Erasmus Mundus European Master in
Language & Communication Technologies (LCT)
UFR Mathématiques et Informatique
Université Nancy 2
Department of Computational Linguistics
Saarland University

Identification of Accessibility Constraints in Discourse

Thesis Submission for a
Master of Science in
Cognitive Science & Applications (SCA)
Specialized in
Traitement Automatique des Langues
(Natural Language Processing)

Sai Qian
Defense on
30th, June, 2009

Supervisor: Maxime AMBLARD
Co-Supervisor: Sylvain POGODALLA
Co-Supervisor: Manfred PINKAL

Composition of Jury:
Maxime AMBLARD
Patrick BLACKBURN
Guy PERRIER
Fabienne VENANT

Acknowledgement

Firstly, I really want to show my gratitude to my supervisor: Maxime Amblard. Since last December we had the first meeting about the topic, he has always been helping me. Through discussing together with him, I obtained lots of new ideas and motivations in this area. Also thanks to his precious time in reviewing the report, I avoided numbers of silly mistakes in the pre-version of the report.

I shall thank Patrick Blackburn and Guy Perrier. As the main persons in charge of the Erasmus Mundus LCT program in Nancy, they provided our students great assistance through the whole year, both in study and personal life, especially during the hardest time when we first arrived in Nancy. Thanks to them, we shall be able to always study in a pleasant atmosphere. Also thank my co-supervisor in Nancy, Sylvain Pogodalla a lot. He offered me a great deal of technique helps in ACG toolkit implementation. At the same time, I want to thank all the teachers in Nancy, including Fabienne Venant, Philippe de Groote, Bruno Guillaume, Carlos Areces, Claire Gardent and Kamel Smaili.

I want to take the opportunity to thank all my first year teachers in Saarbruecken, Germany, especially Valia Kordoni and Manfred Pinkal. Valia is the first person who leads me into the door of NLP and offers me lots of encouragements, where I find my interest really lies in. Professor Pinkal is my co-supervisor in Saarbruecken for this thesis, who not only gives me a great perspective in semantics last year in Germany, but also guides me to think deeper and broader in my topic through the meeting this April.

Finally, I shall thank my parents. Although I could not meet them during the two years' studying abroad, they are always the persons who support me most, no matter how hard the time it was. I love you!

Abstract

Since Richard Montague developed his theory: Montague Grammar, in [Mon73], logicians and linguists began to notice the idea that describes natural language in formal logic. However, in the early 80s, several dynamic semantic approaches came up to cope with Montague Grammar's inability in discourse processing. Those dynamic approaches, although are able to explain some linguistic phenomena to some extent, have their own disadvantages compared with traditional Montague Semantics. There is a recent work [dG06] trying to express the dynamics of discourse in a Montagovian style. In this thesis, various accessibility constrains in dynamics will be investigated and solutions for handling each constraint under the new framework is respectively proposed.

Key words:
Compositionality, Montague Grammar, Discourse Representation Theory, Accessibility Constraint, Anaphora Resolution, Dynamics, ACG

Contents

1 Introduction

The meaning of a complex expression is determined by the meanings of its constituents and the structure they are combined.

—Frege's Principle

Frege's Principle, also known as the principle of compositionality, is one important concept in areas such as philosophy, linguistics, logic and computer science. Richard Montague, a pioneer in modern semantics, is one of the first people trying to use this idea for semantically interpreting natural language. He eliminates the gap between natural language and formal logic, lifting semantics to a new level. In Montague Grammar, the meaning of a sentence depends on its truth condition. In other words, to know the meaning of a declarative sentence is to know the world that makes it true. The semantic interpretation, which evaluates the truth conditions, could be constructed through semantic values of each elements in the sentence and a series of semantic rules. Thus the compositionality is realized through simply typed lambda calculus [Chu40].

However, in the early 1980s, linguists and philosophers found that Montague Grammar was not designed to construct logical formulas for discourses. In other words, multi-sentence processing could not achieve similar success as single sentence treatment. For example, in Montague Semantics, discourse like "*John loves Mary. He wants to invite her for dinner*" would encounter the so-called "cross-sentential anaphora" problem, because pronouns "*he*" and "*her*" will both be considered as free variables when two sentences are combined, which makes it invalid to match them with their antecedents "*John*" and "*Mary*" in the first sentence. Also, Montague Semantics fails in explaining the so-called "donkey sentences", such as "*Every farmer who owns a donkey beats it*", in which the indefinite expression "*a donkey*" should be treated as a universal quantification, rather than the conventional existential quantification. In order to cope with the problems of Montague Semantics, some approaches departuring from a dynamic point of view appeared, such as Kamp's Discourse Representation Theory (DRT) [Kam81], Groenendijk & Stokhof's Dynamic Predicate Logic (DPL) [GS91]. Those dynamic solutions view the content of a discourse as a relation between contexts. The meaning of the discourse is a set of states, each new piece of context has the potential to update the current state, which is completely different from the model theoretical feature in Montague Grammar. Although the dynamic approaches can explain some interesting linguistic phenomena, like the above "cross-sentential anaphora" problem and "donkey sentences", which Montague Grammar fails to, they are not perfections. For instance, DRT loses compositionality [1] and could not handle quantification and coordination phenomena as elegant as in Montague Grammar; DPL assigns new meanings to the traditional logic operators, which somehow tampers the systematic type theory in [Chu40].

As a result, some semanticists raised the question of how to combine Montague Semantics with dynamic approaches. A recent work [dG06] proposes to use standard computational techniques and the classical typed lambda calculus to expressed DRT, especially the concept of accessibility of DRT, in a non representation way. The successfully constructed accessibility constraints can serve to aid another challenging topic in natural language processing: Anaphora Resolution. In this thesis, the accessibility constraints in various linguistic phenomena will be investigated.

[1] This point will be explained in section 3.3

Some new operators and mechanisms, which model and satisfy these constraints based on the new approach, will be proposed.

The rest of the thesis is organized as follows: in Chapter 2, I will introduce anaphora resolution, including the basic concepts, terminologies, the factors and some approaches in this area. In Chapter 3, Kamp's Discourse Representation Theory (DRT) will be discussed. Afterwards, some examples will be presented to explain one important concept in DRT: Accessibility. Chapter 4 mainly focuses on the new proposal that expresses the ideology of DRT in Montague Semantics style [dG06], which will be followed by some simple examples and the process to obtain their interpretations. Chapter 5 is the main part of the thesis. We will select several linguistic phenomena such as conditional phrases, negation sentences, binding theory and treatment of proper names. The accessibility constraints in those cases will be studied and solutions for each of them will be proposed. In Chapter 6, I will present the implementation of the above accessibility constraints in Abstract Category Grammars (ACG). The last chapter is the conclusion.

2 Anaphora Resolution

Being a very difficult and challenging task in Natural Language Processing, anaphora resolution always attracts lots of researchers, who are searching for better and more delicate approaches on this topic. In this chapter, I will briefly introduce the basic concepts, terminologies and the general techniques in anaphora resolution.

2.1 Concepts & Terminologies

The word "anaphora" can be referred back to the Greek word "$\alpha\nu\alpha\varphi o\rho\alpha$", which focuses on words by repeating them in the previous context. The entry (word, concept, etc.) being referred back to in the previous contexts is called *antecedent*, the related reference in the current clause is called *anaphora*. In the following example:

(1) *John loves Mary. He wants to invites her for dinner.*

"*John*" and "*Mary*" are antecedents respectively for pronominal anaphora "*he*" and "*her*". Actually, there exists a co-referential relation between the anaphora and its antecedent, both of which point to the same object in the model of the discourse. The task of anaphora resolution is to find out the co-referential chains in the discourse. In other words, to extract the correct pairs among numbers of anaphora and antecedents. Generally speaking, there are two different ways to classify anaphora. On one hand, from the "position" point of view, anaphora can be distinguished as *intra-sentential* (where the appearances of anaphora and antecedent appear within the same sentence) and *inter-sentential* (where the anaphora and its antecedent appear in different sentences). For example, in Example 1, both "*he*" and "*she*" belong to the latter case, while in the following example:

(2) *John went out with his dog.*

"*his*" is a *intra-sentential anaphora*, which forms a co-referential relation with "*John*". On the other hand, from the "appearance" point of view, anaphora can be classified into *NP anaphora*, *VP anaphora*, *S anaphora* and *One anaphora*. *VP* and *S anaphora* refer to those cases in which the antecedents are verb phrases or complete sentences. For instance, in Example (3) and (4),

(3) *John went to the park. Mary did too.*

(4) *It is rainy outside. That is too bad.*

"*went to the park*" (VP) and "*It is rainy outside*" (S) are respectively the antecedent for "*did*" and "*that*" in the following sentences. While for *One anaphora*, it means the form of the anaphora is word "*one*", which is a very common usage for "*one*" in English. Example (5) is a illustration for *One anaphora*:

(5) *John ate an apple after lunch. Mary also ate one.*

The most widespread and investigated type of anaphora is *NP anaphora*, in which people usually focus on the pronominal anaphora and definite NP anaphora. Pronominal anaphora refers to those sentences where anaphoric pronouns appear, such as "*he*", "*she*", "*it*" and "*they*" in English. Example (1) is just this case. Definite anaphora is another more complicated *NP anaphora*. The anaphora start with definite determiner "*the*", pointing back to some pre-existing semantically related concepts. One point to notice is that for definite NP pronoun, the anaphora might not be equal to the antecedent, it also works if they are only semantically related. Example (6) and (7) can explain the difference between various definite NP anaphora.

(6) *John went into a restaurant. He had dinner in the restaurant.*

(7) *John went into a restaurant. He talked to the waitress.*

In (6), "*the restaurant*" in the second sentence exactly refers to "*a restaurant*" in the first; while in (7), it is also obvious that "*the waitress*" refers to the one working in the restaurant which John goes into, although no waitress has been introduced before.

2.2 Factors for Anaphora Resolution

When people began to notice the phenomenon of anaphora, they found it is not that difficult for human being to interpret correctly the anaphora in unambiguous sentences, even for complex situations. However, it is not easy to let computer acquire such skill. So what exactly determines which antecedent an anaphora links to?

From a theoretical point of view, all the pre-existing concepts, entries could be the potential candidates for future reference. But naturally, only one or several candidates could be the correct antecedent. The resolution process usually depends on a series of linguistic factors, such as morphological features (gender, number agreement), syntactic features (binding theory, syntactic parallelism), semantic and pragmatic features (semantic consistency, semantic parallelism, centering). Some of the factors can block the improper candidates, narrowing the number of potential candidates; the others can help rank all proper candidates based on their

possibilities to be the correct one. As a result, we can divide them into *constraint-based factors* and *preference-based factors*.

2.2.1 Constraint-based Factors

The most well-known constraint factor is the gender, number agreement. Only the antecedent with the same gender, number as the anaphora could be a possible potential candidate. Take Example (1), the agreement constraint can simply block "*John*" to be the antecedent for "*she*", and "*Mary*" to be the antecedent for "*he*". It plays an important role especially in some "gender-active" languages, such as French, German.

On the semantic level, consistency is another important constraint factor. Look at Example (8).

(8) *John bought a hamburger and a basketball on the way back home. He ate it afterwards.*

In this example, only "*hamburger*" can be the antecedent for "*it*" in the second sentence. Because of the semantic meaning of verb "*eat*", its object should only be something eatable, so "*basketball*" is out of consideration in this case. Although the complexity will grow exponentially in some more complicated situations, especially when we take into account the semantic senses for lots of lexical entries, consistency is still considered as one effective factor.

2.2.2 Preference-based Factors

Unlike constraint factors, preference factors play a different role in anaphora resolution system. Good preference factors definitely can help us find the better resolution. Parallelism (both syntactic and semantic) is one of them. The following example is a good illustration:

(9) a. *John gave Mike a book. He also gave Mary one.*
b. *John gave Mike a book. Mary also gave him one.*

Naturally, people will prefer to interpret "*he*" as "*John*" in (9-a) and "*him*" as "*Mike*" in (9-b), although both "*John*" and "*Mike*" are possible candidates (no constraint factor can block either of them) for being the antecedent.

Also some other theories like *centering*, *topic shifting* are also used as preference factors, which server to make at least some distinctions between all possible candidates. They will not be discussed in detailed here, more information could refer to [Hir81].

2.3 Approaches to Anaphora Resolution

Over the past several decades, scientists and researches have tried various approaches for anaphora resolution problem. Some traditional ones involve only pure linguistic knowledge, from syntactic structure, syntax-semantic interface to pragmatic level. Afterwards, more approaches based on statistics and probability theory come up, which prove to have a relatively good result. However, most of them require a large amount of training data, which needs

an intensive laboring job from linguists and philosophers. Recently, some computer scientists establish a knowledge poor paradigm for anaphora resolution, which is mainly based on some machine learning algorithms. In this report, we will not focus on any specific approach, however, more information concerning the detailed processes and results of various approaches could be found in [Rus99].

Generally speaking, the progress on anaphora resolution so far is tremendous, especially in some approaches where people combine linguistic knowledge with statistic model. However, questions like the relationships between different resolution factors and each factor's effect over different genres of materials are still open topics in future.

3 Discourse Representation Theory & Accessibility

There is in my opinion no important theoretical difference between natural languages and the artificial languages of logicians.

—Richard Montague (1970)

Montague Semantics, as one of the most prominent approaches in Natural Language Semantics, is mainly based on type theory and lambda calculus, combining with First Order Logic (FOL). In Montague Semantics, the meaning of a sentence is represented by its FOL formula translation. To know the meaning is to verify the truth condition, namely the world that makes the formula true. Words of different categories (syntactic) are assigned different forms of λ-terms in this framework. We can take Example (10) as a simple illustration.

(10) *A man is happy.*

In order to simplify the process, "*is happy*" will be treated as a single predicate taking one argument, the subject. According to the basic FOL, the λ-expressions for all elements of the sentence are:

$[\![a]\!] = \lambda P \lambda Q \exists x(P(x) \wedge Q(x))$
$[\![man]\!] = \lambda x.\mathbf{man}(x)$
$[\![is\_happy]\!] = \lambda x.\mathbf{is\_happy}(x)$

Based on Context Free Grammar (CFG), we could make use of lambda calculus to "combine" each part of the sentence. Finally, the interpretation for the whole sentence is as follows:

$[\![is\_happy]\!]([\![a]\!][\![man]\!]) \rightarrow_\beta \exists x(\mathbf{man}(x) \wedge \mathbf{is\_happy}(x))$ [2]

However, because Montague Semantics is only designed to construct the semantic interpretation of single sentence, people began to search for other solutions that are able to deal with the multi-sentence cases, in other words, discourse processing. These solutions process natural language from a dynamic point of view. Meaning of discourse is considered as a set of states. Each new piece of discourse will shift the state from one to another, so as to evaluate the meaning of the

[2] The denotation "$\rightarrow_\beta$" means the conventional β-reduction in lambda calculus. Similarly, the denotation "$\rightarrow_\alpha$" in future means α-conversion.

whole discourse. Some of the famous dynamic approaches are Heim's File Change Semantics (FCS) [Hei83], Groenendijk & Stokhof's Dynamic Predicate Logic (DPL) [GS91] and Dynamic Montague Grammar (DMG) [GS89]. Kamp's Discourse Representation Theory (DRT) [Kam81] is also one of them. In this chapter, I will first focus on Kamp's DRT theory and then explain the concept of accessibility in DRT.

3.1 Discourse Representation Theory (DRT)

Before introducing DRT, let's first look at two examples:

(11) *A man is happy. He laughs.*

(12) *Every farmer who owns a donkey beats it.*

In standard Montague Semantics, the first part of Example (11) will be interpreted as "$\exists x.(\mathbf{man}(x) \wedge \mathbf{is\_happy}(x))$", while the second part is "$\mathbf{laugh}(x)$". If we simply use conjunction to connect the two parts, the formula will be "$\exists x.(\mathbf{man}(x) \wedge \mathbf{is\_happy}(x)) \wedge \mathbf{laugh}(x)$". Thus, it would fall into the unbounded variable problem in FOL: variable "x" in "$\mathbf{laugh}(x)$" becomes a free variable. For Example (12), based on the traditional treatment for quantifiers, the typical construction of the sentence would be: "$\forall x.(\mathbf{farmer}(x) \wedge \exists y(\mathbf{donkey}(y) \wedge \mathbf{own}(x, y)) \rightarrow \mathbf{beat}(x, y))$". Again, in this case, variable "y" appears freely at the end of the formula. Also the interpretation does not correctly express the meaning of the donkey sentence.

Generally speaking, DRT focuses on solving the above problems in a representational way (because of its famous "box-structure"). In DRT, Discourse Representation Structure (DRS) is the basic element. A DRS is regarded as the mental representation built in hearers' minds during communication. A DRS consists of two parts, the so called discourse referent and DRS conditions. Objects in the discourse referent list are called reference markers in DRT, they are actually treated as free variables. New variables could be introduced by all kinds of NPs, such as indefinite expressions, definite expressions, proper names and pronouns, etc. However, in DRT, variables introduced by definite NPs and pronouns are handled such that they always have to find a link with the pre-existing variables. The discourse conditions could be defined as follows.

Definition Discourse Condition:

1. If R is a n-arity relation symbol, x_1, $x_2 \cdots x_n$ are reference markers, then $R(x_1, x_2 \cdots x_n)$ is a condition;
2. If K_1 and K_2 are DRSs, then $K_1 \Rightarrow K_2$, $K_1 \vee K_2$ are conditions;
3. If K is a DRS, then $\neg K$ is a condition;
4. If x_1, x_2 are reference markers, $x_1 = x_2$ is a condition (the specialty of equal symbol "$=$");
5. Nothing else is a condition except the above 4.

Because all the conditions appearing at the same level are considered to hold a conjunction relation between each other by default, conjunction is not defined as a condition in standard

DRT. Other logic operators, like implication, disjunction and negation are all realized through the condition definition. In DRT, discourse processing is incremental. Each DRS is regarded as the interpretation of the piece of discourse being processed. A new DRS could be added to the current one in order to update the information of the whole discourse. The merge operation in DRT is simple, just putting together the discourse referent and conditions of two DRSs. But sometimes we need to add some other conditions, which denotes the equality of two variables, in order to represent the co-referential relation between discourse referents (in some versions of DRT, this process is named "resolving α-DRSs").

Now, let's look at a specific DRS example, this is the corresponding DRS for the first part of Example (11):

x
$man(x)$
$is\_happy(x)$

The variable "x" is introduced by the indefinite NP "*a man*", and discourse conditions here are exactly the same as in Montague Semantics. The DRS for the second part of Example (11) is:

y
$laugh(y)$

The variable "y" is introduced by anaphoric pronoun "*he*". After merging the two above DRSs together, we will obtain the DRS for the whole discourse:

x, y
$man(x)$
$is\_happy(x)$
$laugh(y)$
$x = y$

We might notice everything keeps the same in the final DRS, except for the new condition "$x = y$", which indicates the anaphoric link between the anaphora and its antecedent.

3.2 Accessibility

In this part, I will introduce one of the most important concepts in DRT: Accessibility. Accessibility is a commonly used word in linguistics, [vH00] and [WK05] respectively from the semantic level and the syntactic level described it. But in DRT, accessibility has its own definition. It denotes the referential relation of discourse referents between two DRSs. Before going to the detail, let's first take a look at the concept of subordination.

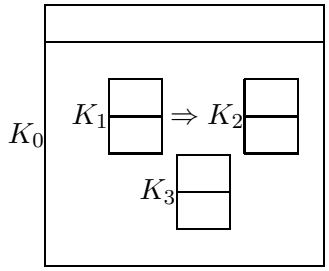
3.2.1 Subordination

To understand the essence of accessibility, we have to know the subordination relation in DRT. This is the formal definition:

Definition Subordination: Let K_1 and K_2 be DRSs, K_1 subordinates K_2 if and only if one of the following conditions holds:

1. K_1 contains a DRS condition of the form "$\neg K_2$";
2. K_1 contains a DRS condition of the form "$K_2 \Rightarrow K_3$", where K_3 is another DRS;
3. "$K_1 \Rightarrow K_2$" is a DRS condition of some other DRS K_3
4. K_1 contains a DRS condition of the form "$K_2 \vee K_3$", where K_3 is another DRS
5. Some DRS K_3 subordinates K_2, and K_1 subordinates K_3

For example, in the following DRS, K_0 subordinates K_1, K_2 and K_3; K_1 subordinates K_2; but not subordination relation between K_1 and K_3, or K_2 and K_3.



3.2.2 More Details on Accessibility

In DRT, accessibility is defined as follows:

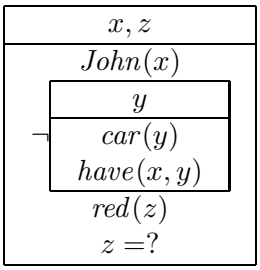
Definition Accessibility: Discourse referents of a DRS K_1 are accessible (reachable) for another DRS K_2 only when:

1. K_1 subordinates K_2;
2. K_1 equals K_2.

Accessibility provides the constraints for anaphoric links between numbers of variables in discourse referent. In standard DRT, no specific algorithm for anaphora resolution was provided. As a result, when constructing DRSs, if we want to find the right antecedent for the current anaphora, we should pick up a "suitable" candidate from all accessible referents. Accessibility correctly predicates some interesting linguistic phenomena, such as negation, conditional phrase and donkey sentence. I will give two simple examples here.

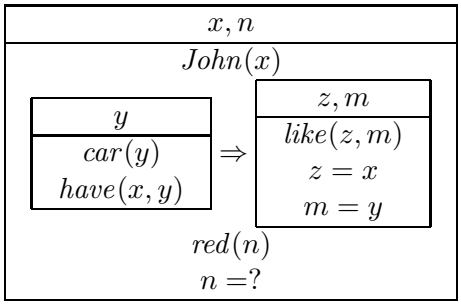
(13) **John does not have a car. It is red.*

Corresponding DRS:



(14) **If John has a car, he likes it. It is red.*

Corresponding DRS:



In Example (13), anaphoric pronoun "*it*" in the second sentence can not refer back to "*car*" in the first part. In DRT, we can explain it such that variable introduced by "*a car*" is embedded in the negation condition, so it is not accessible for other outer DRSs. In Example (14), for the same reason, the variable introduced by "*a car*" is only accessible to the first "*it*", but not the second. Accessibility correctly predicates the real situation in natural language. Actually, the concept of accessibility in DRT models the discourse structure factor for anaphora resolution. Although it is only one of many effective factors, representing the discourse structure and referent relation in the process of discourse processing is undoubtedly of vital importance and meaning. Just simply by checking the accessibility of previous variables, we are able to largely decrease the searching scope for the correct antecedent. In one variant of DRT [LA01], the authors introduced a new DRT paradigm involving the discourse relation. A new concept of "Right Frontier Accessibility" has been raised there, which is able to explain a series of interesting discourse phenomena that traditional DRT could not. But in this thesis, I will stick to the original concept in DRT.

3.3 Compositionality

We often hear people say DRT is not compositional, but not many talk about why. Actually, there are mainly 3 reasons that support this proposition.

1. Firstly, based on DRT, the meaning of the whole sentence could be composed by the meaning of each single sentence by merging every DRS. However, the meaning of a single sentence could not be built upon smaller pieces of natural language.

2. The second reason is the variable renaming problem. As we know, when we merge two DRSs, inevitablely we would encounter variable renaming, which has already been well

defined in simply typed lambda calculus, while lacking of proofs in DRT. If the renaming process is not well standardized, we might not easily "compositionally" carry out the merge operation.

3. The last reason is the way of treating anaphoric pronoun. In DRT, when a pronoun appears, not only a variable is introduced, an anaphoric link between the new variable and its antecedent should be built, while the process for building the link is not part of the meaning of the pronoun. Contradict to the statement of compositionality, some constituents in DRT, such as pronouns, could not introduce the standard function-argument combination, so it is not a compositional system.

4 A New Approach to Dynamics

As introduced in the previous chapter, DRT is quite powerful in discourse processing, especially in explaining the unbounded variable problems and donkey sentences. However, some phenomena such as quantification, coordination, which are well handled in traditional Montague Semantics, are not treated as well in DRT. Further more, as a formal semantic approach, unlike Montague Semantics, DRT is not compositional. Although in [KR93], a translation from DRT to FOL was provided, the essence of DRT, such as structural representation was abandoned.As a result, lots of researchers are searching for means that combine DRT and traditional Montague Semantics, making use of advantages from each of them, one of the most known might be [Mus96].

Recently, there is a new approach [dG06], proposing to use the standard computational techniques and typed lambda-calculus to express the DRT discourse structural descriptive power in a non-representation way. I will mainly explain the new framework in this chapter.

4.1 Outline

By introducing sets of reference markers, DRT tackles the problem of extending the scope of quantifiers to the following sentences in discourse. These reference markers act both as existential quantifiers and free variables. Because of their special status, variable renaming is very important when combining DRT with Montague Semantics.

In [dG06], the author adds a concept of "context" into the traditional Montague Semantics, which extends the power of Montague Semantics, making it possible to express discourse dynamics. Based on DRT and Church's simply typed-calculus, all concepts in the new framework keep the same. The general idea is to use the classical mathematical logic tools to deal with dynamic phenomena, which are often considered not in the scope of Montague Semantics. There is no any kind of dynamic notion added in. Instead, only the simply typed lambda calculus is used to express the dynamics. Doing this has 3 main advantages:

1. The new framework is not claimed to commit with any other specific theory, which means the new approach is not restricted to express the meaning of expressions;
2. The concept of variable (free or bound) is the same as in standard mathematics;
3. It would be more persuasive to explain some phenomenon with a widely accepted theory than with a newly developed theory.

4.2 Core Idea

In this section, I am going to introduce the core idea of this new approach.

4.2.1 Left Context

In DRT, every new sentence is processed under the environment of all preceding sentences. There are two more steps in the new approach.

1. A sentence is processed not only based on the preceding sentences, but also the following ones, namely the left and right contexts;
2. The two kinds of contexts would be abstracted over the meaning of the sentences.

In Church's simple type theory, there are only two atomic types: ι, denoting the type of individual; o, denoting the type of proposition.[3] The new approach adds one more atomic type γ, to express the type of the left contexts, thus the notion of context is realized. Consequently, as the right context could be interpreted as a proposition given its left context, its type should be $\gamma \to o$. For the same reason, the whole discourse could be interpreted as a proposition given both its left and right context. Assuming s and t is respectively the syntactic category for sentences and contexts, their semantic interpretations could be:

$[\![s]\!] = \gamma \to (\gamma \to o) \to o$
$[\![t]\!] = \gamma \to (\gamma \to o) \to o$

In order to compose the meanings of sentences to get the meaning of a discourse, the following equation is proposed:

$$[\![D.S]\!] = \lambda e\phi.[\![D]\!]e(\lambda e'.[\![S]\!]e'\phi)^{4} \tag{1}$$

In which "D" is a piece of preceeding discourse and "S" is the currently processing sentence. We can view the meaning of the equation as follows: the semantic type of a updating context "$D.S$" should be $\gamma \to (\gamma \to o) \to o$, so it would take two parameters of type γ and $\gamma \to o$, namely the left and right context. The main formula should get the previous context "D" involved, which is also of type $\gamma \to (\gamma \to o) \to o$. It then takes one left context of type γ and one expression of type $\gamma \to o$ as arguments. Because the current sentence "S", which is typed $\gamma \to (\gamma \to o) \to o$, also contributes to the latter expression, it must appear somewhere in it. The expression (type $\gamma \to o$) has another left context of type γ in the λ-part, and its body takes a left context of type γ and a right context of type $\gamma \to o$ as arguments. This explains the meaning of the updating equation. While turning to DRT, assume "$x_1, x_2, x_3 \cdots$" are reference markers, and "$C_1, C_2, C_3 \cdots$" are conditions, the corresponding λ-term in the new framework to a general DRS should be:

$\lambda e\phi.\exists x_1 \cdots x_n.C_1 \wedge \cdots C_m \wedge \phi e'$ [5].

[3] Here, we stick to Church's original denotation, using symbol ι and o. They are the same as Montague's denotation: e refers to individual, t refers to proposition, which most people are more famaliar with.

[4] Formula $[\![D.S]\!]$ can simply be considered as the interpretation of the discourse plus the following sentence. The dot "." is just the representation for full stop symbol between declarative sentences.

[5] Here, "e'" is a left context made of "e" and the variables "$x_1, x_2, x_3 \cdots$" The method to construct it depends on the specific structure of the context, which would be illustrated by following examples.

4.2.2 List Structure

The left context (type γ) in the new framework functions for aiding pronominal anaphor resolution. A kind of choice operator is assumed to resolve the anaphora problem in this framework. Some oracles are used to represent the choice operators, such as "$sel_{he}, sel_{she} \cdots$". The type of those oracles is $\gamma \to \iota$ because they take a left context as argument and would return a resolved individual. In order to let the context be updated, another operator "::" is introduced. The main purpose of "::" is to add new individuals to the updating left context, thus it would be possible for coming choice operators to select individual from the new left context. Its type should be $\iota \to \gamma \to \gamma$, because it must take an individual and a left context as argument, and return an updated left context. For instance, formula "$a :: e$" actually meant "$\{a\} \bigcup e$" in application. The list structure, which contains variables introduced by previous context, actually models the discourse referent in DRT.

4.3 Examples

Let's first look at the interpretation for the following two sentences based on the new framework:

(15) *John loves Mary. He smiles at her.*

(16) *John has a car. It is red.*

In Montague semantics, noun phrase (NP) is known as type $(\iota \to o) \to o$. However, because of the novel concept of context, the author modifies the type as $(\iota \to \gamma \to o) \to \gamma \to o$, each occurrence of o also gets a γ involved. The first γ could be understood to give the power for the current NP to update the context; while the second γ renders the possibility for pronouns to access the updated context. Based on the type designing, it is not difficult to obtain the following λ-expressions:

$[\![John]\!] = \lambda\psi e.\psi\mathbf{j}(\mathbf{j} :: e)$
$[\![Mary]\!] = \lambda\psi e.\psi\mathbf{m}(\mathbf{m} :: e)$
$[\![he]\!] = \lambda\psi e.\psi(sel_{he}e)e$
$[\![her]\!] = \lambda\psi e.\psi(sel_{her}e)e$

We might notice that only full NPs have the ability to update the context with the related variables in the list, while pronominal are handled as choice operators, which could only select antecedents from the current context.

As for transitive verbs like "*love*" and "*smile_at*", they are of type "$[\![NP]\!] \to [\![NP]\!] \to [\![s]\!]$". "$NP$" and "$s$" here are both interpreted the same as described above. As a consequence, the following λ-expressions could be constructed:

$[\![love]\!] = \lambda ose\phi.s(\lambda xe.o(\lambda ye.\mathbf{love}(x, y) \wedge \phi e)e)e$
$[\![smile\_at]\!] = \lambda ose\phi.s(\lambda xe.o(\lambda ye.\mathbf{smile\_\ at}(x, y) \wedge \phi e)e)e$

Compared with Montague's interpretation [6], the appearance of the conjunct "ϕe" is a salient difference, it models the right context of the sentence with the transitive verb. Also, the two arguments "e" at the end of the formula render respectively the subject and the object the

[6] In Montague's framework, transitive verb such as "*love*" is interpreted as $[\![loves]\!] = \lambda os.s(\lambda x.o(\lambda y.\mathbf{love}(x, y)))$, "$s$" refers to subject, "$o$" refers to object.

ability to update the list. Based on the above expressions, semantic interpretation for the first part of Example (15) is as follows:

$([\![loves]\!][\![Mary]\!])[\![John]\!] \rightarrow_\beta \lambda e\phi.\mathbf{love}(\mathbf{j},\mathbf{m}) \wedge \phi(\mathbf{m} :: \mathbf{j} :: e)$

The interpretation for the second part is:

$([\![smile\_at]\!][\![her]\!])[\![he]\!] \rightarrow_\beta \lambda e\phi.\mathbf{smile\_\ at}(sel_{he}(e), sel_{her}(e)) \wedge \phi(e)$

After combining the two parts by using Equation 1, we will get the result:

$[\![D.S]\!] \rightarrow_\beta \lambda e\phi.(\mathbf{love}(\mathbf{j},\mathbf{m}) \wedge \mathbf{smile\_\ at}(sel_{he}(\mathbf{m} :: \mathbf{j} :: e), sel_{her}(\mathbf{m} :: \mathbf{j} :: e)) \wedge \phi(\mathbf{m} :: \mathbf{j} :: e))$

For the type of noun, the author keeps the same as the way in Montague's interpretation: $[\![N]\!] = \iota \rightarrow o$. Then the type for determiner is $[\![N]\!] \rightarrow [\![NP]\!]$, namely $(\iota \rightarrow o) \rightarrow (\iota \rightarrow \gamma \rightarrow o) \rightarrow \gamma \rightarrow o$. So the λ-expressions for nouns and determiners are exemplified as:

$[\![car]\!] = \lambda x.\mathbf{car}(x)$
$[\![a]\!] = \lambda n\psi e.\exists x.nx \wedge \psi x(x :: e)$
$[\![every]\!] = \lambda n\psi e.\forall x.nx \rightarrow \psi x(x :: e)$

With the same calculus, the interpretation for Example (16) is as follows:

$([\![has]\!]([\![a]\!][\![car]\!]))[\![John]\!] \rightarrow_\beta \lambda e\phi.(\exists y.(\mathbf{car}(y) \wedge \mathbf{have}(\mathbf{j},y) \wedge \phi(\mathbf{j} :: y :: e)))$
$[\![is\_red]\!][\![it]\!] \rightarrow_\beta \lambda e\phi.\mathbf{is\_\ red}(sel_{it}e) \wedge \phi e$
$[\![D.S]\!] \rightarrow_\beta \lambda e\phi.(\exists y.(\mathbf{car}(y) \wedge \mathbf{have}(\mathbf{j},y) \wedge \mathbf{is\_\ red}(sel_{it}(\mathbf{j} :: y :: e)) \wedge \phi(\mathbf{j} :: y :: e)))$

As we can see in both examples, the new approach provides a list of possible variables for the anaphoric pronouns to choose from. The elements in the list are similar to those accessible referents in DRT. Assume the choice operator can make correct selections within the list (namely it can find the appropriate accessible referent in DRT), we will have a good result for anaphora resolution. At this stage, the above two examples have showed us the new style of discourse processing in a traditional Montague Semantic way, while keeping the characteristic of referent structure expressed in DRT.

4.4 Updated Version

(17) **Every man loves a woman. He smiles at her.*

Although the above mechanism could express some extent of dynamic, it is not that satisfactory. Example (17), the problematic discourse, which would be blocked in DRT, is still valid in the above framework because after lambda calculus, both "*man*" and "*woman*" are possible to be accessed by the following pronouns. Further more, in the above version, nouns are interpreted as properties, which makes it impossible to handle complex noun expressions like "*man who loves a woman*", "*farmer who owns a donkey*", etc. When processing nouns, the notion of context should also be considered. In order to solve those problems, the previous solution is updated to a more systematized one.

According to Montague Semantics, we have the following interpretations: $[\![s]\!] = o$, $[\![N]\!] = \iota \rightarrow o$, $[\![NP]\!] = (\iota \rightarrow o) \rightarrow o$, which could be rephrased as: $[\![s]\!] = o$, $[\![N]\!] = \iota \rightarrow [\![s]\!]$, $[\![NP]\!] = (\iota \rightarrow [\![s]\!]) \rightarrow [\![s]\!]$. Based on what has been discussed previously, after adding the notion of context to Montague semantics, "$[\![s]\!] = o$" becomes "$[\![s]\!] = \gamma \rightarrow (\gamma \rightarrow o) \rightarrow o$". As a result, the interpretations for "N" and "NP" are modified as:

$[\![N]\!] = \iota \to \gamma \to (\gamma \to o) \to o$
$[\![NP]\!] = (\iota \to \gamma \to (\gamma \to o) \to o) \to \gamma \to (\gamma \to o) \to o$

In the same way, the type for transitive verbs, determiners, relative pronouns are also fixed, they are:

$[\![V]\!] = [\![NP]\!] \to [\![NP]\!] \to [\![s]\!] = ((\iota \to [\![s]\!]) \to [\![s]\!]) \to ((\iota \to [\![s]\!]) \to [\![s]\!]) \to [\![s]\!]$
$[\![DET]\!] = [\![N]\!] \to [\![NP]\!] = (\iota \to [\![s]\!]) \to ((\iota \to [\![s]\!]) \to [\![s]\!])$
$[\![REL\_PRO]\!] = ([\![NP]\!] \to [\![s]\!]) \to [\![N]\!] \to [\![N]\!]$

All the "$[\![s]\!]$" in the above equations could be replaced by the new interpretation of "s" after involving the notion of context (both left and right). A final look of λ-expression for all categories of words could be found in the Appendix A.

As an ending, let's look at the processing for Example (12), the donkey sentence, under the new framework. The common FOL interpretation for this sentence is:

$\forall x.(\mathbf{farmer}(x) \to \forall y.((\mathbf{donkey}(y) \wedge \mathbf{own}(x, y)) \to \mathbf{beat}(x, y)))$

As described previously, the special points for Example (12) are:

1. The anaphora "*it*" acts like inter-sentential anaphora;
2. The existential quantifier "*a*" should be semantically interpreted as a universal quantifier;
3. Noun phrases "*every farmer*" and "*a donkey*" should not be accessible for further processing.

Then, by applying the λ-term for each word in the above sentence, we could obtain the final result after β-reduction:

$([\![beat]\!][\![it]\!])([\![every]\!]([\![who]\!]([\![owns]\!]([\![a]\!][\![donkey]\!]))[\![farmer]\!]))$
$\to_\beta \lambda e\phi.(\forall x.\mathbf{farmer}(x) \to (\forall y.(\mathbf{donkey}(y) \wedge \mathbf{own}(x, y) \to \mathbf{beat}(x, sel_{it}(x :: y :: e))))) \wedge \phi e$

This exactly expresses the real meaning of the donkey sentence, and all the three special "donkey constraints" are also satisfied.

5 More Accessibility Constraints

At a first glance, the new framework, that proposes a Montagovian flavor of dynamics, successfully combines the way of multi-sentence processing in DRT with the traditional λ-calculus and type theory. Also, the concept of accessibility in DRT is modeled by the list structure, which provides possible antecedents for future anaphora to refer back. However, in the original paper [dG06], not many examples are provided. We are obviously not content with only some simple discourses such as Example (15) and (16). So how does the new approach work in other more complex cases? In this chapter, I will extend the accessibility constraints to several linguistic phenomena based on the updated version of the framework in Section 4.4, and propose solutions, together with detailed examples for each of them.

5.1 Scope Ambiguity

As shown in Chapter 4, the first version of the new approach interprets universal quantifier "*every*" as:

$[\![every]\!] = \lambda n\psi e.\forall x.(nx \rightarrow \psi x(x :: e))$

However, this is not enough to explain Example (17), in which both "*man*" and "*woman*" are not accessible for future anaphora. As a result, in the updated version, a new interpretation for "*every*" is proposed:

$[\![every]\!] = \lambda n\psi e\phi.(\forall x.\neg(nxe(\lambda e.\neg(\psi x(x :: e)(\lambda e.\mathbf{T}))))) \wedge \phi e$ [7]

In the above expression, the standard logic implication is replaced by its negation version ($\phi \rightarrow \psi = \neg\phi \vee \psi = \neg(\phi \wedge \neg\psi)$). And one main difference from the previous version is the introduction of "$\lambda e.\mathbf{T}$", which functions to block the variables quantified by the universal quantifier. The motivation of doing this is based on the way of handling conditional discourses in DRT. Take the DRS structure in Section 3.2.1 for example, discourse referents in K_1 are quantified by universal quantifier by default. They are only accessible for K_2, so we should eliminate them from the list structure for future referential link. Now, let's look at the detailed steps for interpret Example (17).

$[\![a]\!][\![woman]\!]$
$= \lambda n\psi e\phi.\exists x.nxe(\lambda e.\psi x(x :: e)\phi)(\lambda x'e'\phi'.\mathbf{woman}(x') \wedge \phi'e')$
$\rightarrow_\beta \lambda\psi e\phi.\exists x.(\lambda x'e'\phi'.\mathbf{woman}(x') \wedge \phi'e')xe(\lambda e.\psi x(x :: e)\phi)$
$\rightarrow_\beta \lambda\psi e\phi.\exists x.(\lambda\phi'.\mathbf{woman}(x) \wedge \phi'e)(\lambda e.\psi x(x :: e)\phi)$
$\rightarrow_\beta \lambda\psi e\phi.\exists x.(\mathbf{woman}(x) \wedge (\lambda e.\psi x(x :: e)\phi)e)$
$\rightarrow_\beta \lambda\psi e\phi.\exists x.(\mathbf{woman}(x) \wedge \psi x(x :: e)\phi)$

$[\![loves]\!]([\![a]\!][\![woman]\!])$
$= \lambda os.s(\lambda x.o(\lambda ye\phi.\mathbf{love}(x, y) \wedge \phi e))(\lambda\psi e'\phi.'\exists x'.(\mathbf{woman}(x') \wedge \psi x'(x' :: e')\phi'))$
$\rightarrow_\beta \lambda s.s(\lambda x.(\lambda\psi e'\phi.'\exists x'.(\mathbf{woman}(x') \wedge \psi x'(x' :: e')\phi'))(\lambda ye\phi.\mathbf{love}(x, y) \wedge \phi e))$
$\rightarrow_\beta \lambda s.s(\lambda x.(\lambda e'\phi.'\exists x'.(\mathbf{woman}(x') \wedge (\lambda ye\phi.\mathbf{love}(x, y) \wedge \phi e))x'(x' :: e')\phi'))$
$\rightarrow_\beta \lambda s.s(\lambda x.(\lambda e'\phi.'\exists x'.(\mathbf{woman}(x') \wedge (\lambda e\phi.\mathbf{love}(x, x') \wedge \phi e))(x' :: e')\phi'))$
$\rightarrow_\beta \lambda s.s(\lambda x.(\lambda e'\phi.'\exists x'.(\mathbf{woman}(x') \wedge (\lambda\phi.\mathbf{love}(x, x') \wedge \phi(x' :: e')))\phi'))$
$\rightarrow_\beta \lambda s.s(\lambda x.(\lambda e'\phi.'\exists x'.(\mathbf{woman}(x') \wedge \mathbf{love}(x, x') \wedge \phi'(x' :: e'))))$
$\rightarrow_\alpha \lambda s.s(\lambda x.(\lambda e\phi.\exists y.(\mathbf{woman}(y) \wedge \mathbf{love}(x, y) \wedge \phi(y :: e))))$

$[\![every]\!][\![man]\!]$
$= \lambda n\psi e\phi.((\forall x.\neg(nxe(\lambda e.\neg(\psi x(x :: e)(\lambda e.\mathbf{T})))))) \wedge \phi e)(\lambda x'e'\phi'.\mathbf{man}(x') \wedge \phi'e')$
$\rightarrow_\beta \lambda\psi e\phi.((\forall x.\neg((\lambda x'e'\phi'.\mathbf{man}(x') \wedge \phi'e')xe(\lambda e.\neg(\psi x(x :: e)(\lambda e.\mathbf{T})))))) \wedge \phi e)$
$\rightarrow_\beta \lambda\psi e\phi.((\forall x.\neg((\lambda\phi'.\mathbf{man}(x) \wedge \phi'e)(\lambda e.\neg(\psi x(x :: e)(\lambda e.\mathbf{T})))))) \wedge \phi e)$
$\rightarrow_\beta \lambda\psi e\phi.((\forall x.\neg(\mathbf{man}(x) \wedge (\lambda e.\neg(\psi x(x :: e)(\lambda e.\mathbf{T})))e)) \wedge \phi e)$
$\rightarrow_\beta \lambda\psi e\phi.((\forall x.\neg(\mathbf{man}(x) \wedge \neg(\psi x(x :: e)(\lambda e.\mathbf{T})))) \wedge \phi e)$
$\triangleq \lambda\psi e\phi.(\forall x.\mathbf{man}(x) \rightarrow \psi x(x :: e)(\lambda e.\mathbf{T})) \wedge \phi e$

$([\![loves]\!]([\![a]\!][\![woman]\!]))([\![every]\!][\![man]\!])$
$= \lambda s.s(\lambda x.(\lambda e\phi.\exists y.(\mathbf{woman}(y) \wedge \mathbf{love}(x, y) \wedge \phi(y :: e))))(\lambda\psi e'\phi'.(\forall x'.\mathbf{man}(x') \rightarrow \psi x'(x' :: e')(\lambda e'.\mathbf{T})) \wedge \phi'e')$
$\rightarrow_\beta (\lambda\psi e'\phi'.(\forall x'.\mathbf{man}(x') \rightarrow \psi x'(x' :: e')(\lambda e'.\mathbf{T})) \wedge \phi'e')(\lambda x.(\lambda e\phi.\exists y.(\mathbf{woman}(y) \wedge \mathbf{love}(x, y) \wedge$

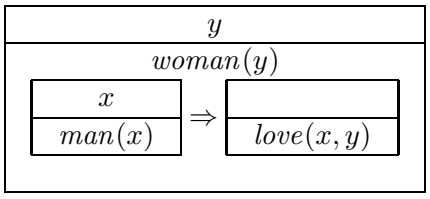
[7] "$\mathbf{T}$" stands for the normal sense"True" in logic. In FOL calculus, "$A \wedge B \wedge \mathbf{T}$" is equal to "$A \wedge B$". Here, "$\lambda e.\mathbf{T}$" takes any argument will return "$\mathbf{T}$".

$\phi(y :: e))))$
$\rightarrow_\beta \lambda e'\phi'.(\forall x'.\mathbf{man}(x') \rightarrow (\lambda x.(\lambda e\phi.\exists y.(\mathbf{woman}(y) \wedge \mathbf{love}(x,y) \wedge \phi(y :: e))))x'(x' :: e')(\lambda e'.\mathbf{T})) \wedge \phi' e'$
$\rightarrow_\beta (\lambda e'\phi'.(\forall x'.\mathbf{man}(x') \rightarrow (\lambda e\phi.\exists y.(\mathbf{woman}(y) \wedge \mathbf{love}(x',y) \wedge \phi(y :: e)))(x' :: e')(\lambda e'.\mathbf{T})) \wedge \phi' e')$
$\rightarrow_\beta \lambda e'\phi'.(\forall x'.\mathbf{man}(x') \rightarrow (\lambda\phi.\exists y.(\mathbf{woman}(y) \wedge \mathbf{love}(x',y) \wedge \phi(y :: (x' :: e'))))(\lambda e'.\mathbf{T})) \wedge \phi' e'$
$\rightarrow_\beta \lambda e'\phi'.(\forall x'.\mathbf{man}(x') \rightarrow \exists y.(\mathbf{woman}(y) \wedge \mathbf{love}(x',y) \wedge (\lambda e'.\mathbf{T})(y :: x' :: e'))) \wedge \phi' e'$
$\rightarrow_\beta \lambda e'\phi'.(\forall x'.\mathbf{man}(x') \rightarrow \exists y.(\mathbf{woman}(y) \wedge \mathbf{love}(x',y) \wedge \mathbf{T})) \wedge \phi' e'$
$\rightarrow_\alpha \lambda e\phi.(\forall x.\mathbf{man}(x) \rightarrow \exists y.(\mathbf{woman}(y) \wedge \mathbf{love}(x,y))) \wedge \phi e$

$([\![smile\_at]\!][\![her]\!])[\![he]\!] \rightarrow_\beta \lambda e\phi.\mathbf{smile\_\ at}(sel_{he}(e), sel_{her}(e)) \wedge \phi e$

$[\![S_1.S_2]\!] = \lambda e\phi.[\![S_1]\!]e(\lambda e'.[\![S_2]\!]e'\phi)$
$= \lambda e\phi.(\lambda e''\phi''.(\forall x.\mathbf{man}(x) \rightarrow \exists y.(\mathbf{woman}(y) \wedge \mathbf{love}(x,y))) \wedge \phi'' e'')e(\lambda e'.(\lambda e'''\phi'''.\mathbf{smile\_\ at}(sel_{he}(e'''), sel_{her}(e''')) \wedge \phi''' e''')e'\phi)$
$\rightarrow_\beta \lambda e\phi.(\lambda\phi''.(\forall x.\mathbf{man}(x) \rightarrow \exists y.(\mathbf{woman}(y) \wedge \mathbf{love}(x,y))) \wedge \phi'' e)(\lambda e'.\mathbf{smile\_\ at}(sel_{he}(e'), sel_{her}(e')) \wedge \phi e')$
$\rightarrow_\beta \lambda e\phi.((\forall x.\mathbf{man}(x) \rightarrow \exists y.(\mathbf{woman}(y) \wedge \mathbf{love}(x,y))) \wedge (\lambda e'.\mathbf{smile\_\ at}(sel_{he}(e'), sel_{her}(e')) \wedge \phi e')e)$
$\rightarrow_\beta \lambda e\phi.(\forall x.\mathbf{man}(x) \rightarrow \exists y.(\mathbf{woman}(y) \wedge \mathbf{love}(x,y))) \wedge \mathbf{smile\_\ at}(sel_{he}(e), sel_{her}(e)) \wedge \phi e$

Both "sel_{he}" and "sel_{her}" have no element to select from the list. The interpretation denotes the sentence is problematic, just the same as we understand it in real world: we do not know whom "*he*" and "*her*" refer to. However, only the *de dicto* reading (also known as subject wide scope reading) is considered here for Example (17). Its *de re* reading (also known as object wide scope reading), in which we could interpret as "*there exists a woman that every man loves*", the variable introduced by "*a woman*" should not be blocked, but only the one introduced by "*every man*". The DRS corresponding to the *de re* reading is the following:



The way we handle this point here is by using another semantic interpretation for transitive verb "*love*", so as to distinguish between the *de dicto* and *de re* readings. For this purpose, we propose the following formula:

$[\![love^*]\!] = \lambda os.o(\lambda y.s(\lambda xe\phi.\mathbf{love}(x,y) \wedge \phi e))$

It will result the correct interpretation for *de re* reading:

$[\![love^*]\!]([\![a]\!][\![woman]\!])$
$= \lambda os.o(\lambda y.s(\lambda xe\phi.\mathbf{love}(x,y) \wedge \phi e))(\lambda\psi e'\phi'.\exists x'.(\mathbf{woman}(x') \wedge \psi x'(x' :: e')\phi'))$
$\rightarrow_\beta \lambda s.(\lambda\psi e'\phi'.\exists x'.(\mathbf{woman}(x') \wedge \psi x'(x' :: e')\phi'))(\lambda y.s(\lambda xe\phi.\mathbf{love}(x,y) \wedge \phi e))$
$\rightarrow_\beta \lambda s.(\lambda e'\phi'.\exists x'.(\mathbf{woman}(x') \wedge (\lambda y.s(\lambda xe\phi.\mathbf{love}(x,y) \wedge \phi e))x'(x' :: e')\phi'))$
$\rightarrow_\beta \lambda s.(\lambda e'\phi'.\exists y.(\mathbf{woman}(y) \wedge s(\lambda xe\phi.\mathbf{love}(x,y) \wedge \phi e)(y :: e')\phi'))$

$([\![love^*]\!]([\![a]\!][\![woman]\!]))([\![every]\!][\![man]\!])$
$= \lambda s.(\lambda e'\phi'.\exists y.(\mathbf{woman}(y) \wedge s(\lambda xe\phi.\mathbf{love}(x,y) \wedge \phi e)(y :: e')\phi'))(\lambda\psi e''\phi''.(\forall x'.\mathbf{man}(x') \rightarrow \psi x'(x' :: e'')(\lambda e''.\mathbf{T})) \wedge \phi'' e'')$
$\rightarrow_\beta \lambda e'\phi'.\exists y.(\mathbf{woman}(y) \wedge (\lambda\psi e''\phi''.(\forall x'.\mathbf{man}(x') \rightarrow \psi x'(x' :: e'')(\lambda e''.\mathbf{T})) \wedge \phi'' e'')(\lambda xe\phi.\mathbf{love}(x,y) \wedge$

$\phi e)(y :: e')\phi')$
$\rightarrow_\beta \lambda e'\phi'.\exists y.(\mathbf{woman}(y) \wedge (\lambda e''\phi''.(\forall x'.\mathbf{man}(x') \rightarrow (\lambda x e\phi.\mathbf{love}(x,y) \wedge \phi e)x'(x' :: e'')(\lambda e''.\mathbf{T})) \wedge \phi'' e'')(y :: e')\phi')$
$\rightarrow_\beta \lambda e'\phi'.\exists y.(\mathbf{woman}(y) \wedge (\lambda e''\phi''.(\forall x'.\mathbf{man}(x') \rightarrow (\lambda\phi.\mathbf{love}(x',y) \wedge \phi(x' :: e''))(\lambda e''.\mathbf{T})) \wedge \phi'' e'')(y :: e')\phi')$
$\rightarrow_\beta \lambda e'\phi'.\exists y.(\mathbf{woman}(y) \wedge (\lambda e''\phi''.(\forall x'.\mathbf{man}(x') \rightarrow \mathbf{love}(x',y) \wedge (\lambda e''.\mathbf{T})(x' :: e'')) \wedge \phi'' e'')(y :: e')\phi')$
$\rightarrow_\beta \lambda e'\phi'.\exists y.(\mathbf{woman}(y) \wedge (\lambda e''\phi''.(\forall x'.\mathbf{man}(x') \rightarrow \mathbf{love}(x',y) \wedge \mathbf{T}) \wedge \phi'' e'')(y :: e')\phi')$
$\rightarrow_\beta \lambda e'\phi'.\exists y.(\mathbf{woman}(y) \wedge (\forall x'.\mathbf{man}(x') \rightarrow \mathbf{love}(x',y) \wedge \mathbf{T}) \wedge \phi'(y :: e'))$
$\rightarrow_\alpha \lambda e\phi.\exists y.(\mathbf{woman}(y) \wedge (\forall x.\mathbf{man}(x) \rightarrow \mathbf{love}(x,y) \wedge \mathbf{T}) \wedge \phi(y :: e))$

In this case, only variable "y", which links to "*a woman*", is accessible for future processing. This is just what the corresponding DRS expresses. Consequently, the new approach can successfully resolve the quantification scope ambiguity in simple lambda calculus, while keeping the accessibility concept of DRT at the same time.

However, here we solve the scope ambiguity by posing two different interpretations for the main verb, another possible solution is by using the technique of Nested Cooper Storage [Kel88]. Whenever we see a quantified NP, instead of applying it directly in the calculus, we store it somewhere. After processing the whole sentence, by activating the stored NPs in different orders, we could obtain various interpretations of different scopes.

5.2 Conditional Sentences

In DRT, conditional sentences like "$if \cdots then \cdots$" are treated as two DRSs connected by a conditional relation symbol. Thus, not only the referents in the condition will be quantified by universal quantifier, also they can only be accessible for the result, while not other future discourses. Again, the DRS structure in Chapter 3.2.1 is a good illustration, referents in K_1 can not be antecedents out the scope of the condition "$K_1 \Rightarrow K_2$". Now let's consider the following sentence in the new framework:

(18) *If a man loves a woman, then he smiles at her.*

It is just a conditional version of the declarative discourse "*A man loves a woman, he smiles at her*". If we ignore the effect brought about by the conditional symbol "$if \cdots then \cdots$", the interpretation of this declarative sentence will be:

$([\![love]\!]([\![a]\!][\![woman]\!]))([\![a]\!][\![man]\!])$
$\rightarrow_\beta \lambda e\phi.\exists x.(\mathbf{man}(x) \wedge \exists y.(\mathbf{woman}(y) \wedge \mathbf{love}(x,y) \wedge \phi(y :: x :: e)))$

$([\![smile\_at]\!][\![her]\!])[\![he]\!] \rightarrow_\beta \lambda e\phi.\mathbf{smile\_at}(sel_{he}(e), sel_{her}(e)) \wedge \phi e$

$[\![S_1.S_2]\!] = \lambda e\phi.[\![S_1]\!]e(\lambda e'.[\![S_2]\!]e'\phi)$
$= \lambda e\phi.(\lambda e''\phi''.\exists x.(\mathbf{man}(x) \wedge \exists y.(\mathbf{woman}(y) \wedge \mathbf{love}(x,y) \wedge \phi''(y :: x :: e'')))e(\lambda e'.(\lambda e'''\phi'''.\mathbf{smile\_at}(sel_{he}(e'''), sel_{her}(e''')) \wedge \phi'''e''')e'\phi)$
$\rightarrow_\beta \lambda e\phi.(\lambda\phi''.\exists x.(\mathbf{man}(x) \wedge \exists y.(\mathbf{woman}(y) \wedge \mathbf{love}(x,y) \wedge \phi''(y :: x :: e))))(\lambda e'.\mathbf{smile\_at}(sel_{he}(e'), sel_{her}(e')) \wedge \phi e')$
$\rightarrow_\beta \lambda e\phi.(\exists x.(\mathbf{man}(x) \wedge \exists y.(\mathbf{woman}(y) \wedge \mathbf{love}(x,y) \wedge (\lambda e'.\mathbf{smile\_at}(sel_{he}(e'), sel_{her}(e')) \wedge \phi e')(y :: x :: e))))$
$\rightarrow_\beta \lambda e\phi.(\exists x.(\mathbf{man}(x) \wedge \exists y.(\mathbf{woman}(y) \wedge \mathbf{love}(x,y) \wedge \mathbf{smile\_at}(sel_{he}(y :: x :: e), sel_{her}(y :: x ::$

$e)) \wedge \phi(y :: x :: e))))$

According to the current result, we have 2 steps more to go:

1. The condition "$if \cdots then \cdots$" needs to be expressed in the formula;
2. Variables introduced in the condition should be blocked.

In order to achieve both steps, we propose a new formula other than Equation 1, to combine two discourses, through which the conditional constraints could be realized.

$$[\![D.S]\!] = \lambda e\phi.(\neg[\![D]\!]e(\lambda e'.\neg[\![S]\!]e'(\lambda e''.\mathbf{T}))) \wedge \phi e \tag{2}$$

This formula also could be rewritten as a "condition operator", which has the following form:

$$[\![con\_op]\!] = \lambda DSe\phi.(\neg[\![D]\!]e(\lambda e'.\neg[\![S]\!]e'(\lambda e''.\mathbf{T}))) \wedge \phi e \tag{3}$$

The "condition operator" takes two pieces of discourse as argument, and will return the interpretation of the whole discourse. It is motivated by the Morgan Law:

$S_1 \rightarrow S_2 \triangleq \neg S_1 \vee S_2 \triangleq \neg(S_1 \wedge \neg S_2)$.

Normal conjunction in the new framework is realized through Formula 1. By adding two negations in different positions, we hope to obtain the implication relation. Now let's try to test on Example (18).

$[\![con\_op]\!][\![S_1]\!][\![S_2]\!]$
$= (\lambda DSe\phi.(\neg[\![D]\!]e(\lambda e'.\neg[\![S]\!]e'(\lambda e''.\mathbf{T}))) \wedge \phi e)[\![S_1]\!][\![S_2]\!]$
$\rightarrow_\beta \lambda e\phi.(\neg(\lambda e''\phi''.\exists x.(\mathbf{man}(x) \wedge \exists y.(\mathbf{woman}(y) \wedge \mathbf{love}(x,y) \wedge \phi''(y :: x :: e'')))) e(\lambda e'.\neg(\lambda e'''\phi'''.\mathbf{smile\_at}$
$(sel_{he}(e'''), sel_{her}(e''')) \wedge \phi'''e''')e'(\lambda e''.\mathbf{T}))) \wedge \phi e$
$\rightarrow_\beta \lambda e\phi.(\neg(\lambda\phi''.\exists x.(\mathbf{man}(x) \wedge \exists y.(\mathbf{woman}(y) \wedge \mathbf{love}(x,y) \wedge \phi''(y :: x :: e)))) (\lambda e'.\neg(\lambda\phi'''.\mathbf{smile\_at}$
$(sel_{he}(e'), sel_{her}(e')) \wedge \phi'''e')(\lambda e''.\mathbf{T}))) \wedge \phi e$
$\rightarrow_\beta \lambda e\phi.(\neg(\lambda\phi''.\exists x.(\mathbf{man}(x) \wedge \exists y.(\mathbf{woman}(y) \wedge \mathbf{love}(x,y) \wedge \phi''(y :: x :: e)))) (\lambda e'.\neg(\mathbf{smile\_at}$
$(sel_{he}(e'), sel_{her}(e')) \wedge (\lambda e''.\mathbf{T})e'))) \wedge \phi e$
$\rightarrow_\beta \lambda e\phi.(\neg(\lambda\phi''.\exists x.(\mathbf{man}(x) \wedge \exists y.(\mathbf{woman}(y) \wedge \mathbf{love}(x,y) \wedge \phi''(y :: x :: e)))) (\lambda e'.\neg(\mathbf{smile\_at}$
$(sel_{he}(e'), sel_{her}(e')) \wedge \mathbf{T}))) \wedge \phi e$
$\rightarrow_\beta \lambda e\phi.(\neg(\exists x.(\mathbf{man}(x) \wedge \exists y.(\mathbf{woman}(y) \wedge \mathbf{love}(x,y) \wedge (\lambda e'.\neg(\mathbf{smile\_at}(sel_{he}(e'), sel_{her}(e')) \wedge$
$\mathbf{T}))(y :: x :: e))))) \wedge \phi e$
$\rightarrow_\beta \lambda e\phi.(\neg(\exists x.(\mathbf{man}(x) \wedge \exists y.(\mathbf{woman}(y) \wedge \mathbf{love}(x,y) \wedge (\neg(\mathbf{smile\_at}(sel_{he}(y :: x :: e), sel_{her}(y ::$
$x :: e)) \wedge \mathbf{T})))))) \wedge \phi e$
$\triangleq \lambda e\phi.(\neg\exists x\exists y.(\mathbf{man}(x) \wedge \mathbf{woman}(y) \wedge \mathbf{love}(x,y) \wedge \neg(\mathbf{smile\_at}(sel_{he}(y :: x :: e), sel_{her}(y :: x ::$
$e)) \wedge \mathbf{T}))) \wedge \phi e$
$\triangleq \lambda e\phi.\forall x\forall y.((\mathbf{man}(x) \wedge \mathbf{woman}(y) \wedge \mathbf{love}(x,y)) \rightarrow \mathbf{smile\_at}(sel_{he}(y :: x :: e), sel_{her}(y :: x ::$
$e))) \wedge \phi e$

Assume the selection function "sel_{he}" and "sel_{her}" can retrieve the correct element from the list, the final result will be:

$\lambda e\phi.\forall x\forall y.((\mathbf{man}(x) \wedge \mathbf{woman}(y) \wedge \mathbf{love}(x,y)) \rightarrow \mathbf{smile\_at}(x,y)) \wedge \phi e$

As discussed previously, the two negations in Formula 3 work for the purpose of changing the normal conjunction into implication. While the "$\lambda e''.\mathbf{T}$" in 3 serves to eliminate elements in

the current list, so as to block variables introduced in conditional phrase for future reference. Its function is the same as the "$\lambda e.\mathbf{T}$" in the interpretation of "*every*".

Accordingly, by applying the declarative forms to the "condition operator", we are able to reach the correct interpretation of the conditional phrase under the new framework.

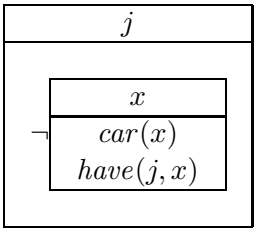
5.3 Negation

In natural languages, negation allows the speaker to express the opposite situation of what he says, which is one of the most common and important phenomena in linguistics. It is generally accepted that negation has the power to reduce the accessibility of concept. Through some interesting experiments, [SH08] and [Kau01] exemplify this point from a psycholinguistic point of view. Here, I will adopt the way that DRT handles negation: assuming that negation encapsulates the negated item in a sub-DRS, treating it as another condition for the universal DRS. In this way, the discourse referents embedded in the negation condition will be blocked for future co-reference, such that sentences like Example (13) will be considered problematic.

First, let's take a look at the interpretation for the first part of Example (16):

$$
\begin{aligned}
&([\![has]\!]([\![a]\!][\![car]\!]))[\![John]\!] \\
&= \lambda s.s(\lambda x.(\lambda e\phi.\exists y.(\mathbf{car}(y) \wedge \mathbf{have}(x,y) \wedge \phi(y :: e))))(\lambda\psi e'\phi'.\psi\mathbf{j}e'(\lambda e'.\phi'(\mathbf{j} :: e'))) \\
&\rightarrow_\beta (\lambda\psi e'\phi'.\psi\mathbf{j}e'(\lambda e'.\phi'(\mathbf{j} :: e')))(\lambda x.(\lambda e\phi.\exists y.(\mathbf{car}(y) \wedge \mathbf{have}(x,y) \wedge \phi(y :: e)))) \\
&\rightarrow_\beta \lambda e'\phi'.(\lambda x.(\lambda e\phi.\exists y.(\mathbf{car}(y) \wedge \mathbf{have}(x,y) \wedge \phi(y :: e))))\mathbf{j}e'(\lambda e'.\phi'(\mathbf{j} :: e')) \\
&\rightarrow_\beta \lambda e'\phi'.(\lambda e\phi.\exists y.(\mathbf{car}(y) \wedge \mathbf{have}(\mathbf{j},y) \wedge \phi(y :: e)))e'(\lambda e'.\phi'(\mathbf{j} :: e')) \\
&\rightarrow_\beta \lambda e'\phi'.(\exists y.(\mathbf{car}(y) \wedge \mathbf{have}(\mathbf{j},y) \wedge (\lambda e'.\phi'(\mathbf{j} :: e'))(y :: e'))) \\
&\rightarrow_\beta \lambda e'\phi'.(\exists y.(\mathbf{car}(y) \wedge \mathbf{have}(\mathbf{j},y) \wedge \phi'(\mathbf{j} :: (y :: e')))) \\
&\rightarrow_\alpha \lambda e\phi.\exists y.(\mathbf{car}(y) \wedge \mathbf{have}(\mathbf{j},y) \wedge \phi(\mathbf{j} :: y :: e))
\end{aligned}
$$

Corresponding DRS to the first part of Example (13):



In order to interpret its negated form, namely Example (13), again we have 2 more steps to go.

1. The negation operator should appear in the expression;
2. The variable introduced by a car should be blocked.

In [dG06], no specific solution for negation condition has been provided. But in [dG07], the author provides the following formula to handle negation:

$$\sim A \triangleq \lambda e\phi.\neg(Ae(\lambda e.\mathbf{T})) \wedge \phi e \tag{4}$$

The operator "$\sim$" will take the interpretation of a whole discourse as argument, add negation inside the original expression, at the same time block the variables. One problem with Formula

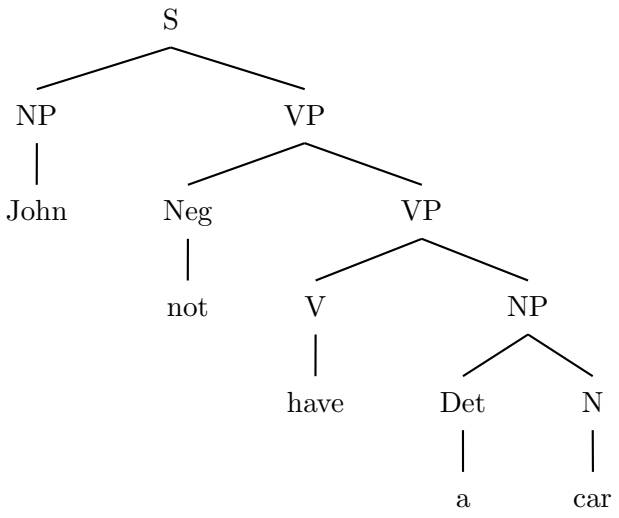


Figure 1: Syntactic Tree for Example (13) [8]

4 is that it will block all variables introduced within the scope of the sentence. So for example, in Example (13), both "*John*" and "*a car*" will be deleted from the list, which is not what we want. In [Pog08], another proposal is given, but it complicates the system by introducing some new concepts (a new connective and *Sub*, *Coord* discourse relation). Here, I would like to propose a new approach, which shall tackle the negation problem without too much difficulty. Figure 1 is the syntactic structure for first part of Example (13). Because the negation actually works on VP, not the whole sentence, so I suggest a negation operator, which takes a VP as argument, returns another VP, but negated:

$$[\![neg\_op]\!] = \lambda\psi s.s(\lambda x.(\lambda e\phi.(\neg(\psi(\lambda Q.Qx)e(\lambda e'.\mathbf{T}))) \wedge \phi e)) \tag{5}$$

Then, the interpretation for "*does not have a car*" would be:

$[\![neg\_op]\!]([\![have]\!]([\![a]\!][\![car]\!]))$
$= \lambda\psi s.s(\lambda x.(\lambda e\phi.(\neg(\phi(\lambda Q.Qx)e(\lambda e'.\mathbf{T})))\wedge\phi e))(\lambda s'.s'(\lambda x'.(\lambda e'\phi'.\exists y.(\mathbf{car}(y)\wedge\mathbf{have}(x',y)\wedge\phi'(y :: e')))))$
$\rightarrow_\beta \lambda s.s(\lambda x.(\lambda e\phi.(\neg((\lambda s'.s'(\lambda x'.(\lambda e'\phi'.\exists y.(\mathbf{car}(y)\wedge\mathbf{have}(x',y)\wedge\phi'(y :: e')))))(\lambda Q.Qx)e(\lambda e'.\mathbf{T})))\wedge\phi e))$
$\rightarrow_\beta \lambda s.s(\lambda x.(\lambda e\phi.(\neg((\lambda Q.Qx)(\lambda x'.(\lambda e'\phi'.\exists y.(\mathbf{car}(y)\wedge\mathbf{have}(x',y)\wedge\phi'(y :: e'))))e(\lambda e'.\mathbf{T})))\wedge\phi e))$
$\rightarrow_\beta \lambda s.s(\lambda x.(\lambda e\phi.(\neg(((\lambda x'.(\lambda e'\phi'.\exists y.(\mathbf{car}(y) \wedge \mathbf{have}(x',y) \wedge \phi'(y :: e'))))x)e(\lambda e'.\mathbf{T}))) \wedge \phi e))$
$\rightarrow_\beta \lambda s.s(\lambda x.(\lambda e\phi.(\neg(\lambda\phi'.\exists y.(\mathbf{car}(y) \wedge \mathbf{have}(x,y) \wedge \phi'(y :: e))(\lambda e'.\mathbf{T}))) \wedge \phi e))$
$\rightarrow_\beta \lambda s.s(\lambda x.(\lambda e\phi.(\neg(\exists y.(\mathbf{car}(y) \wedge \mathbf{have}(x,y) \wedge (\lambda e'.\mathbf{T})(y :: e)))) \wedge \phi e))$
$\rightarrow_\beta \lambda s.s(\lambda x.(\lambda e\phi.(\neg(\exists y.(\mathbf{car}(y) \wedge \mathbf{have}(x,y) \wedge \mathbf{T}))) \wedge \phi e))$
$\triangleq \lambda s.s(\lambda x.(\lambda e\phi.(\neg(\exists y.(\mathbf{car}(y) \wedge \mathbf{have}(x,y)))) \wedge \phi e))$

The interpretation for the whole sentence is:

$([\![neg\_op]\!]([\![have]\!]([\![a]\!][\![car]\!])))[\![John]\!]$
$= \lambda s.s(\lambda x.(\lambda e\phi.(\neg(\exists y.(\mathbf{car}(y) \wedge \mathbf{have}(x,y)))) \wedge \phi e))(\lambda\psi e'\phi'.\psi\mathbf{j}e'(\lambda e'.\phi'(\mathbf{j} :: e')))$
$\rightarrow_\beta (\lambda\psi e'\phi'.\psi\mathbf{j}e'(\lambda e'.\phi'(\mathbf{j} :: e')))(\lambda x.(\lambda e\phi.(\neg(\exists y.(\mathbf{car}(y) \wedge \mathbf{have}(x,y)))) \wedge \phi e))$

[8] This is not a real "one-on-one" syntax tree, we just represent the basic syntactic relations between all the words. Phrase "*does not*" is replaced by a negation symbol in the tree.

$\rightarrow_\beta \lambda e'\phi'.(\lambda x.(\lambda e\phi.(\neg(\exists y.(\mathbf{car}(y) \wedge \mathbf{have}(x,y)))) \wedge \phi e))\mathbf{j}e'(\lambda e'.\phi'(\mathbf{j} :: e'))$
$\rightarrow_\beta \lambda e'\phi'.((\lambda\phi.(\neg(\exists y.(\mathbf{car}(y) \wedge \mathbf{have}(\mathbf{j},y)))) \wedge \phi e'))(\lambda e'.\phi'(\mathbf{j} :: e'))$
$\rightarrow_\beta \lambda e'\phi'.(\neg(\exists y.(\mathbf{car}(y) \wedge \mathbf{have}(\mathbf{j},y)))) \wedge (\lambda e'.\phi'(\mathbf{j} :: e'))e'$
$\rightarrow_\beta \lambda e'\phi'.(\neg(\exists y.(\mathbf{car}(y) \wedge \mathbf{have}(\mathbf{j},y)))) \wedge (\phi'(\mathbf{j} :: e'))$
$\rightarrow_\alpha \lambda e\phi.(\neg(\exists y.(\mathbf{car}(y) \wedge \mathbf{have}(\mathbf{j},y)))) \wedge \phi(\mathbf{j} :: e)$

So far, both steps described earlier for negation have been achieved. By using Formula 5, namely the "negation operator", the negation accessibility constraint in DRT can be well modeled in the new framework only with simple lambda calculus.

5.4 Binding Theory

On the syntactic level, there is another important constraint factor for anaphora resolution: the Binding Theory (BT). Being a controversial domain of theoretical linguistics [Bon06], BT was first raised by Chomsky [Cho81] as the form of *Government and Binding Theory*. A systematic and complete investigation on BT could refer to [Bür05]. As the theory develops, some variants of BT appears, such as Reinhart's Bound-Variable Approach and Reinhart & Reuland's "Reflexivity" Approach, a breif introduction on them could be found in [Bon06]. Here, I will only focus on the original version. First, I simplify introducing BT by its 3 principles, known as Principle A, B and C:

- **Principle A**: A reflexive anaphora must be bound within its local domain.
- **Principle B**: A non-reflexive anaphora must never be bound within its local domain.
- **Principle C**: A referential expression (such as proper name) must never be bound.

They can explain respectively Example (19), (20) and (21).

(19) *John talks to himself.*

(20) *John talks to him.*

(21) *He talks to John.*

In Example (19), the anaphora "*himself*" could only refer to the subject of the sentence "*John*", because of the special use of reflexive pronoun in English. But conventionally, in Example (20), pronoun "*him*" should not take "*John*" as antecedent. If the object really co-refers with the subject, then a reflexive pronoun should be used. In the last sentence, appearing in the object position, there is no possibility for "*John*" to co-refer with the subject "*he*", which might else refer to some other pre-existing concepts or proper names. Although BT mainly focus on intra-sentential anaphora, we would like to know how general and adaptabe the new framework is. In [Hui04], the author implemented the BT in DRT. Here, I would like to describe the accessibility constraint in BT under the new framework.

5.4.1 Principle A

Reflexive pronoun is a special kind of anaphora. As described in Principle A, it can only refer back to the antecedent within its local domain. Here, we can understand the "local domain"

as the domain covered by the main verb, which takes the reflexive pronoun as object, at the same time takes its antecedent as subject. Thus, we assume that, for reflexive pronouns, they do not need to select antecedents from the list structure. The subject in the local domain is the one whom it co-refers with. As a result, we propose the following interpretation for reflexive pronoun "*himself*":

$[\![himself]\!] = \lambda\psi e\phi.\exists x.\psi x e\phi$

While for verbs that takes a reflexive pronouns as object, we will add one more condition in their λ-expressions, for example:

$[\![talk\_to]\!] = \lambda os.s(\lambda x.o(\lambda ye\phi.\mathbf{talk\_to}(x,y) \wedge x = y \wedge \phi e))$

On account of this, the interpretation of Example (19) can be constructed as:

$[\![talk\_to]\!][\![himself]\!]$
$= \lambda os.s(\lambda x.o(\lambda ye\phi.\mathbf{talk\_to}(x,y) \wedge x = y \wedge \phi e))(\lambda\psi e'\phi'.\exists x'.\psi x' e'\phi')$
$\rightarrow_\beta \lambda s.s(\lambda x.(\lambda\psi e'\phi'.\exists x'.\psi x' e'\phi')(\lambda ye\phi.\mathbf{talk\_to}(x,y) \wedge x = y \wedge \phi e))$
$\rightarrow_\beta \lambda s.s(\lambda x.(\lambda e'\phi'.\exists x'.(\lambda ye\phi.\mathbf{talk\_to}(x,y) \wedge x = y \wedge \phi e)x' e'\phi'))$
$\rightarrow_\beta \lambda s.s(\lambda x.(\lambda e'\phi'.\exists x'.(\mathbf{talk\_to}(x,x') \wedge x = x' \wedge \phi' e')))$
$\rightarrow_\alpha \lambda s.s(\lambda x.(\lambda e\phi.\exists y.(\mathbf{talk\_to}(x,y) \wedge x = y \wedge \phi e)))$

$([\![talk\_to]\!][\![himself]\!])[\![John]\!]$
$= \lambda s.s(\lambda x.(\lambda e\phi.\exists y.(\mathbf{talk\_to}(x,y) \wedge x = y \wedge \phi e)))(\lambda\psi e'\phi'.\psi\mathbf{j}e'(\lambda e'.\phi'(\mathbf{j} :: e')))$
$\rightarrow_\beta (\lambda\psi e'\phi'.\psi\mathbf{j}e'(\lambda e'.\phi'(\mathbf{j} :: e')))(\lambda x.(\lambda e\phi.\exists y.(\mathbf{talk\_to}(x,y) \wedge x = y \wedge \phi e)))$
$\rightarrow_\beta \lambda e'\phi'.(\lambda x.(\lambda e\phi.\exists y.(\mathbf{talk\_to}(x,y) \wedge x = y \wedge \phi e)))\mathbf{j}e'(\lambda e'.\phi'(\mathbf{j} :: e'))$
$\rightarrow_\beta \lambda e'\phi'.(\lambda\phi.\exists y.(\mathbf{talk\_to}(\mathbf{j},y) \wedge \mathbf{j} = y \wedge \phi e'))(\lambda e'.\phi'(\mathbf{j} :: e'))$
$\rightarrow_\beta \lambda e'\phi'.(\exists y.(\mathbf{talk\_to}(\mathbf{j},y) \wedge \mathbf{j} = y \wedge (\lambda e'.\phi'(\mathbf{j} :: e'))e'))$
$\rightarrow_\beta \lambda e'\phi'.(\exists y.(\mathbf{talk\_to}(\mathbf{j},y) \wedge \mathbf{j} = y \wedge \phi'(\mathbf{j} :: e')))$
$\rightarrow_\alpha \lambda e\phi.\exists y.(\mathbf{talk\_to}(\mathbf{j},y) \wedge \mathbf{j} = y \wedge \phi(\mathbf{j} :: e))$

The above expression correctly expresses the meaning of the sentence. The co-referential relation is illustrated by "$John = y$". And "*John*", the possible referent for future processing is added in the selection list. One point to notice here is that we add the equality relation in the expression of the main verb while not in the reflexive pronoun. Therefore, most transitive verb may have 2 forms of expression, one for reflexive pronoun, one for the rest. We need the information from the context to trigger which form to use. This might run into the danger of expression overflow, so it might not be a perfect solution.

5.4.2 Principle B & C

In order to see how the new framework works for Principle B and C, let's first construct the interpretation for Example (20):

$([\![talk\_to]\!][\![him]\!])[\![John]\!]$
$= \lambda s.s(\lambda x.(\lambda e\phi.\mathbf{talk\_to}(x, sel_{him}e) \wedge \phi e))(\lambda\psi e'\phi'.\psi\mathbf{j}e'(\lambda e'.\phi'(j :: e')))$
$\rightarrow_\beta (\lambda\psi e'\phi'.\psi\mathbf{j}e'(\lambda e'.\phi'(j :: e')))(\lambda x.(\lambda e\phi.\mathbf{talk\_to}(x, sel_{him}e) \wedge \phi e))$
$\rightarrow_\beta \lambda e'\phi'.(\lambda x.(\lambda e\phi.\mathbf{talk\_to}(x, sel_{him}e) \wedge \phi e))\mathbf{j}e'(\lambda e'.\phi'(j :: e'))$
$\rightarrow_\beta \lambda e'\phi'.(\lambda\phi.\mathbf{talk\_to}(\mathbf{j}, sel_{him}e') \wedge \phi e')(\lambda e'.\phi'(j :: e'))$
$\rightarrow_\beta \lambda e'\phi'.(\mathbf{talk\_to}(\mathbf{j}, sel_{him}e') \wedge (\lambda e'.\phi'(j :: e'))e')$
$\rightarrow_{\beta,\alpha} \lambda e\phi.(\mathbf{talk\_to}(\mathbf{j}, sel_{him}e) \wedge \phi(j :: e))$

We can see that through the normal construction, pronoun "*him*" can not select from the list containing "*John*". In other words, "*him*" could not be bound to "*John*". This is just what Principle B tells us. Now, if we do the same thing for Example (21), the result coincides with what Principle C describes: "*John*" can not be bound to the subject "*he*".

$(\llbracket talk\_to \rrbracket \llbracket John \rrbracket) \llbracket he \rrbracket$
$= \lambda s.s(\lambda x.(\lambda e\phi.\mathbf{talk\_to}(x, \mathbf{j}) \wedge \phi(\mathbf{j} :: e)))(\lambda \psi e' \phi'.\psi(sel_{he} e')e'\phi')$
$\rightarrow_\beta (\lambda \psi e' \phi'.\psi(sel_{he} e')e'\phi')(\lambda x.(\lambda e\phi.\mathbf{talk\_to}(x, \mathbf{j}) \wedge \phi(\mathbf{j} :: e)))$
$\rightarrow_\beta \lambda e' \phi'.(\lambda x.(\lambda e\phi.\mathbf{talk\_to}(x, \mathbf{j}) \wedge \phi(\mathbf{j} :: e)))(sel_{he} e')e'\phi'$
$\rightarrow_\beta \lambda e' \phi'.(\mathbf{talk\_to}(sel_{he} e', \mathbf{j}) \wedge \phi'(\mathbf{j} :: e'))$
$\rightarrow_\alpha \lambda e\phi.(\mathbf{talk\_to}(sel_{he} e, \mathbf{j}) \wedge \phi(\mathbf{j} :: e))$

To sum up, we can say the new framework is adaptive enough for BT. For Principle A, one modification in the verb interpretation is needed; while for Principle B and C, everything keeps the same. The reason is that the way to construct interpretation for single sentence will block the possibility of co-referential relation within the same domain (the domain covered by the main verb). This constraint is actually carried out by Formula 1, which achieves the "dynamic" for the whole system.

5.5 Proper Names

Earlier in this chapter, when we described the way that the new framework handles conditional sentences and negations, a "conditional operator" and a "negation operator" have been introduced. Accordingly, the interpretations for Example (18) and (13) can correctly be constructed. However, if we turn to Example (22) and (23), the solution will become problematic.

(22) *If John loves Mary, then he smiles at her.*

(23) *John does not like Mary. She often makes him angry.*

We can see that when the context involves proper names, the proper names should be accessible for future reference. In DRT, proper names are treated as universal variables. It means they are accessible for all sub-DRSs. However, in our previous proposal, for conditional sentences, the "conditional operator" will block every variables introduced within the conditional phrase; for negation cases, the variables introduced by the object will be blocked. This will definitely cause problem when proper names appear in those "blocking positions". In order to solve this problem, we here propose another operator "proper name operator", which only works at the last stage of processing, adding variables related to proper names back into the list. We can take Example (22) as a detailed demonstration.

$(\llbracket love \rrbracket \llbracket Mary \rrbracket) \llbracket John \rrbracket \rightarrow_\beta \lambda e\phi.(\mathbf{love}(\mathbf{j}, \mathbf{m}) \wedge \phi(\mathbf{j} :: \mathbf{m} :: e))$

$(\llbracket smile\_at \rrbracket \llbracket her \rrbracket) \llbracket he \rrbracket \rightarrow_\beta \lambda e\phi.(\mathbf{smile\_\ at}(sel_{he}(e), sel_{her}(e)) \wedge \phi e)$

$\llbracket con\_op \rrbracket \llbracket S_1 \rrbracket \llbracket S_2 \rrbracket$
$= (\lambda DSe\phi.(\neg \llbracket D \rrbracket e(\lambda e'.\neg \llbracket S \rrbracket e'(\lambda e''.\mathbf{T}))) \wedge \phi e) \llbracket S_1 \rrbracket \llbracket S_2 \rrbracket$
$\rightarrow_\beta \lambda e\phi.(\neg(\lambda e''\phi''.(\mathbf{love}(\mathbf{j}, \mathbf{m}) \wedge \phi''(\mathbf{j} :: \mathbf{m} :: e')))e(\lambda e'.\neg(\lambda e'''\phi'''.(\mathbf{smile\_\ at}(sel_{he}(e'''), sel_{her}(e''')) \wedge \phi'''e'''))e'(\lambda e''.\mathbf{T}))) \wedge \phi e$
$\rightarrow_\beta \lambda e\phi.(\neg(\lambda \phi''.(\mathbf{love}(\mathbf{j}, \mathbf{m}) \wedge \phi''(\mathbf{j} :: \mathbf{m} :: e)))(\lambda e'.\neg(\lambda \phi'''.(\mathbf{smile\_\ at}(sel_{he}(e'), sel_{her}(e')) \wedge \phi'''e'))(\lambda e''.\mathbf{T}))) \wedge \phi e$

$\rightarrow_\beta \lambda e\phi.(\neg(\lambda\phi''.(\mathbf{love}(\mathbf{j},\mathbf{m})\wedge\phi''(\mathbf{j}::\mathbf{m}::e)))(\lambda e'.\neg((\mathbf{smile\_\ at}(sel_{he}(e'),sel_{her}(e'))\wedge(\lambda e''.\mathbf{T})e'))))\wedge\phi e$
$\rightarrow_\beta \lambda e\phi.(\neg(\lambda\phi''.(\mathbf{love}(\mathbf{j},\mathbf{m})\wedge\phi''(\mathbf{j}::\mathbf{m}::e)))(\lambda e'.\neg((\mathbf{smile\_\ at}(sel_{he}(e'),sel_{her}(e'))\wedge\mathbf{T}))))\wedge\phi e$
$\rightarrow_\beta \lambda e\phi.(\neg((\mathbf{love}(\mathbf{j},\mathbf{m})\wedge(\lambda e'.\neg((\mathbf{smile\_\ at}(sel_{he}(e'),sel_{her}(e'))\wedge\mathbf{T})))(\mathbf{j}::\mathbf{m}::e))))\wedge\phi e$
$\rightarrow_\beta \lambda e\phi.(\neg((\mathbf{love}(\mathbf{j},\mathbf{m})\wedge(\neg((\mathbf{smile\_\ at}(sel_{he}(\mathbf{j}::\mathbf{m}::e),sel_{her}(\mathbf{j}::\mathbf{m}::e))\wedge\mathbf{T}))))))\wedge\phi e$
$\rightarrow_\beta \lambda e\phi.(\neg(\mathbf{love}(\mathbf{j},\mathbf{m})\wedge(\neg\mathbf{smile\_\ at}(\mathbf{j},\mathbf{m}))))\wedge\phi e$
$\rightarrow_\beta \lambda e\phi.(\mathbf{love}(\mathbf{j},\mathbf{m})\rightarrow\mathbf{smile\_\ at}(\mathbf{j},\mathbf{m}))\wedge\phi e$

Now, let's look at the expression for proper name operator:

$$[\![pname\_op]\!]=\lambda\psi e\phi.(\psi e(\lambda e.\phi(a::b::\cdots::e))) \tag{6}$$

In the list "$a::b::\cdots::e$", "a", "b"$\cdots$ are all variables introduced by the proper names in the current context. The function of Formula 6 is just to add those variables back to the list. Like in Example (22) and (23), the list is "$\mathbf{j}::\mathbf{m}::e$". The proper name list is not difficult to obtain because when designing the system, we can simply add one module to distinguish between proper names and other kinds of NPs. So the result of Example (22) after applying "proper name operator" is:

$[\![pname\_op]\!]([\![con_op]\!][\![S]\!])$
$=\lambda\psi e\phi.(\psi e(\lambda e.\phi(\mathbf{j}::\mathbf{m}::e)))(\lambda e'\phi'.((\mathbf{love}(\mathbf{j},\mathbf{m})\rightarrow\mathbf{smile\_\ at}(\mathbf{j},\mathbf{m}))\wedge\phi' e'))$
$\rightarrow_\beta \lambda e\phi.((\lambda e'\phi'.((\mathbf{love}(\mathbf{j},\mathbf{m})\rightarrow\mathbf{smile\_\ at}(\mathbf{j},\mathbf{m}))\wedge\phi' e'))e(\lambda e.\phi(\mathbf{j}::\mathbf{m}::e)))$
$\rightarrow_\beta \lambda e\phi.(((\mathbf{love}(\mathbf{j},\mathbf{m})\rightarrow\mathbf{smile\_\ at}(\mathbf{j},\mathbf{m}))\wedge(\lambda e.\phi(\mathbf{j}::\mathbf{m}::e))e))$
$\rightarrow_\beta \lambda e\phi.((\mathbf{love}(\mathbf{j},\mathbf{m})\rightarrow\mathbf{smile\_\ at}(\mathbf{j},\mathbf{m}))\wedge\phi(\mathbf{j}::\mathbf{m}::e))$

As a result, both "*John*" and "*Mary*", namely the proper names of the conditional phrase, have been inserted back into the list in such way. They become accessible for future reference again. It is exactly the same case for negation. If there is a proper name in the object position, like "*Mary*" in Example (23), it should be handled universally. The "negation operator" will first delete it from the list, and by applying the "proper name operator", it will be added back.

5.6 Chapter Summary

In this chapter, we mainly investigated the accessibility constraints for several linguistic phenomena and proposed the corresponding solutions based on the new framework. I will briefly summarize the interpretation construction process for different phenomena here as a conclusion for this chapter.

1. Construction for Scope Ambiguity
   Trigger: Quantified NPs (*existential* or *universal*)
   Operation: Two interpretations for the main verb (subject wide scope & object wide scope) are activated, users can select one of the underspecified semantic representation

2. Construction for Conditional Phrase
   Trigger: Linguistic clues for conditional phrases (such as "*if...then...*")
   Operation: Apply the two part declarative forms of the conditional phrase to "condition operator" (3), then apply the result to the "proper name operator" (6)

3. Construction for Negation
   Trigger: Linguistic clues for negation phrase (such as "*not*")
   Operator: Apply the declarative form of the negation phrase to "negation operator" (5), then apply the result to the "proper name operator" (6)

4. Construction for Binding Theory
   Trigger: Appearance of reflexive pronoun (such as "*X-self*" in English)
   Operator: The reflexive interpretation of the main verb is activated, which contains the equality relation between the subject and object

6 Implementation of Accessibility Constraints in ACG

Abstract Categorial Grammars (ACG), which was first raised in [dG01], is a new categorial formalism based on [Gir87]. ACG is established not as a completely new grammatical formalism, rather, it should be considered as a kernel of grammatical framework that has the ability to encode the other already existing grammars [dG02]. Generally speaking, ACG provides a direct match between two languages (abstract language and object language) by using some simple logical function and compositional rules. [dG01] offers a complete definition for ACG, [dG02] and [dGP04] prove ACG's expressive power by exemplifying with Tree Adjoining Grammar (TAG) and Context Free Grammar (CFG). In the first part of this chapter, I will briefly introduce ACG. After that, the implementation of accessibility constraints described in the last chapter will be carried out.

6.1 Definition of ACG

In order to define ACG, we need to know about several other important background concepts.

Concept 1 Linear Implicative Types $I(A)$:
Let A be a set of atomic types, the linear implicative types build on A is:
- If $a \in A$, then $a \in I(A)$;
- If $\alpha, \beta \in I(A)$, then $(\alpha \to \beta) \in I(A)$.

Concept 2 Vocabulary (Higher-order Linear Signature):
A vocabulary is a triple $\Sigma = \langle A, C, \tau \rangle$, in which:
- A is a finite set of atomic types;
- C is a finite set of constants;
- $\tau : C \to I(A)$ is a function that assigns a linear implicative types in $I(A)$ to each constant in C.

Concept 3 Linear λ-terms:
Let X be an infinite countable set of λ-variables, given a vocabulary $\Sigma = \langle A, C, \tau \rangle$, the set $\Lambda(\Sigma)$ of linear λ-terms built upon it is:
- If $c \in C$, then $c \in \Lambda(\Sigma)$;
- If $x \in X$, then $x \in \Lambda(\Sigma)$;
- If $x \in X$, $t \in \lambda(\sigma)$, and x occurs free in t exactly once, then $(\lambda x.t) \in \Lambda(\Sigma)$;
- If $t, u \in \Lambda(\Sigma)$, and the sets of free variables of t and u are disjoint, then $(tu) \in \Lambda(\Sigma)$.

Concept 4 Lexicon:
For two vocabularies $\Sigma_1 = \langle A_1, C_1, \tau_1 \rangle$ and $\Sigma_2 = \langle A_2, C_2, \tau_2 \rangle$, the lexicon L from Σ_1 to Σ_2 is defined as a pair $\langle F, G \rangle$, in which:
- $F : A_1 \to I(A_2)$ is a function that interprets the atomic types of Σ_1 as linear implicative types built upon A_2;

- $G : C_1 \rightarrow \Lambda(A_2)$ is a function that interprets the constants of Σ_1 as linear λ-terms built upon Σ_2;

Based on these concepts, we can define ACG as follows:

Definition Abstract Categorial Grammar (ACG):
An ACG is a quadruple $G = \langle \Sigma_1, \Sigma_2, L, s \rangle$, in which:
- $\Sigma_1 = \langle A_1, C_1, \tau_1 \rangle$ and $\Sigma_2 = \langle A_2, C_2, \tau_2 \rangle$ are two vocabularies, Σ_1 is called the abstract vocabulary and Σ_2 is called the object vocabulary;
- $L : \Sigma_1 \rightarrow \Sigma_2$ is a lexicon mapping from the abstract vocabulary to the object vocabulary;
- $s \in I(A_1)$ is a type of the abstract vocabulary, it is called the distinguished type of the grammar.

To sum up, ACG generates two languages (vocabularies/higher-order linear signatures): the abstract language, which could be thought as the abstract grammatical models; and the object language, which could be thought as the concrete concepts and meanings related to the abstract models. For example, by using ACG, we can build the connection between the grammatical formalism (such as CFG) and the real appearance of the language (such as the strings of English).

6.2 Implementation in ACG Toolkit

6.2.1 ACG Desgining

Currently, a toolkit for developing ACG signatures and lexicons is available here:

http://www.loria.fr/equipes/calligramme/acg/.

This toolkit is developed in the CALLIGRAMME team in LORIA\INRIA Grand Est Nancy. In order to test the portability of the new framework in [dG06] and the accessibility constraints in the previous chapter, we implemented them in the toolkit. The test ACG $\langle \Sigma_1, \Sigma_2, L, s \rangle$ is designed as follows.

$\Sigma_1 = \langle A_1, C_1, \tau_1 \rangle$, the abstract signature, is named "syntax vocabulary", its structure is described in Figure 2.

$\Sigma_2 = \langle A_2, C_2, \tau_2 \rangle$, the object signature, is named "logic vocabulary", its structure is described in Figure 3.

Finally, the lexicon $L : \Sigma_1 \rightarrow \Sigma_2$, is constructed as in Figure 4.

In the above ACG, we use "v" to replace the symbol "γ" in order to denote the type of left context. At the same time, "l" (left context) and "r" (right context) respectively serves as "e" and "ϕ" in the calculus introduced previously. Also, the list connector "::" is replaced by "@". There is not specific reason for doing this, just for the convenience of input to the toolkit. If we input a well formed expression under the syntax vocabulary, the system will first check the validity of its type. If correct, it will create the corresponding semantic interpretation automatically. For instance, with input:

```
input>>LOVE MARY JOHN : s
```

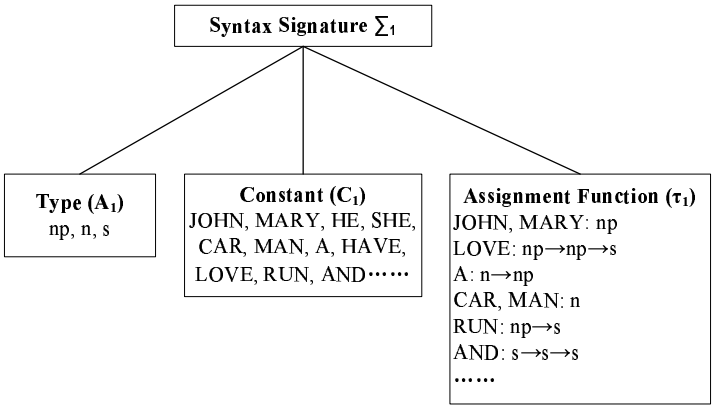


Figure 2: Syntax Signature Σ_1 (Simplied)

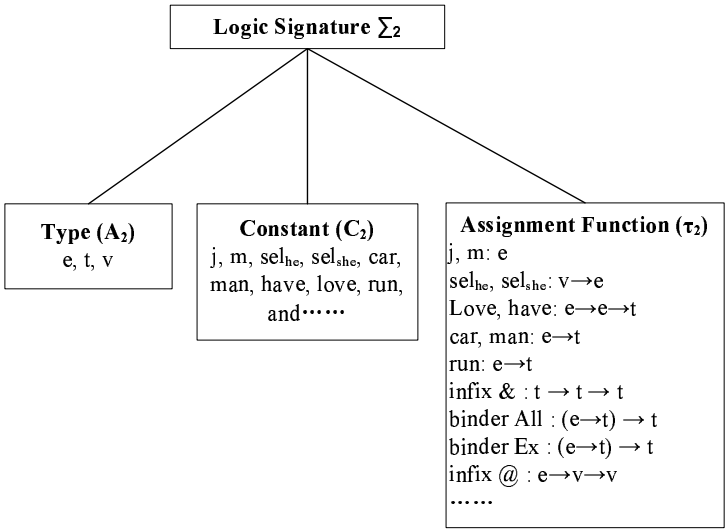


Figure 3: Logic Signature Σ_2(Simplied)

[9] the system will return the correct result after lambda calculus:

```
output<<lambda l r. (love j m) & (r (m @ (j @ l))) : v->(v->t)->t
```

However, if we input:

```
input>>LOVE MARY JOHN : np
```

an error message will appear, because there is a mistake with the type of the input expression. Now let's try an example with two sentences connected by "AND", which is expressed by Formula 1. With the input:

[9]Input for a ACG system includes two part, the expression and its type under that signature. "*s*" here denotes the type of "*LOVE MARY JOHN*" in the syntax signature.

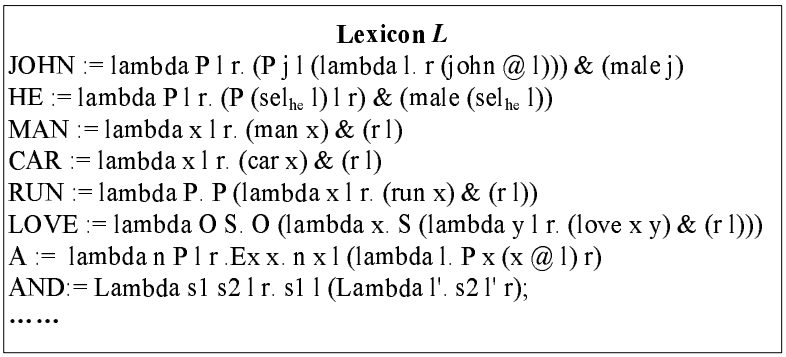


Figure 4: Lexicon L(Simplied)

```
input>>AND (LOVE MARY JOHN) (SMILE HE) : s
```

The system will return:

```
output<<lambda l r. (love j m) & ((smile (selhe (m @ (j @ l)))) & (r (m @ (j @ l))))
: v->(v->t)->t
```

Note that both "**j**" and "**m**" are in the list structure for selection function "sel_{he}" to choose from, also they are stored for future processing. A more detailed description of the ACG designing (the syntax signature, logic signature and the lexicon) could be found in the Appendix C.1, C.2 and C.3. More examples and the input, output format can refer to the Appendix C.4.

6.2.2 Rewriting & Selection

We might notice that in the above section, with numbers of parenthesis and other symbols, the output is not that readable. That is because in the current ACG toolkit, no standard rewriting program exists. Which means the system could not even recognize some easy syntax of logical formulas, such as "$(A \wedge (B \wedge C))$" equals to "$(A \wedge B \wedge C)$". Since directly modifying the code of ACG might corrupt the whole structure of the toolkit, I used Python script to build a rewriting program for ACG output, rendering the toolkit a more readable result format. The rewrting program mainly focuses on 3 kinds of problems, namely the "redundant parenthesis problem", the "@ operator problem" and the "truth value **T** problem". More details on the structure of the program can refer to the Appendix B.1. After processing, the system is able to modify the previous output into:

```
output<<lambda l r. (love j m) & (smile (selhe (m @ j @ l))) & (r (m @ j @ l))
: v->(v->t)->t
```

Finally, in order to activate the selection function in the new framework, which takes a list as argument and returns an element in the list, we propose to add the gender information in the interpretations for NPs. For example, the λ-term for "*John*", "*Mary*", "*he*" will respectively be:

$[\![John]\!] = \lambda\psi e\phi.\psi \mathbf{j} e(\lambda e.\phi(\mathbf{j} :: e)) \wedge \mathbf{male}(\mathbf{j})$
$[\![Mary]\!] = \lambda\psi e\phi.\psi \mathbf{j} e(\lambda e.\phi(\mathbf{m} :: e)) \wedge \mathbf{female}(\mathbf{m})$
$[\![he]\!] = \lambda\psi e\phi.\psi(sel_{he} e) e\phi \wedge \mathbf{male}(sel_{he} e)$

Also under the Python environment, I constructed another script for selection function based on gender. It will only assign male object to "sel_{he}", female object to "sel_{she}". Combining together with the rewriting program, once again look at the previous example, if we input

```
input>>AND (LOVE MARY JOHN) (SMILE HE) : s
```

the output

```
output<<lambda l r. (((love j m) & (((smile (selhe (m @ (j @ l)))) & (r (m @ (j
@ l)))) & (male (selh (m @ (j @ l)))))) & (male j)) & (female m) : v->(v->t)->t
```

will be returned, after the rewriting and selection program, a final look of the semantic interpretation could be achieved:

```
output<<lambda l r. (love j m) & (smile j) & (r (m @ j @ l)) & (male j) &
(female m) : v->(v->t)->t
```

More details about the structure of the selection program could refer to the Appendix B.2.

7 Conclusions

In this thesis, we first briefly discussed the task of anaphora resolution, DRT and one of its key concepts: Accessibility. After that, we introduced a recent work which can successfully express discourse dynamics based on the traditional Montague Grammar, with only a concept of "context" (*left context* and *right context*) introduced, everything else keeps the same as in the classical way.

Then we proved the flexibility of [dG06]'s approach through modeling various types of accessibility constraints. We proposed two interpretations for the main verb, which would either take the subject wide scope or the object wide scope, in order to distinguish the the *de re* and *de dicto* readings in quantification scope ambiguity. The accessibility for conditional phrase and negation phrase was respectively satisfied by applying "conditional operator" and "negation operator". Like the special treatment for proper names in DRT, we also presented the "proper name operator" to handle proper names in the new framework. Binding Theory, as an additional proof, appeared to work in the new framework too. Finally, those theoretical proposals were implemented in the ACG toolkit, which could be a really good illustration for the concept of compositionality. To sum up, the framework in [dG06] resolves the compositionality problem when expressing DRT in the Montague Grammar. Based on the various proposed operators and mechanisms for different linguistic phenomena, we can say [dG06]'s idea is rather flexible in modeling accessibility constraints.

At the end of the thesis, there are some comments and possible future work.

In Chapter 6, in order to obtain a better result for the implementation, we proposed a kind of interpretation with extra lexical information (gender). Although the examples given were rather simple, but the idea ignites that we could "plug in" more information in the λ-expressions. If presenting in a DRT box style, this idea could be:

$L_1(Discourse_Referent)$
$L_2(Lexical_Information)$
$L_3(Conditions)$

In Level 1 and Level 3, we have the conventional discourse referent and conditions. But in Level 2, the so-called "world knowledge" is included. If we consider DRS as a complete model in speakers' minds, the "world knowledge" level should be an essential part of this model. For example, when we hear "*John*", not only will we put variable "**j**" in the referent list, also we know that in most cases, "*John*" refers to a male object in English. The set of lexical information not only contains simple features like gender, number, but also some complex ones such as semantic consistency. All these information, together with the accessibility constraints, would strongly support anaphora resolution. Because all those information could be treated as normal conditions, they also hold a conjunction relation between each other. As a result, it is obviously implementable in [dG06]'s framework. As mentioned previously in Section 3.2.2, one of the similar existing work is [LA01], but the idea there is to embed rhetoric structure and discourse relation into DRT.

As showed in section 5.4, the way to handle Principle A of BT is to raise two interpretations for the main verb. Accordingly, the appearance of reflexive pronoun will be a trigger for different interpretation. However, this treatment is not that satisfactory. On one hand, we might run into the danger of formula flooding; on the other hand, it is not a general enough solution. This could also be one topic for future work.

Finally, some more complex linguistic phenomena, such as temporal information, modality, definite NPs are not mentioned here. For instance, anaphora resolution for definite NPs, which requires more discourse and lexical information, is quite different from for indefinite NPs. The λ-term for definite determiner "*the*" need further inverstigated and improved. The adaptability of the new framework in all the above situations could be explored in future.

References

[Bon06] Roberto Bonato. *An Integrated Computational Approach to Binding Theory.* PhD thesis, Università degli Studi di Verona, Université Bordeaux 1, 2006.

[Bür05] Daniel Büring. *Binding Theory.* Cambridge. Cambridge University Press, Cambridge, 2005.

[Cho81] Noam Chomsky. *Lectures on Government and Binding.* Foris, Dordrecht, 1981.

[Chu40] Alonzo Chuch. A formulation of the simple type theory of types. *Journal of Symbolic Logic*, 1940.

[dG01] Philippe de Groote. Towards abstract categorial grammars. In *Association for Computational Linguistics, 39th Annual Meeting and 10th Conference of the European Chapter, Proceedings of the Conference*, pages 148–155, 2001.

[dG02] Philippe de Groote. Tree-adjoining grammars as abstract categorial grammars. In *TAG+6, Proceedings of the sixth International Workshop on Tree Adjoining Grammars and Related Frameworks*, pages 145–150. Università di Venezia, 2002.

[dG06] Philippe de Groote. Towards a montagovian account of dynamics. *Proceedings of Semantics and Linguistic Theory XVI*, 2006.

[dG07] Philippe de Groote. Yet another dynamic logic. Presentation at the 4th Lambda Calculus and Formal Grammar workshop, September 2007.

[dGP04] Philippe de Groote and Sylvain Pogodalla. On the expressive power of abstract categorial grammars: Representing context-free formalisms. *Journal of Logic, Language and Information*, 13(4):421–438, 2004.

[Gir87] Jean-Yves Girard. Linear logic. *Theoretical Computer Science*, 50:1–102, 1987.

[GS89] Jeroen Groenendijk and Martin Stokhof. Dynamic montague grammar. In *Papers from the Second Symposium on Logic and Language*, pages 3–48. Akademiai Kiadoo, 1989.

[GS91] Jeroen Groenendijk and Martin Stokhof. Dynamic predicate logic. *Linguistics and Philosophy*, 14(1):39–100, 1991.

[Hei83] Irene Heim. File change semantics and the familiarity theory of definiteness. In Rainer Bäuerle, Christoph Schwarze, and Arnim von Stechow, editors, *Meaning, Use, and Interpretation of Language*, pages 164–189. Walter de Gruyter, Berlin, 1983.

[Hir81] Graeme Hirst. Discourse-oriented anaphora resolution in natural language understanding: A review, 1981.

[Hui04] Janneke Huitink. Implementing the binding theory in drt. 2004.

[Kam81] Hans Kamp. A theory of truth and semantic representation. In J. Groenendijk, T. Janssen, and M. Stokhof, editors, *Formal Methods in the Study of Language*, Mathematical Centre Tracts 135, pages 277–322. Mathematisch Centrum, Amsterdam, 1981.

[Kau01] Barbara Kaup. Negation and its impact on the accessibility of text information. *Memory & Cognition*, 29(7):960–967, 2001.

[Kel88] William R Keller. Nested cooper storage: The proper treatment of quantification in ordinary noun phrases. In U. Reyle and C. Rohrer, editors, *Natural Language Parsing and Linguistic Theories*, pages 432–447. Reidel, Dordrecht, 1988.

[KR93] Hans Kamp and Uwe Reyle. *From Discourse to Logic: Introduction to Model-theoretic Semantics, Logic and Discourse Representation Theory*. Kluwer Academic Publishers, 1993.

[LA01] Alex Lascarides and Nicholas Asher. Segmented discourse representation theory: Dynamic semantics with discourse structure. *COMPUTING MEANING*, 3, 2001.

[Mon73] Richard Montague. The proper treatment of quantification in ordinary english. In J. Hintikka, J. Moravcsik, and P. Suppes, editors, *Approaches to Natural Language*. Reidel, Dordrecht, 1973. Reprinted in [Mon74].

[Mon74] Richard Montague. English as a formal language. In R.H.Thomason, editor, *Formal Philosophy: Selected Papers of Richard Montague*. Yale University Press, New Haven, Connecticut, 1974.

[Mus96] Reinhard Muskens. Combining montague semantics and discourse representation, 1996.

[Pog08] Sylvain Pogodalla. Exploring a type-theoretic approach to accessibility constraint modelling. Technical report, LORIA/INRIA, May 2008.

[Rus99] Mitkov Ruslan. Anaphora resolution: the state of the art. 1999.

[SH08] Noa Shuval and Barbara Hemforth. Accessibility of negated constituents in reading and listening. *Intercultural Pragmatics*, 5(4):445–469, November 2008.

[vH00] Klaus von Heusinger. Anaphora, antecedent and accessibility. *Theoretical Linguistics*, 26(1-2):75–93, 2000.

[WK05] Gregory Ward and Andrew Kehler. *Syntactic Form and Discourse Accessibility*, chapter 3, pages 365–385. John Benjamins Publishing Co, 2005.

Appendices

A λ-Expressions

Category	Example	Type	λ-Expression
Noun	*man*	$\iota \rightarrow [\![s]\!]$	$[\![man]\!] = \lambda xe\phi.\mathbf{man}(x) \wedge \phi e$
	car		$[\![car]\!] = \lambda xe\phi.\mathbf{car}(x) \wedge \phi e$
Noun Phrase	*John*	$(\iota \rightarrow [\![s]\!]) \rightarrow [\![s]\!]$	$[\![John]\!] = \lambda\psi e\phi.\mathbf{j}e(\lambda e.\phi(\mathbf{j} :: e))$
	he		$[\![he]\!] = \lambda\psi e\phi.\psi(sel_{he}e)e\phi$
Transitive Verb	*love*	$[\![np]\!] \rightarrow [\![np]\!] \rightarrow [\![s]\!]$	$[\![love]\!] = \lambda os.s(\lambda x.o(\lambda ye\phi.\mathbf{love}(x, y) \wedge \phi e))$
	*love**		$[\![love*]\!] = \lambda os.o(\lambda y.s(\lambda xe\phi.\mathbf{love}(x, y) \wedge \phi e))$
Intransitive Verb	*smile*	$[\![np]\!] \rightarrow [\![s]\!]$	$[\![smile]\!] = \lambda s.s(\lambda xe\phi.\mathbf{smile}(x) \wedge \phi e)$
Determiner	*a/some*	$[\![n]\!] \rightarrow [\![np]\!]$	$[\![a]\!] = \lambda n\psi e\phi.\exists x.nxe(\lambda e.\psi x(x :: e)\phi)$
	every		$[\![every]\!] = \lambda n\psi e\phi.\forall x.\neg(nxe(\lambda e.\neg(\psi x(x :: e)(\lambda e.\mathbf{T})))) \wedge \phi e$
Relative Pronoun	*who*	$([\![np]\!] \rightarrow [\![s]\!]) \rightarrow [\![n]\!] \rightarrow [\![n]\!]$	$[\![who]\!] = \lambda rnxe\phi.nxe(\lambda e.r(\lambda\psi.\psi x)e\phi)$
Adjective	*red*	$[\![n]\!] \rightarrow [\![n]\!]$	$[\![red]\!] = \lambda\psi xe\phi.(\mathbf{red}(x) \wedge \psi xe\phi)$

Table 1: λ-Expression for Words of Different Categories [10]

Name	Type	λ-Expression
Normal Connector	$[\![s]\!] \rightarrow [\![s]\!] \rightarrow [\![s]\!]$	$[\![n\_con]\!] = \lambda S_1S_2e\phi.[\![S_1]\!]e(\lambda e'.[\![S_2]\!]e'\phi)$
Condition Operator	$[\![s]\!] \rightarrow [\![s]\!] \rightarrow [\![s]\!]$	$[\![con\_op]\!] = \lambda S_1S_2e\phi.\neg([\![S_1]\!]e(\lambda e'.\neg[\![S_2]\!]e'\phi))$
Negation Operator	$[\![s]\!] \rightarrow [\![s]\!]$	$[\![neg\_op]\!] = \lambda\phi s.s(\lambda x.(\lambda e\phi.\neg\phi(\lambda Q.Qxe(\lambda e'.(\mathbf{T} \wedge \phi e)))))$
P_Name Operator	$[\![s]\!] \rightarrow [\![s]\!]$	$[\![pname\_op]\!] = \lambda\psi e\phi.(\phi e(\lambda e.\phi(a :: b :: \cdots :: e)))$

Table 2: λ-Expression for Operators

[10] In order to save space, $[\![s]\!]$ is equal to $\gamma \rightarrow (\gamma \rightarrow o) \rightarrow o$ in all the above type descriptions, and $[\![n]\!]$, $[\![np]\!]$ can refer to the type for noun and noun phrase. The basic types are only ι, o and γ.

B Rewriting & Selection Program

B.1 Rewriting System

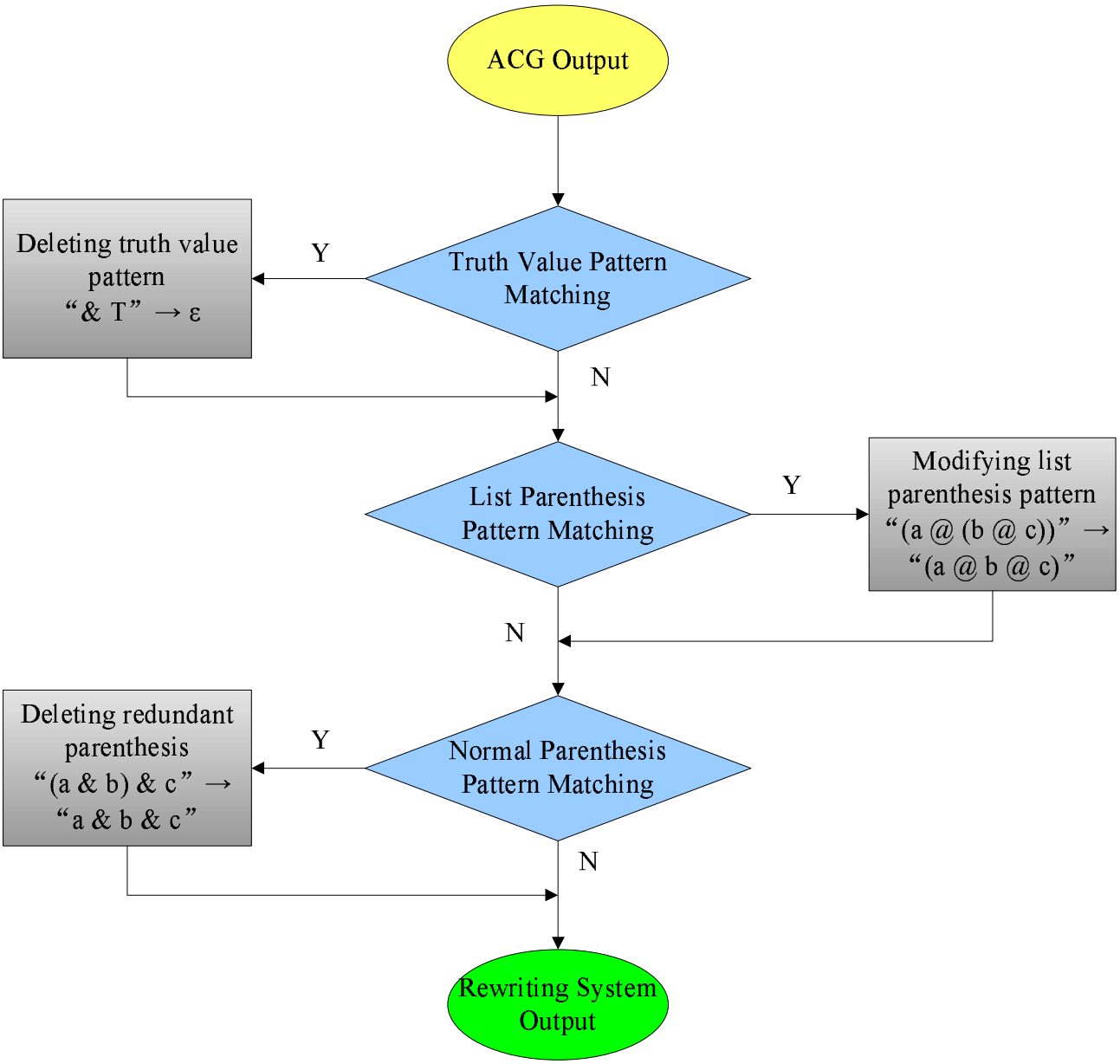


Figure 5: Rewriting System Module

B.2 Selection Function

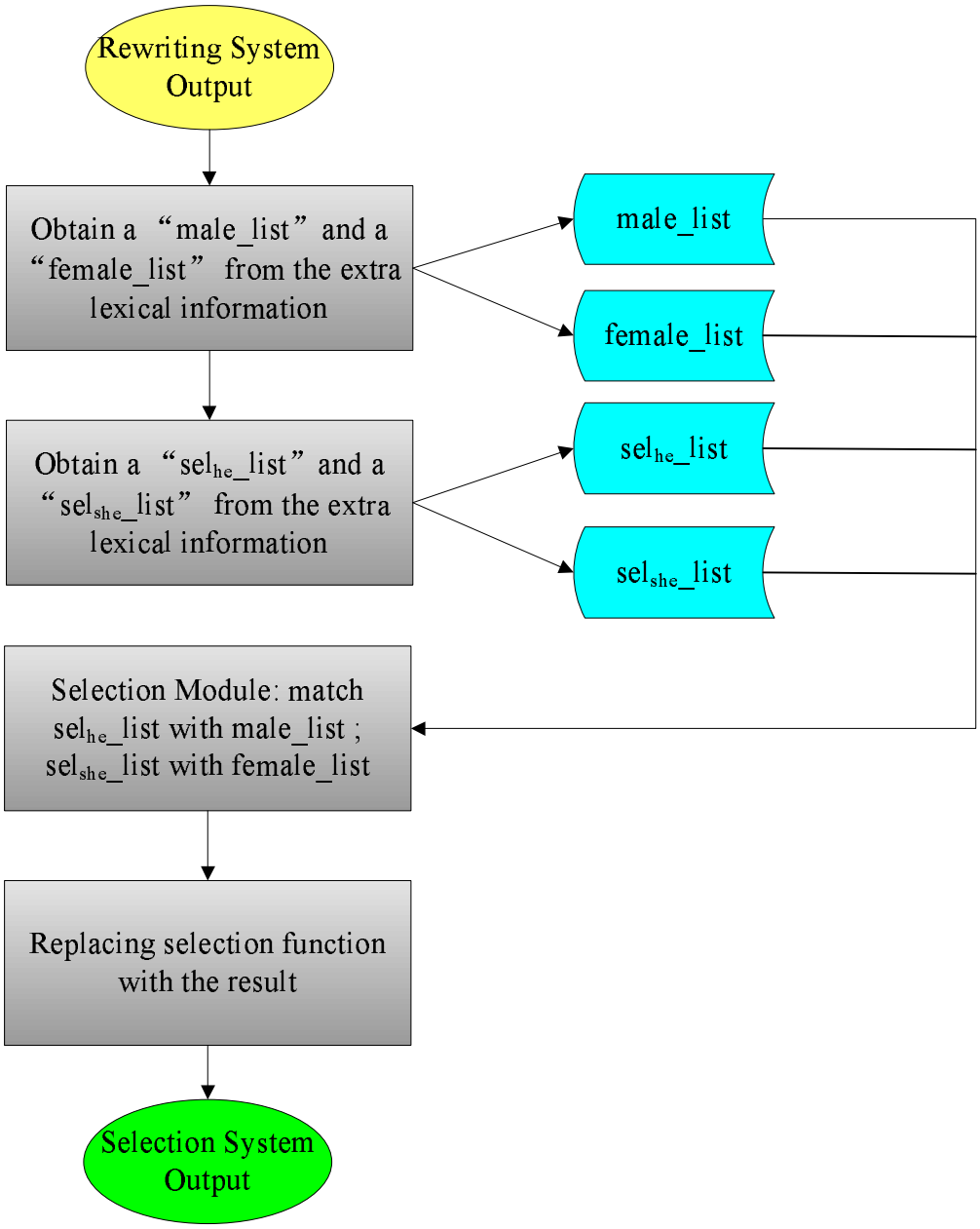


Figure 6: Selection Function Module

C ACG Implementation

C.1 Syntax Signature Σ_1

Type (A_1)	**Constant** (C_1)	**Assignment Function** (τ_1)
n, np, s	*CAR* *MAN* *WOMAN* *FARMER* *DONKEY*	n
	JOHN *MARY* *HE/HIM* *SHE/HER* *IT*	np
	LOVE *LOVE** *OWN* *SMILE_AT* *BEAT*	$np \to np \to s$
	SMILE	$np \to s$
	A *EVERY*	$n \to np$
	WHO	$(np \to s) \to n \to n$
	AND *CON*	$s \to s \to s$
	NEG	$(np \to s) \to (np \to s)$

Table 3: Detailed Syntax Signature Σ_1

C.2 Logic Signature Σ_2

Type (A_2)	Constant (C_2)	Assignment Function (τ_2)
e, t, v	**car** **man** **woman** **farmer** **donkey**	$e \rightarrow t$
	j **m**	e
	$sel_{he/him}$ $sel_{she/her}$ sel_{it}	$v \rightarrow t$
	love **own** **smile_at** **beat**	$e \rightarrow e \rightarrow t$
	smile	$e \rightarrow t$
	Infix &	$t \rightarrow t \rightarrow t$
	Infix @	$e \rightarrow v \rightarrow v$
	Binder **All** Binder **Ex**	$(e \rightarrow t) \rightarrow t$
	Binder **Ne**	$t \rightarrow t$

Table 4: Detailed Logic Signature Σ_2 [11]

[11] For the sake of system input, we replace the original type symbols, which are represented by latin characters, with the common character. ι, o, γ are respectively changed into e, t, v.

C.3 Lexicon L

Function	Σ_1	Σ_2
$F : A_1 \Rightarrow I(A_2)$	n np s	$e \rightarrow [\![s]\!]$ $(e \rightarrow [\![s]\!]) \rightarrow [\![s]\!]$ $v \rightarrow (v \rightarrow t) \rightarrow t$
$G : C_1 \Rightarrow \Lambda(A_2)$	*CAR* *MAN* *WOMAN* *FARMER* *DONKEY*	lambda x l r. (**N**(x)) & (r l) ***N*** refers to **car/man/woman/...**
	JOHN *MARY*	lambda P l r.(P **N** l (lambda l. r(**N** @ l))) ***N*** refers to **j/m**
	$sel_{he/him}$ $sel_{she/her}$ sel_{it}	lambda P l r. (P(sel_N l)l r) sel_N refers to $sel_{he/him}$ or $sel_{she/her}$ or sel_{it}
	LOVE *LOVE** *OWN* *SMILE_AT* *BEAT*	lambda O S. O (lambda x. S(lambda y l r. **N**(x,y)&(rl))) ***N*** refers to **love/own/beat...**
	SMILE	lambda P. P(lambda x l r. (**smile**(x)&(r l)))
	A	lambda n P l r. Ex x. n x l(lambda l. Px(x@l)r)
	EVERY	lambda n S l r. (All x. Ne (n x l (Lambda l. Ne (S x (x @ l)(Lambda l. T))))) & (r l)
	AND	lambda s1 s2 l r. s1 l (lambda l'. s2 l' r)
	CON	lambda s1 s2 l r. (Ne (s1 l(lambda l'. Ne(s2 l'(lambda l". T)))))& (r l);
	NEG	lambda P S. S (lambda x.(lambda l r.((Ne(P (lambda Q. Q x) l (lambda l'. T)))&(r l))

Table 5: Detailed Lexicon L [12]

[12]Because character "e" has already been used to represent type, in order not to confuse, we use "l" (left context) and "r" (right context) to replace the previous "e" and "ϕ" in the λ-term. Also, other main changes include: the ψ symbol is replaced by "P"; list connector "::" is replaced by @ symbol; negation symbol "$\neg$" is replaced by "Ne".

C.4 Test Data

C.4.1 No Lexical Information

[13]

1. input>>LOVE (A WOMAN) (EVERY MAN):s
   output1<<lambda l r.(All x. Ne((man x) & (Ne(Ex x'. (woman x') &
   ((love x x') & T))))) & (r l): v->(v->t)->t

2. input>>LOVE* (A WOMAN) (EVERY MAN):s
   output1<<lambda l r. Ex x. (woman x) & ((All x'. Ne((man x') &
   (Ne((love x' x) & T)))) & (r (x@l))): v->(v->t)->t

3. input>>CON (LOVE (A WOMAN) (A MAN)) (SMILE HE):s
   output1<<lambda l r.(Ne(Ex x.(man x)&(Ex x'.(woman x')&((love x x')&
   (Ne((run(selhe(x'@(x@l))))&T))))))&(r l): v->(v->t)->t
   output2<<lambda l r.(Ne(Ex x.(man x)&(Ex x'.(woman x')&(love x x')&
   (Ne(run(selhe(x'@x@l)))))))&(r l): v->(v->t)->t

4. input>>(NEG (HAVE (A CAR))) JOHN:s
   output1<<lambda l r. (Ne (Ex x. (car x) & ((have j x) & T))) &
   (r (j @ l)): v->(v->t)->t
   output2<<lambda l r. (Ne (Ex x. (car x) & (have j x))) & (r (j @ l))
   : v->(v->t)->t

5. input>>AND (NEG (HAVE (A CAR)) JOHN) (IS_RED IT):s
   output1<<lambda l r.(Ne(Ex x.(car x) & ((have j x) & T))) & ((is_red
   (selit (j @ l))) & (r (j @ l))): v->(v->t)->t
   output2<<lambda l r.(Ne(Ex x.(car x) & (have j x))) & (is_red (selit
   (j @ l))) & (r (j @ l)): v->(v->t)->t

C.4.2 With Lexical Information

1. input>>:LOVE MARY JOHN: s
   output1<<lambda l r.(((love j m)&(r(j@(m@l))))&(female m))
   &(male j): v->(v->t)->t
   output2<<lambda l r.(love j m)&(r(j@m@l))&(female m)&
   (male j): v->(v->t)->t

[13] "No Lexical Information" means the gender information is not added into the λ-term. For example, "*John*" is interpreted as "$[\![JOHN]\!] = lambdaPlr.(P\mathbf{j}l(lambdal.r(\mathbf{j}@l)))$". And output1 denotes the result directly from the ACG system, output2 denotes the result post-processed by the rewriting and selection program.

14

2. input>>:AND (LOVE MARY JOHN) (SMILE HE): s
   output1<<lambda l r.(((love j m)&(((smile(selhe(j@(m@l))))&
   (r(j@(m@l))))&(male(selhe(j@(m@l))))))&(female m))&(male j)
   : v->(v->t)->t
   output2<<lambda l r.(love j m)&(smile j)&(r(j@m@l))&(female m)
   &(male j): v->(v->t)->t

[14]When gender information is added, the λ-term for "*John*" is "$[\![JOHN]\!]$ = $lambdaPlr.(P\mathbf{j}l(lambdal.r(\mathbf{j}@l)))\&male(\mathbf{j})$". Similarly, the gender information for "*Mary*" and "*he*" is added by simple conjunction in the original formula, that is how the result comes out.