A UNIFICATION-BASED SEMANTIC INTERPRETATION FOR COORDINATE CONSTRUCTS

Jong C. Park
University of Pennsylvania
Computer and Information Science
200 South 33rd Street
Philadephia, PA 19104-6389 USA
Internet: park@linc.cis.upenn.edu

Abstract

This paper shows that a first-order unification-based semantic interpretation for various coordinate constructs is possible without an explicit use of lambda expressions if we slightly modify the standard Montagovian semantics of coordination. This modification, along with partial execution, completely eliminates the lambda reduction steps during semantic interpretation.

1 Introduction

Combinatory Categorial Grammar (CCG) has been offered as a theory of coordination in natural language (Steedman [1990]). It has usually been implemented in languages based on first order unification. Moore [1989] however has pointed out that coordination presents problems for first-order unification-based semantic interpretation. We show that it is possible to get over the problem by compiling the lambda reduction steps that are associated with coordination in the lexicon. We show how our first-order unification handles the following examples of coordinate constructs.

(1.1) Harry walks and every farmer walks.
(1.2) A farmer walks and talks.
(1.3) A farmer and every senator talk.
(1.4) Harry finds and a woman cooks a mushroom.
(1.5) Mary gives every dog a bone and some policeman a flower.

We will first start with an illustration of why standard Montagovian semantics of coordination cannot be immediately rendered into a first-order unification strategy. The lexicon must contain multiple entries for the single lexical item "and", since only like categories are supposed to conjoin. For example, the lexical entry for "and" in (1.1) specifies the constraint that the lexical item should expect on both sides sentences to give a sentence.

Moore [1989] predicts that a unification-based semantic interpretation for sentences which involve for example noun phrase coordination won't be possible without an explicit use of lambda expressions, though there are cases where some lambda expressions can be eliminated by directly assigning values to variables embedded in a logical-form expression. The problematic example is shown in (1.6), where proper noun subjects are conjoined.

(1.6) john and bill walk.

The argument is that if we do not change the semantics of "john" from `j` to λ`P.P(j)`, where `P` is a second order variable for property in the Montagovian sense[1], then the single predicate λ`X.walk(X)` should accommodate two different constants `j` and `b` in a single variable `X` at the same time. Since the unification simply blocks in this case, the argument goes, we need to use higher order lambda expressions such as λ`P.P(j)` or λ`P.P(b)`, which when conjoined together, will yield semantics for e.g. "john and bill" as

$$\lambda \texttt{P.(P(j) \& P(b))} .$$

Combined finally with the predicate, this will result in the semantics (1.7), after lambda reduction.

(1.7) `walk(j) & walk(b)`

[1] Montague [1974]. $\lambda P^{\vee} P(j)$ to be exact, taking intensionality into account. The semantics of the predicate "walks" will then be $(^{\wedge}\lambda X.walk(X))$.

Although Moore did not use quantified noun phrases to illustrate the point, his observation generalizes straightforwardly to the sentence (1.3). In this case, the semantics of "and", "every" and "some" (or "a") will be (1.8) a, b, and c, respectively.

(1.8) (a) λQ.λR.λP.(Q(P) & R(P))
(b) λS.λP'.forall(X,S(X)=>P'(X))
(c) λS.λP''.exists(X,S(X)&P''(X))

Thus, after four lambda reduction steps, one for each of Q, R, P' and P'', the semantics of "a farmer and every senator" will be

λP.(exists(X,farmer(X)&P(X)) &
forall(X,senator(X)=>P(X))) ,

as desired.

Moore's paper showed how lambda reduction could be avoided by performing lambda reduction steps at *compile* time, by utilizing the lexicon, instead of doing them at run time. Consider again (1.8a). The reason why this formulation requires *four* subsequent lambda reduction steps, not three, is that the property P should be applied to each of the conjuncts, requiring two separate lambda reduction steps. Suppose that we try to eliminate these two lambda reduction steps at compile time by making the argument of the property P explicit in the lexicon, following the semantics (1.9).

(1.9) λQ.λR.λP.(Q(λX.P(X)) & R(λX.P(X)))

The first-order variable X ranges over the set of individuals, and the hope is that after lambda reduction it will be bound by the quantifiers, such as forall, embedded in the expressions denoted by the variables Q and R. Since the same variable is used for both constructs, however, (1.9) works only for pairs of quantified noun phrases, which don't provide constants, but not for pairs involving proper nouns, which do provide constants. Incidentally, this problem is particular to a unification approach, and there is nothing wrong with the semantics (1.9), which is equivalent to (1.8a). This unification problem cannot be avoided by having two distinct variables Y and Z as in (1.10) either, since there is only one source for the predicate property for the coordinate noun phrases, thus there is no way to isolate the argument of the predicate and assign distinct variables for it at compile time.

(1.10) λQ.λR.λP.(Q(λY.P(Y)) & R(λZ.P(Z)))

The way we propose to eliminate the gap between (1.9) and (1.10) is to introduce some spurious binding which can always be removed subsequently. The suggestion then is to use (1.11) for the semantics of "and" for noun phrase conjunction.

(1.11) Semantics of "and" for NP Conjunction:
λQ.λR.λP.(Q(λY.exists(X,X=Y&P(X)))
& R(λZ.exists(X,X=Z&P(X))))

This satisfies, we believe, the two requirements, one that the predicate have the same form, the other that the variables for each conjunct be kept distinct, at the same time. The rest of the lambda expressions can be eliminated by using the notion of partial execution (Pereira & Shieber [1987]). Details will be shown in Section 3, along with some "more immediate but faulty" solutions. It is surprising that the same idea can be applied to some fairly complicated examples as (1.5), and we believe that the solution proposed is quite general.

In order to show how the idea works, we use a first-order Montagovian Intensional Logic (Jowsey [1987]; Jowsey [1990]) for a semantics. We apply the proposal to CCG, but it could equally well be applied to any lexicon based grammar formalism. We explain briefly how a CCG works in the first part of Section 2. As for the semantics, nothing hinges on a particular choice, and in fact the code we show is devoid of some crucial features of Jowsey's semantics, such as indices for situations or sortal constraints for variable binding. We present the version of Jowsey's semantics that we adopt for our purposes in the second part of Section 2, mainly for completeness. In Section 3, each of the cases in (1.1) through (1.5), or variations thereof, is accounted for by encoding lexical entries of "and", although only (1.3) and (1.5) depend crucially on the technique.

We have a few words for the organization of a semantic interpretation system we are assuming in this paper. We imagine that it consists of two levels, where the second level takes a *scope-neutral* logical form to produce every possible, genuinely ambiguous, scoping possibilities in parallel and the first level produces this scope-neutral logical form from the source sentence. We assume that our second level, which we leave for future research, will not be very different from the one in Hobbs & Shieber [1987] or Pereira & Shieber [1987]. The goal of this paper is to show how the scope-neutral logical forms are de-

rived from natural language sentences with coordinate constructs. Our "scope-neutral" logical form, which we call "canonical" logical form (CLF), *syntactically* reflects derivation-*dependent* order of quantifiers since they are derived by a derivation-dependent sequence of combination. We emphasize that this derivation-dependence is an artifact of our illustrative example, and that it is not an inherent consequence of our technique.

2 Background Formalisms

A Combinatory Categorial Grammar

The minimal version of CCG we need to process our examples contains four reduction rules, (2.1) through (2.4), and two type raising rules, (2.5) and (2.6), along with a lexicon where each lexical item is assigned one or more categories. For the reasons why we need these, the reader is referred to Steedman [1990].

(2.1) Function Application (>): `X/Y Y => X`
(2.2) Function Application (<): `Y X\Y => X`
(2.3) Function Composition (>B): `X/Y Y/Z => X/Z`[2]
(2.4) Function Composition (<B): `Y\Z X\Y => X\Z`
(2.5) Type Raising, Subject (>T): `np => s/(s\np)`
(2.6) Type Raising, Backward (<T): `np => X\(X/np)`

The present fragment is restricted to the *basic* categories `n`, `np` and `s`.[3] Derived categories, or *categories*, are recursively defined to be basic categories combined by directional symbols (/ or \). Given a category `X/Y` or `X\Y`, we call `X` the *range* category and `Y` the *domain* category. Parentheses may be used to change the left-associative default. The semantics part to be explained shortly, (2.7a) through (2.7e) show examples of a common noun, a proper noun, a quantifier, an intransitive verb, a sentential conjunction, respectively.

(2.7) Sample Lexicon
(a) `cat(farmer, n:X^farmer(X)).`
(b) `cat(harry, np:A1^(h^B)^B).`
(c) `cat(every, np:(X^A)^(X^B)^forall(X,A=>B)`
`/n:X^A).`
(d) `cat(walks, s:S\np:(X^A)^(X^walk(X))^S).`
(e) `cat(and, (s:(S1 & S2)\s:S1)/s:S2).`[4]

A First-Order Montague Semantics

In this section, we will focus on describing how Jowsey has arrived at the first-order formalism that we adopt for our purposes, and for further details, the reader is referred to Jowsey [1987] and Jowsey [1990]. The reader can safely skip this section on a first reading since the semantics we use for presentation in Section 3 lacks many of the new features in this section.

Montague's PTQ analysis (Dowty, Wall & Peters [1981]) defines an intensional logic with the basic types `e`, `t` and `s`, where `e` is the type of entities, `t` the type of truth values and `s` the type of indices. Derived types `<a,b>` and `<s,a>` are recursively defined over the basic types. A name, which is of type `e`, denotes an individual; individual concepts are names relativized over indices, or functions from indices to the set of individuals. Individual concepts are of type `<s,e>`. A predicate denotes a set of individuals, or a (characteristic) function from the set of individuals to truth values. Properties are intensional predicates, or functions from indices to the characteristic functions. Properties are of type `<s,<e,t>>`, or `<e,<s,t>>`. A formula denotes a truth value, and propositions are intensional formulas, thus of type `<s,t>`.

By excluding individual concepts, we can ensure that only truth values are relativized over indices, and thus a modal (omega-order) logic will suffice to capture the semantics. For this purpose, Jowsey defines two basic types `e` and `o`, where `o` corresponds to the type `<s,t>`, and then he defines derived types `<a,b>`, where `a` and `b` range over basic types and derived types. The logic is then made into first-order by relying on a fixed number of sorts and eliminating recursively defined types. These sorts include `e`, `s`, `o`, `p` and `q`, which correspond to the types `e`, `s`, `<s,t>`, `<e,<s,t>>` and `<<e,<s,t>>,<s,t>>` respectively in an omega-order logic.

For a full exposition of the logic, the reader is referred to Jowsey [1990]. For our presentation, we

[2] In Steedman [1990], this rule is conditioned by $Z \neq$ `s\np` in order to prevent such constructs as "*[Harry] but [I doubt whether Fred] went home" or "*[I think that Fred] and [Harry] went home."

[3] For simplicity, we do not show variables for gender, case, tense, and number. Larger fragment would include pp, etc.

[4] The category `(s\s)/s` has the potential danger of allowing the following construct, if combined with the rule <B: "*Mary finds a man who $[\text{walks}]_{s\backslash np}$ $[\text{and he talks}]_{s\backslash s}$." The suggestion in Steedman [1990] is to add a new pair of reduction rules, X $[X]_{\&}$ => X and conj X => $[X]_{\&}$, together with the category of "and" as conj. Thus, the category of "and harry talks" is now $[s]_{\&}$, blocking the unwanted combination.

will simplify the semantics and drop intensionality altogether. We also drop the sortal constraint, since our examples do not include belief operators and hence the only variables left are of sort e.

3 A First-Order Unification

We will follow the standard technique of combining the syntactic information and the semantic information as in (3.1), where up-arrow symbols ('^')[5] are used to give structures to the semantic information for partial execution (Pereira & Shieber [1987]), which has the effect of performing some lambda reduction steps at compile time.

(3.1) Basic Categories
(a) n: $(d_e \hat{} d_o)$
(b) np: $(d_e \hat{} d_o) \hat{} (d_e \hat{} r_o) \hat{} s_o$
(c) s: s_o

The term d_o in (3.1a) and (3.1b) encodes domain constraint for the variable d_e. Likewise, the term r_o in (3.1b) specifies range constraint for d_e. The term s_o in (3.1b) and (3.1c) encodes the sentential constraint associated with a sentence. In order to avoid possible confusion, we shall henceforth call categories without semantic information "syntactic" categories.

In this section, we will develop lexical entries for those coordinate constructs in (1.1) through (1.5), or variations thereof. For each case, we will start with "more immediate but faulty" solutions and present what we believe to be the correct solution in the last. (For those who want to skip to the correct lexical entries for each of the cases, they are the ones not commented out with %.) We have seen the lexical entry for sentential conjunction in (2.7d). The lexical entry for predicate conjunction can be similarly encoded, as in (3.2).

(3.2) Lexical Entry for Predicate Conjunction

```
cat(and, ((s:S\np:A^(X^(B1 & B2))^S)
           \(s:S1\np:A^(X^B1)^S1))
          /(s:S2\np:A^(X^B2)^S2)).
```

When the conjoined predicates are combined with the subject noun phrase, the subject NP provides only the domain constraint, through A in the first line. The range constraints in the last two NP categories guarantee that B1 and B2 will bear the same variable X in them, so that they can be safely put as the range constraint of the first NP category. The CLF for (1.2) from (3.2) is shown in (3.3).

(3.3)
```
exists(X1,farmer(X1)&(walk(X1)&
talk(X1)))
```

Let us turn to noun phrase coordination, e.g., (1.3). The first try, on the model of predicate conjunction, would be:

(3.4) Lexical Entry for NP Conjunction:

```
%cat(and, (np:A^(X^D)^(B & C)
%          \np:A1^(Y^D)^B)
%         /np:A2^(Z^D)^C).
```

The intention is to collect the two domain constraints via A1 and A2, to get the range constraint from D in the first line, and then to combine them by joining the two sentential constraints B and C of the domain categories. This idea however does not work, since the variables Y and Z do not appear in the range constraint D. As a result, (3.4) will give the following ill-formed CLF for (1.3).

```
% exists(X1,farmer(X1)&talk(X3))
% &forall(X2,senator(X2)=>talk(X3))
```

We therefore need to use distinct variables in place of D for the two range constraints which will have the same predicate symbol for their range categories. Using the Prolog predicate univ ('=..'), we can correct (3.4) as follows:[6]

(3.5) Lexical Entry for NP Conjunction:

```
%cat(and, (np:A^(X^D)^(B & C)
%          \np:A1^(Y^B1)^B)
%         /np:A2^(Z^C1)^C) :-
%             D =.. [Pred, X],
%             B1 =.. [Pred, Y],
%             C1 =.. [Pred, Z].
```

This is an explicit case of a first-order simulation of second order variables. Unfortunately, this does not work, for several reasons.[7] First, this handles predicates of arity 1 only, and we need to know the type of each argument if we want to provide a different category for each predicate of different arity. Second, this can not be combined with predicate coordination, for example, such as "john and

[5] Not to be confused with Montague's hacek symbol, '^'.

[6] D =.. [P,X] succeeds if D is unifiable with P(X).

[7] One implementation-dependent reason is that the Prolog requires at least one of the two variables D and Pred to be already instantiated for the univ to work. This can not be expected when the noun phrase conjunction is being processed, since we don't yet know what predicate(s) will follow.

a woman walk and talk," or some complex verbs that may require several predicates, such as "believes", since it assumes only one predicate for the range constraint.

The solution we propose is to use the revised semantics of "and" in (1.11) instead. That is, we expect (3.6) from (1.3):

(3.6) Proposed Semantics of (1.3):

```
exists(X1,farmer(X1)
&(exists(X2,(X2=X1)&talk(X2))))
&forall(X3,senator(X3)
=>(exists(X2,(X2=X3)&talk(X2))))
```

We need to distinguish the variable X2 in the second line from the variable X2 in the fourth line, via something like α conversion, since in the present form, the Prolog will consider them as the same, while they are under distinct quantifiers. In fact, since we are separating the semantic interpretation into two levels, we can further process the CLF at the second semantic interpretation level to eliminate those spurious bindings such as exists(X,(X=␣)&␣) along with variable renaming to derive the logical form (3.7) from (3.6):

(3.7)
```
exists(X1,farmer(X1)&talk(X1))
&forall(X3,senator(X3)=>talk(X3))
```

(3.8) produces the CLF (3.6) for (1.3).

(3.8) Lexical Entry for NP Conjunction:

```
cat(and,
   (np:A^(X^D)^(B & C)
    \np:A1^(Y^(exists(X,(X=Y)&D)))^B)
   /np:A2^(Z^(exists(X,(X=Z)&D)))^C).
```

The reason why we are able to maintain in the two domain categories two different forms of range contraints is that the only place that will unify with the actual range constraint, i.e., the predicate, is the range constraint part of the range category *only*. We note in passing that Jowsey provided yet another approach to noun phrase coordination, a generalized version of his idea as shown below.

(3.8a) Lexical Entry for NP Conjunction :

```
cat(and,
(np:(X^A)^(X^D)^B
 \np:(Y^A1)^(Y^C)^B)
/np:(Z^A2)^(Z^forall(X,(X=Y v X=Z)=>D))^C).
```

For example, (3.8a) will give the following semantics for (1.3).

```
exists(X1,farmer(X1)&forall(X2,senator(X2)
=>forall(X3,(X3=X1 v X3=X2)=>talk(X3))))
```

This approach has its limits, however, as indicated in the footnote 8.

We now turn to some of the non-standard constituent coordination. First, consider (1.4), which is an instance of Right Node Raising (RNR). The CCG syntactic category of the conjunction "and" in this case is (C\C)/C, where C is s/np. (3.9) shows one derivation, among others, for (1.4). The syntactic category of "finds" is (s\np)/np.

(3.9) One derivation for (1.4).

```
harry   finds  and a woman    cooks a mushroom
-----T -----  --- -------T -----  ----------
s/(s\np) ....      s/(s\np)  .....     np
------------>B    --------------->B
    s/np                 s/np
               ------------------>
                  (s/np)\(s/np)
-----------------------------------<
            s/np
         ------------------------------>
                        s
```

Assuming that the category of "finds" is as follows,

(3.10) Lexical Entry for "finds":

```
cat(finds, ((s:S\np:A1^(X^A)^S)
          /np:A2^(Y^find(X,Y))^A)).
```

here is the first try for the RNR "and."

(3.11) Lexical Entry for RNR Conjunction:

```
%cat(and, ((s:S/np:A^(X^(B1&B2))^S1)
%          \(s:S/np:A^(X^B1)^S1)
%         /(s:S3/np:A^(X^B2)^S2).
```

For example, (3.11) will produce the CLF (3.12) for the sentence "harry finds and mary cooks a mushroom."

(3.12)
```
exists(X1,mushroom(X1)&find(h,X1)&
      cook(m,X1))
```

However, this works only for pairs of proper nouns. For example, for the sentence "every man finds and a woman cooks a mushroom," it will give the ill-formed CLF (3.13) where the domain constraint for the noun phrase "a woman" is gone and X3 is therefore unbound. This happens because the sentential constraint S2 is not utilized for the final sentential constraint.

(3.13)
```
%forall(X1,man(X1)=>exists(X2,
      %mushroom(X2)&find(X1,X2)
      %&cook(X3,X2)))
```

Putting the two sentential constraints S1 and S2 together as follows does not work at all, since the relation between S and S0 is completely undefined, unlike the ones between S1 and B1 and between S2 and B2.

```
%cat(and, ((s:S/np:A^(X^(S1&S2))^S0)
%          \(s:S1/np:A1^(X^B1)^B1))
%         /(s:S2/np:A2^(X^B2)^B2)).
```

This problem is corrected in (3.14), which will produce the CLF (3.15) for (1.4):

(3.14) Lexical Entry for RNR Conjunction.

```
cat(and, ((s:S/np:A^(X^(S1&S2))^S)
          \(s:S1/np:A1^(X^B1)^B1))
         /(s:S2/np:A2^(X^B2)^B2)).
```

(3.15) Semantics of (1.4) from (3.14):

```
exists(X1,mushroom(X1)&find(h,X1))
&exists(X2,woman(X2)&cook(X2,X1)))
```

(1.5) shows another case of non-standard constituent coordination, which we will call an instance of Left Node Raising (LNR). The syntactic category of "and" for LNR is (C\C)/C where C is (s\np)\(((s\np)/np)/np). (3.16) shows one syntactic derivation for (1.5). The syntactic category of "gives" is ((s\np)/np)/np.

(3.16) One derivation for (1.5), fragment.

```
every dog                          a bone
---------<T                        ------<T
((s\np)/np)\(((s\np)/np)/np)
                        (s\np)\((s\np)/np)
-----------------------------------<B
      (s\np)\(((s\np)/np)/np)
```

Again, we assume that the category of "gives" is:

(3.17) Lexical Entry for "gives":

```
cat(gives, ((s:S1\np:A1^(X^S2)^S1)
            /np:A2^(Y^give(X,Z,Y))^B)
           /np:A3^(Z^B)^S2).
```

(3.18) shows the first try for the lexical entry.[8]

(3.18) Lexical Entry for LNR Conjunction.

```
%cat(and,
%  (((s:_\np:_)
%     \(((s:S\np:(X^A)^(X^(S4 & S6))^S)
%        /np:A1^(Y^B)^S1)/np:A2^(Z^S1)^S2))
%    \((s:_\np:_)\(((s:_\np:_)
%         /np:A3^(Y^B)^S3)
%        /np:A4^(Z^S3)^S4)))
%   /((s:_\np:_)\(((s:_\np:_)
%        /np:A5^(Y^B)^S5)
%       /np:A6^(Z^S5)^S6))).
```

It gives the CLF (3.19) for (1.5):

(3.19) Semantics of (1.5) from (3.18):

```
forall(X1,dog(X1)=>exists(X2,bone(X2)
&give(m,X1,X2)))&exists(X1,policeman(X1)
&exists(X2,flower(X2)&give(m,X1,X2)))
```

Unfortunately, (3.18) favors quantified nouns too much, so that when any proper noun is involved in the conjunction the constant for the proper noun will appear incorrectly in the two sentential constraints at the same time. It seems that the only way to resolve this problem is to create *four* variables, Y1, Y2, Z1 and Z2, at the semantics level, similar to idea in (1.11). (3.20) implements this proposal.

(3.20) Lexical Entry for LNR Conjunction.

```
cat(and,
  (((s:_\np:_)
    \(((s:S
       \np:(X^A)^(X^(S4 & S6))^S)
       /np:A1^(Y^B)^S1)
      /np:A2^(Z^S1)^S2))
  \((s:_\np:_)
     \(((s:_\np:_)
        /np:A3^(Y1^
          (exists(Y,(Y=Y1)&B)))^S3)
       /np:
 A4^(Z1^(exists(Z,(Z=Z1)&S3)))^S4)))
  /((s:_\np:_)
    \(((s:_\np:_)
       /np:
 A5^(Y2^(exists(Y,(Y=Y2)&B)))^S5)
      /np:
 A6^(Z2^exists(Z,(Z=Z2)&S5)))^S6))).
```

(3.20) will give the CLF (3.21) for (1.5).

(3.21) Semantics of (1.5) from (3.20):

```
forall(X1,dog(X1)=>exists(X2,X2=X1
&exists(X3,bone(X3)&exists(X4,X4=X3
&give(m,X2,X4)))))
&exists(X1,policeman(X1)&exists(X2,X2=X1
&exists(X3,flower(X3)&exists(X4,X4=X3
&give(m,X2,X4)))))
```

Using the technique of eliminating spurious bindings, (3.21) may be replaced by a logical form (3.22):

[8] In this case, we can no longer use the disjunctive technique such as `forall(X1,(X1= v X1= )=>give( ,X1, ))` for the CLF, since X1 is now a pair. The problem gets worse when the conjoined pairs do not have the same type of quantifiers, as in (1.5).

(3.22)
```
forall(X1,dog(X1)
    =>exists(X3,bone(X3)&give(m,X1,X3)))
    &exists(X1,policeman(X1)
    &exists(X3,flower(X3)&give(m,X1,X3)))
```

In addition to this, (3.20) gives the CLF (3.23) for (3.24),

(3.23)
```
exists(X1,X1=j&exists(X2,bone(X2)
    &exists(X3,X3=X2 &give(m,X1,X3))))
    &exists(X1,X1=b&exists(X2,flower(X2)
     &exists(X3,X3=X2 &give(m,X1,X3))))
```

(3.24) mary gives john a bone and bill a flower.

for which no CLF could be derived if we were using (3.18). This completes our demonstration for the technique.

The natural question at this point is how many lexical entries we need for the conjunct "and". If natural language makes every possible category conjoinable, the number of entries should be infinite, since function composition can grow categories unboundedly, if it can grow them at all. We predict that in natural language we can limit the conjunction arity to n, where n is the maximum arity in the lexicon.

4 Conclusion

The system described in this paper is implemented in Quintus Prolog. We expect that the approach can be extended to any lexicon-based grammar of the same power as CCG if it provides means for term unification.

The reason we choose to eliminate all the lambda expressions is that it allows uniform treatment within first-order unification, since Jowsey's results suggest that in other respects natural language semantics can be characterized in a first-order logic. As an alternative, we could choose to enforce uniform treatment within second-order unification, using the idea for example in Nadathur & Miller [1988]. Although we leave this possibility for future research, we believe that this option might turn out to be more appropriate in terms of elegance of the approach. And the resulting conceptual clarity might be exploited to design a schema for generating these entries for "and".

5 Acknowledgements

Many thanks are due to Dr. Mark Steedman, whose guidance immensely helped to improve the quality of presentation, as well as the quality of the content. I am also very grateful to Dr. Mark Johnson, who suggested, and took pains of going over in detail, another way of presenting the thesis, that resulted in the material in the introduction section. All errors are however entirely mine. The author was supported by the ARO grant DAAL03-89-C-0031PRI.

References

David R. Dowty, Robert E. Wall & Stanley Peters [1981], *Introduction to Montague Semantics*, D. Reidel Publishing Company.

Jerry R. Hobbs & Stuart M. Shieber [January-June 1987], "An Algorithm for Generating Quantifier Scopings," *Computational Linguistics* 13, 47–63.

Einar Jowsey [1987], "Montague Grammar and First Order Logic," *Edinburgh Working Papers in Cognitive Science: Categorial Grammar, Unification Grammar and Parsing* 1, 143–194.

Einar Jowsey [1990], *Constraining Montague Grammar for Computational Applications*, Doctoral Dissertation, Department of AI, University of Edinburgh.

Richard Montague [1974], in *Formal Philosophy*, Richmond H. Thomason, ed., Yale University Press.

Robert C. Moore [1989], "Unification-Based Semantic Interpretation," *Proceedings of the ACL.*

Gopalan Nadathur & Dale Miller [1988], "An Overview of λ-Prolog," *Proceedings of the Fifth International Logic Programming Conference.*

Fernando C.N. Pereira & Stuart M. Shieber [1987], *Prolog and Natural-Language Ananlysis*, CSLI Lecture Notes Number 10.

Mark J. Steedman [April 1990], "Gapping as Constituent Coordination," *Linguistics and Philosophy* 13, 207–263.