From English to PFO: A Formal Semantic Parser

Jordan Zlatev[1]
Stockholm

Abstract

Pagin and Westerståhl (1993) present a formalism called PFO (**P**redicate logic with **F**lexibly binding **O**perators) which is said to be well-suited for formalizing the semantics of natural languages. Among other things, PFO permits a compositional formalization of "donkey sentences" of the type *If a farmer owns a donkey he beats it.*

In this paper we present a formal procedure and its computer implementation (written in PROLOG) that translates from a limited fragment of English to PFO, i.e. a *formal semantic parser.* The translation is done in two steps: first a DCG grammar delivers a parse tree for the sentence; then a number of translation rules that operate on (sub)trees apply to the analysed sentence in all possible orders which may give rise to different "interpretations". For example the sentence *Every man does not love a woman* receives 6 different formalizations corresponding to the 6 possible orders of applying the universal quantification rule, the existence quantification rule and the negation rule.

Other ambiguities which the parser accounts for are those between anaphoric and deictic interpretations of pronouns: for the sentence in the first paragraph the parser will provide a formalization in which the variable for *he* is co-indexed with that for *farmer* (the "anaphoric" interpretation) and a formalization with a new variable (the "deictic" one).

1. Introduction

PFO, which stands for **P**redicate logic with **F**lexibly binding **O**perators, is a logical formalism developed by Peter Pagin and Dag Westerståhl (cf. Pagin & Westerståhl 1993, hence P & W). Its novelty consists in the fact that it permits a compositional formalization of certain problematic natural language constructions not by extending the semantics of first-order predicate logic (PL) as in e.g. Discourse Representation Theory (DRT, Kamp 1981), but by changing its syntax.

Section 2 reviews the motivation for developing PFO, presents it in brief, compares it to PL and shows how the first, but not the second allows for a compositional formalization of "donkey sentences". This section is closely based on P & W, sections 1–4.

---

[1]The research reported in this paper was done while participating in the project *Logic with Flexibly-Binding Operators* at the Department of Philosophy, Stockholm University during the 92-93 academic year. The rule system presented in section 3 was established after numerous discussions with Peter Pagin and Dag Westerståhl.

Proceedings of NODALIDA 1993, pages 317-329

However, Pagin and Westerståhl do not present a formal procedure for translating from sentences in a natural language such as English into PFO. It has been the task of the work reported in this paper to describe such a procedure for a limited fragment of English (comparable to the fragment presented in the classical "PTQ" paper of Richard Montague (1974) though without intensional contexts). **Section 3** will thus present a formalization of the translation from English to PFO for a number of basic linguistic constructions.

Since it is to be entirely formal, this procedure should be equally well performable by a computer program and the programming language PROLOG makes it quite straightforward to express the translation rules as computer code. Implementing the translation procedure as a computer program was a convenient way to check for the consistency of the rules, their ordering, interaction etc. Its purpose has been one of a "debugging device". It is both the translation procedure from section 3 and its implementation, which we briefly present in **section 4**, that we refer to as a "formal semantic parser".[1]

Finally, **section 5** will briefly point out some engineering and theoretical conclusions that derive from the project of implementing a translation procedure English-to-PFO.

2. A brief presentation of the PFO formalism

Through PFO, P & W challenge "... the view that certain natural language constructions with anaphoric pronouns cannot be *compositionally formalized* in predicate logic, at least not in any reasonable way". [p.189, my italics]

The principle of compositionality stating that "the meaning of a complex expression is a function of the meaning of its parts" is both vague and controversial and something more will be said about it in section 5. But the notion of a "compositional formalization" is quite straightforward: if X is a constituent of Y in NL (natural language) and X is formalized as X_{FL} and Y as Y_{FL} in FL (formal language), then X_{FL} is to be a constituent of Y_{FL} in FL.

The "natural language constructions" that do not seem to fulfil this requirement include the so-called "donkey sentences", brought to the attention of the linguistic community first by Geach (1962). Consider (1),

---

[1]Strictly speaking, as sections 3 and 4 make clear, both the formalization procedure and the implementation consist of a *syntactic parser*, which delivers a phrase structure tree, and a *translator* that in a number of consecutive steps transforms the parse tree into a PFO formula. By "formal semantic parser" we mean both parts.

which is *not* a donkey sentence, and its compositional formalization in PL, (1_{PL}).

(1) If Bill owns a car he is rich
(1_{PL}) $\exists y(car(y) \wedge owns(b,y)) \rightarrow rich(b)$

But (2), which is a donkey sentence, constitutes a problem. (2*), which is derived by analogy to (1_{PL}) is not a sentence (a well-formed formula) in PL: *y* in *drives(b,y)* is not bound. (2?), which is derived by extending the scope of $\exists$, does not have the right meaning. The "right" formalization is, of course, (2_{PL}) but it is not compositional: it does not have as constituent *$\exists$y(car(y) $\wedge$ owns(b,y))* , which is the formalization of *Bill owns a car* .

(2) If Bill owns a car he drives it
(2*) $\exists y(car(y) \wedge owns(b,y)) \rightarrow drives(b,y)$
(2?) $\exists y((car(y) \wedge owns(b,y)) \rightarrow drives(b,y))$
(2_{PL}) $\forall y((car(y) \wedge owns(b,y)) \rightarrow drives(b,y))$

PFO differs from PL in the following three respects:

(a) The *variable-binding operators* of PFO are *binary* rather than *unary*. [X,Y] expresses universal quantification and (X,Y) expresses existential quantification. Furthermore PFO fuses variable-binding and sentential operators, so that [X,Y] also expresses *material implication* between Y and Y and (X,Y) expresses *conjunction*. (3) and (4) would therefore formalize the following way in PFO and PL respectively.

(3) A man sleeps
(3_{PL}) $\exists x(man(x) \wedge sleeps(x))$
(3_{PFO}) (man x, x sleeps)

(4) Every man sleeps
(4_{PL}) $\forall x(man(x) \rightarrow sleeps(x))$
(4_{PFO}) [man x, x sleeps]

The PFO formalizations are both simpler and, in a sense, closer to natural language in providing a "subject" and "predicate" part, and not having to complement with conjunction and implication operators that have no correlate in the sentences.

(b) Variable-binding is *unselective* (PFO), rather than *selective* (PL) which means that all variables common to two immediate subformulas get bound, without any need for explicit indication. So e.g. [Px, (Qy, Rxy)] corresponds to $\forall x(Px \rightarrow \exists y(Qy \wedge Rxy))$.

(c) Finally, quantification priority is from the *outside in*, rather than from the *inside out*, so that e.g. [Px, (Qx, Rxy)]—notice the slight difference from (b) above—will correspond to $\forall x(Px \rightarrow (Qx \wedge Rxy))$.

There is much more to be said about PFO: P & W present a formal specification of its syntax and semantics, show how to perform natural deduction with it and compare it with "dynamic" logics such as DRT. Here I will end this brief presentation by returning to the donkey sentence (2) and show how in PFO it gets formalized analogously to (1), i.e. compositionally.

First both (1) and (2) get translated into an intermediary stage, which is the result of formalizing the *if-(then)* construction.

($1_{PFO'}$) [b owns a car, he is rich]
($2_{PFO'}$) [b owns a car, he drives it]

Then the first subformula in both is transformed according to the formalization rule for indefinite phrases in object position (cf. 3.2) and the pronoun *he* is substituted with the same constant as that for *Bill*.

($1_{PFO''}$) [(car y, b owns y), b is rich]
($2_{PFO''}$) [(car y, b owns y), b drives it]

And finally a pronoun interpretation rule applies to ($2_{PFO''}$) producing

($2_{PFO'''}$) [(car y, b owns y), b drives y]

The last contains as constituents the PFO formalizations of the constituents of (2), and indeed looks very similar to ($1_{PFO''}$) while getting a different kind of interpretation due to the different way of doing variable-binding in PFO. Now to the main subject of this paper: the exact rules and derivational procedure for e.g. arriving from (2) to ($2_{PFO'''}$), i.e. from English to PFO.

3. Formalizing the translation from English to PFO

It turned out convenient to divide the formalization of the translation procedure English-to-PFO into two stages: (a) a syntactic analysis of the English sentence and (b) a translation of the parse tree produced by (a) into a PFO formula. The main advantage of this modular design is that the translation rules of stage (b) can be "structure-dependent"[1]: i.e. their application can depend on non-terminal as well as on terminal symbols.

[1]Cf. Chomsky (1975) for an argument for the necessity of "structure-dependent" rules.

3.1 Syntactic analysis

The grammar used for parsing the English sentences is a context-free phrase structure grammar with the small generalization allowed by adding the morphosyntactic *features*[1] CASE (with values `nom` and `acc`) for pro-nouns and FIN(iteness) (with values `fin, inf`) for verbs. These serve as constraints on the phrase structure rules, disallowing sentences such as:

**Him* loves Mary. *John loves *he*.
*Pedro *own* a donkey. *Pedro does not *owns* a horse.

A third feature, GEN(der) (with values `fem, masc, neutr`) is marked in the lexicon for nouns and pronouns. It does not play a role in the syntactic analysis, but it does in the translation rules that deal with pronoun interpretation (cf. next subsection).

The only peculiarity of the grammar worth mentioning is the use of two noun phrase subcategories with corresponding symbols in the grammar NPs and NPq. The latter includes noun phrases that have *every* or *no* as determiners (such as *every man* or *no woman that sleeps*) while the first includes pronouns, proper names and noun phrases with determiners *a* and *the*. The reason for this is semantic: NPq:s involve rules of universal quantification for their formalization and the translation rules described in the next section require this distinction in order to avoid producing incorrect formalizations for sentences that involve disjunction. The following rules from the grammar see to it that if at least **one** of the noun phrases in a disjunction is an NPq, the whole disjoint noun phrase is an NPq.

NPs -> NPs or NPs
NPq -> NPs or NPq | NPq or NPs | NPq or NPq

The grammar in its entirety is given in *Appendix A*.

3.2 Translation rules

Once an English sentence is analysed with the help of the grammar, it is available to the translation rules. This is how the first rule used in the translation procedure looks like:

(R1) <<every <CN>cn>npq <VP>vp>s ⇒
[<<CN>cn <x>np>s, <<x>np <VP>vp>s]

[1] As in *unification-based grammars* (cf. Shieber 1986).

All rules have this common form: The left-hand side of the translation symbol, ⇒, is a *structural description.* The right-hand side is a *PFO formula.* The brackets "<" and ">" mark phrase structure (in order to avoid confusion with the PFO operators), with an index on the right specifying the syntactic category. Symbols in capital letters stand for *phrase-structure variables*, i.e. any part of the phrase-structure tree that has the category specified by its index. (As can be seen, the variable symbols and their indices coincide, so to simplify the notation we will abbreviate $<W>_W$ as W in the following). The structural description part always contains reference to some "logical word" such *every, if, or* etc. or to a pronoun, while the PFO formula has *PFO-variables* of syntactic category NP; the significance of this will be seen in a moment. (The marking of a PFO-variable with $<...>_{np}$ will also be omitted for abbreviation.)

Notice also that the structural description requires that the input to a translation rule be of syntactic category S, which is also the category of the two subformulas on the right-hand side. Rules can operate on the subformulas produced by other rules. They can apply in all possible orders and when we have reached a PFO formula on which no other rule can apply, we have a *PFO formalization* of the initial English sentence.

One (negative) consequence of the fact that rules apply on whole sentences is that rules that refer to a noun phrase in their structural description need to come in pairs: one for when this noun phrase is "subject" as in (R1) and one when it is "object", such as (R2).

(R2) $<NP <Vtr <every\ CN>_{npq} vp>_s \Rightarrow [<CN\ x>_s, <NP <Vtr\ x>vp>s]$

These are the rules of *universal quantification.* The rules of *existence quantification,* (R3) and (R4), introduce one more complication: U and W are *anonymous phrase-structure variables*, they can be instantiated by any part of the phrase-structure that otherwise fulfils the structural description.

(R3) $<<U <a\ CN>_{nps}\ W>_{np}\ VP>_s \Rightarrow (<CN\ x>_s, <<U\ x\ W>_{np}\ VP>_s)$
(R4) $<NP <U <a\ CN>_{nps}\ W>_{vp}>_s \Rightarrow (<CN\ x>_s, <NP <U\ x\ W>vp>s)$

The purpose of these variables is to allow the existential quantifier of an (indefinitly) embedded indefinite noun phrase to have a wider scope than a linearly preceding universal quantifier. If (R1) applies to (5) first (after the sentence is syntactically analysed) it will produce the PFO formula ($5_{PFO'}$).

(5) Every man who owns a donkey sleeps
($5_{PFO'}$) $[<<man\ who <owns <a\ donkey>_{nps}>vp>_{cn}\ x>_s, <x <sleeps>vp>s]$

Now (R5), which provides a conjunctive interpretation of relative clauses can apply to the left subformula to produce ($5_{PFO''}$).

(R5) <<N Comp VP>$_{CN}$ x>$_S$ => (<N x>$_S$, <x VP>$_S$)
($5_{PFO''}$) [(<man x>s, <*x* <*owns* <*a donkey*>*nps*>*vp*>*s*), <x <sleeps>$_{VP}$>$_S$]

Finally (R4) can apply, with U instantiated as *owns* and W as nil, to the italicized subformula — remember that x is of category NP! — to yield ($5_{PFO'''}$) which is equivalent to (5_{PL}).

($5_{PFO'''}$) [(<man x>$_S$, (<donkey y>$_S$, <x <owns y>$_{VP}$>$_S$)), <x <sleeps>$_{VP}$>$_S$]
(5_{PL}) $\forall x \exists y((man(x) \wedge donkey(y) \wedge owns(x,y)) \rightarrow sleeps(x))$

However, (5) has another interpretation, which would correspond to the PL sentence obtained by exchanging the places of quantifiers. The corresponding PFO formalization can be obtained by starting with (R3) with U = *every man who owns* and then (R1) and (R5):

R3: (<donkey x>$_S$, <<every man who owns x >$_{npq}$ <sleeps>$_{VP}$>$_S$)
R1: (<donkey x>$_S$, [<<man who owns x >$_{CN}$ y>$_S$, <y <sleeps>$_{VP}$>$_S$])
R5: (<donkey x>$_S$, [(<man y>$_S$, <y owns x >$_S$) , <y <sleeps>$_{VP}$>$_S$])

Similarly, by applying (R1) + (R4) + (R10) ((R10) is one of the two rules for negation) in the six possible orders, six different formalizations of e.g. *Every man does not love a woman* will be derived, corresponding to the six different possible orderings of the quantifiers $\forall$, $\exists$ and the negation operator $\neg$ in PL.

(R10) <NP <Aux not W>$_{VP}$>$_S$ $\Rightarrow$ [<NP <Aux W>vp>s, $\perp$]

The list of rules for the fragment includes rules for *definite noun phrases* and *disjunctions*, which are somewhat more complex, but introduce nothing essentially new. The rules for translating pronouns to variables, e.g. the rule needed to transform ($2_{PFO''}$) to ($2_{PFO'''}$) above, however, differ more. Their task is to produce a formalization which corresponds to an *anaphoric interpretation* (a variable which is co-indexed with the variable of a possible antecedent) whenever it is syntactically and semantically possible and/or a *deictic interpretation* (a new variable). (6) and (7) demonstrate cases when syntactic respectively semantic constraints do not permit an anaphoric interpretation of the final pronoun.

(6) If Pedro owns a donkey, he beats her
(7) If Pedro owns every donkey, he beats it

The first constraint is enforced through the GEN feature, mentioned in section 3.1. The subject pronoun rule is (R17).

(R17) <<Pron:*GEN*>$_{nps}$> VP>$_{s}$ ⇒ <*x* VP>$_{s}$
IFF <<N:*GEN*>$_{cn}$ *x*>$_{s}$
<*y* VP>$_{s}$

The "IFF <structure>" statement serves as a constraint on whether the variable *x* can be used: it is possible only if the specified structure exists as a subformula somewhere in the current PFO formula (i.e. the one that the structural description is a part of as well). In the case of (R17) this means that the current PFO formula should have a noun with the same value for the GEN feature and the same PFO-variable *x*. This condition will not be fulfilled for (6) so the only part of the rule applicable will be the part that introduces *y*, a new variable.

The semantic constraint necessary is somewhat more complex. The anaphoric PFO formalization of (7) is (7_{PFO}) which will indeed be produced by the translation rules.

(7_{PFO}) *[[donkey x, p owns y], p beats x]

This formalization can be disallowed through a constraint such as the one discussed by Pagin & Westerståhl stating in effect that a variable that is quantified within [X,Y] is not to be used outside [X,Y]. This, however, has been more difficult to express procedurally than one can imagine. It is not as simple as to say that PFO-variables introduced by universal quantification rules such as (R1), (R2) and (R10) are not "reusable". Example (8) has an anaphoric interpretation, ($8_{PFO'}$), despite of that.

(8) Every man loves a woman that pleases him
($8_{PFO'}$) ([man x, ((woman y, y pleases x), x loves y]

But neither can a pronoun always co-refer with a noun that is within the same sentence; a different order of applying the translation rules (e.g. (R4) + (R5) + (R1)) will produce ($8_{PFO''}$) in which *him* cannot be anaphoric.

($8_{PFO''}$) ((woman y, y pleases him), [man x, x loves y])

What seems to be necessary is a mechanism that "remembers" when an universal quantification has been introduced in a PFO-formula and allows co-referece with the universally quantified variable *only among subformulas* of that formula.

4. Computer implementation

As mentioned in the introduction, the purpose of the computer implementation in PROLOG has been mainly one of a debugging device, and therefore the implementation is quite crude. Here we will only present the basics of the notation and "trace" the derivation of the classical donkey sentence *If a man owns a donkey then he beats it.*

The syntactic analysis is performed by a standard *Definite Clause Grammar* (DCG) (cf. Pereira and Warren 1980) which straightforwardly uses the rules in *Appendix A* and PROLOG's built in top-down interpreter with unification to produce a syntactic tree (with nouns and pronouns marked for their GEN feature), in the form of a PROLOG *list.*

```
[s,if,[s,[nps,[[dets,a],[cn,[n,farmer,masc]]]],
     [vp,[vtr,owns],[nps,[[dets,a],[cn,[n,donkey,neutr]]]]]],
   then,
   [s,[nps,[pron,he,masc]],[vp,[vtr,beats],[nps,[pron,it,neutr]]]]]
```

The implementation of the translation rules to apply on this structure also consists of an *input list* and *output list,* which specify the structural description and PFO formula respectively. The following is e.g. the implementation of (R1).

```
r1(SD,PFO_formula) :-
   SD = [s,[npq,[[detq, every],CN]], VP],
   PFO_formula = [all, [s,CN,[np,X]], [s,[np,X],VP]].
```

The only difference from (R1) is that non-terminal symbols are specified in the first position of their respective (sub)list and that PROLOG's square brackets which specify the boundaries of a list are used both to mark phrase structure and, together with the "modifiers" `all` and `exist`, PFO operators. When a translation rule such as as (R3) and (R4) has "anonymous phrase structure variables" this is dealt with in the following way. A four-place predicate, `mem`, looks recursively for a certain constituent within a tree, then, having found it, substitutes it with a formalization and returns the new tree:

```
mem(<Constituent>, <Tree>, <Formalization>, <New_tree>)
```

With its help the following is a faithful implementation of (R4).

```
r4(In,Out) :-
   In = [s,NP1, [vp|Rest]],
   mem([nps,[[dets,a],CN]],Rest,[np, X],NewRest),
   Out = [exist, [s,CN,[np,X]], [s,NP1,[vp|NewRest]]].
```

Let us now trace the gradual transformation of the analysed sentence into a PFO formalization. First the *If-(then)* rule (R16) applies to produce:

```
[all,
       [s,[nps,[[dets,a],[cn,[n,farmer,masc]]]],
            [vp,[vtr,owns],[nps,[[dets,a],n,[n,donkey,neutr]]]]]],
       [s,[nps,[pron,he,masc]],[vp,[vtr,beats],
                 [nps,[pron,it,neutr]]]]]
```

Then the existence quantification rule for subject-NP's (R3) applies to the first subformula of the above:

```
[all,
 [exist,[s,[cn,[n,farmer,masc]],[np,x1]],
       [s,[np,x1],[vp,[vtr,owns],[nps,[[dets,a],
                                       [cn,[n,donkey,neutr]]]]]]],
 [s,[nps,[pron,he,masc]],[vp,[vtr,beats],[nps,[pron,it,neutr]]]]]
```

Then the existence quantification rule for object-NP's (R4) applies to the second subformula of the first subformula of the above to produce:

```
[all,
 [exist,[s,[cn,[n,farmer,masc]],[np,x1]],
            [exist,[s,[cn,[n,donkey,neutr]],[np,x3]],
                      [s,[np,x1],[vp,[vtr,owns],[np,x3]]]]],
 [s,[nps,[pron,he,masc]],[vp,[vtr,beats],[nps,[pron,it,neutr]]]]]
```

The subject pronoun interpretation rule (R17) applies to the italicized subformula, finds a possible anaphor, `[cn,[n,farmer,masc]]`, with the right GEN feature and substitutes *`[nps,[pron,he,masc]]`* with the corresponding variable.

```
[all,
 [exist,[s,[cn,[n,farmer,masc]],[np,x1]],
            [exist,[s,[cn,[n,donkey,neutr]],[np,x3]],
                 [s,[np,x1],[vp,[vtr,owns],[np,x3]]]]],
 [s,[np,x1],[vp,[vtr,beats],[nps,[pron,it,neutr]]]]]
```

Finally we come to the last pronoun, which according to an object pronoun rule (R18) can be substituted with an "old" variable, `[np,x3]`, to yield an anaphoric interpretation, or with a "new" variable, `[np,x5]`, to yield a deictic interpretation.

Apart from some cosmetic details added here for perspicuity, this derivation illustrates the performance of the parser (which also yields a large number of equivalent formalizations).

5. PFO and natural language processing

The project of formalizing and implementing the translation procedure English-to-PFO has lended some support to the claim that PFO is well-suited for formalizing natural language semantics. The rules required for carrying out the formalization procedure are quite simple, yet efficient. The "toy implementation" showed that the translation rules do not involve unpredictable interactions. There is no need for any restrictions on the

order of application independent of the structural description, unlike in "classical" transformation grammar. So from the perspective of (applied) natural language processing PFO may prove to be an attractive formalism because (a) due to its compositional nature it *minimizes ambiguity*, e.g. there is no need for different treatments of *a car* in (1) and (2) and (b) does this without extensively extending first-order predicate logic, i.e. in a relatively *constrained formalism.*

However, the particular kind of compositionality that characterizes PFO, compositionality *on the sentence level*, also showed a few drawbacks. The necessity of having "subject"-"object" pairs of rules was cumbersome in itself, but the possible positions of a noun phrase in a sentence is far greater than that. The formalism must therefore be extended to below-sentence compositionality before it can be truly useful for linguistic description. On the other side, the compositional treatment of "donkey anaphora" in a formalism with "a single, uniform notion of semantic content" (P & W, p. 120) seemed to make it harder to specify the semantic constraint on binding. P & W *do* make a clear specification, but they do it *declaratively*, while the lack of any intermediate structures such as the DRS's of DRT make it necessary for the formalization procedure itself to embody this constraint. As pointed out at the end of 3.2. what seems to be called for is a "short term memory" that keeps track of which rule has applied where in the PFO-formula. But this seems to go against the "single, uniform notion of semantic content".

Finally, it should be reminded once again that "semantic compositionality" is not an unproblematic notion. In one sense—that simple expressions combine to produce complex expressions—it seems to be all-encompassing and thus vacuous. In the other, formal, sense defined in section 2 as a relation between a natural and a formal language it may be too strong a constraint. Modification (e.g. *fake gun*), polysemy, intensional contexts and many other natural language phenomena seem not to be easily coerced into it. If PFO can be extended to deal with some of these other phenomena this would present an even greater challenge.

References

Chomsky, N. 1975. *Reflections on Language*. Maurice Temple Smith Ltd.

Geach, P. Th. 1962. *Reference and Generality*. Ithaca, N.Y.

Kamp, H. 1981. *A theory of truth and semantic representation*. In Groenendijk, Janssen and Stokhof (Eds.), *Formal Models in the Study of Language*. Amsterdam.

Montague, R. .1974. *The Proper Treatment of Quantification in Ordinary English*. In *Formal Philosophy: Selected Papers of Richard Montague*, Thomason, R. (Ed.) New Haven, Conn.

Pagin, P. and D. Westerståhl. 1993. *Predicate Logic with Flexibly Binding Operators and Natural Language Semantics*, in JOURNAL OF LANGUAGE, LOGIC AND INFORMATION Vol 2: pp. 89–129, Kluwer.

Pereira, F. and D. Warren. 1980. *Definite Clause Grammars for Language Analysis*, ARTIFICIAL INTELLIGENCE, 13.

Shieber, S. .1986. *An Introduction to Unification-Based Approaches to Grammar*. CSLI Lecture Notes No.4, Stanford, CA.

Appendix A

The context-free grammar with morphosyntactic features for syntactically parsing the fragment of English sentences. Features are marked within square brackets, with **&** signifying conjunction, **=/** "is different from" and **|** disjunction.

```
S -> S or S
S -> if S then S
S -> NP[HEAD = pron & CASE = nom] VP
S -> NP[HEAD =/ pron] VP

NP -> NPs | NPq

NPs -> Dets, CN | PN | Pron
NPq -> Detq CN
NPs -> NPs or NPs
NPq -> NPs or NPq | NPq or NPs | NPq or NPq

CN -> N | N Comp VP

VP -> Vitr
VP -> Aux Neg Vitr[FIN = inf]
VP -> Vtr NP[HEAD = pron & CASE = acc]
VP -> Vtr NP[HEAD =/= pron]
VP -> Aux Neg Vtr[FIN = inf] NP[HEAD = pron & CASE = acc]
VP -> Aux Neg Vtr[FIN = inf] NP[HEAD =/= pron]
VP -> Cop Adj
VP -> Cop Neg Adj

PN -> bill | pedro
N -> farmer[GEN = masc] | donkey[GEN = neutr] | woman[GEN = fem]

Vitr -> sleeps[FIN = fin] | sleep[FIN = inf]
Vtr -> owns[FIN = fin] | loves[FIN = fin] | beats[FIN = fin]
Vtr -> own[FIN = inf] | love[FIN = inf] | beat[FIN = inf]

Dets -> a | the
Detq -> every | no

Pron -> it[GEN = neutr]
Pron -> he[GEN = masc & CASE = nom] | she[GEN = fem & CASE = nom]
Pron -> her[GEN = fem & CASE = acc] | him[GEN = masc & CASE = acc]

Aux -> does
Neg -> not
Comp -> who | that
Cop -> is
Adj -> tired | rich
```