ is a use of $[NP \ Smith]$, 

**LEX-TVERB-TENSED** If $u$ is a use of $[V \ hired]$, 

\[
[u] = \begin{array}{c}
\text{ds} \rightarrow DS, \langle do, u \rangle \rightarrow X, \langle \text{exploits}, u \rangle \rightarrow R \\
\text{DS} \rightarrow \text{ref\hspace{1pt}u\hspace{1pt}X, exploits(u,R)} \\
\text{R} \rightarrow \text{named(X,“Smith”)}
\end{array}
\]
We will define the combination operation $A \{ B \}$ in the next section; briefly, it creates an assignment that is appropriate for both $A$ and $B$, and it returns the abstraction obtained by this assignment and the application of the content of $A$ to the content of $B$.

For readability’s sake, in most of the examples below we will use the names assigned to the indices and some naming conventions introduced in the next section to indicate what each parameter does, so as to be able to specify only those restrictions that are absolutely necessary to understand the meaning we are assigning to a sentence constituent. In particular, we will more often than not omit the restrictions on the discourse situation, and simply introduce the parameter $DS$ for each meaning without specifying any restrictions on it.

The (sub) utterances associated with an utterance of the sentence *Smith hired Jones* are associated by the grammar above with the following meanings (forcing the notation slightly, we write here and elsewhere, for example, $[[NP \ Smith]]$ where we should write $[[u]]$, with $u$ the particular use of that syntactic construct):

---

**PS-TVP** If $u$ is a use of $[VP \ V NP]$, 

$[u] = [V][[NP]]$

**PS-S** If $u$ is a use of $[S \ NP \ VP]$, 

$[u] = [VP][[NP]]$

**PS-ASTN** If $u$ is a use of $[ASTN \ S]$, 

$[u] = \lambda a. \exists^* \beta$, where $[S] = \lambda a. \beta$
1. [[NP Smith]]: as specified by LEX-PN.
2. [[V hired]]: as specified by LEX-TVERB-TENSED.
3. [[NP Jones]]: like the content of Smith, *mutatis mutandis*.

```
| ds → DS, (utti-time, u) → N, (descr-sit, u₂) → S,  
| (rt, u₂) → T, (do, u₂) → X₂, (exploits, u₂) → R₂,  
| subj → Y |
|---------|------------------|
| S       |                  |
|         |                  |
| R₂      |                  |
|         | named(X₂, "Jones") |
| U₂      |                  |
|         | utti-time(U₂, N)  |
| T ⊲ N   |                  |
```

4. [[VP hired Jones]] =

```
hire(Y, X₂, T)  
```

```
| ds → DS, (utti-time, u) → N, (descr-sit, u₂) → S,  
| (rt, u₂) → T, (do, u₂) → X₂, (exploits, u₂) → R₂,  
| (do, u₁) → X₁, (exploits, u₁) → R₁,  |
| subj → Y |
|---------|------------------|
| S       |                  |
|         |                  |
| R₁      |                  |
|         | named(X₁, "Smith") |
| R₂      |                  |
|         | named(X₂, "Jones") |
| U₂      |                  |
|         | utti-time(U₂, N)  |
| T ⊲ N   |                  |
```

5. [[S Smith hired Jones]] =

```
hire(X₁, X₂, T)  
```

```
| ds → DS, (utti-time, u) → N, (descr-sit, u₂) → S,  
| (rt, u₂) → T, (do, u₂) → X₂, (exploits, u₂) → R₂,  
| (do, u₁) → X₁, (exploits, u₁) → R₁,  |
| subj → Y |
|---------|------------------|
| S       |                  |
|         |                  |
| R₁      |                  |
|         | named(X₁, "Smith") |
| R₂      |                  |
|         | named(X₂, "Jones") |
| U₂      |                  |
|         | utti-time(U₂, N)  |
| T ⊲ N   |                  |
```

6. I [[ASTN [S Smith hired Jones]]]: this is (34b).

In order to ensure that the rules above apply correctly, all meanings discussed below are assumed to be abstractions over contextually-supplied values; at the very least, these abstractions involve the indices ‘ds’ and ⟨utti-time, u⟩.
3.2.1.1 Inferences with EKN

To give a basic feeling of what can be inferred from EKN representations, we will consider now a proof that (35b) follows from (34b).11

(35) a. Someone hired Jones.

b. \[ \exists^* \left( \begin{array}{c}
X_1 \\
\text{person}(X_1) \\
\text{hire}(X_1, X_2, T) \\
\end{array} \right) \]

In order to prove that (35b) follows from (34b), we have to show that for every assignment \( f \) such that \( f \in \text{Ext}((34b)) \), i.e., for every way of fixing the context-dependent aspects of \( \text{Smith hired Jones} \), there is an assignment \( f' \in \text{Ext}((35b)) \), that is, there is a way of fixing the context-dependent values of (35b) such that the content of the sentence is a true proposition.

The way one does this is similar to what one would do in DRT. Let \( U \) be a lambda situation theoretic universe, and let \( f \) be an assignment such that \( f \in \text{Ext}((34b)) \), i.e., an assignment that assigns to DS, N, S, T, \( X_1 \), \( R_1 \), \( R_2 \), \( X_2 \) values such that the restricted proposition

---

11 Running a bit ahead of ourselves, we have already used the translation for indefinites, whereby an indefinite like someone introduces a parameter that gets bound by the abstract that is existentially quantified over to get the content of the utterance Someone hired Jones.
is a true proposition. Then, we can construct an assignment $f'$ over $U$ that makes the following proposition true:

$f'$ assigns to $T$, $S$, $R_2$, $X_2$, and $N$ the same values assigned to them by $f$. The proposition thus obtained is true if there is an assignment $ff'$ that assigns to $X_1$ a value that makes the resulting proposition true: this is the value that $f$ assigns to $X_1$.\(^{12}\)

### 3.2.2 Notation and Operations

We now introduce in more detail the notation used to define the grammar.

\(^{12}\)Strictly speaking, to prove the entailment we would also need to have person($X_1$) in the restriction of (34b).
3.2.2.1 Parameter Sorts

The letters and symbols used for parameters in the grammar encode information about their sort, according to the following conventions:

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Sort</th>
</tr>
</thead>
<tbody>
<tr>
<td>S, MS</td>
<td>situation (mental state)</td>
</tr>
<tr>
<td>T</td>
<td>time</td>
</tr>
<tr>
<td>X, Y, Xi</td>
<td>individual</td>
</tr>
</tbody>
</table>
| P          | \(([X] \rightarrow \text{proposition}) \rightarrow \text{proposition}\)  
            | i.e. type of types of individuals, a noun-phrase “content” |
| Prpn       | proposition  
            | i.e. a sentence content |
| P          | \([X] \rightarrow \text{infon}\)  
            | i.e. property of individuals |
| Q          | \([X] \rightarrow \text{proposition}\)  
            | i.e. type of individuals |
| MProp      | \([S, T] \rightarrow \text{proposition}\)  
            | i.e. a type of situations and times, a “Montague proposition” |

The sortal restrictions on parameters could be expressed by including propositions like \((Prpn : \text{proposition})\) as restrictions, but adopting the convention above makes the notation easier to read.

3.2.2.2 Index Assignments

If ζ is an abstract with role indices \(r_1, \ldots, r_n\) then \(f\) is an index assignment for \(ζ\) if \(\{r_1, \ldots, r_n\} \subseteq \text{dom}(f)\), \(f\) is 1–1 and for any \(r \in \text{dom}(f)\), \(f(r)\) is a parameter which is not a parameter of \(ζ\). An example of an index assignment for

\[
\begin{array}{c}
\text{desc-sit} \rightarrow S, \langle rt, u \rangle \rightarrow T \\
S \\
\text{meet}(s, k, T)
\end{array}
\]

is:

\[
\begin{bmatrix}
\text{desc-sit} \rightarrow S' \\
\langle rt, u \rangle \rightarrow T
\end{bmatrix}
\]

If \(Z\) is a set of abstracts, \(f\) is a \textit{minimal index assignment} (MIA) for \(Z\) iff \(f\) is an index assignment for each \(ζ \in Z\), and \(\text{dom}(f) = \bigcup_{ζ \in Z} \text{roles}(ζ)\), the union of the role indices of each \(ζ \in Z\).
3.2.2.3 Application to partial assignments

In order to make the rules easier to read we introduce an abbreviation for application to partial assignments, i.e. those assignments that don't provide a value for all the roles of an abstract. This abbreviation, which is discussed in [Barwise and Cooper, 1993], is defined in terms of the normal notion of application to total assignments.

If \( \text{dom}(f) \cap \text{roles}(\zeta) \subset \text{roles}(\zeta) \) (i.e., the assignment only supplies values for a proper subset of the roles), then we write \( \zeta.f \) for \( \lambda f''(\zeta,f'') \) where \( f' \) is an extension of \( f \) which assigns unique parameters other than those in \( f \) or \( \zeta \) to each role in \( \text{roles}(\zeta) - \text{dom}(f) \) and \( f'' \) is \( f' \) restricted to \( \text{roles}(\zeta) - \text{dom}(f) \).

For example,

\[
\begin{array}{c}
\text{desc-sit} \rightarrow S, (rt,u) \rightarrow T \\
S \\
\text{meet}(s,k,T) \\
\text{.}[\{rt,u\} \rightarrow t]
\end{array}
\]

is

\[
\begin{array}{c}
\text{desc-sit} \rightarrow S \\
S \\
\text{meet}(s,k,t)
\end{array}
\]

3.2.2.4 Conventions for non-abstracts and empty assignments

We will adopt the following conventions. Let \( \zeta \) be an abstract. Then

- \( \zeta.f = \zeta \) if \( \text{dom}(f) \cap \text{roles}(\zeta) = \emptyset \)
- \( \lambda\emptyset \zeta = \zeta \)
  
  Where \( \emptyset \) is the empty assignment.
- if \( f' \) is a restriction of \( f \) to the roles of \( \zeta \), then \( \lambda f\zeta = \lambda f'\zeta \)

This will allow us to give general definitions where in some cases we want to apply a type to an assignment (in order to obtain content from meaning) and in other cases there will be
no abstraction over context parameters. According to this convention a proposition may also
be regarded as a zero-place type and the result of applying it to any assignment will be the
proposition itself.

3.2.2.5 Combination ("Linguistic application")

Let $\alpha$ and $\beta$ be abstracts. Then the combination of $\alpha$ and $\beta$, written $\alpha \{\beta\}$, is defined as
follows:

$$\alpha \{\beta\} = \lambda f(\alpha.f[\beta.f])$$

where $f$ is a mia for $\{\alpha, \beta\}$

3.2.2.6 Proposition and Type Merging

This operation is used to define the meaning of coordinated sentences and the meaning of
'sequencing' in discourse. If its two arguments are two propositions, it simply returns the
conjunction of these two. If they are two types, or a proposition and a type, it returns the
type obtained by merging the assignment(s) and conjoining the propositions.

1. If $p_1$ and $p_2$ are propositions, then $p_1 \oplus p_2 = p_1 \land p_2$.

   In the special case in which $p_1$ and $p_2$ are propositions about the same situation $s$, $p_1 = (s = \Phi)$
   and $p_2 = (s = \Psi)$, the extension of the conjunction of $p_1$ and $p_2$ is the same
   as the extension of the the proposition whose described situation is $s$, asserting that $s$
is of the type specified by $\Phi \land \Psi$. As this proposition is more useful, we define:

   $$p_1 \oplus p_2 = \begin{array}{c}
   s \\
   \Phi \\
   \Psi
   \end{array}$$

   We also extend the operation to the case in which one, or both, of the objects being
merged are restricted, by assuming that the $p_1 \oplus p_2$, in case one or both of $p_1$ and $p_2$ is
restricted, is a restricted proposition, the restriction being the union of the restrictions
of $p_1$ and $p_2$.

2. Let $t_1$ and $t_2$ be two types obtained by abstracting the parametric propositions $p_1$ and $p_2$ over the assignments $g_1$ and $g_2$. The merge of the types is defined as follows:

   $$t_1 \oplus t_2 = \lambda g. [t_1.g \oplus t_2.g]$$

   where $g$ is a MIA for $\{g_1, g_2\}$.

   $t_1 \oplus t_2$ is undefined in case there is no MIA for $g_1$ and $g_2$.  

153
3. Let \( p \) be a proposition and \( t \) a type. Then

\[
p \oplus t = \lambda a. [t.a \oplus p]
\]

### 3.2.2.7 Situation theory for syntax

Our rules will be of a form like

**Rule name** If \( u \) is a use of \([X \ Y \ Z]\),

with constituents \( u_1 \) and \( u_2 \) respectively . . .

Here \([X \ Y \ Z]\) represents a type in the situation theoretic sense. It is a type of utterance situation. We will not go into detail here about the interpretation of labelled bracketings and feature structures in situation theory but just give some examples to give concreteness to our rules. We will use \([\cdot ]\) here to represent the situation theoretic interpretation of the syntactic formalism.

[[S NP VP]] is to be

```
U
```

```
U1, U2
```

```
cat(U, s)
daughters(U, \{U1, U2\})
precede(U1, U2)
```

[[NP].[U1]]
```
[[VP].[U2]]
```

where

```
NP
```
```
U
```
```
cat(U, np)
```

and
Features such as *cat* and *daughters* are declared to be functional, i.e. no utterance (situation) may support more than one infon with this relation. We accommodate features on nodes by adding additional feature insons. For example,

In introducing feature structures as values involves introducing a parameter for the value and a restriction on it defined in terms of the interpretation of the embedded feature structure. Thus

We encode indices for reentrancy by introducing special role indices in the situation theoretic objects. Here we will use negative integers for those indices used in the feature structure and will use the positive integers otherwise. Thus
Thus feature structures correspond to either a type or a type-abstract. When the feature structure is a type-abstract we call its roles the context roles of the feature structure. We can define context roles or “croles” as follows:

If $\zeta$ is a type (or not an abstract) then $\text{croles}(\zeta) = \emptyset$. Otherwise $\text{croles}(\zeta) = \text{roles}(\zeta)$.

In combining feature structures we always add new roles for those in the individual feature structures which are indexed by positive integers (i.e. those that are not labelled in order to obtain reentrancy). We will do this by means of concatenation of assignments whose domains are initial chains of the positive integers. We will represent this as $\downarrow$. Recall that $[X, Y]$ is an abbreviation for the assignment $[1 \rightarrow X, 2 \rightarrow Y]$. Then $[X, Y] \uparrow [W, Z]$ will be $[X, Y, W, Z]$. To achieve reentrancy we will always make the roles labelled by the same negative integers fall together when we combine feature structures. By way of illustration here is a general rule for binary branching labelled bracketings.

$[[X \ Y \ Z]]$ is
\[ \lambda g \cup (f \cup f' \cup [U_1, U_2]) (\text{cat}(U, x)\text{ daughters}(U, \{U_1, U_2\})) \]

where:

\[ \text{dom}(g) = \{ r \mid r \in \text{croles}([Y]) \cup \text{croles}([Z]) \text{ and } r < 0 \} \text{ and } \forall r \in \text{dom}(g), g(r) \text{ is a unique parameter not of } [Y] \text{ or } [Z] \]

\( f \) is a mia for \([Y]\) and \( f' \) is a mia for \([Z]\), \( \text{ran}(f) \) and \( \text{ran}(f') \) are disjoint and do not include any parameters in \([Y]\) or \([Z]\).

Unification is defined in such a system in terms of conjunction.

\[ \zeta \cup \xi = \]

\[ \lambda g \cup (f \cup f') (\zeta.g.f.[X] \xi.g.f'.[X]) \]

if this is a consistent type. Undefined otherwise. The requirements placed on \( g, f \) and \( f' \) are exactly similar to those used in the syntax interpretation rule above. Note that it would be straightforward to let unification be defined even when an inconsistent type results, thus allowing local contradictions which would not lose any information.

### 3.2.3 The Gawron & Peters Fragment

By way of illustration of this style of syntax-semantics interface, we give a presentation of the fragment from [Gawron and Peters, 1990]. Gawron and Peters present meanings as a relation between varying numbers of arguments. For example, their rule for the proper name \( \text{John} \) is as follows.

\[ \text{NP} \rightarrow \text{John} \]

\[ \text{c[\text{NP}]_{DO, RES} iff (C \models \langle \text{REFREL, NP, DO, RES} \models \langle \text{NAMED, DO, "John"} \rangle \rangle)} \]
We will in general make our meanings have roles corresponding to those of Gawron and Peters’ meaning relation except that we will not have the role $C$ for circumstances. Our assignments to the roles in abstracts will have most of the technical function of Gawron and Peters’ circumstances situation.

Instead of treating meanings as relations with a DO-role for “described object,” we will represent Gawron and Peters’ meanings as abstracts which when they are applied to an appropriate assignment (representing the context) return as a value what Gawron and Peters would call the described object. It turns out that this variant is more convenient for creating compositional interpretations given the way we have set things up.

3.2.3.1 Lexicon

269. If $u$ is a use of type $[John]_{NP}$ then $[[u]] = X$.

270. If $u$ is a use of type $[he]_{NP}$ then if $u \models \langle covary, u, < role, u' >; 1 \rangle$, $[[u]] = X$.

Otherwise, $[[u]] = X$.

271. If $u$ is a use of type $[himself]_{NP}$ and if $u \models \langle covary, u, < role, u' >; 1 \rangle$, $[[u]] = [R, \text{male}]$.
Otherwise, \([u]\) is undefined.

272. If \(u\) is a use of type \([\text{she}]\) \(\text{NP} \quad \text{[+def]}\) then if \(\models \langle \text{covary, } u, < \text{role, } u' >; 1 \rangle\), \([u]\) =

\[
\begin{array}{c}
\langle \text{role, } u' \rangle \rightarrow X, \langle \text{res, } u \rangle \rightarrow R
\end{array}
\]

\[
\begin{array}{c}
R
\end{array}
\]

\[
\begin{array}{c}
\text{male}(X)
\end{array}
\]

Otherwise \([u]\) =

\[
\begin{array}{c}
\langle \text{ref, } u \rangle \rightarrow X, \langle \text{res, } u \rangle \rightarrow R
\end{array}
\]

\[
\begin{array}{c}
R
\end{array}
\]

\[
\begin{array}{c}
\text{female}(X)
\end{array}
\]

273. If \(u\) is a use of type \([\text{herself}]\) \(\text{NP} \quad \text{[+def]}\) and if \(\models \langle \text{covary, } u, < \text{role, } u' >; 1 \rangle\), \([u]\) = then \([u]\) =

\[
\begin{array}{c}
\langle \text{role, } u' \rangle \rightarrow X, \langle \text{res, } u \rangle \rightarrow R
\end{array}
\]

\[
\begin{array}{c}
R
\end{array}
\]

\[
\begin{array}{c}
\text{R. female}
\end{array}
\]

\[
\begin{array}{c}
\text{unique}
\end{array}
\]

\[
\begin{array}{c}
R
\end{array}
\]

\[
\begin{array}{c}
\text{female}(X)
\end{array}
\]

Otherwise, \([u]\) is undefined.

274. If \(u\) is a use of type \([\text{his}]\) \(\text{NP} \quad \text{[+def]}\) case: poss then if \(\models \langle \text{covary, } u, < \text{role, } u' >; 1 \rangle\), \([u]\) =
275. If \( u \) is a use of type [her] then if \( u \models \langle \text{covary, } u, \langle \text{role, } u' \rangle; 1 \rangle \), \([u] = \)

\[
\begin{bmatrix}
\text{NP} \\
+ \text{def}
\end{bmatrix}
\begin{bmatrix}
\text{case: poss}
\end{bmatrix}
\]

276. If \( u \) is a use of type [student]_{CN} then \([u] = \)

\[
\begin{bmatrix}
\text{student}(X) \\
\text{utt-time}(u, L)
\end{bmatrix}
\]
277. If \( u \) is a use of type \([\text{paper}]_{CN}\) then \([u] =\)

\[
\begin{align*}
(\text{utt}, u) &\rightarrow L \\
\text{instance} &\rightarrow X \\
\text{paper}(X) &\rightarrow \text{utt-time}(u, L)
\end{align*}
\]

278. If \( u \) is a use of type \([\text{revised}]_{V}^{(+\text{tns})}\) then \([u] =\)

\[
\begin{align*}
(\text{utt}, u) &\rightarrow L \\
<\text{subj}, u> &\rightarrow X, <\text{obj}, u> \rightarrow Y, <\text{tns}, u> \rightarrow Z \\
\text{revising} \left[ \begin{array}{c}
\text{reviser} \rightarrow X \\
\text{revised} \rightarrow Y \\
\text{loc} \rightarrow Z
\end{array} \right] &\rightarrow \text{utt-time}(u, L)
\end{align*}
\]

279. If \( u \) is a use of type \([\text{revising}]_{V}^{(-\text{tns})}\) then \([u] =\)

\[
\begin{align*}
(\text{utt}, u) &\rightarrow L \\
<\text{subj}, u> &\rightarrow X, <\text{obj}, u> \rightarrow Y, <\text{tns}, u> \rightarrow Z \\
\text{revising} \left[ \begin{array}{c}
\text{reviser} \rightarrow X \\
\text{revised} \rightarrow Y \\
\text{loc} \rightarrow Z
\end{array} \right] &\rightarrow \text{utt-time}(u, L)
\end{align*}
\]

280. If \( u \) is a use of type \([\text{revise}]_{V}^{(-\text{tns})^{\text{bse}}^{\text{form: prp}}}\) then \([u] =\)

\[
\begin{align*}
(\text{utt}, u) &\rightarrow L \\
<\text{subj}, u> &\rightarrow X, <\text{obj}, u> \rightarrow Y, <\text{tns}, u> \rightarrow Z \\
\text{revising} \left[ \begin{array}{c}
\text{reviser} \rightarrow X \\
\text{revised} \rightarrow Y \\
\text{loc} \rightarrow Z
\end{array} \right] &\rightarrow \text{utt-time}(u, L)
\end{align*}
\]

281. If \( u \) is a use of type \([\text{is}]_{V}^{(+\text{tns})^{\text{XCOMP: form: prp}}}^{(+\text{aux})}\) then \([u] =\)

\[
\begin{align*}
(\text{utt}, u) &\rightarrow L, <\text{ref-time}, u> \rightarrow T \\
\text{P} &\rightarrow L \circ T
\end{align*}
\]

\(\circ\) denotes temporal overlap
282. If \( u \) is a use of type [is] then \([u] = \varepsilon\):

\[
\begin{align*}
\text{+tns} & \quad \text{XCOMP: [form: bse]} \\
\text{+aux} & \quad \text{<utt-time, } u \text{ > } \rightarrow \text{L, <ref-time, } u \text{ > } \rightarrow \text{T}
\end{align*}
\]

\[
P \quad L < T
\]

283. If \( u \) is a use of type [not] then \([u] = -P(\begin{array}{c}
\text{subj } \rightarrow \text{X}, \\
\text{tns } \rightarrow \text{Y}
\end{array})\):

\[
\begin{array}{c}
\text{subj } \rightarrow \text{X, tns } \rightarrow \text{Y}
\end{array}
\]

284. If \( u \) is a use of type [every]_{\text{QuantDet}} then \([u] = \text{every}\):

\[
<\text{rr, } u > \rightarrow \text{Type}
\]

285. If \( u \) is a use of type [most]_{\text{QuantDet}} then \([u] = \text{most}\):

\[
<\text{rr, } u > \rightarrow \text{Type}
\]

286. If \( u \) is a use of type [the]_{\text{RefDet}} with [def] then \([u] = \text{Type } = \text{unique}\):

\[
P \quad \text{Type } = \text{unique}
\]

287. If \( u \) is a use of type [a]_{\text{RefDet}} with [-def] then \([u] = \text{Type } = \text{true}\):

\[
P \quad \text{Type } = \text{true}
\]

288. If \( u \) is a use of type [before]_{\text{Adv}} then \([u] = \text{before}\):

\[
\text{instance } \rightarrow \text{X, object } \rightarrow \text{Y}
\]

3.2.3.2 Phrase Structure

255. If \( u \) is a use of type \([S_1, \ldots, S_n]_{\text{DISCOURSE}}\) with constituents \( u_1, \ldots, u_n \), respectively, then \([u] = \lambda \phi' \cup [\text{desc-sit, } u > \rightarrow S] (\begin{array}{c}
\text{S} \\
\text{Closure}(u, [u_1], f \land \ldots \land [u_n], f)
\end{array})\)

where \( f \) is a mia for \([u_1], \ldots, [u_n] \) and \( u \models (\text{asserted}, u, [u]) \) and \( f' \) is the restriction of \( f \) to the parameters free in \( \text{Closure}(u, [u_1], f \land \ldots \land [u_n], f) \).
256. If \(u\) is a use of type \([\text{NP VP}]\) with constituents \(u_1\) and \(u_2\), respectively, then \([u] = \lambda f^{'} \cup \left< \text{rt}, u \rightarrow T \right> \left( \begin{array}{c} \text{Closure} \left( f^{'} \left( \begin{array}{c} \left< \text{subj}, u \rightarrow \left[ u_1 \right] \right>, f \right), f \right) \\ T = f'(\langle r t, u_2 \rangle) \end{array} \right) \right)\)

where \(f\) is a mia for \(\{[u_1], [u_2]\}\), and \(f^{'}\) is the restriction of \(f\) to the parameters free in \(\text{Closure}(u, [u_2], f \left( \begin{array}{c} \left< \text{subj}, u \rightarrow \left[ u_1 \right] \right>, f \right), f)\).

257. If \(u\) is a use of type \([\text{V NP}]\) with constituents \(u_1\) and \(u_2\), respectively, then \([u] = \lambda f^{'} \cup \left< \text{rt}, u \rightarrow T \right> \left( \begin{array}{c} \left< \text{subj}, u \rightarrow X \right>, \left< \text{tens}, u \rightarrow Y \right> \\ \text{Closure} \left( f^{'} \left( \begin{array}{c} \left< \text{subj}, u \rightarrow X \right> \\ \left< \text{obj}, u \rightarrow \left[ u_2 \right] \right> \\ \left< \text{tens}, u \rightarrow Y \right> \\ \left< \text{subj}, u \rightarrow \left[ u_1 \right] \right> \right), f \right), f \right) \\ T = f'(\langle r t, u_2 \rangle) \end{array} \right)\)

Where \(f\) is a mia for \(\{[u_1], [u_2]\}\) such that if \(f\) is defined on \((\text{subj}, u)\), then \(f(\langle \text{subj}, u \rangle) = X\) and \(f^{'}\) is the restriction of \(f\) to parameters free in \(\text{Closure}(u, [u_2], f \left( \begin{array}{c} \left< \text{subj}, u \rightarrow X \right> \\ \left< \text{obj}, u \rightarrow \left[ u_2 \right] \right> \\ \left< \text{tens}, u \rightarrow Y \right> \\ \left< \text{subj}, u \rightarrow \left[ u_1 \right] \right> \right), f)\).

258. If \(u\) is a use of type \([\text{NP 's}]\) with constituent \(u_1\) then \([u] = \lambda f \cup \left< \text{res}, u \rightarrow R \right> \left( \begin{array}{c} \left[ u_1 \right], f \\ \text{R} = f'(\langle \text{res}, u_1 \rangle) \end{array} \right)\)

where \(f\) is a mia for \(\{[u_1]\}\).

259. If \(u\) is a use of type \([\text{REFDET NOM}]\) with constituents \(u_1\) and \(u_2\), respectively, then \([u] = \lambda f \cup \left< \text{do, u} \rightarrow X \right>, \left< \text{res}, u \rightarrow R \right> \left( \begin{array}{c} X \\ \left[ u_2 \right], f \left( \text{instance}: X \right) \\ \text{R, } [u_2], f \left( \text{res}, u_1 \rangle \right) \end{array} \right)\)

where \(f\) is a mia for \(\{[u_1], [u_2]\}\).

260. If \(u\) is a use of type \([\text{NP Nom}]\) with constituents \(u_1\) and \(u_2\), respectively, then \([u] = \left[ \begin{array}{c} \text{CASE: Poss} \\ + \text{def} \\ \text{R, } [u_2], f \left( \text{res}, u_1 \rangle \right) \end{array} \right)\)
\[ \lambda f \cup \begin{cases} <do, u> \rightarrow X \\ <res, u> \rightarrow R \\ <posrel, u> \rightarrow Rel \end{cases} \]

where \( f \) is a mia for \( \{[u_1], [u_2]\} \).

261. If \( u \) is a use of type \([\text{QUANTDET NOM}]_N^P \) with constituents \( u_1 \) and \( u_2 \), respectively, then \([u] = \)

\[ \lambda f \cup \begin{cases} <do, u> \rightarrow Par \\ <ass-type, u> \rightarrow T \\ <qf, u> \rightarrow Q \\ <res, u> \rightarrow R \end{cases} \]

where \( f \) is a mia for \( \{[u_2]\} \).\textsuperscript{13}

262. If \( u \) is a use of type \([\text{NOM}]_V^P \) with constituents \( u_1 \) and \( u_2 \), respectively, then

\[ [u] = \]

\[ \lambda f \]

\[ \text{Closure} \left( \begin{array}{c} \text{instance} \rightarrow X \\ \text{instance} \rightarrow X \end{array} \right) \]

\[ \left( \begin{array}{c} \text{instance} \rightarrow X \\ \text{instance} \rightarrow X \end{array} \right) \]

\[ \left( \begin{array}{c} \text{instance} \rightarrow X \\ \text{instance} \rightarrow X \end{array} \right) \]

263. If \( u \) is a use of type \([\text{CN}]_N^P \) with constituents \( u_1 \) and \( u_2 \), respectively, then \([u] = \)

\[ \lambda f \]

\[ \left( \begin{array}{c} \text{instance} \rightarrow X \\ \text{instance} \rightarrow X \end{array} \right) \]

\[ \left( \begin{array}{c} \text{instance} \rightarrow X \\ \text{instance} \rightarrow X \end{array} \right) \]

\[ \left( \begin{array}{c} \text{instance} \rightarrow X \\ \text{instance} \rightarrow X \end{array} \right) \]

\[ \left( \begin{array}{c} \text{instance} \rightarrow X \\ \text{instance} \rightarrow X \end{array} \right) \]

where \( f \) is a mia for \( \{[u_1]\} \).

\textsuperscript{13}Modified to include parameter (\textit{Par}) as described object.
264. If \( u \) is a use of type \([ VP \quad VP \quad VP \quad [XCOMP: u_2]]\) with constituents \( u_1 \) and \( u_2 \), respectively, then \([u] = \)

\[
\lambda f' \cup \lbrack \langle rt, u \rangle \rbrack \to \big[ \langle subj, u \rangle \to X, \langle tns, u \rangle \to Y \big]
\]

\[
\text{Closure} \left( u, \left[ \left[ \langle subj, u_2 \rangle \to X \right], f' \right] \right), f \bigg| T = f(\langle rt, u_1 \rangle)
\]

where \( f \) is a mia for \([u_1], [u_2] \) and \( f' \) is the restriction of \( f \) to parameters free in \( \text{Closure} (u, [u_1], f([u_2]), f) \).

265. If \( u \) is a use of type \([ NOT \quad VP \quad VP \quad [XCOMP: u_2]]\) with constituents \( u_1 \) and \( u_2 \), respectively, then \([u] = \)

\[
\lambda f' \cup \lbrack \langle rt, u \rangle \rbrack \to \big[ \langle subj, u \rangle \to X, \langle tns, u \rangle \to Y \big]
\]

\[
\text{Closure} \left( u, \left[ \left[ \langle subj, u_2 \rangle \to X \right], f' \right] \right), f \bigg| T = f(\langle rt, u_1 \rangle)
\]

where \( f \) is a mia for \([u_2] \) and \( f' \) is the restriction of \( f \) to parameters free in \( \text{Closure} (u, [u_1], f([u_2]), f) \).

266. If \( u \) is a use of type \([ V \quad VP \quad [XCOMP: u_2]]\) with constituent \( u_1 \) then \([u] = \)

\[
\lambda f \cup \lbrack \langle rt, u \rangle \rbrack \to \big( \langle subj, u \rangle \to X, \langle tns, u \rangle \to Y \big)
\]

\[
P \bigg| T = f(\langle rt, u_1 \rangle)
\]

where \( f \) is a mia for \([u_1] \).

267. If \( u \) is a use of type \([ VP \quad AdvP \quad VP \quad [XCOMP: u_2]]\) with constituents \( u_1 \) and \( u_2 \), respectively, then \([u] = \)

\[
\lambda f' \cup \lbrack \langle rt, u \rangle \rbrack \to \big[ \langle subj, u \rangle \to X, \langle tns, u \rangle \to Y \big]
\]

\[
\text{Closure} \left( u, \left[ \left[ \langle subj, u_1 \rangle \to X \right], f' \right] \right), f \bigg| T = f(\langle rt, u_1 \rangle)
\]

where \( f \) is a mia for \([u_1], [u_2] \) and \( f' \) is the restriction of \( f \) to the parameters free in \( \text{Closure} (u, [u_1], f([u_2]), f) \).

268. If \( u \) is a use of type \([ Adv \quad S \quad AdvP \quad [XCOMP: u_2]]\) with constituents \( u_1 \) and \( u_2 \), respectively, then \([u] = \)

\[
\lambda f \bigg( \langle subj, u \rangle \to X \bigg)
\]

\[
\left[ u_1 \right] \left[ \langle subj, u \rangle \to X \right], f \bigg| S \bigg| T = f(\langle rt, u_2 \rangle)
\]

\[
\exists \bigg[ s \bigg] \left[ \langle subj, u \rangle \to X \right], f \bigg| T = f(\langle rt, u_2 \rangle)
\]

\[
\bigg[ u_2 \bigg] \bigg| T = f(\langle rt, u_2 \rangle)
\]

\[
165
\]
where \( f \) is a mia for \( \{[u_2]\} \).

### 3.2.3.3 Definition of Closure

1. If \( u \models \langle \text{scope-over, } u_{\text{NP}_1}, u' \rangle \), \( u_{\text{NP}} \) is a quantified NP and \( u' \) is either a quantified NP or \( u' = u \) and and \( f \) is an assignment then \( \text{closure}(u, \sigma, f; u_{\text{NP}_1}) = \)

\[
\begin{pmatrix}
    f(<\text{qf, } u_{\text{NP}_1}>) & f(<\text{asc-typ, } u_{\text{NP}_1}>) \\
    \text{closure}(u, \sigma, f; u')
\end{pmatrix}
\]

2. If \( u \models \langle \text{scope-over, } u_{\text{NP}_1}, u_{\text{NP}_2} \rangle \land \ldots \land \langle \text{scope-over, } u_{\text{NP}_{n-1}}, u' \rangle \), where \( u_{\text{NP}_1}, \ldots, u_{\text{NP}_{n-1}} \) are all non-quantified NPs (definites or indefinites), \( u_{\text{NP}_1} \) is a quantified NP. \( u' \) is either a quantified NP or \( u' = u \), and \( f \) is an assignment then \( \text{closure}(u, \sigma, f; u_{\text{NP}_1}) = f(<\text{qf, } u_{\text{NP}_1}>) \cdot f(<\text{asc-typ, } u_{\text{NP}_1}>) \) where:

\[
\exists \begin{pmatrix}
    \langle\text{do, } u_{\text{NP}_1} \rangle & \rightarrow f(\langle\text{do, } u_{\text{NP}_2} \rangle), \ldots, \langle\text{do, } u_{\text{NP}_{n-1}} \rangle & \rightarrow f(\langle\text{do, } u_{\text{NP}_{n-1}} \rangle) \\
    \text{closure}(u, \sigma, f; u_{\text{NP}_1})
\end{pmatrix}
\]

3. If \( u \models \langle \text{scope-over, } u_{\text{NP}_1}, u' \rangle \), \( u_{\text{NP}} \) is a non-quantified NP and \( u' \) is either an NP or \( u' = u \) and and \( f \) is an assignment then \( \text{closure}(u, \sigma, f; u_{\text{NP}_1}) = \text{closure}(u, \sigma, f; u') \)

4. \( \text{closure}(u, \sigma, f; u) = \sigma \)
Chapter 4

Property Theory

In the following the first section is concerned with the motivation for a theory of propositions, properties and truth (property theory). It presents one such formal theory (PT), together with definitions of some first-order types, of use in natural language semantics. The second section is about the syntax-semantics interface. It notes that a PTQ-style treatment of natural language can be re-implemented within this formal theory. It also shows how natural language might be interpreted via some intermediate, underspecified form, within PT. Deliverable 9 contains pointers to treatments of various phenomena in the semantics of natural language.

4.1 Semantic Tools

The motivation for a theory of propositions, properties and truth (commonly known as property theory) involves issues in intensionality and types. A basic premise of property theory is that if we wish to obtain an highly intensional theory then propositions and properties should be taken seriously as primitive notions which are independent of any set-theoretic interpretation. Equality between propositions should be as weak as syntactic equality between their representations. Traditionally, this syntactic view is regarded as problematic since it can lead to an inconsistency via the logical paradoxes. The need to avoid such inconsistency is often taken as motivation for using a strongly typed theory. However, by adopting a weak representationalist view, where not all terms of the appropriate form represent propositions, the paradoxes can be avoided. This allows the formalisation of a weakly typed, highly intensional theory in which unproblematic instances of self-predication can be represented, together with universal properties, which are prohibited in strongly typed theories, such as self-identity. Types can be re-introduced into the theory as first-order predicates.

Property theory is not an empirical theory about natural language semantics, except in so far as it provides a treatment of intuitions about propositions and truth. In this sense, as a theory for natural language semantics it should be seen on a par with syntactic formalisms such as Generalised Phrase Structure Grammars, and Categorial Grammar. Clearly such theories
were devised and revised to cope with issues in syntax, such as movement, just as property theory was constructed to cope with issues in semantics such as paradoxes and self-application. Formalisms for natural language syntax do not, by themselves, constitute a grammar. They provide a framework in which to explore treatments of empirical issues. In the same way, theories of propositions, properties and truth provide a vocabulary for implementing semantic theories and a formal space in which to explore issues in natural language semantics.

The version of property theory presented here adopts a first-order axiomatic approach to proposition- hood and truth. The axioms are incomplete with respect to the proposition- hood, and hence the truth of, paradoxical terms. In addition to providing a treatment of propositions, properties and truth, this axiomatic approach can be taken as a methodology for natural language semantics. Some basic intuitions will be captured by the way phrases in natural language are represented in the theory. To cover additional phenomena, we can, if necessary, add definitions and axioms which are strong enough to capture salient intuitions. In conjunction with a rule such as modus ponens, inferences in natural language are then mirrored by corresponding inferences in the representation. So, the theory can be strengthened to cover various semantic treatments in a uniform framework. The axioms may remain incomplete with respect to problematic or controversial issues. In this respect, the weakness of the theory is a strength. In addition, the weak axioms do not impose requirements, such as strong types, which can create additional complications for the treatment of some semantic phenomena.

Let us consider in more detail the relevant issues in intensionality and types which have been used to motivate property theory. A theory is said to be intensional if two objects can be distinguished, even though they are equated under some other mode of analysis. Given two levels of interpretation, one level is more intensional (less extensional) than the other if it embodies a weaker sense of equality. In terms of a theory of sets, we might contrive two notions of equality: one, when we consider sets to be defined by the members they have; another, weaker equality, when we take sets to be defined by the procedure used to determine their members. The latter sense of equality gives rise to an intensional view of sets; sets which are equated because they have the same members may be distinguished if the procedures used to determine their members are different.

Intensionality is required in the semantics of propositional attitudes. The simplest view of propositions is perhaps to claim that they denote truth-values. However, an individual believes in propositions which are distinct from truth-values. Further, if the equality relation between propositions is given by the equality between the denoted truth values, then there would only be two propositions. The traditional approach is to say instead that a proposition is something which evaluates to a truth-value given a “state of affairs”, or “world”. In other “possible worlds”, or states of affairs, it may evaluate to a different truth-value. Thus, a proposition is equated with the set of worlds in which it holds.

We may question whether such an approach—where intensionality is defined in terms of an extensional set theory—is sufficiently fine-grained; if two propositions necessarily hold in the same states of affairs, then they cannot be distinguished. If an agent believes a proposition,

\footnote{For example, by defining dependent types, property-theoretic representations can be given sufficient internal structure—in the form of witnesses to propositions—to cope with Geach’s “donkey” sentences.}
she is committed to believing all its logical consequences. We may not wish to characterise belief in this way. For example, all propositions in mathematics become equated. It seems too strong to say that if an agent knows some mathematical truth, then that agent knows all mathematical truths. Clearly we need to be able to distinguish mathematical propositions which are equated on a possible worlds analysis. There are also problems for the representation of properties on this account: a property will be represented by a function from individuals to propositions. Some properties, such as those of ‘being bought’ and ‘being sold’ become equated, yet we might like them to be different.

Greater intensionality would be obtained if propositions were taken to be equal only if they are represented by the same syntactic object. This a representationalist view of propositions. However, this approach is potentially problematic. If the assertion of a proposition is equivalent to asserting its truth, then it is possible to construct paradoxical statements. To illustrate this, if we maintain the following equivalences (effectively the Tarski Biconditional):

\[
\begin{align*}
\text{s} &\iff \text{It is the case that } s, \\
px &\iff \text{It is the case that } p \text{ holds of } x.
\end{align*}
\]

and define a property \( R \) as:

\[
Rx =_{\text{def}} \text{It is the case that } x \text{ does not hold of } x.
\]

then from the definition, we can deduce that:

\[
RR \iff \text{It is the case that } R \text{ does not hold of } R.
\]

but from the Tarski Biconditional above:

\[
RR \iff \text{It is the case that } R \text{ holds of } R.
\]

Thus, we obtain the contradiction:

\[
\text{It is the case that } R \text{ holds of } R \iff \text{It is the case that } R \text{ does not hold of } R.
\]

A solution to this problem would be to prevent expressions of the form \( RR \). This is one purpose of strongly typed theories. An object’s type specifies the type of those objects it may be applied to, and the type of the resultant object. If \( f \in \langle a, b \rangle \) (that is \( f \) is of type \( \langle a, b \rangle \)) and \( g \in a \), then \( fg \in b \). In the semantics of natural language, types may correspond with syntactic categories.

To avoid paradoxical assertions requires a stronger notion of types, where objects must belong to one, and only one type, and syntactically, objects can only be applied to arguments of the
appropriate type. There must be no notion of equality (or containment) across the types. Such *monomorphic* types refer to higher order functions. Strong typing prevents the formation of expressions such as:

\[ RR \]

because it cannot be typed: if we assume \( R \in \langle S, \text{prop} \rangle \) (that is, it produces a proposition when applied to something of type \( S \)) then for \( RR \) to be a proposition, it must be the case that \( R \in \langle \langle S, \text{prop}, \text{prop} \rangle \), but this means \( R \) belongs to more than one type. This is the approach taken in a possible worlds style analysis, which we have rejected because it does not have sufficiently fine-grained intensionality.

However, if we are to adopt a theory in which propositions are equated only if they are represented by the same syntactic object, then the use of strong types to avoid the paradoxes may lead to other problems. To express beliefs of the form:

\[
\text{Mary believes}_m \ P
\]

the representation “believes\( _m \)” must be of the type \( \langle S, \langle \epsilon, \text{prop} \rangle \rangle \), where \( P \in S \), and \( \text{Mary'} \in \epsilon \). We can say that “believes\( _m \)” is in the meta-language of object language sentences of type \( S \). To express:

\[
\text{John believes}_j \text{ that everything Mary believes}_m \text{ is true.}
\]

requires that “believes\( _j \)” for John is in the meta-language for statements about Mary’s beliefs. If, however, we also have the expression:

\[
\text{Mary believes}_m \text{ that everything John believes}_j \text{ is true.}
\]

then we cannot express the appropriate content: this last sentence requires that “believes\( _m \)” is in the meta-language of sentences containing “believes\( _j \)”, which contradicts the requirements of the preceding example. In other words, we can conclude from these last two sentences that “believes\( _m \)” and “believes\( _j \)” are higher than each other in the object/metalanguage hierarchy, which is inconsistent.

The problem is created by the use of strong typing which was introduced to avoid the expression of the logical paradoxes. However, there is an alternative way of avoiding the paradoxes, which is to give up Tarski’s Biconditional:

\[
s \rightarrow \text{It is the case that } s.
\]

\(^2\)Chierchia notes that even in some strongly typed theories, such as Dynamic Montague Grammar, the paradoxes may accidentally be reintroduced in the semantics of dynamic binding [Chierchia, 1991b].

\(^3\)This argument is taken from [Turner, 1992].
We can achieve this by saying that only some of the objects in the language represent propositions. It is only for such objects that the biconditional holds.

Among its advantages, such a weakly typed theory allows the expression of universal properties (such as being self identical), and the simple expression of gerunds and infinitives. As an example of the latter, in the following sentences:

\[
\begin{align*}
\text{John likes to play tennis.} \\
\text{John likes playing tennis.}
\end{align*}
\]

we might wish to take “to play tennis”, and “playing tennis” to denote properties, or at least terms which are systematically related to the property denoted by the finite verb phrase in:

\[
\text{John is playing tennis.}
\]

If we have a weakly typed framework in which properties are taken to be just another kind of individual, then in:

\[
\begin{align*}
\text{John likes Mary.} \\
\text{John likes playing tennis.}
\end{align*}
\]

the verb “likes” can be represented by an object of the same type.

Chierchia notes that if we take “is fun” and “being fun” to denote essentially the same property, then:

\[
\text{Being fun is fun.}
\]

predicates a property of itself (or, from a Fregean perspective, it predicates a property of the \textit{individual correlate} of itself). With this example, we have a legitimate instance of self-predication, which would be ruled out in a strongly typed theory.

The theory of propositions, properties and truth to be presented here is an axiomatic theory. The axioms concerning proposition-hood are deliberately too weak to prove the proposition-hood of paradoxical terms. As mentioned before, in natural language semantics, an axiomatic approach can provide a methodology, in which the semantic theory can be strengthened till it is just strong enough to capture the desired intuitions, but no stronger. The axioms can be left incomplete with respect to controversial intuitions. Further, the axiomatic theory also lends itself to the characterisation of felicitous discourse, where only meaningful discourse is sufficient to satisfy the requirements of proposition-hood. For example, a semantic theory could be constructed where the axioms of truth need never apply to sentences with false presuppositions.
4.1.1 The Basic Theory\(^4\)

The particular version of property theory to be introduced here is PT, Ray Turner’s axiomatisation of Aczel’s Frege Structures [Turner, 1990; Turner, 1992; Aczel, 1980]. Other formalisations of property theory will not be discussed, except to mention that equating predication with \(\lambda\)-application may prove problematic for some examples [Turner, 1989; Bealer, 1989]. A short example will be used to illustrate how the paradoxes are avoided.

4.1.1.1 General Framework\(^5\)

Conceptually, PT can be split into two components, or levels. The first is a language of terms, which consists of the untyped \(\lambda\)-calculus, embellished with logical constants. A restricted class of these terms will correspond to propositions. When combined appropriately using the logical constants, other propositions result. As an example, given the propositions \(t, s\), the ‘conjunction’ of these, \(t \land s\), is also a proposition, where \(\land\) is a logical constant.

Some of the propositions will, further, be true propositions. When combining propositions with the logical constants, the truth of the resultant proposition will depend upon the truth of the constituent propositions. Considering the previous example, if \(t, s\) are both propositions, then \(t \land s\) will be a true proposition if and only if \(t\) and \(s\) are true propositions.

There may be terms that form propositions when applied to another term. These terms are the properties. The act of predication is modelled by \(\lambda\)-application.

The essential point to note is that this is a highly intensional theory as the notion of equality is that of the \(\lambda\)-calculus: propositions are not to be equated just because they are always true together; similarly, properties are not to be equated just because they hold of the same terms (i.e. form true propositions with the same terms).

There are problems with the theory so far: the logical constants have no proof theory; and the notions of being a proposition, or a true proposition, cannot be expressed within this language of terms. That is, although we can consider terms as propositions, or true propositions, and comprehend how the proposition-hood and truth of a term depends upon the proposition-hood and truth of its constituent terms, we cannot express these notions formally within the language of terms; some meta-language is required. This is the purpose of the second component of PT: the language of well formed formulae (wff). This is a first-order language where the terms which can be quantified over are those of the \(\lambda\)-calculus extended with logical constants, as discussed above. The language of wff has two predicates, \(P\) for ‘is a proposition’, and \(T\) for ‘is a true proposition’. Clearly, this gives the formal means for axiomatising the behaviour of propositions and true propositions. For example, the informal discussion concerning the behaviour of the logical constant \(\land\) can be formalised as follows:

\(^4\)This is taken from [Fox, 1994] which in turn is from [Fox, 1993] and [Turner, 1992].
\(^5\)Verbatim from [Fox, 1994]
“given the propositions \( t, s \), the conjunction of these \( t \land s \) is also a proposition”:

\[
P(t) \land P(s) \rightarrow P(t \land s)
\]

“if \( t, s \) are both propositions, then \( t \land s \) will be a true proposition if and only if \( t \) and \( s \) are true propositions”:

\[
P(t) \land P(s) \rightarrow (T(t \land s) \equiv (T(t) \land T(s)))
\]

Axioms concerning \( T \) must be restricted so that only terms that are propositions are considered.

The distinction between a wff which expresses the truth conditions of a propositional term, and the term itself, can be taken to be akin to that between extension and intension in Montague semantics ([Dowty et al., 1981]). In that theory, however, intensions are derived from extensions. As a consequence, the equality of intensions is that of the extensions, so the intensions of propositions will be equated if they are always true together, and properties (the intensions of predicates) will be equated if they always hold of the same objects. This is in contrast to PT, where the intensions are basic. Propositions in the language of terms may have the same truth conditions when \( T \) is applied, but this does not force them to be the same proposition, so we might have:

\[
T(s) \equiv T(t)
\]

but that does not mean that the terms are equal:

\[
s = t
\]

Similarly, in the language of wff, properties may hold of the same terms, yet they may be distinct. The \( \lambda \)-equality of terms is thus weaker than the notion of logical equivalence obtained when considering truth conditions in the meta-language.

It can be seen that PT characterises a Frege Structure [Aczel, 1980]: two classes of \( \lambda \)-terms are defined by \( P \) and \( T \) as below:

Diagram: A Frege Structure
4.1.1.2 The Formal Theory

The following presents a formalisation of the languages of terms and wff, together with the axioms that provide the closure conditions for P and T.

The Language of terms

Basic Vocabulary:

<table>
<thead>
<tr>
<th>Individual variables:</th>
<th>x, y, z, ...</th>
</tr>
</thead>
<tbody>
<tr>
<td>Individual constants:</td>
<td>c, d, e, ...</td>
</tr>
<tr>
<td>Logical constants:</td>
<td>( \lor, \land, \neg, \Rightarrow, \Xi, \Theta )</td>
</tr>
</tbody>
</table>

Inductive Definition of Terms:

(i) Every variable or constant is a term.
(ii) If \( t \) is a term and \( x \) is a variable then \( \lambda x.t \) is a term.
(iii) If \( t \) and \( t' \) are terms then \( t(t') \) is a term.

The Language of Wff

Inductive Definition of Wff:

(i) If \( t \) and \( s \) are terms then \( s = t, P(t), T(t) \) are atomic wff.
(ii) If \( \varphi \) and \( \varphi' \) are wff then \( \varphi \land \varphi', \varphi \lor \varphi', \varphi \rightarrow \varphi', \sim \varphi \) are wff.
(iii) If \( \varphi \) is a wff and \( x \) a variable then \( \exists x \varphi \) and \( \forall x \varphi \) are wff.

The theory is governed by the following axioms:

**Axioms of The \( \lambda \beta \)-Calculus**

\[
\lambda x.t = \lambda y.t[y/x] \text{ if } y \text{ not free in } t \\
(\lambda x.t)t' = t[t'/x]
\]

This defines the equivalence of terms.
The closure conditions for proposition-hood are given by the following axioms:

**Axioms of Propositions**

1. \( P(t) \land P(s) \rightarrow P(t \land s) \)
2. \( P(t) \land P(s) \rightarrow P(t \lor s) \)
3. \( P(t) \land (T(t) \rightarrow P(s)) \rightarrow P(t \implies s) \)
4. \( P(t) \rightarrow P(\neg t) \)
5. \( \forall x P(t) \rightarrow P(\Theta x.t) \)
6. \( \forall x P(t) \rightarrow P(\Xi x.t) \)
7. \( P(s \approx t) \)

Truth conditions can be given for those terms that are propositions:

**Axioms of Truth**

1. \( P(t) \land P(s) \rightarrow (T(t \land s) \rightarrow T(t) \& T(s)) \)
2. \( P(t) \land P(s) \rightarrow (T(t \lor s) \rightarrow T(t) \lor T(s)) \)
3. \( P(t) \& (T(t) \rightarrow P(s)) \rightarrow (T(t \implies s) \rightarrow T(t) \rightarrow T(s)) \)
4. \( P(t) \rightarrow (T(\neg t) \rightarrow \sim T(t)) \)
5. \( \forall x P(t) \rightarrow (T(\Theta x.t) \rightarrow \forall x T(t)) \)
6. \( \forall x P(t) \rightarrow (T(\Xi x.t) \rightarrow \exists x T(t)) \)
7. \( T(t \approx s) \rightarrow t = s \)
8. \( T(t) \rightarrow P(t) \)

The last axiom states that only propositions may have truth conditions.

Note that the quantified propositions \( \Theta x.t \), \( \Xi x.t \) can be written as \( \Theta x(t) \), \( \Xi x(t) \), where the \( \lambda \)-abstraction is implicit.
This basic theory is very weak. The general approach for analysing semantic phenomena is to amend the theory either definitionally, as is done when adding dependent type constructors, or by strengthening it with more axioms and primitive notions, such as for events and plurals. This is, of course, in addition to obtaining appropriate representations for natural language phrases.

4.1.2 Definitions of Types

Here, it is shown how various types can be defined in the theory, which can include the concepts of quantifier, determiner, and function space. These can be used when giving a Montagovian-style analysis with PT. In addition, definitions for Martin-Löf’s dependent type in PT are presented [Martin-Löf, 1982; Martin-Löf, 1984; Turner, 1990]. These operators can be used in a constructive analysis of generalised quantifiers [Sundholm, 1989], which are useful in the treatment of “donkey” sentences [Ranta, 1991; Davila-Perez, 1994; Turner, 1994] as illustrated in section 4.4.1 in Deliverable 9.

The notions of \( n \)-place relations can be defined recursively:

\[
\begin{align*}
\text{(i)} & \quad \text{Rel}_0(t) \leftrightarrow P(t) \\
\text{(ii)} & \quad \text{Rel}_n(\lambda x.t) \leftrightarrow \text{Rel}_{n-1}(t)
\end{align*}
\]

We can write \( \text{Rel}_1(t) \) as \( Pty(t) \) and and \( \lambda x.t \) as \( \{ x : t \} \). In keeping with this set-like notation, we can write \( T(tx) \) as \( xet \), especially if \( t \) is a property.

Following Turner [Turner, 1992] we can define the empty and universal property:

\[
\begin{align*}
\nabla & \overset{\text{def}}{=} \{ x : (x \approx x) \} \\
\Omega & \overset{\text{def}}{=} \{ x : \neg(x \approx x) \}
\end{align*}
\]

We can also give definitions for intersection \( \cap \), union \( \cup \), difference \( \setminus \), cartesian product \( \otimes \), disjoint union \( \sqcup \), and function space \( \mapsto \) operators:

\[
\begin{align*}
\cap & \overset{\text{def}}{=} \lambda f. \lambda g. \{ x : fx \land gx \} \\
\cup & \overset{\text{def}}{=} \lambda f. \lambda g. \{ x : fx \lor gx \} \\
\setminus & \overset{\text{def}}{=} \lambda f. \lambda g. \{ x : fx \land \neg gx \} \\
\otimes & \overset{\text{def}}{=} \lambda f. \lambda g. \{ z : \exists x \exists y (z \approx (x, y) \land fx \land gy) \} \\
\sqcup & \overset{\text{def}}{=} \lambda f. \lambda g. \{ z : (\text{fst}(z) \approx 0 \land f(\text{snd}(z))) \lor (\text{fst}(z) \approx 1 \land g(\text{snd}(z))) \} \\
\mapsto & \overset{\text{def}}{=} \lambda f. \lambda g. \{ z : \Theta x(f x \Rightarrow g(z)) \}
\end{align*}
\]
which trivially lead to the following theorems:

\[
\begin{align*}
\forall z. (t \land s) & \iff \forall z. (t \land \forall z s) \\
\forall z. (t \lor s) & \iff \forall z. (t \lor \forall z s) \\
\forall z. (t \rightarrow s) & \iff \forall z. (t \rightarrow \neg z \forall z s) \\
\forall z. (t \otimes s) & \iff \forall z. (\text{fst}(z) \land \neg \text{snd}(z) \land z) \\
\forall z. (t \oplus s) & \iff (\text{fst}(z) = 0 \land \text{snd}(z) \land z) \lor (\text{fst}(z) = 1 \land \text{snd}(z) \land z) \\
\forall z. (t \twoheadrightarrow s) & \iff \forall z. (\text{fst}(z) \land \neg \text{snd}(z)) \\
\end{align*}
\]

where $\langle \cdot, \cdot \rangle$, $\text{fst}$, $\text{snd}$ have their usual definitions:

\[
\begin{align*}
\text{fst} & = \text{def} \lambda p. \lambda x. y. x \\
\text{snd} & = \text{def} \lambda p. \lambda x. y. y \\
\langle x, y \rangle & = \text{def} \lambda z. z(x)(y)
\end{align*}
\]

so that:

\[
\begin{align*}
\text{fst}(\langle x, y \rangle) & =_{\beta} x \\
\text{snd}(\langle x, y \rangle) & =_{\beta} y
\end{align*}
\]

### 4.1.2.1 Natural Language Types

It is straightforward to define first-order types which correspond to the notions of determiner, quantifier and functional types in natural language semantics:

\[
\begin{align*}
(i) & \quad \text{Quant}(f) =_{\text{def}} \forall x. (Pty(x) \rightarrow P(f x)) \\
(ii) & \quad \text{Det}(f) =_{\text{def}} \forall x. (Pty(x) \rightarrow \text{Quant}(f)) \\
(iii) & \quad (R \rightarrow S)(f) =_{\text{def}} \forall x. (Rx \rightarrow S(f x))
\end{align*}
\]

The semantic types of the representations of lexical items has to be declared in order to be able to prove the proposition-hood of well formed sentences.

### 4.1.2.2 Dependent Types

Later it will prove useful to have a notion of dependent types [Martin-Löf, 1982; Martin-Löf, 1984; Turner, 1990], for a treatment of Geach’s “donkey” sentences, and other anaphoric puzzles:

\[
\begin{align*}
\Pi & =_{\text{def}} \lambda f. \lambda g. \{ h : \Theta x. (f x \Rightarrow g x(h x)) \} \\
\Sigma & =_{\text{def}} \lambda f. \lambda g. \{ h : f(\text{fst}(h)) \land g(\text{snd}(h)) \}
\end{align*}
\]
These definitions support the following theorems:

If \( \text{Pty}(f) \) and \( \forall x(x \notin f \rightarrow \text{Pty}(gx)) \) then:

\[
\begin{align*}
\text{Pty}(\Pi fg) \\
\text{Pty}(\Sigma fg)
\end{align*}
\]

and:

\[
\begin{align*}
h \in \Pi fg & \rightarrow \forall x(x \notin f \rightarrow hx \in gx) \\
h \in \Sigma fg & \rightarrow \text{fst}(h) \in f \& \text{snd}(h) \in g(\text{fst}(h))
\end{align*}
\]

To paraphrase these definitions, \( h \in \Pi fg \) means that \( h \) is a function which takes an element (or ‘proof’/‘witness’) of \( f \) and gives a ‘proof’ of \( g \) applied to that element of \( f \). The expression \( h \in \Sigma fg \) means that \( h \) is a pair, where the first component of the pair is an element of \( f \), and the second is an element of \( g \) applied to that element of \( f \).

The use of these dependent type operators will become clearer later, but as can be seen, with both of these types the evaluation of \( g \) is dependent upon the chosen element, or ‘proof’, of \( f \). In some sense then, the meaning of \( g \) depends upon the context created by \( f \). Notice also that we can use \( \Sigma \) to conjoin sequences of propositions, so the interpretation of a sentence will then depend upon the interpretation of preceding sentences.\(^6\)

\section*{4.1.3 Model\(^7\)}

Here, a simple domain-theoretic model of the \( \lambda \)-calculus is sketched, extended to model a Frege Structure. First of all we need a model for the \( \lambda \)-calculus. This can be used to build a model of PT.

\subsection*{4.1.3.1 A Model of the \( \lambda \)-Calculus}

Following an existing approach [Scott, 1973], we can build a model of the \( \lambda \)-calculus from \textit{domains} consisting of \textit{complete lattices}. In the limit we have a domain \( D_\infty \) isomorphic to its own continuous function space, so \( D_\infty \cong [D_\infty \rightarrow D_\infty] \). We can define mappings \( \Phi : D \rightarrow [D \rightarrow D] \) and \( \Psi : [D \rightarrow D] \rightarrow D \).

\begin{definition}
A Scott Model is a triple \( \mathcal{D} = \langle D, \Phi, \Psi \rangle \) with \( D \) a domain and \( \Phi, \Psi \) as above.
\end{definition}

\(^6\)The term \( \Sigma \lambda x.g \ (x \ not \ free \ in \ g) \) is equivalent to \( f \odot g \).

\(^7\)Culled from [Fox, 1993]
The terms of \( \lambda \)-calculus can be interpreted in such a structure relative to both an assignment function \( g \), which assigns elements of \( D \) to variables, and an interpretation function \( i \), which assigns elements of \( D \) to constants. The function \( g[d/x] \) is the function \( g \) except that \( d \) is bound to \( x \). Reference to \( D \) is dropped in the following, and \( i \) is assumed to be fixed:

\[
\begin{align*}
I[x]_g &= g(x) \\
I[c]_g &= i(c) \\
I[\lambda x t]_g &= \Psi(\lambda d. I[t]_g[d/x]) \\
I[t(t')]_g &= \Phi(I[t]_g)(I[t']_g)
\end{align*}
\]

### 4.1.3.2 A Model of PT

Following [Aczel, 1980]:

**Definition 2** A model for PT shall be taken to be a Frege structure \( \mathcal{M} = (\mathcal{D}, T, P) \) where \( \mathcal{D} \) is a model of the Lambda Calculus and

\[
\begin{align*}
T : D &\rightarrow \{0, 1\} \\
P : D &\rightarrow \{0, 1\}
\end{align*}
\]

Where \( T \) and \( P \) satisfy the structural requirements in [Aczel, 1980].

The characteristic functions \( T \) and \( P \) provide the extensions of the truth predicate, and the proposition predicate, respectively. The structural requirements of \( T, P \) given by Aczel [Aczel, 1980] (which are not repeated here) verify the appropriate axioms of PT. As an example, the function \( T \) characterises a subset of \( P \). Thus the terms have a subclass consisting of terms that correspond to propositions; this subclass, in turn, has a subclass of terms corresponding to the true propositions.

The language of wff can now be given truth conditions.

\[
\begin{align*}
\mathcal{M} \models_{\mathcal{I}} s = t & \text{ iff } I[t]_g = I[s]_g \\
\mathcal{M} \models_{\mathcal{I}} T(t) & \text{ iff } T(I[t]_g) = 1 \\
\mathcal{M} \models_{\mathcal{I}} P(t) & \text{ iff } P(I[t]_g) = 1 \\
\mathcal{M} \models_{\mathcal{I}} \varphi \& \psi & \text{ iff } \mathcal{M} \models_{\mathcal{I}} \varphi \text{ and } \mathcal{M} \models_{\mathcal{I}} \psi \\
\mathcal{M} \models_{\mathcal{I}} \varphi \lor \psi & \text{ iff } \mathcal{M} \models_{\mathcal{I}} \varphi \text{ or } \mathcal{M} \models_{\mathcal{I}} \psi \\
\mathcal{M} \models_{\mathcal{I}} \varphi \rightarrow \psi & \text{ iff } \mathcal{M} \models_{\mathcal{I}} \varphi \text{ implies } \mathcal{M} \models_{\mathcal{I}} \psi \\
\mathcal{M} \models_{\mathcal{I}} \neg \varphi & \text{ iff } \mathcal{M} \models_{\mathcal{I}} \text{ not } \varphi \\
\mathcal{M} \models_{\mathcal{I}} \forall x \varphi & \text{ iff } \text{ for all } d \in D \mathcal{M} \models_{\mathcal{I}}[d/x] \varphi \\
\mathcal{M} \models_{\mathcal{I}} \exists x \varphi & \text{ iff } \text{ for some } d \in D \mathcal{M} \models_{\mathcal{I}}[d/x] \varphi
\end{align*}
\]

A wff \( \varphi \) of PT is valid in a model \( \mathcal{M} \) iff \( \mathcal{M} \models_{\mathcal{I}} \varphi \) for all assignment functions \( g \).
4.2 Syntax-semantics Interface

There are many possible approaches to the representation of natural language (NL) semantics in PT. In this section, a static PTQ-style treatment given by Turner is mentioned [Turner, 1992]. It will also be shown how it is possible to express the relationship between NL sentences and their truth conditions less directly, via underspecified terms. These terms are underspecified in the sense that they do not interpret NL constructions directly with logical constants, and thus they do not directly inherit their fixed truth conditional, and scoping, behaviour. This is not an alternative to other approaches, but is something which could be adopted by them.

Later, in section 4.4.1 in Deliverable 9, two alternatives to a static PTQ-like analyses are introduced. One is due to Chierchia, where ideas from Dynamic Montague Grammar [Groenendijk and Stokhof, 1990; Groenendijk and Stokhof, 1991] are added to PT [Chierchia, 1984]. In the second alternative, quantifiers are represented with dependent type constructors as has been used in constructive semantics for natural language [Sundholm, 1989; Ranta, 1991; Davila-Perez, 1994; Turner, 1994]. This gives a dynamic theory with no changes to the model.

In implementing theories of natural language semantics in PT, the basic approach is to find a compositional representation of sentences as terms, and, if necessary, to add more axioms to achieve the desired truth conditions for sentences and discourse. For the PTQ fragment, no new axioms are required, given a suitable compositional representation. For the underspecified representations, and for theories of certain semantic phenomena such as plurals and mass terms, the bulk of the work is in providing additional axioms which obtain suitable truth conditions.

The freedom that an axiomatic approach gives may seem a heavy responsibility: it is be possible to create an inconsistent theory with some apparently simple and desirable additions. However, no semantic framework can guarantee the construction of only consistent theories. All theories may result in unwanted consequences (indeed, as mentioned in section 4.1, property theory is intended to address some of the unwanted consequences of Montagovian, model-theoretic Intensional Logic, and syntactic treatments of propositional attitudes). One methodology, which can help avoid inconsistency, is to adopt the weakest axioms possible which have the desired effect. Further, if the additional axioms are mainly concerned with truth conditions, and guarded with requirements that relevant terms are propositions, then the chances of reintroducing the paradoxes are reduced.

PT provides no specific inference theory other than first-order theorem proving with \(\lambda\)-equality. In a particular application there may be methods that can be employed to reduce the complexity of the inference process, and perhaps make it decidable. The basic axioms of the theory concerning proposition-hood P and truth T can typically be implemented as Horn clauses. Provided that lemmas in the model concerning which terms constitute propositions, and true propositions, reflect the form used in the antecedents of the axioms, then the theory is decidable for those terms which match the axioms ‘syntactic’ coverage, and simple proofs need not be computationally complex. As for more complex inferences, presumably there comes a point with all semantic formalisms where specific inference mechanisms are not
enough, and something like the full power of first-order theorem proving is required.

4.2.1 PTQ-like Interpretation

It has been shown in several papers how a PTQ-style treatment carries over to property-theoretic semantics [Turner, 1992; Chierchia and Turner, 1988; Kamareddine, 1988; Chierchia, 1991b]. The essential difference, compared to the standard Intensional Logic version of PTQ [Dowty et al., 1981], is that all terms in PT are already intensional, so there is no need for the operator $\wedge$ which is used in PTQ to derive intensions from extensions. A similar fragment has been treated using context-free attribute-value grammar (implemented with a bi-directional chart parser [Steel and De Roeck, 1987]) [De Roeck et al., 1991a; De Roeck et al., 1991b; De Roeck et al., 1991c].

The semantic representation of a sentence is an intensional term. If this term can be shown to correspond with a proposition via the axioms for P, then its truth conditions can be found using the axioms for T. As an illustration, the sentence:

\[\text{Every boy laughed.}\]

could be represented as the term:

\[\Theta x(\text{boy}^\prime x \Rightarrow \text{laughed}^\prime x)\]

This object is independent of any truth conditions. To find the truth conditions of the sentence, we must first show that the term representing it is a proposition, that is:

\[P(\Theta x(\text{boy}^\prime x \Rightarrow \text{laughed}^\prime x))\]

This is an expression in the language of wff. According to the axioms for P, this will hold if:

\[\forall x(P(\text{boy}^\prime x) \rightarrow P(\text{laughed}^\prime x))\]

This can be proved if the terms boy', laughed' have been declared to be properties.

If the sentence is a proposition, then its truth conditions are given by:

\[T(\Theta x(\text{boy}^\prime x \wedge \text{laughed}^\prime x))\]

According to the axioms, this holds if and only if:

\[\forall x(T(\text{boy}^\prime x) \rightarrow T(\text{laughed}^\prime x))\]

Not all sentences will express propositions. As an example, the axioms should not allow the representation of:

\[\text{This sentence is false.}\]

to be a proposition, otherwise the theory would fall foul of the paradoxes.
Not all logical constants in the representations of sentences will be interpreted as logical connectives in the truth conditions. The sentence:

\[ \text{Mary believes that every boy laughed.} \]

might be represented by the term:

\[ \text{believe}'(\Theta x(\text{boy}'x \Rightarrow \text{laughed}'x))\text{mary}' \]

If this is a proposition, then in its truth conditions T will not apply to “every boy laughed”:

\[ T(\text{believe}'(\Theta x(\text{boy}'x \Rightarrow \text{laughed}'x))\text{mary}') \]

This corresponds to the idea that the object of a belief is an intensional proposition, not a truth value, or set of possible worlds.

### 4.2.2 Underspecified Semantics

It is possible to achieve the effect of underspecified representation by compositionally translating to some “neutral” term. The truth conditions can then be found by strengthening the basic theory with additional axioms [Fox, 1993]. As a brief example of underspecification with respect to quantifier scoping, the compositional representation of:

\[ \text{Every man loves a woman.} \]

might be the term:

\[ (\text{love}'(a'\text{woman}')((\text{every}'\text{man}')) \]

which just indicates the predicate-argument structure of the original sentence. We could allow this representation to subsume the two possible scopings of the quantifiers.

To prove that this expression is a proposition, we could type the atomic terms directly in PT so that it is provably a proposition from the axioms of P, or indirectly, by showing that it has truth conditions, and so must be a proposition. Note that to prove its proposition-hood directly requires that the lexical items belong to types which are different to those used in a PTQ-style analysis.

We can ensure that its truth conditions are effectively disjunctive so that they can be satisfied in two ways. Assuming the relevant expressions are propositions, we need to be able to derive (abbreviating the names of the properties):

\[\begin{align*}
T(\Theta x m'x \Rightarrow (\exists y w'y \land \ell'yx)) & \rightarrow T((\ell')(a'w')(\text{every}'m')) \\
T(\exists y w'y \land (\Theta x m'x \Rightarrow \ell'yx)) & \rightarrow T((\ell')(a'w')(\text{every}'m'))
\end{align*}\]
This can be achieved in a variety of ways. As an example, we could use the following four axioms for simple sentences with transitive verbs (an axiom for each quantifier in each argument position):

\[
\begin{align*}
P_s(\Theta x(nx \Rightarrow px)) & \land \quad T(\Theta x(nx \Rightarrow px)) \rightarrow T(p(every'n)) \\
T_s(\Xi x(nx \land px)) & \land \quad T(\Xi x(nx \land px)) \rightarrow T(p(a'n)) \\
T_s(\Theta x(nx \Rightarrow rxt)) & \land \quad T(\Theta x(nx \Rightarrow rxt)) \rightarrow T(r(every'n)t) \\
T_s(\Xi x(nx \land rxt)) & \land \quad T(\Xi x(nx \land rxt)) \rightarrow T(r(a'n)t)
\end{align*}
\]

where \( P_s \) characterises those propositions derived from natural language sentences, and \( P_s(t) \rightarrow P(t) \). Restricting the axioms to this subclass of \( P \) avoids forcing all propositions of this form to have this special behaviour. It may be possible to generalise this to arbitrary numbers of noun phrases, perhaps by encoding arguments of verbs as nested pairs.

In effect, this approach models the process of semantic interpretation of sentences (or parse trees) inside PT.
Chapter 5

Monotonic Semantics

This chapter describes an approach to the semantic interpretation of natural language known as monotonic semantic interpretation. The approach is exemplified by consideration of the Quasi Logical Form (QLF) notation used in SRI Cambridge's Core Language Engine.

5.1 Semantic Tools

5.1.1 Introduction and Motivation

Sentences taken out of context often have a multiplicity of possible interpretations. These can arise from a variety of sources, e.g.: structural/syntactic ambiguity; lexical ambiguity; different relative scopes of operators and quantifiers; different antecedents for anaphors and ellipses; implicit relations, e.g. possessives, compound nouns; vague relations and properties, etc.

In the theoretical linguistics literature the tradition has been, where possible, to define rules which produce for a sentence all possible alternative interpretations, leaving it to some other, usually only vaguely sketched, contextual component to decide which is the appropriate interpretation.

It has long been recognised in computational linguistics that this approach is practically implausible: it is a ‘generate and test’ solution of the type which has in many different areas of NLP and AI led to an overwhelming search problem. Many also suspect it to be theoretically implausible too. An alternative approach has been to try to develop representations that are underspecified: that represent the purely linguistic contribution of the sentence to its meaning, leaving contextual constraints on full interpretation to a later stage.

In the specific cases of phenomena like quantifier scope, or pronoun interpretation, such a move is by no means novel. In an early system like that of Woods' [Woods, 1977], for
example, quantifier scoping was handled by a set of rules operating on a canonical ordering of quantifiers (essentially as determined by the syntax), to produce alternative scopings where necessary. However, in Woods’ approach, and in some others using some ‘quasi logical form’ notions, e.g. [Schubert and Pelletier, 1982; Hobbs and Shieber, 1987], these manipulations are destructive (in the technical, programming language, sense): they move chunks of logical form around, delete some things, and add some other things. The semantics of these operations, and the representations they operate on, is seldom if ever specified: it is only the output representation that is semantically transparent. For many practical purposes this may not matter, but it does matter for some. For example, if as well as analysing sentences with pronouns or multiple quantifiers, one wants to generate them from a more explicit structure, it is as well to have some means, at least in principle, of knowing that the generated sentences can have the correct interpretation.

Monotonic semantics is an approach to semantic interpretation that focuses on underspecification — the ability to represent multiple possible interpretations by a single structure which is neutral between them all — and the non-destructive resolution of underspecification. Non-destructive resolution is achieved by the use of unification to instantiate meta-variables in semantic representations. The meta-variables (which are distinct from logical variables) represent those aspects of meaning that are contextually underspecified.

Unification-based syntactic formalisms have a number of computationally beneficial properties, emerging primarily from the monotonic behaviour of unification: independence of specific processing architectures; the absence of complex, order-dependent interactions between phenomena; potential reversibility; the possibility of producing partial analyses. At a practical level, by using unification in contextual resolution we aim to derive similar computational advantages in semantic and contextual processing.

At a more theoretical level, a question arises in unification-based syntactic formalisms of whether feature structures are partial syntactic objects, or partial descriptions of (complete) syntactic objects [Johnson, 1988; Keller, 1993]. A similar question arises here. Underspecified semantic representations in monotonic semantics are best seen as partial descriptions of a certain kind of semantic object (semantic compositions), and not as partial semantic objects themselves:

Traditionally, semantic interpretation has been viewed as a matter of composing the meanings of a sentence’s constituents to derive the meaning of the sentence as a whole. We might baptise this view interpretation as composition. Monotonic semantics introduces an extra level of indirection, and adopts the view of interpretation as description. That is, semantic interpretation is a process of building a description of the final composition, rather than actually performing the composition.

In the absence of context we may not know the precise meanings of certain constituents (e.g. pronouns), or the precise way in which certain constituents are to be composed together (e.g. the scope of quantifiers). That is, interpretation on the basis of the syntactic structure of a sentence alone will usually furnish only a partial description of the intended semantic composition. Contextual resolution serves to fill the gaps in this partial description. At
some point, contextual resolution may furnish enough information to allow us to identify and proceed with the composition. However, performing the composition is best seen as part of semantic evaluation—determining whether the sentence is true, updating context, or whatever—as distinct from interpretation, which determines what a sentence means but not whether it is true etc.

We will argue below that viewing interpretation as description has a good deal to recommend it, both in providing a monotonic, order-independent model of semantic interpretation, and in providing a reasonably well motivated level of intermediate semantic representation. The former is of computational interest, and the latter facilitates the analysis of context-dependent phenomena like anaphora and ellipsis.

These claims will be backed by discussion of the Quasi Logical Form (QLF) formalism developed at SRI Cambridge [Alshawi, 1990; Alshawi, 1992; Alshawi and Crouch, 1992]. This should not be taken to imply that monotonic semantics is simply an alternative name for quasi logical form. Monotonic semantics is a general approach to semantic interpretation, while QLF represents a particular implementation of this approach.

This chapter is structured as follows. Section 5.1.2 gives more background about the role of interpretation in semantics and the requirements for and advantages of monotonicity. Section 5.1.3 introduces the Quasi Logical Form (QLF) notation, with some examples illustrating its application. Section 5.1.4 provides a semantics for QLF. Section 5.1.5 comments on QLF and its semantics, sometimes in a critical manner. Section 5.1.6 speculates, in the light of these comments, on alternatives to QLF as an implementation of monotonic semantics. Section 5.2 describes the syntax-semantics interface for QLF. Also note that Deliverable 9 gives an analyses of a variety of linguistic phenomena for QLF, as specified in FraCaS deliverable D2.

5.1.2 Interpretation and Monotonicity

Interpretation

Two issues are of primary concern to any theory of the semantics of natural language:

**Interpretation** The way in which a sentence (or utterance of a sentence) is mapped onto its meaning.

**Model Theory** What these meanings are. This is usually presented in terms of some kind of model theory, which gives a formalisation of the entailment relations holding between the meanings of various sentences (or utterances).

Montague semantics is a prime example of a theory addressing both the above. On the interpretation side, it provides a strictly compositional mapping from the syntactic structure
of a sentence to its meaning. And its intensional, higher-order model theory accounts for some entailment relations holding between sentences containing e.g. intensional verbs.

While interpretation and model theory are not completely independent of one another,\(^1\) there is scope for varying one while leaving the other fixed. For example, Property Theory revises montagovian model theory to give a finer grained account of intensionality, but typically leaves the strictly compositional syntax-semantics mapping as it is [Turner, 1992]. Cooper storage [Cooper, 1983] is one proposal for relaxing strict compositionality of the interpretive mapping (so as to permit semantic ambiguity without syntactic ambiguity), but can be employed with fairly standard montagovian model theory.

Matters of intensionality aside, the major departures from Montague semantics have been motivated by the need to account for the contextual dependency of sentence meanings. In some cases e.g. Situation Theory, this has prompted substantial changes both to the model theory and to the interpretive mapping. Others, like Discourse Representation Theory, have made major changes to the interpretive mapping but lesser (though still substantial) changes to the model theory. Some varieties of dynamic semantics, e.g. Dynamic Montague Grammar, have attempted to retain as much as possible of the compositional interpretive mapping and have adjusted the notion of meaning/model theory accordingly.

Monotonic semantic interpretation, as its name suggests, focuses squarely on the interpretive mapping as the place to deal with most, if not all, contextual dependency. No real innovation in model theory is attempted. Thus QLF tends to be quite conservative in the assumptions it makes about the underlying model theory, sticking mainly to a ‘Montague without the intensions’ approach.

This should not be taken to imply that we feel no model-theoretic innovation is needed for dealing with natural language. A property theoretic treatment of intensionality appears promising, for example, and there seem to be no good reasons why it could not be incorporated within a monotonic approach to interpretation. However, we do suspect that some of the model-theoretic innovations motivated on contextual grounds are misguided.

QLF’s employment of a ‘vanilla’ higher-order model theory is best be seen as a methodological device: hold the model theory constant while seeing how much mileage can be got through varying the interpretive mapping. At the end of this it may become apparent that (a) some phenomena still require model theoretic alteration, e.g. intensionality, and/or (b) changes to the interpretive mapping necessitate adjustments to the model theory. Having not yet reached the end of this process, we will refrain from discussing (a) and (b).

\(^1\)For example, consider non-intensional Montague-style grammars producing only first-order analyses of sentences. To preserve compositionality, individual words frequently have to be given higher-order meanings, even though sentence meanings built from them have a first-order \(\beta\)-redex. There is perhaps a sense in which the higher-order meanings of words reflect the logical structure of semantic interpretation, as opposed to the meanings derived through interpretation.
Monotonicity and Semantic Composition

Monotonicity and Unification  The monotonicity of unification has been one of the primary reasons for the success of unification-based grammar formalisms in computational linguistics. Monotonicity permits and/or facilitates:

- Order independence of unification/parsing operations:
  In other words, the formalism does not enforce a particular processing architecture for parsing; one can proceed left to right, right to left, outwards from syntactic heads, top down, bottom up, and so forth.

- Simpler interactions:
  The same order independence means that one does not have to worry about order-dependent interactions between different components of the grammar (no complex cyclic applications of rules, etc). This makes grammars written within a unification-based formalism more perspicuous

- Reversibility for generation:
  Again, order-independence allows grammars to be used in both directions, for both parsing and generation. (Note that monotonicity alone is generally not enough to ensure effective reversibility: information flow in both directions must be sufficient to constrain search to reasonable bounds)

- Production of partial analyses:
  Unification is used to monotonically build up information about the syntactic structure of the sentence. Intermediate stages provide partial descriptions / analyses of the syntactic structure, and these can be useful in their own right. Fuller analyses are obtained by adding more and more information to the partial analyses, but never removing information that is already there.

These properties all concern the construction of syntactic representations. As will be clear to those familiar with unification-based grammatical formalisms, this is a picture that follows quite naturally from the view of feature structures as descriptions of syntactic objects rather than being syntactic objects themselves. An underspecified feature structure is thought of as a partial description of a large, possibly infinite set of fully specified syntactic objects which it subsumes. Monotonicity ensures that construction of these descriptions is confluent, i.e. order-independent. That is, where one decides to start constructing a feature structure, and the order in which different parts of it are built up, does not affect the range of possible outcomes.

The above monotonicity properties are of great desirability in computational semantics. We are now going argue that to obtain these properties, we must resist the temptation to identify semantic interpretation with semantic composition. Instead we should view interpretation as building descriptions of semantic compositions.
**Order-Dependence of Composition**  Semantic composition is highly order-dependent. For example, the order in which an N-ary predicate is applied to its individual arguments makes a substantial difference to the final interpretation (e.g., the difference between *John loves Mary* and *Mary loves John*). Similarly, the order in which quantifiers are scoped during composition can make a big difference to the outcome. So if we wish to have monotonic, order-independent semantic interpretation, composition is simply the wrong kind of process.

This is not to deny that semantic composition — the way that the meanings of the parts are composed together to give the meaning of the whole — is a vital part of semantics. It is only to deny that we should identify composition with interpretation, if we want a computationally useful account of interpretation. Semantic composition is the means by which semantic values (i.e., meanings) for sentences are constructed. But the monotonicity properties we are interested in concern the construction of semantic representations or descriptions. Semantic values and semantic representations are not the same thing.

This can be brought out by considering Montague semantics, where there is no (ineliminable) intermediate level of semantic representation, for the simple reason that the syntactic structure of a sentence is all the semantic representation that is required. A strict syntactic-semantic homomorphism means that syntactic structure furnishes a complete specification / description of the meanings of the constituents and the way they are composed together. But this is not to say that syntactic structures (i.e., semantic representations) are meanings (i.e., semantic values like propositions resulting from the composition). This would be to confuse a description of an object with the object itself.

There is no reason why syntactic structures in Montague semantics should not be constructed monotonically through the use of, say, a unification-based grammar. And if syntactic structures are semantic representations, we also get a form of monotonic semantic interpretation. Looking at things in this way separates composition out from interpretation, and the overall mapping from sentence to meaning goes in two stages. Semantic interpretation is handled by parsing, and builds up what is in effect a full description of the intended semantic composition. The composition described is then is then carried out to construct the semantic value of the sentence. Performing the composition adds no significant new information: the semantic value is already fixed by the syntactic structure, and the composition merely constructs it.

The Montagovian assumption that syntax alone determines the meaning of sentences is of course widely held to be implausible. A common response has been to continue to run semantic composition off syntactic structure, but to relax the syntactic-semantic homomorphism, e.g. Cooper storage or Pereira’s categorial semantics [Pereira, 1990]. Syntactic structure is thus still used as a (form of) semantic representation\(^2\), but one that does not uniquely fix the meaning of a sentence because it does not uniquely specify one semantic composition. One actually has to perform the composition, making decisions where there is scope for choice, in order to determine what the sentence means. Because of this, composition becomes part of the process of fixing the meaning of the sentence, i.e. part of semantic interpretation. Indeed, categorial semantics explicitly equates semantic interpretation and composition.

\(^2\)Following the Montagovian tradition that there should be no other intermediate levels of representation.
What this means in practice is that one typically builds up some sort of initial semantic representation in parallel with the syntactic one, perhaps using unification to construct both. But one then has to perform various destructive operations on the initial representation, e.g. raising quantifiers, replacing pronouns by their referents [Woods, 1977; Schubert and Peletier, 1982; Pereira and Pollack, 1991; Dalrymple et al., 1991]. These destructive manipulations reflect those areas where semantic composition is underdetermined by the initial semantic representation.

Moreover, the order in which these manipulations are performed typically affects the semantic value finally obtained. In Dalrymple, Shieber and Pereira’s (DSP’s) treatment of ellipsis [Dalrymple et al., 1991] for example, the order in which quantifiers are scoped and ellipses resolved affects the final interpretation. So long as fixing quantifier scopes and ellipsis resolvers are seen as things to be done during composition, this order dependence is what one would expect. However, it forces a particular processing architecture on any system wishing to employ DSP’s analysis: scoping must be interleaved with ellipsis resolution. One cannot have a pipe-lined architecture where scoping is done at one stage and ellipsis resolution at another. Unfortunately, there are good practical reasons for preferring this kind of pipe-lined architecture.

**Intermediate Semantic Representations** Making composition part of interpretation (i.e. relaxing strict compositionality in the syntax-semantics mapping) makes interpretation order-dependent and non-monotonic. But there are alternative ways of dealing with the fact that syntax alone does not fix the meaning of a sentence.

The most obvious is to introduce an intermediate level of semantic representation, distinct from syntactic structure. The syntactic structure of a sentence contributes to the intermediate representation, but does not completely fix it. Further contextual resolution fleshes out the intermediate representation. Once this is done, a strictly compositional mapping from the intermediate representation to meanings can be provided. This is essentially the policy adopted in Discourse Representation Theory.

Introducing an intermediate level of semantic representation (a) does not alone guarantee monotonicity in interpretation and (b) is felt by some to be dangerously arbitrary. On the first count, there is nothing to prevent the use of an order-dependent algorithm for constructing the intermediate representation. This is a particular danger if the construction algorithm is directly modelled on some form of semantic composition.

On the second count, the form of the intermediate representation is potentially underdetermined by sentence structure at one end and meaning at the other. There are plausible arguments, such as Fodor’s [Fodor, 1975], that meanings must be ‘computational’ and given in some kind of formal representation with which the mind can compute. But it is not at all clear what empirical evidence can be brought to bear on determining what form the representation should take.
**Interpretation as Description Building**  A variant on the theme of using intermediate semantic representations stems from a descriptive view of feature structures in unification-based grammatical formalisms. We take the intermediate level of semantic representation to be a description of the intended semantic composition.

As with feature structures, the description may be incomplete. In particular, the syntactic structure of a sentence typically only provides enough information for a partial description of the semantic composition. Contextual resolution fills the description out by adding further components to it.

Of crucial importance here is that the order in which different parts of a description are built up should not influence the object being described. For example, we might build some description in the order shown below:

1. Give quantifier $Q_1$ wide scope with respect to operator $O$.
   Then,
2. Give quantifier $Q_2$ narrow scope with respect to operator $O$.

This describes the same set of objects as the differently ordered description:

1. Give quantifier $Q_2$ narrow scope with respect to operator $O$.
   Then,
2. Give quantifier $Q_1$ wide scope with respect to operator $O$.

The order-independence of constructing the description contrasts with the order-dependence of the operations in the semantic composition begin described. Thus,

1. Scope quantifier $Q_1$
   Then,
2. Scope quantifier $Q_2$

describes a different scoping from

1. Scope quantifier $Q_2$
   Then,
2. Scope quantifier $Q_1$

So, if we view semantic interpretation as description building, exploiting information from
both syntax and context, one is naturally drawn to an order-insensitive model of interpretation. Unification would be the obvious tool for building the description up.

Another virtue to this approach is that if the intermediate level of semantic representation is a description of semantic objects like compositions, then it (arguably) appears a lot less arbitrary.

Semantic interpretation viewed as description building is not an entirely novel idea. One could (probably) construe certain flavours of DRT in this light. And Nerbonne’s constraint-based semantics [Nerbonne, 1991] explicitly builds descriptions of logical forms via unification and feature structures. Indeed, given that conventional logical forms have a strict homomorphism from their structure to their meaning, Nerbonne’s approach can be seen as one of building descriptions of semantic compositions rather than just logical forms. Although he does not present things in this way, this move would deflect objections on the grounds that describing logical forms make illicit reference to arbitrary syntactic aspects of the logical form.

**Semantic Monotonicity**  Up till now we have been concerned with a form of syntactic monotonicity: monotonicity in building up semantic representations. But there is also a semantic analogue

Semantic interpretation can be seen as a matter of combining several sources of information to arrive at the meaning of a sentence. Syntactic structure is one vital source of information, and context (e.g. contextual salience of various objects or properties, contextual plausibility of various meanings) is another. As more information is brought to bear during interpretation, more possible readings for a string of words are eliminated. Once eliminated, further information should never cause a reading to be reinstated.

More specifically, an underspecified/unresolved interpretation for a sentence is one where there are models where the sentence counts as true under one (more complete) interpretation/reading and false under another. Resolution monotonically decreases the number of models where the sentence can be both true and false (under different interpretations); i.e. it monotonically increases the number of models where the sentence is either true under all interpretations (definitely true) or false under all interpretations (definitely false). Of course, one could contrive to build the descriptions in an order-dependent way, though this would be perverse. And in practice, it is much easier to start by building the description on the basis of syntax and then adding the contextual parts than it is to go the other way round. But this is because syntax typically provides more information about the intended composition than context.

For example, the sentence *He slept* can be seen, in the absence of context, as meaning that some male, salient in context, slept. Contextual resolution might further specify the meaning of *he*, so that we can refine it to saying that John is a contextually salient male and that John slept. An alternative further specification might be that Bill is a contextually salient male and that Bill slept. Both these interpretation are subsumed by the original underspecified one.

---

2 Of course, one could contrive to build the descriptions in an order-dependent way, though this would be perverse. And in practice, it is much easier to start by building the description on the basis of syntax and then adding the contextual parts than it is to go the other way round. But this is because syntax typically provides more information about the intended composition than context.

3 Supervaluation [van Fraassen, 1966] proves to be a powerful tool for dealing with different interpretations within a given model, and plays a central role in specifying the semantics for QLF.
Thus in a model where John slept but Bill didn’t, He slept would be true under one interpretation but false under another. If we resolve the sentence to mean that John slept, then the sentence becomes definitely true in that model.

Another example is Every man likes some woman. Without resolving quantifier scope this is true in a model if either there is some woman that every man likes, or if every man likes at least one woman. It is false if either there is no single woman that every man likes, or if there is some man who likes no woman. As before, the sentence can be both true and false (on different scopings) in the same model.

This example is complicated by the fact that the $\exists w \forall m. l(m, w)$ scoping entails the $\forall m. \exists w. l(m, w)$ scoping. Suppose that we resolve the sentence to give the existential wide scope. This does not increase the number of models where the sentence counts as definitely true, since every model in which one woman is liked by all men is also a model where all men like at least one woman. However, the resolution does increase the number of models where it is definitely false: any model in which different men like different women. It is thus important to keep track not only of those models where a sentence counts as true, but also those where it counts as false.

Pictorially, the monotonic effects of resolution are illustrated in figure 5.1. The first part shows a set of models and indicates the models where an unresolved sentence counts as true under at least one interpretation and the models where it counts as false under at least one interpretation. There is an overlap between these two subsets: where the sentence is both true and false, though under different interpretations. The second part illustrates the effects of resolution. Fixing on the intended interpretation eliminates models where the sentence can be true (or false) under at least one interpretation, and correspondingly increases the number of models where it is either definitely true or false. At full resolution, all models will be such that the sentence is either definitely true or definitely false in that model.

It is important to bear in mind that what resolution eliminates is possible semantic compositions; elimination of models where the sentence can count as true or false is a by-product of this. Eliminating compositions does not always eliminate models. For example, a sentence like A man likes a woman has two possible compositions, depending on which quantifier is scoped first. But both compositions lead to the same meaning. So this is a case where resolving the scope of the quantifiers, while eliminating possible compositions, does not eliminate any further models.

The fact that it is compositions that are monotonically eliminated, and not models, is crucial to understanding the connections between the sentences

(i) He slept.
(ii) A man slept.
(iii) John slept.

Given that John is a contextually salient man, then (iii) entails both (i) and (ii). Despite this, (iii) only counts as a possible resolved reading for (i). For (i) we have a partially described

193
A. Unresolved

\[ \begin{array}{c|c|c}
\text{True} & \text{True and false} & \text{False} \\
\end{array} \]

B. Resolved

\[ \begin{array}{c|c}
\text{True} & \text{False} \\
\end{array} \]

Figure 5.1: Monotonic Effects of Resolution
composition saying that the predicate corresponding to \textit{slept} must be applied to some term whose meaning is not fully specified but which must denote some contextually salient male. Sentence (iii) corresponds to a particular way of fleshing this composition out. Sentence (ii) on the other hand already describes the composition fully. The meaning of \textit{a man} is already determinate.

Putting things somewhat differently, (i) can mean ‘John slept’ or it can mean ‘Bill slept’, or it can mean ... But (ii) means ‘John or Bill or ... slept’. That is, the difference is between (i) a disjunction of meanings i.e. alternative compositions or (ii) a disjunctive meaning i.e. a single composition covering a range of models. Unless we distinguish between meanings (sets of models etc) and compositions, it is hard to draw this distinction. Yet it a distinction that needs to be drawn [Poesio, 1994b; Milward, 1991].

5.1.3 QLF Syntax

Quasi Logical Form (QLF) grew during the course of developing the Core Language Engine (CLE) and a broad syntactic and semantic coverage of English (and latterly other languages). While it was devised in a conscious attempt to tackle underspecification and to employ unification-based methods to resolve it, its development was spurred more by practical considerations than theoretical. The advantage of this is that QLF is a semantic formalism that can be shown to have clear practical application. The disadvantage is that theory and practice are not always in step. In some cases, sound practice still awaits adequate theoretical description. In other cases theoretical considerations suggest revisions to the formalism that have yet to be implemented.6

One should therefore bear in mind in what follows that theoretical work on QLF is still in progress, even though practical work is often much further advanced than for other semantic frameworks. Indeed one of the aims of this document is to advance the theoretical understanding of QLF.

In this section we will describe the syntax of the QLF notation, and in the next we will describe its semantics. The material from these sections is derived from [Alshawi and Crouch, 1992], although it differs in a number of details.

A QLF term must be one of the following

- a term variable: \textit{x, y, ...}
- a term index: \textit{+i, +j, ...}
- a constant term: \textit{7, mary, ...}
- an expressions of the form:
  \textit{term(Idx, Cat, Restr, Quant, Reft)}

\footnote{We are ignoring the contextual indeterminacy of such things as tense.}
\footnote{It is unclear, for example, that QLF is really the optimum language for describing semantic compositions.}
The term index, \texttt{Idx}, uniquely identifies the term expression. \texttt{Cat} is a list of feature-value equations, for example \texttt{<type=pro,num=sing,\ldots>}. \texttt{Restr} is a one-place predicate of entities. For a resolved term, \texttt{Quant} will be a generalized quantifier/determiner (a cardinality predicate holding of the extensions of two properties) and \texttt{Reft}, the term’s ‘referent’, will be a predicate specifying a contextual restriction on the range of quantification. For an ‘unresolved’ term, \texttt{Quant} and \texttt{Reft} may be meta-variables (?P, ?Q, ?R).

Predicates may either be atomic (e.g. \texttt{man}, \texttt{dog}, \texttt{like}) or constructed using lambda abstraction. A lambda abstract takes the form \texttt{Var^Body} and \texttt{Body} is a formula or an abstraction and \texttt{Var} is a variable ranging over individuals or relations.

A QLF formula must be one of the following

- the application of a predicate to arguments:
  \texttt{Predicate(Argument1,\ldots,Argumentn)}

- an expression of the form:
  \texttt{form(Idx,Category,Restriction,Resolution)}

- a formula with scoping constraints:
  \texttt{Scope:Formula}

- a formula with a re-interpretation:
  \texttt{Formula:\{Term1/Term2,Term3/Term4,\ldots\}}

\texttt{Predicate} is a first or higher-order predicate, including the usual logical operators \texttt{and}, \texttt{not}, etc. An argument may be a term, a formula or predicate.

In forms, \texttt{Restriction} is a higher-order predicate. \texttt{Resolution} is a either a meta-variable or a contextually predicate. The meaning of the form results from applying its restriction to its resolution.

\texttt{Scope} is either a meta-variable when scoping information is underspecified or a (possibly empty) list of term indices e.g. \texttt{[+]i,+]j} if term \texttt{+i} outscopes \texttt{+j}. The terms identified by the indices must occur within \texttt{Formula}.

Our current notation does not allow for partial orderings of quantifier scopes: there is no means of specifying an ordering like \texttt{[{+]i,+]j},k\}]. However, it would be possible to add such a notation, with a corresponding complication of the evaluation rules given later.

The degree to which a QLF is unresolved corresponds approximately to the extent to which meta-variables (appearing above in the positions marked by \texttt{Quant, Reft, Scope, and Resolution}) are instantiated to the appropriate kind of object level expressions.

We will say more about re-interpretations in the section on ellipsis.
Some Examples

In order to illustrate the syntax of QLF and give a rough indication of its intended semantics, we will provide some examples of English sentences and their (approximate) QLFs, both before and after contextual resolution. It will hopefully become evident that the notation is closer to (the syntactic structure of) natural language than is the case for traditional logical formalisms. For example, terms usually correspond to noun phrases, with information about whether e.g. they are pronominal, quantified or proper names included in the term’s category. This makes the QLF representation easier to read than it might at first seem, once its initial unfamiliarity is overcome.

Quantification: Every boy met a tall girl illustrates the representation of quantification. The basic QLF analysis might be (ignoring tense):

\[
\begin{align*}
?S &: \text{meet} (\text{term}(+b, \text{type}=q, \text{lex}=\text{every}), \text{boy}, ?Q, ?X), \\
& \quad \text{term}(+g, \text{type}=q, \text{lex}=a), \\
& \quad y^\land (\text{girl}(y), \text{tall}(y)), ?P, ?R).
\end{align*}
\]

A resolved structure could be obtained by instantiating the quantifier meta-variables \(?Q\) and \(?P\) to \(\forall\) and \(\exists\), and the scoping meta-variable \(?S\) to \([+b, +g]\) for the ‘\(\forall\exists\)’ reading:

\[
\begin{align*}
 [+b, +g]:
& \quad \text{meet} (\text{term}(+b, \text{type}=q, \text{lex}=\text{every}), \\
& \quad \quad \text{boy}, \forall x \cdot x = x), \\
& \quad \text{term}(+g, \text{type}=q, \text{lex}=a), \\
& \quad \quad y^\land (\text{girl}(y), \text{tall}(y)), \exists x \cdot x = x).
\end{align*}
\]

Both terms have been resolved to have no further contextual restriction on their range of quantification, \(x^\sim x=x\), i.e. they are restricted to range over self-identical objects.

In a restriction-body notation for generalized quantifiers, the truth conditional content of this resolved expression corresponds to

\[
\begin{align*}
\forall (b, \text{boy}(b), \\
& \quad \exists (g, \text{and}(\text{girl}(g), \text{tall}(g)), \\
& \quad \quad \text{meet}(b, g))).
\end{align*}
\]

The benefits of being able to resolve determiners to quantifiers are discussed in [Alshawi, 1996]. For example, ‘any’ can be resolved to ‘\(\forall\)’ (‘any knife will do’) or ‘\(\exists\)’ (‘if any person arrives,’); determiners like some (plural) could be resolved to collective or distributive quantifiers; three could be interpreted as meaning either ‘exactly three’ or ‘at least three’, and if need be, bare plurals like dogs could be variously interpreted as meaning ‘some dogs’, ‘all dogs’ or ‘most dogs’.
Anaphora: *Every boy claims he met her* illustrates the treatment of anaphora (in a context where Mary is assumed to be salient)*^8^*

Unresolved:

?S1: claim(
  term(+b,<type=q, lex=every>, boy, ?Q1, ?R1),
  ?S2: meet(term(+h1,<type=pro, lex=he>,
    male, ?Q2, ?R2),
  term(+h2,<type=pro, lex=her>,
    female, ?Q3, ?R3))).

Resolved:

[+b]: claim(
  term(+b,<type=q, lex=every>,
    boy, forall, x^x=x),
  []: meet(term(+h1,<type=pro, lex=he>,
    male, exists, x^x=+b),
  term(+h2,<type=pro, lex=her>,
    female, exists, x^x=mary))).

The pronominal term for *her* is resolved so that it existentially quantifies over female objects identical to *mary*. The ‘bound variable’ pronoun *he* has a referent coindexed with its antecedent, *+b*. The category and restriction of the pronoun serves to identify possible antecedents, and hence determine possible quantifiers and contextual restrictions.

The scope of *+h2* is left unspecified, since exactly the same truth conditions arise if it is given wide or narrow scope with respect to *every boy* or *he*. Similarly for the bound pronoun *+h1*, although in this case constraints on interpretability (see p. 205) will ensure that it receives narrow scope with respect to its antecedent.

Vague Relations: An unresolved QLF expression representing the noun phrase *a woman on a bus* might be a *term* containing a *form* that arises from the prepositional phrase modification:

term(+w,<lex=a, ...>,
  x^x and (woman(x),
  form(+f,<type=prep, lex=on>,
    r^r(x, term(+b,<lex=a, ...>,
      bus, ?Q2, ?R))),
  ?F)),
  ?Q1, ?V).

Informally, the *form* is resolved by applying its restriction, r^r(...), to some appropriate contextually inferred predicate. The resolution is marked by instantiating the *form*’s resolvent meta-variable, ?F, to this predicate. In this case, the appropriate predicate might be *inside*, so that meaning of the formula as a whole corresponds to

*^8^Here we simplify the issues arising out of the semantics of intensional, sentential complement verbs like *claim*. 
inside(x,term(+b,<lex=a...>,bus,?Q2,?B)).

**Tense:** One way of treating tense is by means of a temporal relation form in the restriction of an event term. For *John slept* we might have:

```plaintext
?S:sleep(term(+e,<type=event>,
   e"form(+f,<type=trel,tense=past>,
      r"and(event(e),r(e)),
      ?T),
   ?Q1,?E),
   term(+j,<type=name>,
      J"name(j,'John'),?Q2,?J)).
```

Since the tense on the temporal relation category is past, the resolution says that the event occurred before a particular speech time, t7:

```plaintext
[+e]:
   sleep(
      term(+e,<type=event>,
         e"form(+f,<type=trel,tense=past>,
            r"and(event(e),r(e)),
            e1"precede(e1,t7)),
         exists,x"x=e),
      term(+j,<type=name>,
         x"name(x,'John'),exists,x"x=john)).
```

The resolved form corresponds to `and(event(e),precede(e,t7))`.

QLF is not committed to an event based treatment of tense. An alternative is to treat the verbal predication `sleep(....)` as a temporal form, whose category specifies tense and aspect information:

```plaintext
?S:form(+v,<type=verb,tense=past,lex=sleep>,
   p"p(term(+j,<type=name>,
      x"name(x,'John'),?Q,?J)
   )
).
```

According to taste (or perhaps to sound theoretical considerations!) the tensed verb form can be resolved in a variety of different ways, e.g. introducing a past tense operator, or introducing an event term plus temporal relation to give an equivalent resolution to the above. These two approaches would lead respectively to the following sorts of resolution:

```plaintext
?S:form(+v,<type=verb,tense=past,lex=sleep>,
   p"p(term(+j,<type=name>,
      x"name(x,'John'),?Q,?J)
      x"past(sleep(x))
).
```
(tense operator version), and

$$\begin{align*}
\text{[+e]:&form(\langle v, \langle \text{type=verb, tense=past, lex=sleep} \rangle,} \\
&\quad \text{p}'\text{p(term(\langle j, \langle \text{type=name} \rangle,} \\
&\quad \quad \text{x'\text{name(x, 'John')}, \text{?Q, ?J)} \\
&\quad \text{x'sleep(term(\langle e, \langle \text{type=event} \rangle,} \\
&\quad \quad \text{e'\text{and(event(e), precede(e, t7)} \\
&\quad \quad \exists x'x'=e),} \\
&\quad \text{x) )}
\end{align*}$$

(event version), where the last corresponds to the earlier more explicitly event-based analysis.

### 5.1.4 QLF Semantics

In this section we outline the semantics of the QLF language in a way that is as close as possible to classical approaches that provide the semantics in terms of a function from models to truth values. The main difference is that denotation functions will be partial functions for some unresolved QLF formulas, reflecting the intuition that these are 'partial interpretations'. Moreover, the denotation function is not built up directly. The recursive part of the semantic definition specifies a partial valuation relation, and a supervaluation style construction is used to derive a partial denotation function from the values that the relation assigns to the top level QLF formula.

The denotation of a QLF expression will be extended monotonically as it is further resolved, a fully resolved formula receiving a total function as its denotation. Note that the semantics is not intended to describe the process of resolution, although it forms the basis on which resolution operates, as described in section 5.2.2. The semantics described here is a more detailed version of that presented in [Alshawi and Crouch, 1992], although it is still subject to revision. Section 5.1.5 discusses some problems and possible modifications.

We will start by defining the valuation relation:

$$V(\text{QLF, M,Ctx, S, g, Subs, v})$$

where 
- \text{QLF} is a QLF expression
- \text{M} is a model
- \text{Ctx} is a context
- \text{S} is a salience relation
- \text{g} is an assignment of values to variables
- \text{Subs} is a set of reinterpretations (substitutions)
- \text{v} is a value assigned to the QLF expression
In what follows, we will usually omit explicit reference to the model, context and salience relation.

A QLF model is a higher-order model, as described in [Dowty et al., 1981]. That is, a pair \((O, F)\) where \(O\) is a domain of entities and \(F\) is a function from the non-logical constants with a range for constants of different types as follows:

- **Entities**: \(D_e = O\)
- **Formulas**: \(D_I = \{0, 1\}\)
- **Type \(\alpha \rightarrow \beta\)**: \(D_{\alpha \rightarrow \beta} = D_\beta^{D_\alpha}\)

For now, we will leave open what constitutes a context: the matter is addressed below. The salience relation, \(S\), relates QLF categories, term or form restrictions and contexts (Ctx) to quantifiers and other (contextually salient or inferred) properties that can be used to resolve terms and forms. The salience relation is also used in connection with scope constraints.

The assignment function, \(g\), maps variables of a given type \(\alpha\) onto objects of the appropriate type, \(D_\alpha\).

Reinterpretations (or substitutions) correspond in a direct way to reinterpretations as defined in the QLF syntax. That is, a reinterpretation is of the form \(new/old\), where \(old\) is some QLF expression that should be evaluated as though it were the expression \(new\). What kinds of QLF expression can be reinterpreted as what other kinds of QLF expression is something of an open question. The most conservative position is that terms can be reinterpreted as other terms, term indices or variables, term indices can be reinterpreted as other indices or variables, and forms can be reinterpreted as other forms. That is, non-logical constants, variables and so forth are not open to reinterpretation.

We will need to provide some operations on sets of reinterpretations:

- \(Subs_1 \uplus Subs_2\) combines two sets of reinterpretations. This is like set union, except that where \(Subs_1\) and \(substs_2\) both reinterpret a particular item, the reinterpretation from \(Subs_1\) is retained and not that in \(Subs_2\).

- \(newexpr(Old, Subs)\) returns \(New\) if \(Old/\)New \(\in Subs\) and otherwise \(Old\).

Finally, we assume a subsumption ordering over QLF expressions, \(\sqsupseteq\). Basically \(QLF_1 \sqsupseteq QLF_2\) if \(QLF_2\) is like \(QLF_1\) but has (possibly) more meta-variables instantiated. The extent to which QLF could be given a semantics without reference to the subsumption ordering is addressed below. In particular, an additional assignment to meta-variables might serve in its place.

We now give a recursive definition of the valuation relation \(V\) (suppressing the model argument, context and salience arguments, \(M\), Ctx and \(S\)):
1. Constant symbols, $c$: $\mathcal{V}(c, g, \text{Subs}, v)$ iff $F(c) = v$

2. Variables, $x$: $\mathcal{V}(x, g, \text{Subs}, v)$ iff $g(x) = v$

3. Reinterpretation:
   $\mathcal{V}(QLF_1, g, \text{Subs}, v)$ iff $\mathcal{V}(QLF_2, g, \text{Subs}, v)$
   where $QLF_2 | QLF_1 \in \text{Subs}$

4. Merging reinterpretations:
   $\mathcal{V}(QLF: \text{Subs}_1, g, \text{Subs}_2, v)$ if $\mathcal{V}(QLF, g, \text{Subs}_1 \uplus \text{Subs}_2, v)$

5. Application:
   $\mathcal{V}(p(\text{arg}_1, \ldots, \text{arg}_n), g, \text{Subs}, P(\text{ARG}_1, \ldots, \text{ARG}_n))$ if $p(\text{arg}_1, \ldots, \text{arg}_n) \models p'(\text{arg}'_1, \ldots, \text{arg}'_n)$, and
   $\mathcal{V}(p', g, \text{Subs}, P), \mathcal{V}(\text{arg}'_1, g, \text{Subs}, \text{ARG}_1), \ldots, \mathcal{V}(\text{arg}'_n, g, \text{Subs}, \text{ARG}_n)$

6. Abstraction:
   $\mathcal{V}(x^\phi, g, \text{Subs}, h)$ if $\phi \models \phi'$ and $h$ is such that $\mathcal{V}(\phi', g^c, \text{Subs}, v)$ iff $h(k, v)$

7. Conjunction:
   $\mathcal{V}(\text{and}(\phi, \psi), g, \text{Subs}_1)$ if $\phi \models \phi'$ and $\psi \models \psi'$, $\mathcal{V}(\phi', g, \text{Subs}, 1)$ and $\mathcal{V}(\psi', g, \text{Subs}, 1)$;
   $\mathcal{V}(\text{and}(\phi, \psi), g, \text{Subs}, 0)$ if $\phi \models \phi'$ and $\psi \models \psi'$, $\mathcal{V}(\phi', g, \text{Subs}, 0)$ or $\mathcal{V}(\psi', g, \text{Subs}, 0)$

8. Disjunction:
   $\mathcal{V}(\text{or}(\phi, \psi), g, \text{Subs}_1)$ if $\phi \models \phi'$ or $\psi \models \psi'$, $\mathcal{V}(\phi', g, \text{Subs}, 1)$ or $\mathcal{V}(\psi', g, \text{Subs}, 1)$;
   $\mathcal{V}(\text{or}(\phi, \psi), g, \text{Subs}, 0)$ if $\phi \models \phi'$ or $\psi \models \psi'$, $\mathcal{V}(\phi', g, \text{Subs}, 0)$ or $\mathcal{V}(\psi', g, \text{Subs}, 0)$

9. Negation:
   $\mathcal{V}(\text{not}(\phi), g, \text{Subs}_1)$ if $\phi \models \phi'$ and $\mathcal{V}(\phi', g, \text{Subs}, 0)$;
   $\mathcal{V}(\text{not}(\phi), g, \text{Subs}, 0)$ if $\phi \models \phi'$ and $\mathcal{V}(\phi', g, \text{Subs}, 1)$

10. Unscoped term:
    $\mathcal{V}(\phi, g, \text{Subs}, v)$ if $\mathcal{V}(Q'(R', \phi'), g, \text{Subs}, v)$
    where $\phi$ is a formula containing the term, $T_0$, $\text{term}(I_0, C_0, R_0, Q_0, P_0)$,
    and where (a) $\text{newexpr}(T_0, \text{Subs}) = T = \text{term}(I, C, R, Q, P)$
    (b) $S(C, R, \text{Ctz}, Q')$ and $Q \models Q'$,
    (c) $S(C, R, \text{Ctz}, P')$ and $P \models P'$,
    (d) $R'$ is $x^\wedge \text{and}(R(x), P'(x)) \{x/I\}$
    (e) $\phi'$ is $x^\wedge \phi : \{x/T, x/I\}$

202
11. Scoped formula:
\[ V(Scope : \phi, g, Subs, v) \] if \[ V(Q'(R', \phi'), g, Subs, v) \]
where \( Scope, Ctx[I, J, \ldots] \) and \( Scope \supseteq [I, J, \ldots] \)
and \( \phi \) is a formula containing the term, \( T_0, \text{term}(I_0, C_0, R_0, Q_0, P_0) \)

and where (a) \( \text{newexpr}(T_0, Subs) = T = \text{term}(I, C, R, Q, P) \)
(b) \( S(C, R, Ctx, Q') \) and \( Q \supseteq Q' \)
(c) \( S(C, R, Ctx, P') \) and \( P \supseteq P' \)
(d) \( R' \) is \( x^\wedge \text{and}(R(x), P(x)) : \{x/I\} \)
(e) \( \phi' \) is \( x^\wedge[J, \ldots] : \phi : \{x/T, x/I\} \)

12. Form:
\[ V(\text{form}(I, C, R, P), g, Subs, v) \] if \[ V(R(P'), g, Subs, v) \]
where \( S(C, R, Ctx, P') \) and \( P \supseteq P' \)

13. The membership of the relation \( V \) is defined solely by the above.

Given the valuation relation, \( V \), we can now define a valuation function on QLF formulas through a supervaluation style construction [van Fraassen, 1966]. The denotation of a formula \( \phi \) relative to a model \( M \), assignment \( g \), context \( Ctx \), and salience relation \( S \) — \( [\phi]^{M, g, Ctx, S} \) — is defined as follows:

- \( [\phi]^{M, g, \{\}}^{Ctx, S} = 1 \) iff \( V(\phi, M, g, \{\}, Ctx, S, 1) \) but not \( V(\phi, M, g, \{\}, Ctx, S, 0) \)
  (i.e. true under all interpretations)
- \( [\phi]^{M, g, \{\}}^{Ctx, S} = 0 \) iff \( V(\phi, M, g, \{\}, Ctx, S, 0) \) but not \( V(\phi, M, g, \{\}, Ctx, S, 1) \)
  (i.e. false under all interpretations)
- \( [\phi]^{M, g, \{\}}^{Ctx, S} \) is undefined iff \( V(\phi, M, g, \{\}, Ctx, S, 1) \) and \( V(\phi, M, g, \{\}, Ctx, S, 0) \)
  (i.e. true and false under different interpretations)
- \( [\phi]^{M, g, \{\}}^{Ctx, S} \) is uninterpretable iff
  neither \( V(\phi, M, g, \{\}, Ctx, S, 1) \) nor \( V(\phi, M, g, \{\}, Ctx, S, 0) \)
  (i.e. neither true nor false under any interpretation)

**Explanations**

The valuation relation is defined, for the most part, in a non-deterministic way. The use of “if” rather than “iff” in most of the rules means that a QLF expression can often be assigned more than one value, or assigned the same value in more than one way.

The first two rules, for constant symbols and variables are an exception to this. There is only one way of assigning values to constants and variables, and only one value that can be
assigned. These values are provided by the assignment function to constants, \( F \), and the assignment function to variables, \( g \).

The only other deterministic rule is 3, which states that re-interpretations must always be applied when applicable. When a re-interpretation does apply, the QLF expression in question must be evaluated as though it were the QLF expression re-interpreting it.

Non-determinacy in the other rules arises from two sources. First, there may be more than one salient quantifier or property that can be used to instantiate meta-variables in terms or forms. This means that certain rules (e.g. 10, 12, 11) can be applied in more than one way. Second, terms can be scoped at a variety of different points through rule 10. \(^9\) This means that for QLF formulas it will generally be the case that two rules may apply: either scope a term (rule 10) or apply the more specific rule (e.g. 7, 9, 11, 12).

For example, suppose that we have a QLF formula

\[
\text{and}(p(c), q(\text{term}(+i, \ldots))).
\]

Two rules may apply to this formula: the conjunction rule, 7, or the rule for scoping some embedded term, 10. Application of one rule or the other may well end up in assigning different values to the formula, or assigning the same value but in a different way. For this to happen, it is crucial that rules 7 and 10 are expressed with an “if” and not an “iff”.

Going through the remaining evaluation rules, the rule for merging re-intepretations, 4, essentially takes the union of two sets of interpretations, but if there is a clash, the more recent reinterpretation takes precedence.

The rule for application, 5, applies a value of the functor to values of the arguments.

The rule for abstraction states that a lambda abstract \( x^\phi \) non-deterministically denotes a relation, \( h \), between the domain of values for \( x \) and the values of (more specific) versions of the abstract’s body with these values in place of \( x \). Different instantiations of the body will give rise to different relations, and these relations become functions if the body is fully instantiated.

QLF abstraction and application can be shown to support beta-reduction, provided that the functor and argument are independently interpretable.

\[
\mathcal{V}(x^\phi p(x \ldots))(k), g, v) \text{ iff } \mathcal{V}(p(k \ldots), g, v)
\]

However, there are certain cases where reduction does not apply. For example, the application

\(^9\)Rule 10 could be eliminated if we a) allowed partially ordered lists of indices in scope points and a correspondingly more complex subsumption ordering over scope points, and b) ensured that every formula had a scope point.
x^+[i]:p(x) (term(+i,...))

turns out to be uninterpretable, whereas [+i]:p(term(+i,...)) is interpretable. This is because the functor contains an occurrence of the index +i, and none of the QLF evaluation rules provide a way of evaluating the index without the corresponding term also being present in the functor.

A conjunction is true (rule 7) if both conjuncts are true, and is false if either are false (similarly disjunction and negation). Of course, a conjunct may be true under one evaluation but false under another, so that conjunctions may be both true and false.

The rules for conjunction and negation make it possible for the following to be given the value true:

\text{and}(QLF, \text{not}(QLF))

where QLF is some QLF formula containing uninstantiated meta-variables. (Likewise, \text{or}(QLF, \text{not}(QLF)) can be false). This is because the QLF in the two conjuncts may have its meta-variables instantiated differently. An example of this might be every man loves some woman and not every man loves some woman, where the two conjuncts are given different scopings.\(^{10}\)

The rules for terms and scoped formulas are both very similar. They (i) select a term from a formula, (ii) select an (arbitrary) quantifier and referent resolvent that is subsumed by the quantifier and referent arguments of the term (which may be meta variables or instantiated) (iii) discharge all occurrences of the term and its index in the formula and the term's restriction, replacing them by a variable, and (iii) apply the term's quantifier to the discharged restriction and formula. The difference between 10 and 11 is simply that the latter also discharges the head of the scoping list, in this case by removing it rather than by replacing it. (Keep in mind that the discharge and replacement operations take place at the level of the evaluation rules for QLF; they are not applied to the QLF expressions representing natural language meanings themselves).

As with Lewin's scoping algorithm, [Lewin, 1990], there are no constraints built explicitly into the QLF semantics on where a quantification rule for a term may be applied, or indeed on the number of times it may be applied. However, several constraints arise out of (a) the absence of any semantic rules for evaluating isolated terms, term indices or scope lists, and (b) the requirement that a term be selected from a formula so that its quantifier is known.

The emergent conditions on legitimate scoping are

\(^{10}\)This behaviour is justifiable provided that the two QLFs do not share exactly the same meta-variables. But if the conjuncts are identical right down to identity of meta-variables, instantiations to meta-variables in one conjunct ought also to be imposed on the second. Unfortunately, the evaluation rules for conjunction and disjunction do not impose this kind of common instantiation. They allow the possibility of evaluating distinct more specific instances of the two conjuncts. Provided that one does not intend to reason directly with unresolved QLFs, this is a minor technical irritation: the problem does not arise with fully instantiated QLFs. One can avoid this problem in two ways: (a) Thread a separate assignment to meta-variables through the evaluation. (b) Take complete instantiations of the QLF before embarking on evaluation.
1. No term may be quantified more than once: The first application of the quantifier rule discharges the term. Subsequent applications of the rule lower down in the evaluation would fail to select an undischarged term.

2. When a term’s index occurs in a scope list, the quantifier rule for the term must be applied at that point: It must be applied to discharge the head of the scope list, and by (1) above cannot additionally be applied anywhere else.

3. All occurrences of a term’s index must occur within the scope of the application of the term’s quantifier rule: The quantification rule will only discharge indices within the formula to which it is applied. Any occurrences of the index outside the formula will be undischarged, and hence unevaluable.

4. If a term R occurs within the restriction of a term H, and R is to be given wide scope over the restriction, then R must also be given wide scope over H: Otherwise, suppose H is given wide scope over R. Term H will first be discharged, replacing the term, and with it its restriction, in the formula to which the rule is applied. Then the quantification rule for R needs to be applied to the discharged formula, but the formula will not contain an occurrence of the term R, making the rule inapplicable.

The last two constraints have often been attributed to restrictions on free variables and vacuous quantification. The attribution is problematic since open formulas and vacuously quantified formulas are both logically well defined, and without suspect appeal to the syntax of the logical formalism they cannot be ruled out as linguistically ill-formed. By contrast, QLF makes these violations semantically uninterpretable.

Rule 12 selects an (arbitrary) contextually salient or inferred property for the form, which must be subsumed by the form’s resolvent, and applies the form restriction to this property. If the form has been resolved to something contextually inappropriate, the saliency condition on the rule means that the form is unevaluable. The same is true for terms where quantifiers and referents have been resolved to inappropriate values.

Rule 13 makes up for the use of “if” rather than “iff” in the other semantic rules, ensuring that QLF expressions only get assigned the values they do through the application of the semantic rules listed. The set of rules given above are however slightly incomplete; e.g. evaluation rules for identity have not been specified.

5.1.5 QLF Semantics: Comments and Criticisms

5.1.5.1 Compositionality

Evaluation of QLFs is not compositional. This should not be a great surprise. The whole point about unresolved QLFs is that they can have more than one possible semantic value, so

\[11\] Ordinary application is perhaps the wrong thing to use. See section 5.1.5.5.
one could hardly expect there to be a function from (the structure of) the QLF onto semantic values.

Faced with the possibility of more than one semantic value for an expression, compositionality can be preserved by jacking everything up a level, so that evaluation delivers the set of possible values. This is essentially what the supervaluation construction for the valuation function \[ [\phi]^{M,g}_{Ctx,S} \] does. This is a slightly different approach from that adopted in relational treatments of composition, e.g. [Poesio, 1994b; Muskens, 1989]. There, sets of values are build up and combined in the main recursion of the evaluation. The semantics we have given for QLF in effect backtracks over possible evaluations for the QLF, only collecting the results together when doing the top level supervaluation.

This way of presenting the QLF semantics shows how unresolved QLFs correspond to a set of semantic compositions (i.e. ways of evaluating the QLF by piecing its components together). We might say that in a relational treatment of composition, the logical formula gives a complete characterisation of a composition that is somehow partial or underspecified. Whereas for QLF we give a partial or underspecified characterisation of the intended, complete composition.

However, this is not all there is to be said about the non-compositionality of the QLF evaluation relation. For if it was, we would expect evaluation of fully resolved QLFs to be compositional. But the quantification rule \[ \lambda \] for example, does not assign a value to a QLF formula on the basis of the values assigned to its immediate subexpressions. Instead, it relies on the formula containing within it some term expression, but not necessarily as an immediate subexpression, substitutes a variable for the term and its index, and then evaluates another expression built from the term and the formula plus substitutions.

The non-compositionality of the rule arises from two sources (i) the need to find a (non-immediate) subterm in the formula, and (ii) the need to substitute variables for terms and indices. With regard to the second point, there is a parallel problem in substitutional semantics for quantification in first order logic, though in the reverse direction: the semantics relies on substituting constants and other terms for variables. This problem is avoided in objectual treatments of quantification by means of a variable assignment function.

This would correspond to replacing our set of substitutions by an assignment of values (rather than other QLF expressions) to terms and indices. However, introducing a term/index assignment would not avoid having to look inside a formula for occurrences of a term. Here it is possible that some variant of a Cooper storage mechanism could be included in the semantics for QLF.\footnote{Such a storage mechanism would not be the mechanism by which the scopes of quantifiers are resolved in QLF, which is just the instantiation of scope constraints. It would provide the means for interpreting scope resolutions once made.} This would necessitate maintaining a list/store of terms as a separate argument to the valuation relation (along with variable assignments etc). Instead of searching the QLF expression for terms, the semantic rule evaluation rules would search the store. As with the idea of introducing an assignment to terms and indices, the details of this have not been worked out. Of course, a storage mechanism like this would not eliminate non-compositionality
But it is far from clear that this non-compositionality is cause for concern. First, values are assigned to resolved and unresolved QLFs in a fully systematic way. Second, if QLFs are viewed as providing a description of a semantic composition, there is no reason why the structure of the composition should exactly parallel the structure of the description.

This second point is question-begging, since we have not demonstrated that QLFs are descriptions of compositions. At best, the non-compositionality combined with a systematic assignment of values is merely evidence that such a view is plausible (though more evidence is forthcoming below).

5.1.5.2 Uninterpretability

A peculiarity of QLF, when compared with other formal languages, is the possibility of having well formed expressions that are nevertheless uninterpretable (semantically ill-formed). Expressions are uninterpretable either when a term or its index is not discharged by the quantifier rules 10 and 11, or when term and form referents and quantifiers have values not included within the salience relation $S$.

Uninterpretable QLFs make perfect sense if they are construed as well-formed descriptions of illegitimate semantic semantic compositions; in much the same way that a square circle is a well-formed description of an impossible object.

Semantic composition involves determining the meanings of constituents and then composing them together in a certain way. A partial description of a composition may either (a) incompletely specify the meanings of certain constituents, and/or (b) incompletely specify how those meanings are to be composed. Uninterpretability can arise completing the specification of either of these points in an illegitimate way.

When forms or terms are resolved to have inappropriate quantifiers or referents, the QLF assigns an illegitimate meaning to a constituent. One of the reasons for including so much syntactic information in the categories of terms and forms is that there are strong syntactic and lexical constraints on what the legitimate range of meanings is for any expression. When performing reference resolution, care is taken to ensure that only legitimate, meaningful instantiations of the meta-variables are made. However, the QLF semantics builds in no such assumption. One can instantiate the meta-variables to anything one likes. But if the values are not covered by the saliency relation $S$, the QLF becomes uninterpretable.

Uninterpretability through failure to discharge terms and their indices by scoping reflects a description of a composition that attempts to apply operations of semantic composition in an incorrect way. As we will see below, this closely parallels Pereira’s [Pereira, 1990] proof-theoretic account of semantic composition, where quantifier assumptions are introduced and discharged in an illegitimate way.
Salience Relations

One source of non-determinism in the semantics for QLF is the fact that often more than one rule can be applied in more than one way to evaluate a given QLF expression. The other source of non-determinism lies in the relational nature of the salience relation $S$; unresolved terms and forms may have more than one possible resolution.

The salience relation provides the formal basis for reference resolution. The reference resolution process, which is really part of the (natural language) syntax-semantics interface, is discussed below in section 5.2. Here we will just say something about the formal nature of this relation, and raise some potentially thorny questions about what kinds of objects form the arguments to the relation.

First an example. Given a category corresponding to a singular male pronoun (he) and a context $C$ in which the entity john1 is salient, we might have

- $S(<type=pro, num=sing>, male, C, X^X=john1)$
- $S(<type=pro, num=sing>, male, C, exists)$

which means that one way of resolving a term like

term(+i, <type=pro, num=sing>, male, -q, -r)

in the context $C$ is as

term(+i, <type=pro, num=sing>, male, exists, X^X=john1)

The relation $S$ can be seen as forming part of the model against which QLFs are evaluated. In practice, the relation will be defined in terms of a series of axioms relating to the context, e.g.

- $S(<type=pro, num=sing>, Restr, C, X^X=k)$ if
  - salient-object($C, k$), &
  - singular-object($C, k$), &
  - restriction-applies($Restr, C, k$)

which says that $k$ must be an object that is salient in context and singular, and to which the restriction of the term applies.

Expressions or Denotations? Does the saliency relation $S$ relate QLF expressions or denotations of QLF expressions. That is, when $X^X=john$ occurs as a referent argument to $S$,
does it stand for the property of being identical to the individual named by \textit{john}, or does it stand for itself, i.e. the QLF expression $X^X=\text{john}$?

Given the possibility of referents like $X^X=+i$, where $+i$ is a QLF index, we seem drawn to the conclusion that it is QLF expressions that occur as arguments to the saliency relation. Taken in isolation an expression like $X^X=+i$ is uninterpretable, since it contains an undischarged index.

An informal gloss of the meaning of $x^x=+i$ is ‘the property of being identical to whatever is referred to by the term/noun phrase identified by $+i$’. In general, we may build up the meaning of expressions by explicit reference to the way that the meanings of other expressions are composed. This suggests that rather than simply viewing $x^x=+i$ as uninterpretable, it has a conditional denotation, meaning that in the absence of a wider context where the term $+i$ is evaluated the expression is uninterpretable, but within such a context it does have an interpretation.

That is, rather than QLF expressions, the values covered by the saliency relation are descriptions of bits of semantic composition. What is important, however, is not how the composition is described, but what is described. In practice, though, it does not real harm to regard the arguments to the saliency relation as being QLF expressions.

\textbf{What is Context?} One thing context should contain, given the preceding discussion, are QLF expressions corresponding to the referent arguments of the saliency relation. As the variety of expression types is widened, more NL resolution phenomena are covered. A rough summary is:

- constants: intersentential pronouns
- predicate constants: compound nouns, prepositions
- quantifiers: vague determiners
- indices: bound variable, intrasentential pronouns
- predicates built from NP restrictions: one-anaphora
- predicates built from previous QLFs: intersentential ellipsis
- predicates built from current QLF: intrasentential ellipsis

Access to the expressions is furnished by making the QLF of the utterance being interpreted and the QLFs of previous utterances part of context.

In addition, the context needs to support certain kinds of inference, e.g. the ability to determine whether an object satisfies a term’s restriction. In this respect, we really want the context to behave like a model. That is, the context should be a (partial?) model containing the information that is currently known or believed about the model $M$ used in evaluating QLF expressions.
5.1.5.4 Subsumption and Monotonicity

Computationally, it is useful to have two forms of monotonicity acting in tandem. Semantic monotonicity, which monotonically reduces the number of readings by eliminating models in which an expression can be true/false. And syntactic monotonicity (subsumption) that reduces the number of readings by further instantiating QLF expressions. This parallelism is built into the QLF semantics by invoking a subsumption relation, \( \sqsubseteq \).

To show that subsumption and monotonicity coincide, we need to establish that

\[
\text{If } QLF_1 \sqsubseteq QLF_2, \text{ then } \{ m : V(QLF_1, m, g, v) \} \supseteq \{ m : V(QLF_2, m, g, v) \}
\]

(Strict coincidence would require that the relation also hold in the reverse direction, which would demand that for every possible semantic refinement there is a corresponding syntactic one; this is an implausibly stringent condition.) Given the use of the subsumption ordering in every non-deterministic semantic evaluation rule, it is trivial to establish parallelism. The semantics for unresolved QLF expression essentially involves considering possible resolutions of meta-variables, as given by the saliency relation \( S \), and evaluating the QLF with each of these resolutions in place. Consider a QLF expression, \( QLF_1 \), with an unresolved meta-variable, \( ?X \), and another expression \( QLF_2 \) that differs in having instantiated the meta-variable. Suppose that \( ?X \) has been instantiated to something that is not contextually permissible (i.e. not a possible value for \( S \)). Then \( QLF_2 \) will be uninterpretable and have no values in any model, thus trivially satisfying the monotonicity coincidence. So suppose that \( QLF_2 \) does instantiate \( ?X \) to a legitimate resolvent. This resolution will be one of the cases considered in evaluating \( QLF_1 \) — so the values assigned under the resolution will also be covered by the evaluation of \( QLF_1 \).

Given this result, we can easily extend it from the valuation \( V \) to the valuation \( \llbracket \ldots \rrbracket^M_g \) for formulas. That is

\[
\text{If } QLF_1 \sqsubseteq QLF_2, \text{ then } \llbracket QLF_1 \rrbracket^M_g \geq \llbracket QLF_1 \rrbracket^M_g
\]

(where: undefined \( \geq \) definitely true;
undefined \( \geq \) definitely false;
definitely true \( \geq \) uninterpretable;
definitely false \( \geq \) uninterpretable)

That is, if an expression is definitely true in a model, further specification cannot make it false (although it can still make it uninterpretable).

None of this says anything about the nature of the QLF subsumption ordering. We can define a subsumption ordering over QLF expressions which requires identity (up to alphabetic
variance of logical variables) between instantiated parts of the QLF expression, and unification style subsumption between meta-variables. In other words, QLF subsumption is more or less subsumption as defined by Prolog unification, except that only meta-variables may subsume expressions that are non-identical after renaming of logical variables.

There is a question as to whether this unification-style subsumption is optimal. If scope constraints were represented as partial orders of term indices, rather than totally ordered lists as at present, a more sophisticated form of subsumption would be required. Likewise, unification-style subsumption is sensitive to such things as the order of conjuncts in a conjunction, where semantically this makes no difference. However, given the current QLF notation and a restriction to standard orderings of conjuncts etc in the arguments to the salience relation, unification-style subsumption is both adequate and computationally efficient.

Since the subsumption ordering concerns only instantiations to meta-variables, we could in fact do away with subsumption over entire QLF expressions. Instead we could revise the semantics to include a partial assignment to meta-variables, and define a subsumption ordering over these assignments. Instead of evaluating a QLF expression by considering a more specific instance of the expression, one would evaluate it relative to a more specific assignment. Threading of input and output assignments through the evaluation would also ensure common instantiation of identical meta-variables, which is not currently guaranteed.

5.1.5.5 Substitution and Application

Failures of Beta-Reduction We noted earlier the non-equivalence between

\[ x^+[i]:p(x) \ (\text{term}(+i,\ldots)) \]

and

\[ [+i]:p(\text{term}(+i,\ldots)) \]

This poses some problems for the advertised treatment of form resolutions: a contextually salient property to which the form restriction is applied. There are times when it would be useful to have the kind of equivalence above. It may be preferable to treat the connection between a form restriction and its resolvent through explicit substitutions (of the kind used for replacing terms and indices by variables). This needs further work, though.

One possibility is to introduce some further notation looking a bit like abstraction, but framed in purely substitutional terms. Application of these substitutional abstracts simply involves substituting abstracted variables by the arguments they are applied to, and then evaluating the result. Representing this substitutional form of abstraction as a backwards slash we might have
Substitutional Application:

\[ V((x \phi)(QLF), g, Subs, v) \]

\[ \phi \equiv \phi', QLF \equiv QLF', \text{ and } V(\phi': \{QLF'/x\}, g, Subs, v) \]

This is unlike real lambda abstraction, where the abstract is assigned some semantic value (a relation or function from objects to results), and where this semantic value is applied to the values of its arguments. Substitutional abstracts only have a meaning in the context of an application, and none independently of it.

This kind of substitutional application is also needed to make sense of the way that QLFs are built up on the basis of syntax. Moore [Moore, 1989] argues that instantiating unification variables to logical expressions in a unification-based syntax-semantics interface corresponds to a form of beta-reduction. However, this cannot be the case if the variable is instantiated to a QLF term, since terms are not independently interpretable: the reduction fails. The instantiation therefore needs to be seen in more directly substitutional terms.

Substitution and Recomposition  A lambda abstract is a function from the meaning of its argument to some other meaning built around it. We can look at substitutional abstraction in similar terms, expect that instead of constructing meanings it constructs compositions. That is, it is a function from the semantic composition of its argument to some other composition built up around it.

If this view is correct, then substitution and substitutional abstraction is a more appropriate tool for semantic interpretation than functional application and lambda abstraction.

Substitutions can profitably be employed to deal with ellipsis (Section 5.5.7 in Deliverable 9). The idea is that an ellipsis acts as an instruction to recompose the meaning of the antecedent so as to incorporate the material explicitly contained in the ellipsis. The alterations to the original composition needed to include the elliptical material can easily be expressed by means of using substitutions.

5.1.6 Comparisons and Alternatives to QLF

Categorial Semantics

Connections and similarities between monotonic semantics and Montague semantics, aspects of discourse representation theory and Nerbonne's constraint-based semantics have already been noted. But the most interesting comparison is to Pereira's categorial semantics [Pereira, 1990], even though his is an avowedly non-monotonic treatment of semantic interpretation.

Put briefly, in categorial semantics, semantic evaluation is represented as deduction in a functional calculus that derives the meanings of sentences from the meanings of their parts.
Considerable emphasis is placed on the nature of these semantic derivations, as well as on the final results of the derivations (the ‘logical forms’ of sentences).

One significant advantage of this approach is that constraints on legitimate scoping emerge naturally from a consideration of permissible derivations of sentence meaning, rather than arising artificially from syntactic constraints imposed on logical forms. Derivations involving quantified terms first introduce an assumption that allows one to derive a simple term from a quantified term. This assumption is later discharged by the application of a quantifier. Conditions on the appropriate introduction and discharge of assumptions in natural deduction systems impose restrictions on the way that quantifiers may legitimately be applied. For example, a quantifier assumption may not be discharged if it depends on further assumptions that have not themselves been discharged. This prevents the occurrence of free variables in logical form, but without appeal to the syntax of logical form.

The discharge of terms and term indices when evaluating QLF closely parallels the discharge of quantifier assumptions in categorial semantics. Indeed, the terms and the indices are precisely the assumptions introduced by quantified expressions, and which need to be discharged. Furthermore, the different orders in which quantifier assumptions may be discharged in categorical derivation correspond to the choices that the quantifier rules permit for discharging quantified terms.

Where monotonic interpretation and categorical semantics part company is on the degree of explicitness with which semantic derivations are represented. In categorial semantics, derivation is a background process that builds up logical forms, but is not explicitly represented in the semantic formalism. By contrast, the annotation of QLFs with scope lists provides an extra level of information about how the derivations proceed. In particular, they indicate which evaluation rules should be applied where.

QLF thus provides a (usually partial) specification of a semantic derivation, showing (a) what the initial ‘premises’ are (roughly, lexical meanings, although these too may only be partially specified), and (b) the rules by which the ‘premises’ are combined. QLF resolution amounts to further instantiating this specification. This view of QLF can be contrasted with Logical Form as it is normally understood, which represents the results of carrying out a semantic derivation.

**Dynamic Semantics**

Dynamic semantics treats meaning as transition between contexts or information states. The order in which transitions are made can be significant, contributing a degree of non-monotonicity to the semantics. At first sight, the undeniable order-sensitivity of context change seems to sit uneasily with the monotonicity of monotonic semantics.

But the conflict is illusory, since the monotonicity and non-monotonicity exist on different levels. For dynamic semantics, monotonic interpretation would amount to making decisions about which transitions to take when, but would not involve putting those decisions into
action. The monotonicity in monotonic interpretation thus refers to the way in which alternative derivations of sentence meanings may be chosen, but not to the semantic effects of those sentence meanings.

This needs to be qualified somewhat, since resolution depends on context and meaning updates context. But it is not necessary to actually update context to determine what the effects of the update would be. And it is only the latter that is required for resolution. What this means in practice is that context in resolution typically features through ‘call by name’ rather than ‘call by value’.

For instance, semantic evaluation of a sentence containing an indefinite may update context to include some new entity. After context is updated, anaphors can refer directly to the new entity (call by value). But if we do not choose to update the context straight away, we can still refer to whatever entity would be added by evaluating the indefinite (call by name). The use of term indices in coindexing for anaphora (Section 4.5.6 in Deliverable 9) is one way of implementing this call by name use of context.

This being said, much work needs to be done in seeing how smoothly monotonic interpretation can be integrated with dynamic semantics.

### Alternatives to QLF

The first part of this document represents an attempt to force QLF into the mould of something that describes semantic compositions. QLF was never designed with this explicitly in mind; the mould-forcing is a post hoc attempt at rationalisation.

If one were to start again from first principles in monotonic interpretation, it is possible that a different kind of semantic formalism would be chosen. For example, a language explicitly geared to describing semantic derivation trees, e.g. as a set or conjunction of constraints on the derivation tree that gets monotonically enlarged through resolution. Or more speculatively, one might try to apply constructive type theory to the task of describing semantic compositions.

QLF represents a curious mixture when it comes to describing semantic compositions. The basic-predicate argument structure of composition is built into the structure of the QLFs as constructed from the syntactic analysis of the sentence. This contrasts with an unstructured description of predicate-argument structure one might obtain in a language for describing derivation trees. Other aspects of the composition, such as scoping, are not represented in such purely structural terms in QLF.

However, it is possible that QLF constitutes a sensible practical compromise. The structuring of QLF around the basic, syntactically given, predicate-argument may well represent the computationally most economical way of presenting this information. The structure guides processing in a way that inferencing on a flat set of constraints could not. And certainly, QLF has proved its worth in computational application.
So, for exploring the theory of monotonic interpretation it may well be advantageous to provide a flatter, less structured formalism in place of QLF. But for implementation purposes, something like QLF may still be desirable.

5.2 Syntax-semantics Interface

The syntax-semantics interface is usually taken to refer to the mapping from the syntactic structure of a sentence to an initial, non-contextually resolved semantic representation. In monotonic interpretation, contextual resolution is as an important part of the semantics as this initial mapping. In this section, we will therefore construe the syntax-semantics interface more widely to cover both syntactically- and contextually-derived semantics.

This fits in with a view of semantic interpretation as being a process of mapping a string of words onto its (literal) meaning, and where a variety of sources of information contribute to this mapping. The syntactic structure assigned to the word string provides one set of constraints on the string’s meaning, and contextual factors provide another.

5.2.1 Syntax-Semantics Rules

Unresolved QLF expressions can be built up on the basis of syntactic analyses of sentences using techniques that are familiar from unification-based syntax-semantics interfaces. We will therefore not say much about this, except to raise a question about the status of unification variables as they occur in semantic rules. To what extent do these variables correspond to QLF meta-variables, whose possible instantiations are not given by some contextual salience relation, but by the rules of the grammar?

It is also important to point out that, given the way abstraction is defined in the QLF semantics, it is not possible to follow [Moore, 1989] in viewing unification in the semantics as corresponding to functional application. For very often, we will want to unify variable feature values with terms or their indices, and as pointed out in the previous sections beta-reduction does not apply to such uninterpretable objects. As perhaps with the semantics for forms, some more directly substitutional kind of abstraction may well be required to make sense of what is going on here.

5.2.2 Reference Rules and Resolution

In practice reference resolution amounts to instantiating meta-variables in QLFs derived from the initial syntax-semantics mapping. As pointed out earlier, the salience relation $S$ forms the basis for doing this. Part of a grammar and semantics for natural language therefore involves specifying what the salience relation is. This can be done via a set of resolution rules, which essentially take the form indicated previously, e.g.
In practice, predicates like most-salient-object or restriction-applies might be implemented as prolog procedures.

5.2.2.1 Getting the correct resolution

It is not sufficient to produce all possible resolutions for a sentence. In most contexts there will be one correct resolution, and a variety of implausible resolutions.

Reference resolution can be portrayed as solving the following equation

$$\text{Context} \land \text{Assumptions} \models \text{QLF} \equiv \text{RQLF}$$

where QLF is an unresolved QLF and RQLF is a more instantiated version of it. The role of the Assumptions will be seen in due course.

To give a simple illustration, suppose that we have a sentence like *He slept*. Let us assume that the category on the term corresponding to *he* says something like ‘the referent property is that of being identical to a male object that is most salient in context’. Moreover, suppose that John is the only contextually salient male. Under these circumstances, we can show that

$$\text{Context} \models \\text{slept}($$

$$\text{term}($$

$$\text{(+h,\langle type=pro,lex=he\rangle,\text{male},?Q,?H\rangle)}$$

$$\equiv$$

$$\text{slept}($$

$$\text{term}($$

$$\text{(+h,\langle type=pro,lex=he\rangle,\text{male,exists},x^x=\text{john}\rangle))}$$

The assumptions come into force when, for example, there is more than one salient male in context. To select a particular male as the pronoun resolvent, we can make an assumption that he is in fact the most salient male. These assumptions are made at a cost. By making different assumptions, we arrive at different resolutions of the QLF, and these alternatives can be ranked by the cost of the assumptions made. Assumptions can also be used as a way of updating context to reflect resolutions that have been made.

5.2.3 Reversibility

The general statement of the resolution problem,
Context ∧ Assumptions ⊢ QLF ≡ RQLF

is not direction specific. In principle, we can use it derive an unresolved QLF that is equivalent to some other resolved QLF in a given context and making certain assumptions. This can be used to provide informative paraphrases of resolutions.

In the example above, we might have

\[
\text{Context} \models \text{slept(}\langle +n, \text{type}=\text{name}\rangle, x \ → \ \text{nameof}(x, 'John'), ?Q, ?N) \equiv \text{slept(}\langle +h, \text{type}=\text{pro}, \text{lex}=\text{he}\rangle, \text{male}, \text{exists}, x \ → \ x = \text{john})
\]

That is, in a context where John is, or is assumed to be, the most salient male object, the sentence *He slept* is equivalent to *John slept*.

In practice, QLF resolution has not yet been carried out in fully reversible way for anything but a very simple variety of cases. One way of solving resolutions in the reverse direction is to replace the category and restriction of a term or form with meta-variables while keeping the resolvents instantiated, and using the salience relation \(S\) to suggest possible values for the category and restriction. But this technique has limited applicability, since uninstantiating the restriction often leads to a loss of information about the meaning of term of form. With referents like \(x^x = \text{john}\) enough information is retained for reverse resolution, but in other cases the restriction needs to remain at least partially instantiated.
Bibliography


