Using conjunctive normal form for natural language semantics

Steven Abney and Ezra Keshet

January 2, 2013

Contents

1 Introduction

In this paper, we consider the consequences of using *conjunctive normal form* (CNF)—also known as *clausal normal form*—as the semantic metalanguage for the interpretation of natural language discourses. CNF is a quantifier-free subset of first-order predicate calculus (FOPC) that constitutes a normal form in the sense of being inferentially equivalent to general FOPC. It is the standard representation in automated reasoning systems. We will show that the use of CNF predicts the existence of certain classes of eccentric anaphora such as donkey anaphora and telescoping.

Theories of discourse semantics that accommodate donkey anaphora include Discourse Representation Theory (Kamp 1981, Kamp and Reyle 1993) and Dynamic Predicate Logic (Groenendijk and Stokhof 1991). We will compare the CNF approach to these alternatives with respect to their empirical predictions. But of equal interest is the manner in which the predictions arise. CNF is designed for utility in reasoning, and the CNF-based system of interpretation that we describe is designed to enable a two-way connection between language and reasoning. Eccentric anaphora emerges as an accidental consequence of the design. In that sense, the CNF approach goes beyond merely accommodating eccentric anaphora to giving a possible explanation for why it exists in the first place.

1.1 Language as Interface to Reasoning

Natural language users are able to both produce and comprehend language. Information acquired through language comprehension feeds reasoning, as well as decision-making, emotions, and other basic cognitive functions; and these functions in turn generate information which may be conveyed to others via language production. Viewed from a computational perspective, language thus constitutes an input-output system for an intelligent agent. In the input direction, the system converts a sentence into an internal representation that is conducive to reasoning or other uses, and in the output direction, this conversion is reversed.

Conventional linguistic accounts of semantics (Montague 1970a,b) define interpretation functions, often in FOPC, but do not give an account of the connection between those functions and reasoning. An interpretation function typically assigns a set of possible worlds to a sentence: in particular, those possible worlds where the sentence is true. Most work in semantics does not consider how a set of possible worlds can be finitely represented, algorithmically computed, or used in reasoning algorithms.

In the field of artificial intelligence (AI), there is an enormous literature on reasoning algorithms. By far the commonest algorithm is resolution.[1] In

[1]We note that reasoning algorithms other than resolution do exist. We particularly mention model-building algorithms that, at first blush, appear more compatible with model-based interpretation functions. Model building algorithms attempt to satisfy a formula by explicitly constructing a finite model for it. If a formula is satisfiable, its negation is not valid. As such,

the context of AI, the textbook approach to language comprehension involves a pipeline consisting of a parser that converts a sentence to an expression in first-order predicate calculus, followed by conversion to CNF, for processing by an automated reasoner.[2] CNF emerged from early work on logical inference by Whitehead and Russell (1910), Herbrand (1930), Skolem (1934), and Gödel (1939), and its special significance for automated reasoning was established with the introduction of resolution-based theorem provers (Robinson 1965). Resolution is a rule of inference that applies to CNF, not to general FOPC expressions. It is inferentially complete as a sole rule of inference and is therefore the basis for most practical reasoning systems.

To do language production, one must reverse the pipeline just sketched. However, most computational systems for language production do not actually reverse the comprehension pipeline, but rely instead on a production system that is entirely independent of the comprehension system. One difficulty is the conversion from FOPC to CNF, which is not readily inverted. The system of semantic interpretation that we present below is based on the idea of avoiding the FOPC-to-CNF conversion by doing semantic interpretation directly to CNF. We effectively use the same tree structure for both the English sentence and the CNF expression—this is made possible by the homorphism between English logical forms (LFs) and CNF discussed in section 2.1—and the common structure makes it possible to do language production by parsing in reverse. One parses the CNF to construct an LF tree, and one then reads the English sentence off the LF tree (Abney 2012).

1.2 Introducing CNF

An expression in CNF is a conjunction of clauses, where a *clause* is a disjunction of literals. A *literal,* in turn, is an atomic formula or a negated atomic formula, an *atomic formula* being the application of a predicate to zero or more terms. A *term* is a variable, an individual constant, or a function applied to a list of terms. For instance, the following diagram labels the parts of the CNF expression $[A(c) \vee \neg B(f(x))] \wedge C(x)$ with predicates A, B, C, variable x, constant c, and

model building is a useful complement to deduction: deduction searches for a proof that a given formula is valid, and building a model for the negation can provide a counterexample, demonstrating that no proof will be found.

But a model-building approach is not necessarily any more compatible with the standard interpretation functions. Generally, model building algorithms also use CNF internally, or convert expressions to CNF as a step in reducing the first-order model-building problem to a propositional satisfiability problem (McCune 2003).

[2]For example, this is the approach spelled out in Chapter 9 of Russell and Norvig (1995), which is the standard introduction to artificial intelligence.

function f:

$$\overbrace{[\,\overbrace{\underbrace{\underbrace{A}_{\text{predicate}}(\underbrace{c}_{\text{term}})}_{\text{atomic formula}}}^{\text{literal}} \vee \overbrace{\neg\, \underbrace{\underbrace{B}_{\text{predicate}}(\underbrace{f(x)}_{\text{term}})}_{\text{atomic formula}}}^{\text{literal}}\,]}^{\text{clause}} \wedge \overbrace{\overbrace{\underbrace{\underbrace{C}_{\text{predicate}}(\underbrace{x}_{\text{term}})}_{\text{atomic formula}}}^{\text{literal}}}^{\text{clause}}$$

Notice that there are no quantifiers in a CNF expression: all variables that occur are free and are implicitly universally bound. In addition, certain of the function symbols are distinguished as *Skolem functions*; these function symbols are implicitly existentially bound. We will extend the term *variable* to include both the standard variables over individuals and Skolem function symbols, which we think of as variables ranging over functions. The former we call *individual variables* or *universal variables* (because of the implicit universal binding), and the latter we call *Skolem variables.* To distinguish them typographically, we place a dot over Skolem variables.

The interpretation of a CNF expression is the same as for any predicate calculus expression, except for the implicit binding of variables. More precisely, for any CNF expression ϕ, let $x_1, \ldots, x_n$ be the universal variables that occur in ϕ, and let $\dot{y}_1, \ldots, \dot{y}_m$ be the Skolem variables that occur. Then ϕ is interpreted identically to the (second-order) predicate calculus formula

$$\exists \dot{y}_1 \ldots \exists \dot{y}_m \, \forall x_1 \ldots \forall x_n \, \phi.$$

For example, consider the following sentence:

(1) Every farmer owns a donkey.

Its CNF translation is the following:[3]

(2) $\neg \text{farmer}'(x) \vee [\text{donkey}'(\dot{y}(x)) \wedge \text{owns}'(x, \dot{y}(x))]$.

As we will see, variables may be shared across multiple sentences. Hence our comments above regarding implicit quantification provide a definite interpretation only for a complete discourse or dialogue—that is, a complete *database* of CNF clauses. We therefore must address the interpretation of a CNF expression such as (2) that constitutes only a fragment of a database.

In formal definitions of the predicate calculus, an open formula is typically treated as representing (at least indirectly) a function from assignments to truth values. A very similar interpretation applies here: an open formula such as (2) represents a function from assignments to truth values; although in this case the domain of the assignment includes not only individual variables but also Skolem variables.

CNF differs from the standard predicate calculus in that CNF additionally permits an open formula to be interpreted as if it were a sentence (i.e., a formula

[3]We adopt the common convention of using ω' to represent the logical predicate that translates the object-language word ω.

with no free variables). With a slight abuse of terminology, we can think of these two interpretations as the *intension* and *extension* of the formula. The intension is the function that maps assignments to truth values. The extension is a truth value, defined as follows. A CNF expression whose intension is f has the extension TRUE just in case there exists a partial assignment g_0, assigning values just to the Skolem variables such that every extension of g_0 to a total assignment g yields $f(g) = \text{TRUE}$. That is, an open formula is true *qua* sentence just in case there exists some way of fixing the values of Skolem variables, such that every way of assigning values to universal variables makes the formula, *qua* formula, true.

Note incidentally that intensions are easily conjoined: if we have a CNF formula whose intension is f_1, and we add a new clause with intension f_2, the conjoined intension is

$$\lambda g\,[f_1(g) \wedge f_2(g)].$$

Hence sentence meanings are readily conjoined to create discourse meanings.

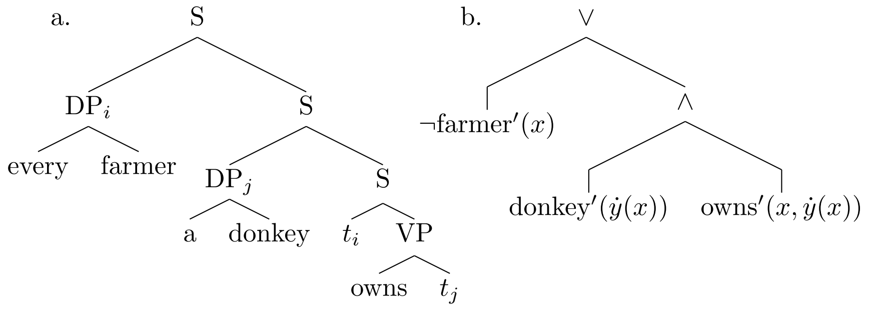
2 CNF as Semantic Metalanguage

In this section, we define a semantic system using CNF as the metalanguage. Instead of a complex pipeline from natural language to CNF, our system is a homomorphism from an LF tree to its corresponding CNF expression. After defining the system, we step through several illustrative examples, and, finally, we present a shorthand that will prove useful in later sections.

2.1 Interpretation function as homomorphism

The domain of our interpretation function consists of standard LF trees, in which quantifiers and negation have been raised. (That is, we treat both quantification and negation as scope-taking operators.) For example, we take the LF of sentence (1) to be something like (3a). Our system assigns it the interpretation (2), whose syntactic structure is shown in (3b):

(3)



A key intuition is that our semantic interpretation function defines a homomorphism from LF trees to CNF expressions. That is, the structure of the CNF expression is a simplification of the LF tree, but it does not involve any restructuring. The nodes of the CNF tree (3b) correspond in an obvious way to nodes in the LF tree (3a): the leaf nodes correspond to the leaves *farmer, donkey,* and *owns* in the LF tree; the root node corresponds to the root node of the LF tree; and the node labeled "$\wedge$" corresponds to the middle S node in the LF tree. The structure of the LF tree is preserved, albeit in simplified form, in the CNF expression.

The next few sections define the semantic interpretation system necessary to achieve this homomorphism. Comprehensive examples of the system are postponed until section 2.7. On a first reading one may wish to skim the examples before launching into the definitions of sections 2.2–2.5.

2.2 Semantic types

Before defining the interpretation function, it will be convenient to introduce a system of semantic types. A type has three parts: a *basic type* $\rho \in \{L, 0, C, \forall, \exists, \wedge, \vee\}$, governing how nodes combine; an *index* i, governing predicate-argument relationships and quantifier-trace relationships; and a *polarity* $s \in \{+1, -1\}$. The combination ρ^s of basic type and polarity is a *signed type.* We usually adopt a slightly modified notation and write a signed type simply as ρ if $s = +1$, and $\overline{\rho}$ if $s = -1$. The combination ρ^s_i of signed type and index is a *full type.*

Basic types. The basic types are summarized in the following table. Unlike the types e, t, $\langle e, t\rangle$, etc., familiar from the typed predicate calculus, our types do not represent the mathematical class of the denotatum of a node, but rather the mode of combination used when interpreting the node.

Node	**Basic Type**
Literals: nouns, verbs, adjectives, prepositions	L
Uninterpreted nodes: all other terminal nodes	0
Copy nodes (share a denotation with one child)	C
Universally quantified DPs	$\forall$
Existentially quantified DPs	$\exists$
Disjunction	$\vee$
Conjunction	$\wedge$

The first two types are used for terminal nodes, and the remaining types are used for nonterminal nodes. The assignment of types to nodes is determined by the rules of section 2.4. Namely, a configuration is interpretable just in case there is an interpretation rule that licenses it. (By *configuration* we mean a local subtree consisting of a node and its children.) The reader may find it useful to think of the interpretation rules as rewrite rules in which semantic types represent the categories.

Polarity. The polarity of a node is also determined by the rules of interpretation given below. The general principle is quite simple, however, and we state it here. Root nodes have positive polarity, and polarity is inherited downward (i.e., the polarity of a node equals the polarity of its parent) with two exceptions:

- In a determiner phrase $[_{\text{DP}}$ D NP$]$ in which the determiner is a universal (*every, all, each,* …), the polarity of the NP is opposite to that of the DP.
- In a negation $[_{\text{S}}$ *not* S$]$, the two S nodes have opposite polarity.

Indices. An *index term* may be a universal variable, a Skolem term consisting of a Skolem function and its arguments, or a name (individual constant). An *index* is a tuple of index terms. For example:

- $\langle\,\rangle$ – an empty index. Used for uninterpreted nodes and saturated nodes, such as complete clauses.
- $\langle u\rangle$ – a single index term. Used for one-place predicates, such as simple nouns and intransitive verbs. Also used for quantified DPs and their traces.
- $\langle u, v\rangle$ – used for two-place predicates, such as transitive verbs.
- $\langle u, v, w\rangle$ – used for three-place predicates, such as ditransitive verbs.

Tuples of any length are allowed.

In most cases, we omit the angle brackets when writing indices. For example, the full type $C_{x,y}$ has index $\langle x, y\rangle$; the full type $\overline{\exists}_x$ has index $\langle x\rangle$; and the full type $\wedge$ has index $\langle\,\rangle$.

2.3 Semantic lexicon and root constraints

The semantic types and interpretations of nodes are determined by four sets of rules: a *semantic lexicon,* a set of *root constraints,* a set of *interpretation rules,* and a set of *index-assignment rules.* The interpretation rules are given in section 2.4; the index-assignment rules are stated in section 2.5; and the lexicon and root constraints are dispensed with here.

Semantic lexicon The semantic lexicon determines the basic type of each word. We do not give an explicit listing, but the general principles are very simple. Content words (nouns, non-copular verbs, adjectives, and prepositions) have type L, and all other words have type 0. The lexicon also determines a CNF predicate ω' corresponding to each content word ω.

Root constraints The root constraints are also simply stated: the root node must have positive polarity and its index must be $\langle\,\rangle$.

2.4 Rules of interpretation

For each interpreted node in an LF tree, the interpretation function $[\![\ ]\!]$ determines a CNF *intension*—that is, a function from assignments to truth values. This intension is coordinated with the intensions of the previous sentences in the current discourse, and the resulting CNF expression (still an intension) is assigned a TRUE/FALSE *extension* as described in Section 1.2.

The interpretation function is defined by the rules given below. In these rules we use "colon" notation to indicate the semantic type of a node. For example, "$\alpha : \overline{\exists}_i$" is a pattern that matches any node α whose semantic type is $\overline{\exists}_i$. We use the variable ω for terminal nodes only; we use α, β, and γ for nodes that may be either terminal or nonterminal. We use the variables σ and τ for basic types *other than* 0. We use variables u and v for individual index terms, and we use i and j for indices. (Recall that indices are tuples of index terms.) The case $i = \langle\,\rangle$ is expressly permitted. Conversely, the omission of an index is significant: for example, a semantic type indicated simply as τ is incompatible with any index except $\langle\,\rangle$. The linear order of a node's children is not important and may be reversed in any of the rules.

As mentioned earlier, for any given node in an LF tree, there is only rule that might possibly apply, the relevant rule being readily determined by the number of children (if any) and their semantic or syntactic types. If the semantic types in the relevant rule do not match the semantic types in the local configuration in the LF tree, the tree is semantically ill-formed. The determination of which rule is the "relevant rule" is made as follows.

- For a terminal node, the only applicable rules are Literals (4) and Zeros (5). The choice between them is determined by the semantic type of the word (L or 0), as assigned by the semantic lexicon.
- For a nonterminal node with a single child, the only applicable rule is Copy (6).
- For branching nodes, rules (7)–(12) apply if there is exactly one child of type 0. The syntactic type of that child determines which rule applies. The relevant syntactic types, along with the notation we use to indicate them, are as follows: identity elements (ω_I) such as copular verbs, negative elements ($\omega_\neg$) such as *not,* conjunction ($\omega_\wedge$), disjunction ($\omega_\vee$), referential elements (ω_{DP}) such as personal pronouns and traces, relative pronouns (ω_{RP}), universal quantificational determiners ($\omega_\forall$) such as *every* and *each,* and existential quantificational determiners ($\omega_\exists$) such as *some* and *a(n).*
- The two remaining rules are Modification (14) and Q-Scope (13), for which *neither* child may be of type 0. Q-Scope requires one child to be quantificational (basic type $\forall$ or $\exists$), and Modification prohibits either child from being quantificational, so again the choice of rule is determined by the types of the children.

2.4.1 Literals

Recall that ω must be a terminal node.

(4) $[\![\omega : L_{u_1,\ldots,u_n}]\!] = \omega'(u_1, \ldots, u_n) \qquad [\![\omega : \overline{L}_{u_1,\ldots,u_n}]\!] = \neg\omega'(u_1, \ldots, u_n)$

Thus, a content word ω with an index i comprising n index terms denotes a CNF literal whose predicate is the semantic constant ω' listed for ω in the lexicon and whose argument list is identical to i. The cases $n = 1$ and $n = 0$ are permitted. Examples are the terminal nodes labeled "L" in any of the trees of section 2.7.

2.4.2 Zeros

A terminal node ω whose type is not L must have type 0. The interpretation of such nodes is undefined.

(5) $[\![\omega : 0_i]\!] = \bot \qquad [\![\omega : \overline{0}_i]\!] = \bot$

Such nodes may, however, contribute to the interpretation less directly. As already mentioned, words of type 0 play a syncategorematic role in many of the following rules. Also, some of the index-assignment rules of section 2.5 apply to DP nodes of type 0. Note that we treat pronouns, proper nouns, and traces as terminal nodes, devoid of internal structure, despite having category DP. Examples are the terminal nodes labeled "0" in any of the trees of section 2.7.

2.4.3 Copy

The following rule applies to any unary-branching node γ whose only child is α.

(6) $\left[\!\!\left[\begin{array}{c}\gamma : C_i \\ | \\ \alpha : \tau_i\end{array}\right]\!\!\right] = [\![\alpha]\!] \qquad \left[\!\!\left[\begin{array}{c}\gamma : \overline{C}_i \\ | \\ \alpha : \overline{\tau}_i\end{array}\right]\!\!\right] = [\![\alpha]\!]$

The index of the child is copied to the parent. Child and parent also share polarity and denotation. An example can be found in (21).

On a first exposure, the reader may find it confusing that the interpretation appears to be the same, regardless of the polarity of the parent node. But remember that if the child α is a terminal node with a negative type $\overline{\tau}_i$, its interpretation $[\![\alpha]\!]$ will be a negated CNF literal. Negations in CNF appear *only* on literals; negative polarity on a nonterminal node indicates that all the literals it contains are to be negated. Similar comments apply to all the following rules.

2.4.4 Identity

In the following rule, ω_I is an identity element, such as a copula.[4]

[4] For the moment we set aside any tense interpretion of the copula. Tense is revisited in section 4.1.

(7) $$\left[\!\!\left[\begin{array}{c}\gamma : C_i \\ \diagup\quad\diagdown \\ \omega_I : 0 \qquad \beta : \tau_i\end{array}\right]\!\!\right] = [\![\beta]\!] \qquad \left[\!\!\left[\begin{array}{c}\gamma : \overline{C}_i \\ \diagup\quad\diagdown \\ \omega_I : \overline{0} \qquad \beta : \overline{\tau}_i\end{array}\right]\!\!\right] = [\![\beta]\!]$$

Again, the parent shares index, polarity, and denotation with one child—the one not of type 0. An example can be found in (24): the word *is* is an identity element.

2.4.5 Negation

In this rule, $\omega_\neg$ represents a negative element. The configuration under consideration is typically the result of operator raising at LF.

(8) $$\left[\!\!\left[\begin{array}{c}\gamma : C_i \\ \diagup\quad\diagdown \\ \omega_\neg : 0 \qquad \beta : \overline{\tau}_i\end{array}\right]\!\!\right] = [\![\beta]\!] \qquad \left[\!\!\left[\begin{array}{c}\gamma : \overline{C}_i \\ \diagup\quad\diagdown \\ \omega_\neg : \overline{0} \qquad \beta : \tau_i\end{array}\right]\!\!\right] = [\![\beta]\!]$$

The only effect of a negative element is to flip the polarity of its complement. An example is found in (25).

2.4.6 Conjunction and disjunction

In these rules, $\omega_\wedge$ represents a conjunction (*and, but*) and $\omega_\vee$ represents a disjunction (*or*).

(9) $$\left[\!\!\left[\begin{array}{c}\gamma : \wedge_i \\ \diagup\quad|\quad\diagdown \\ \alpha : \sigma_i \qquad \omega_\wedge \qquad \beta : \tau_i\end{array}\right]\!\!\right] = [\![\alpha]\!] \wedge [\![\beta]\!] \qquad \left[\!\!\left[\begin{array}{c}\gamma : \overline{\wedge}_i \\ \diagup\quad|\quad\diagdown \\ \alpha : \overline{\sigma}_i \qquad \omega_\wedge \qquad \beta : \overline{\tau}_i\end{array}\right]\!\!\right] = [\![\alpha]\!] \vee [\![\beta]\!]$$

$$\left[\!\!\left[\begin{array}{c}\gamma : \vee_i \\ \diagup\quad|\quad\diagdown \\ \alpha : \sigma_i \qquad \omega_\vee \qquad \beta : \tau_i\end{array}\right]\!\!\right] = [\![\alpha]\!] \vee [\![\beta]\!] \qquad \left[\!\!\left[\begin{array}{c}\gamma : \overline{\vee}_i \\ \diagup\quad|\quad\diagdown \\ \alpha : \overline{\sigma}_i \qquad \omega_\vee \qquad \beta : \overline{\tau}_i\end{array}\right]\!\!\right] = [\![\alpha]\!] \wedge [\![\beta]\!]$$

A negated conjunction ($\overline{\wedge}$) is interpreted as $\vee$, and a negated disjunction ($\overline{\vee}$) is interpreted as $\wedge$.

Some readers may find the ternery branching structure distasteful. We have adopted it for simplicity; defining interpretation rules for a binary-branching structure involves technical issues that would complicate matters to little profit.

2.4.7 Application

This rule corresponds roughly to Functional Application. However, unlike with standard Functional Application, in our system the CNF predicate is not actually applied to its arguments until one interprets the terminal node. Recall that the Literals rule (4) takes its node's index terms to represent the predicate's arguments. Hence the present rule implements function application indirectly, by adding the index term v of the argument to the index i of the parent, yielding the index $\langle i, v\rangle$ for the head.

The variable ω_{DP} represents a referential DP—a proper noun, a pronoun, or a trace. We use $\langle i, v\rangle$ as a shorthand for $\langle u_1, \ldots, u_n, v\rangle$ where $i = \langle u_1, \ldots, u_n\rangle$.

(10)
$$\left[\!\!\left[\begin{array}{c}\gamma : C_i \\ \alpha : \sigma_{i,v} \quad \omega_{\mathrm{DP}} : 0_v\end{array}\right]\!\!\right] = [\![\alpha]\!] \qquad \left[\!\!\left[\begin{array}{c}\gamma : \overline{C}_i \\ \alpha : \overline{\sigma}_{i,v} \quad \omega_{\mathrm{DP}} : \overline{0}_v\end{array}\right]\!\!\right] = [\![\alpha]\!]$$

We can view the rule alternatively from a bottom-up perspective, as stating that a predicate α with a non-empty index $\langle i, v\rangle$ may discharge its last index term v by combining with an uninterpreted node ω_{DP} whose sole index term is v. The resulting parent node has the same interpretation and polarity as the predicate, but its index list i is one term shorter, omitting the discharged term. Examples can be found in any of the trees of section 2.7.

2.4.8 Abstraction

Next, we have a rule akin to Predicate Abstraction. Here, ω_{RP} represents a relative pronoun.

(11)
$$\left[\!\!\left[\begin{array}{c}\gamma : C_u \\ \omega_{\mathrm{RP}} : 0_u \quad \beta : \tau\end{array}\right]\!\!\right] = [\![\beta]\!] \qquad \left[\!\!\left[\begin{array}{c}\gamma : \overline{C}_u \\ \omega_{\mathrm{RP}} : \overline{0}_u \quad \beta : \overline{\tau}\end{array}\right]\!\!\right] = [\![\beta]\!]$$

In words, a relative pronoun (of basic type 0) may combine with a node having an empty index to form a node having a singleton index. Note that the lack of index on τ is significant. The relative pronoun ω_{RP} and its parent share the same index. The denotation of the parent is taken from the other child β. An example can be found in (24).

2.4.9 Q-Restriction

A set of syncategorematic rules are required to capture quantificational DPs. In the following, the parent γ is a quantificational DP; $\omega_\forall$ must be a universal quantificational determiner such as *every, all, each,* etc.; and $\omega_\exists$ must be an existential quantificational determiner such as *a(n), some,* etc.

(12)
$$\left[\!\!\left[\begin{array}{c}\gamma : \exists_u \\ \omega_\exists : 0 \quad \beta : \tau_u\end{array}\right]\!\!\right] = [\![\beta]\!] \qquad \left[\!\!\left[\begin{array}{c}\gamma : \overline{\exists}_u \\ \omega_\exists : \overline{0} \quad \beta : \overline{\tau}_u\end{array}\right]\!\!\right] = [\![\beta]\!]$$
$$\left[\!\!\left[\begin{array}{c}\gamma : \forall_u \\ \omega_\forall : 0 \quad \beta : \overline{\tau}_u\end{array}\right]\!\!\right] = [\![\beta]\!] \qquad \left[\!\!\left[\begin{array}{c}\gamma : \overline{\forall}_u \\ \omega_\forall : \overline{0} \quad \beta : \tau_u\end{array}\right]\!\!\right] = [\![\beta]\!]$$

Existentially quantified DPs carry the basic type $\exists$ while universally quantified DPs carry $\forall$. With existentials, the parent and both children share the same polarity; but universal quantificational determiners flip the polarity of their complements. In all cases, the parent and the restriction (β) share a common index, which must consist of a single index term. The parent always shares a denotation with the restriction. Most of the trees in section 2.7 contain examples.

In rule (12), the passing of the index term is being used in lieu of lambda abstraction. As a general matter, we implement lambda abstraction through the passing of index terms, rather than through the use of a lambda operator in the metalanguage. That may at first seem a mere notational idiosyncracy, but it is actually central to the CNF system. See section 2.6 for discussion.

2.4.10 Q-Scope

Next we need rules to combine quantificational DPs with their nuclear scope.

(13)
$$\left[\!\!\left[\begin{array}{c} \gamma:\wedge \\ \alpha:\exists_i \quad \beta:\tau \end{array}\right]\!\!\right] = [\![\alpha]\!] \wedge [\![\beta]\!] \qquad \left[\!\!\left[\begin{array}{c} \gamma:\overline{\wedge} \\ \alpha:\overline{\exists}_i \quad \beta:\overline{\tau} \end{array}\right]\!\!\right] = [\![\alpha]\!] \vee [\![\beta]\!]$$
$$\left[\!\!\left[\begin{array}{c} \gamma:\vee \\ \alpha:\forall_i \quad \beta:\tau \end{array}\right]\!\!\right] = [\![\alpha]\!] \vee [\![\beta]\!] \qquad \left[\!\!\left[\begin{array}{c} \gamma:\overline{\vee} \\ \alpha:\overline{\forall}_i \quad \beta:\overline{\tau} \end{array}\right]\!\!\right] = [\![\alpha]\!] \wedge [\![\beta]\!]$$

A DP of basic type $\exists$ has a type-$\wedge$ parent (a conjunction) while a DP of basic type $\forall$ has a type-$\vee$ parent (a disjunction). If the polarity is negative, the interpretation of the connective is inverted. That is, "$\overline{\wedge}$" is interpreted as $\vee$ and "$\overline{\vee}$" is interpreted as $\wedge$. Note that the parent and the second child must have an empty index. Most of the trees of section 2.7 contain examples.

2.4.11 Modification

The next rule corresponds to Predicate Modification. In this rule, the children may not be quantified DPs. That is, in (14), σ and τ may be any basic type except 0, $\forall$, or $\exists$.

(14)
$$\left[\!\!\left[\begin{array}{c} \gamma:\wedge_i \\ \alpha:\sigma_i \quad \beta:\tau_i \end{array}\right]\!\!\right] = [\![\alpha]\!] \wedge [\![\beta]\!] \qquad \left[\!\!\left[\begin{array}{c} \gamma:\overline{\wedge}_i \\ \alpha:\overline{\sigma}_i \quad \beta:\overline{\tau}_i \end{array}\right]\!\!\right] = [\![\alpha]\!] \vee [\![\beta]\!]$$

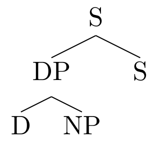
In words, two nodes with matching polarity and indices may combine to form a node of type $\wedge$ with the same polarity and index as the children. The signed semantic type of the parent node determines how the interpretations of its children are combined, with "$\overline{\wedge}$" being interpreted as $\vee$. An example is the combination of *donkey* and the relative clause in (26). Modification also applies when an adjective modifier is combined with a noun, though there are no examples of that in section 2.7.

2.5 Index-assignment rules

The interpretation rules of the previous section determine the basic types and polarities of all nodes. They also propagate indices from one node to another, and that propagation suffices to determine indices for all nodes, *provided* that indices for DPs are initially specified. The rules of this section provide those initial specifications. Before stating the rules, we require a couple of preliminary definitions.

We define the *scope of a quantificational DP* to be the set of nodes dominated by its parent. The rationale is as follows. In a well-formed LF tree, a quantificational DP occurs in the following configuration:

(15)



The NP is the *restriction* of the DP, the lower S is its *nuclear scope,* and everything dominated by the upper S, including everything in both the restriction and nuclear scope, constitutes the *scope* of the DP.

We assume a fixed enumeration of universal variables $x_1, x_2, \ldots$ and a fixed enumeration of Skolem variables $\dot{x}_1, \dot{x}_2, \ldots$.[5] We also enumerate the quantificational DPs in preorder as $\mathrm{DP}_1, \ldots, \mathrm{DP}_n$.[6]

Now we state the index-assignment rules. They determine the indices for DP nodes, and those indices are propagated throughout the tree by the interpretation rules of the previous section.

(16) **Proper Nouns.** The index of a proper noun ω is $\langle \omega' \rangle$. ω' is an individual constant, determined by the semantic lexicon.

(17) **Anaphora.** The index of a trace or pronoun must be the same as the index of its antecedent.

(18) **Universal Variables.** If DP_t (the t-th DP in the preorder enumeration) has signed type $\forall$ or $\overline{\exists}$, then its index is $\langle x_t \rangle$, a singleton index consisting of the t-th universal variable.

(19) **Skolem Variables.** Otherwise the index of DP_t is of form $\langle \dot{x}_t(\ldots) \rangle$, a singleton index consisting of the t-th Skolem function applied to an argument list, and the argument list "$\ldots$" consists of all distinct universal variables that occur in the index terms of outscoping quantificational DPs.[7]

[5]Hence our use, in example trees, of variables that vary unsystematically by letter—x, y, etc.—rather than systematically by subscript—x_1, x_2, x_3, etc.—is not strictly speaking correct; we have opted for readability over rigor.

[6]In preorder, a parent node is ordered before its children, and the k-th child and all its descendents are ordered before the $(k+1)$-st child and any of its descendents.

[7]This definition is slightly nonstandard. In the standard definition, the argument list contains all universal variables that *are* index terms of outscoping quantificational DPs. The relaxation from "are" to "occur in" makes no difference under normal circumstances, but one

For example, in (21) below, the second DP ("a boy") has an index term consisting of Skolem function $\dot{y}$ combined with argument x because it lies in the scope of the first DP ("every girl"), whose index term is x.

In general, in what follows, we write the arguments of Skolem functions as subscripts rather than placing them within parentheses. The motivation is to reduce clutter.

2.6 Lambda abstraction

As a general matter, we have used index-term passing where the standard approach would have lambda abstraction. There is a temptation to view this as a mere notational idiosyncracy and to make the interpretation rules look more familiar by adopting the more common approach of using lambda operators in the metalanguage, to be subsequently eliminated by beta reduction.

There are compelling reasons to resist the temptation. The use of lambda abstraction in any of the rules would have repercussions for the entire system. For example, if we use lambda abstraction in Q-Restriction (12), we would have to replace Q-Scope (13) with standard function application, and replace Modification (14) with set intersection (implemented as conjunction of characteristic functions), and so on.

Introducing lambda abstraction into the semantic translation would destroy the homomorphism mentioned in section 2.1 and discussed in more detail in section 2.7.2 below. The interpretation rules would no longer translate directly to CNF. One would first need to do beta reduction to convert the output of interpretation to CNF, and beta reduction is not readily invertible. We would lose the symmetry between production and comprehension that is fundamental to the reversibility of parsing discussed in section 1.1.

Donkey anaphora would disappear as well. The indices provide a connection between pronouns and quantifiers that is global in scope. By contrast, lambda abstraction combined with beta reduction allows only connections that are compositions of chains of local connections. Eccentric connections by definition involve elements outside of the scope of the highest lambda operator, and would become unavailable if we converted the CNF system to one that uses lambda abstraction.

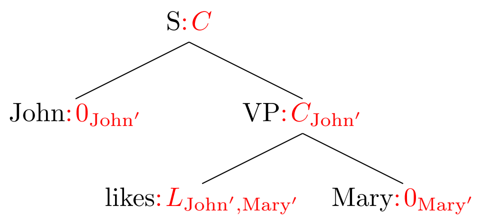
2.7 Examples

2.7.1 Elementary examples

Consider the following tree, with full types written in:

case arises in section 4.3 where this tiny change becomes critical.

(20)



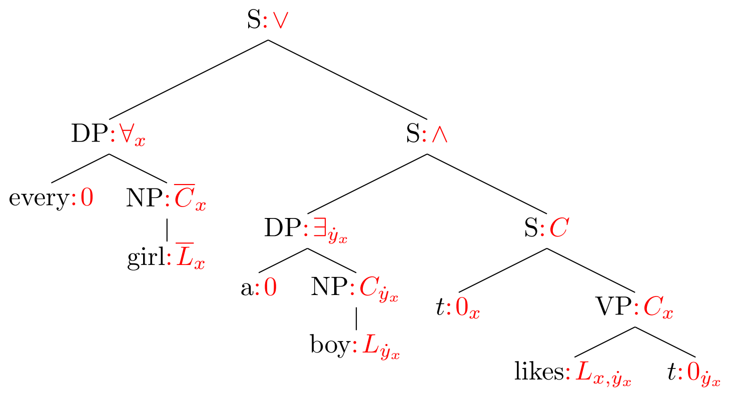
The semantic lexicon determines the semantic types of the terminal nodes: L for the verb, and 0 for the proper nouns. As for the VP and S nodes, each has a sole type-0 child with syntactic category DP, so the only rule that can be used is Application (10). The root constraints require the root node to have positive polarity, and Application guarantees that positive polarity is propagated to the rest of the tree.

Notice that the index $\langle \text{John}', \text{Mary}' \rangle$ on *likes* "becomes" $\langle \text{John}' \rangle$ on VP when the verb combines with the DP node *Mary* having index Mary′. This VP index then "becomes" $\langle\, \rangle$ on the S once the VP combines with the subject DP.

The only nodes in this tree that have interpretations are *likes*, VP, and S, and they all have the same interpretation: $\text{likes}'(\text{John}', \text{Mary}')$. The DP nodes contribute to the interpretation by providing the lexical constants John′ and Mary′, which they bear as index terms, by the Proper Nouns rule (16). The index terms on the verb are constrained by the Application rule to match the index terms of the DP nodes.

Next, consider the following tree, again annotated with full types:

(21)



This tree illustrates the use of several rules. By Q-Scope (13), the top node has a type of $\vee$, meaning that its two daughters are combined via disjunction. Also by Q-Scope, the second highest S has type $\wedge$, meaning that its daughters are combined via conjunction. By Q-Restriction (12), the NP dominating *girl* has negative polarity, because of the universal quantification. The Copy rule (6) propagates that polarity to the N *girl*.

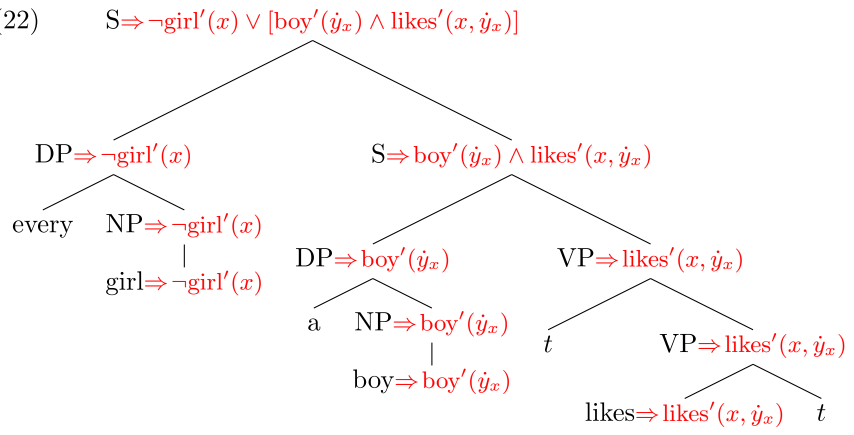
Because the DP dominating *girl* has signed type $\forall$, the Universal Variables rule (18) specifies that its index term is a universal variable, x. The DP dominating *boy* has signed type $\exists$, so the Skolem Variables rule (19) specifies that its index is a Skolem term whose argument list comprises the outscoping universal variables. There is only one outscoping universal variable, x, hence the index term is $\dot{y}(x)$, which we have written as $\dot{y}_x$.

By the Anaphora rule (17), each trace bears the index of its antecedent. Thus, the original sentence for this tree was “every girl likes a boy” and not “a boy likes every girl.”

2.7.2 The essence of the interpretation function

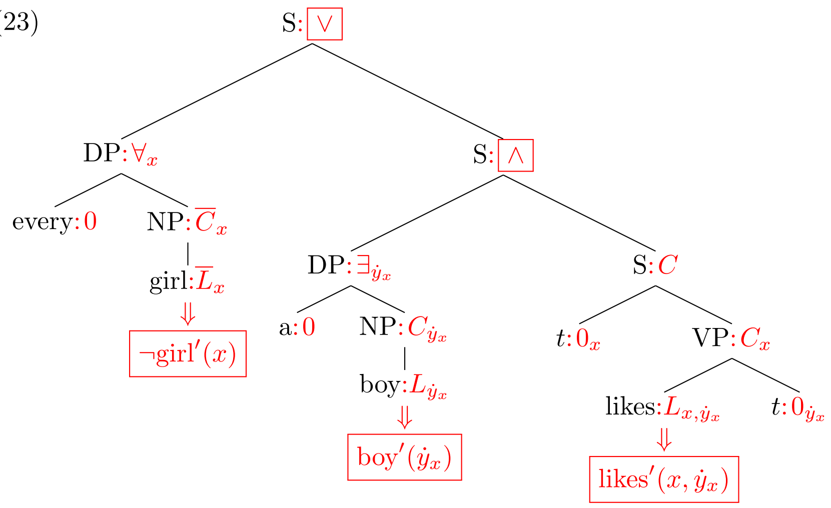
Next, consider the same tree annotated with the node interpretations:

(22)


S⇒ $\neg$girl′$(x) \vee$ [boy′$(\dot{y}_x) \wedge$ likes′$(x, \dot{y}_x)$]
DP⇒ $\neg$girl′(x)
S⇒ boy′$(\dot{y}_x) \wedge$ likes′$(x, \dot{y}_x)$
every
NP⇒ $\neg$girl′(x)
girl⇒ $\neg$girl′(x)
DP⇒ boy′$(\dot{y}_x)$
VP⇒ likes′$(x, \dot{y}_x)$
a
NP⇒ boy′$(\dot{y}_x)$
boy⇒ boy′$(\dot{y}_x)$
t
VP⇒ likes′$(x, \dot{y}_x)$
likes⇒ likes′$(x, \dot{y}_x)$
t

Such an annotated tree is quite repetitious. In fact, almost all of the interpretation rules of section 2.4 are copy rules, in the sense that the denotation of the parent is identical to the denotation of one of the children. The only exceptions are Literals (4), which interprets terminal nodes of type L as literals, and Q-Scope (13) and Modification (14), which interpret nonterminal nodes of type $\wedge$ and $\vee$ as connectives, understanding $\overline{\wedge}$ as $\vee$ and $\overline{\vee}$ as $\wedge$. For the example just given, it suffices to compute the following (where boxes have been placed around the literals’ denotations and the coordinating nodes’ types):

(23)

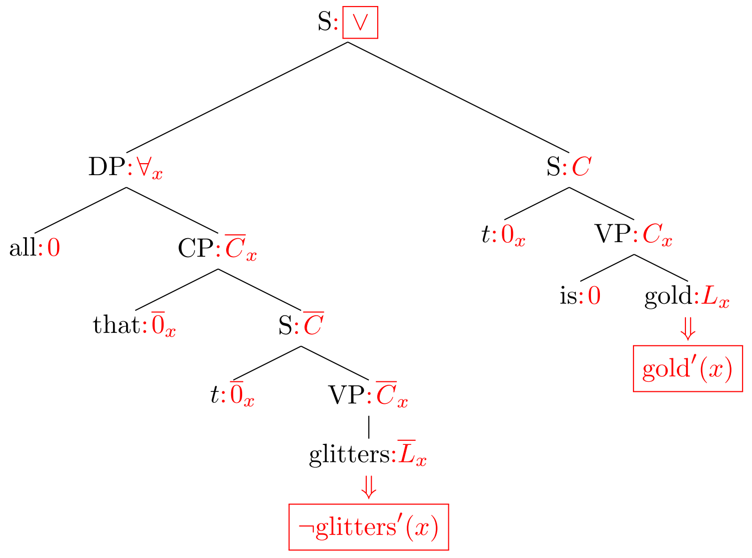


One can then read off the interpretation simply by reading the boxed elements: $\neg\text{girl}'(x) \vee [\text{boy}'(\dot{y}_x) \wedge \text{likes}'(x, \dot{y}_x)]$

2.7.3 More examples

The following provides an example of a relative clause. The tree (24) is the LF for "all that glitters is gold."

(24)

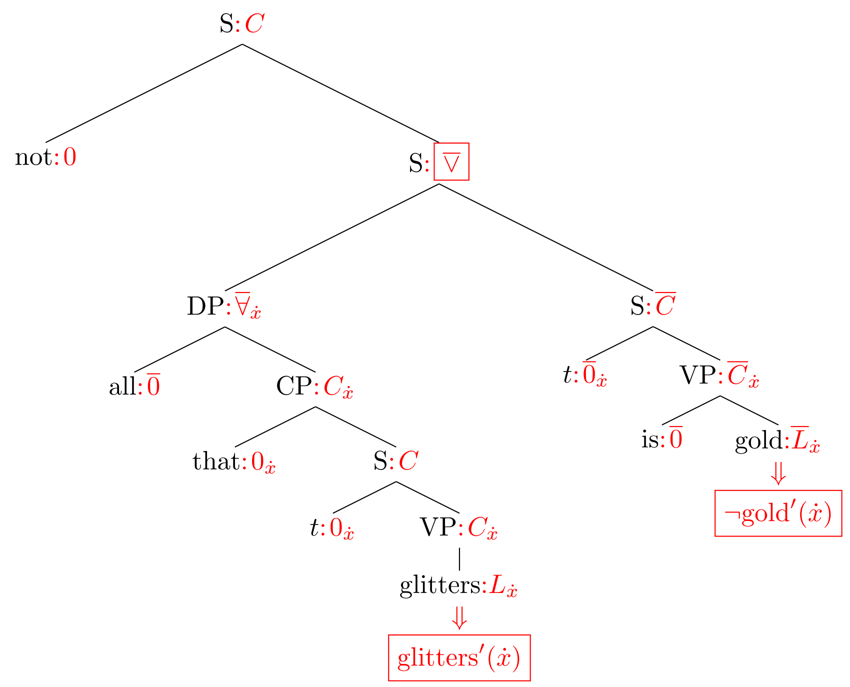


The translation, $\neg\text{glitters}'(x) \vee \text{gold}'(x)$, is logically equivalent to $\text{glitters}'(x) \rightarrow \text{gold}'(x)$, with implicit universal binding of x.

The negation on the CP sibling of *all* is introduced by Q-Restriction (12) and propagates to the entire subtree. The propagation of the index x throughout the tree is also noteworthy. The choice of the variable is determined at the DP by the Universal Variables rule (18). (Properly speaking, the first variable should be x_1, but we have used x instead to reduce subscript clutter.) The variable is propagated to the relative pronoun by Abstraction (11), from the relative pronoun to its trace by Anaphora (17), from the trace to the subordinate-clause VP by Application (10), and from the VP to *glitters* by Copy (6). In the other direction, the variable is propagated from the DP to its trace by Anaphora (17), from the trace to the matrix-clause VP by Application (10), and from the VP to *gold* by Identity (7).

Now let us consider what happens when we negate the sentence: "all that glitters is not gold." This of course has two readings; the following LF assigns wide scope to the negation.

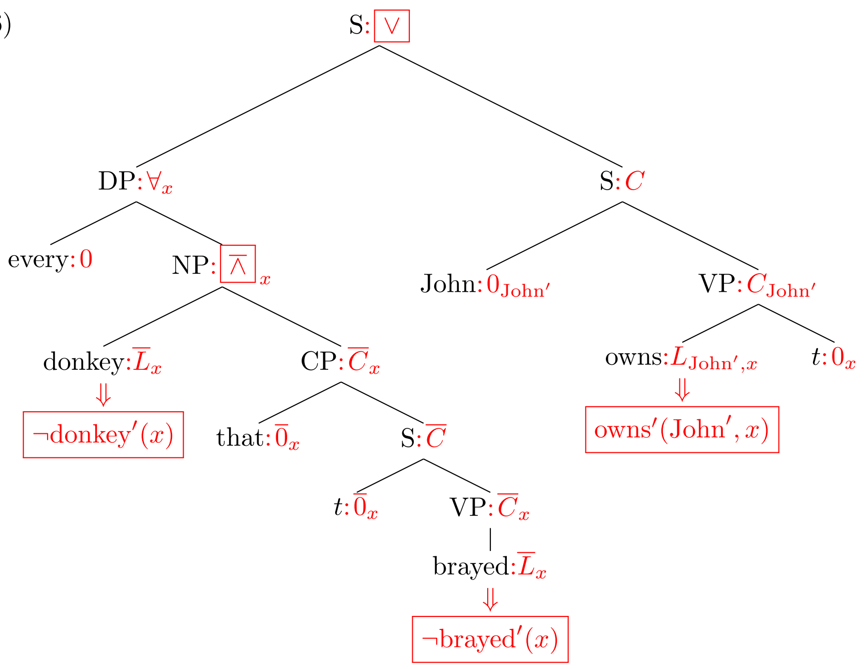
(25)


S:C
not:0
S:$\boxed{\overline{\triangledown}}$
DP:$\overline{\forall}_{\dot{x}}$
S:$\overline{C}$
all:$\overline{0}$
CP:$C_{\dot{x}}$
t:$\overline{0}_{\dot{x}}$
VP:$\overline{C}_{\dot{x}}$
that:$0_{\dot{x}}$
S:C
is:$\overline{0}$
gold:$\overline{L}_{\dot{x}}$
$\Downarrow$
$\boxed{\neg\text{gold}'(\dot{x})}$
t:$0_{\dot{x}}$
VP:$C_{\dot{x}}$
glitters:$L_{\dot{x}}$
$\Downarrow$
$\boxed{\text{glitters}'(\dot{x})}$

This tree illustrates the use of Negation (8); it causes the second-highest S to have negative polarity. The basic type of a node combining a universally quantified DP with its sister is $\vee$, although the negated version $\overline{\triangledown}$, which we have here, is interpreted as $\wedge$. The negation spreads through the whole tree, except that it is canceled by a second negation on the sibling of *all*. The semantic translation is $\text{glitters}'(\dot{x}) \wedge \neg\text{gold}'(\dot{x})$, which can be read intuitively as "there is a thing $\dot{x}$ such that $\dot{x}$ glitters, but $\dot{x}$ is not gold."

Next we consider an example in which there are two literals in the restriction of the quantifier; it illustrates the use of Modification (14).

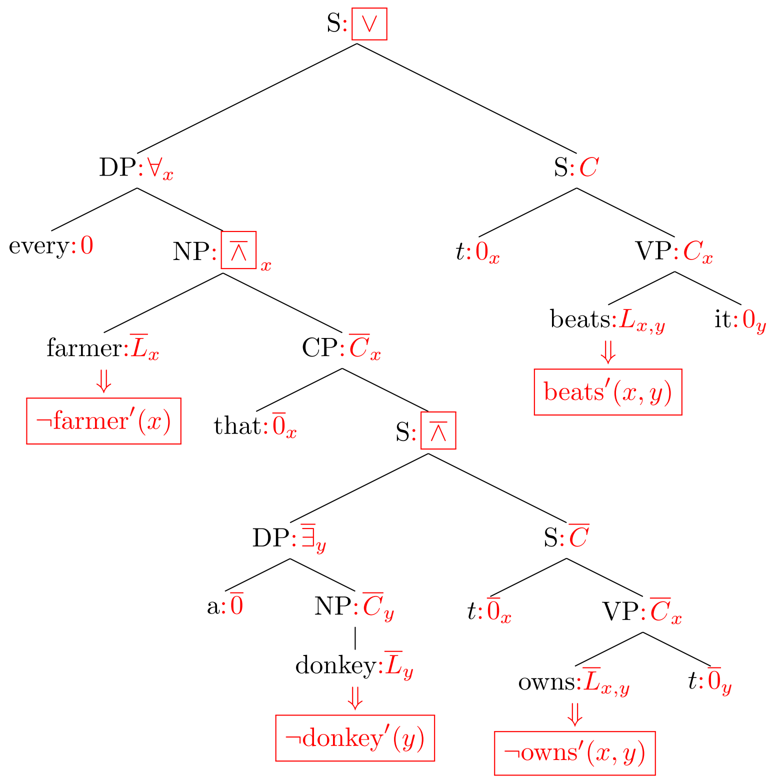
(26)



Modification is used to combine *donkey* with the relative clause. It propagates the variable on NP to both the head noun and the modifier. The translation is $\neg\text{donkey}'(x) \vee \neg\text{brayed}'(x) \vee \text{owns}'(\text{John}', x)$.

At this point, we have illustrated all of the interpretation rules. Now we turn to the classic donkey-anaphora example, "every farmer who owns a donkey beats it" (Geach 1962).

(27)



The upper DP (headed by *farmer*) is of type $\forall$, hence its parent S is of type $\vee$. The lower DP (headed by *donkey*) has basic type $\exists$ because its determiner is existential, hence its parent has basic type $\wedge$. However, both are negated because they occur within the restriction of a universal. The index terms for both DPs are variables, not Skolem terms, because the signed types are universal: $\forall$ and $\overline{\exists}$, respectively. (Note that the basic type of the NP dominating *farmer* is $\wedge$, not because of an existential quantifier, but because it is a case of Modification.)

The only thing that is new is the pronoun. Its antecedent is the DP headed by *donkey;* hence its index is y, by Anaphora (17). This—rather magically—gives the correct interpretation:

$$\neg\mathrm{farmer}'(x) \vee \neg\mathrm{owns}'(x,y) \vee \mathrm{beats}'(x,y)$$

See section 3 below for details on the CNF treatment of anaphora.

2.8 Shorthand

Despite the simplifications we have made, using trees to compute translations consumes considerable space. We can define a shorthand for computing the translation directly from a partially bracketed sentence. Rather than giving a formal definition, we give an informal characterization. Take the LF structure, and add annotations according to the following templates:[8]

(28) $\begin{array}{lll} \text{some}_{\dot{u}}\ \mathit{NP} \wedge & \wedge \text{ that } S & \text{not } \overline{S} \\ \text{every}_u\ \overline{\mathit{NP}} \vee & \text{Adj} \wedge \text{NP} & \end{array}$

For the determiners, "u" or "$\dot{u}$" represents the variable assigned to the determiner's parent node.

For example, consider "every dog that chases every cat is happy." Switching to LF word order and adding annotations, this becomes:

(29) $\text{every}_x\ \overline{\text{dog} \ \wedge\ \text{that every}_y\ \overline{\text{cat}}\ \vee\ \text{chases}}\ \vee\ \text{is happy}$

To read off the interpretation, one makes the following adjustments in a negative context (that is, in any region with an odd number of superscribed lines): add dots to undotted variables, remove dots from dotted variables, place "¬" before each content word, and invert the connectives ∧ and ∨. Then one adds outscoping universal variables as subscripts on Skolem variables; one copies the variables to the argument positions of content words, as appropriate; and finally one deletes everything except the literals and connectives. Grouping follows the LF tree structure. For (29), the result is (30):

(30) $\neg\text{dog}'(x) \vee [\text{cat}'(\dot{y}_x) \wedge \neg\text{chases}'(x, \dot{y}_x)] \vee \text{happy}'(x).$

Note that a dot has been added to the variable y (it has been converted to a Skolem variable) because it appears in a negative context. The resulting Skolem function $\dot{y}$ has argument x because its DP, *every cat,* appears within the scope of *every dog* at LF.

Here is an example with wide-scope *not.* The (a) line is the S-structure, the (b) line is the LF, and the (c) line is the translation.

(31)
a. every dog that chases a cat_1 does not catch it_1

b. $\text{not } \overline{\text{every}_x\ \overline{\text{dog} \ \wedge\ \text{that a}_{\dot{y}} \text{ cat} \wedge t_x \text{ chases } t_{\dot{y}}}\ \vee t_x \text{ catches it}_{\dot{y}}}$

c. $\text{dog}'(\dot{x}) \wedge \text{cat}(\dot{y}) \wedge \text{chases}'(\dot{x}, \dot{y}) \wedge \neg\text{catches}'(\dot{x}, \dot{y})$

There are multiple occurrences of x and $\dot{y}$ in (31b). The occurrences on the determiners (*every*$_x$ and *a*$_{\dot{y}}$) are the *primary occurrences,* and the context of the primary occurrence determines whether the variable remains as it is or switches between dotted and undotted. The secondary occurrences are all on anaphors that take a primary occurrence as antecedent.

The variable x receives a dot because the primary occurrence is in a negative context. The other occurrences of x follow suit, regardless of their own context. That is, all occurrences of x are replaced uniformly with $\dot{x}$. The primary occurrence of $\dot{y}$ occurs in a positive context (double negative is positive), so $\dot{y}$ remains unchanged. All occurrences of $\dot{y}$ remain unchanged, regardless of their own context. In (31b), x outscopes $\dot{y}$, but because x ends up as a dotted

[8]The templates are largely sufficient for the examples we consider, but they are not exhaustive.

variable, it is not an argument of $\dot{y}$: only outscoping *universal* variables appear as arguments of a Skolem function.

3 Anaphora

One of the main motivations for Discourse Representation Theory (see Kamp and Reyle 1993), and its compositional cousin Dynamic Predicate Logic (Groenendijk and Stokhof 1991), is the treatment of pronouns—in particular, cross-sentential anaphora and intra-sentential donkey anaphora. In this section, therefore, we detail our treatment of anaphora and highlight the differences from the DPL approach. (We skip DRT since DPL was designed as a compositional analog of DRT).

3.1 Basic intra-sentential anaphora

The CNF system allows for basic intra-sentential anaphora via the mechanism of indexation. Two co-indexed nodes will have the same referent in this system, whether this referent is determined by a constant, a regular variable, or a Skolem term. The following examples illustrate the treatment of typical referential and bound pronouns:

(32) a. $\text{every}_x\ \overline{\text{dog}} \vee \text{loves himself}_x \quad \Rightarrow \quad \neg\text{dog}'(x) \vee \text{loves}'(x,x)$
b. $\text{some}_{\dot{x}}\ \text{dog} \wedge \text{loves himself}_{\dot{x}} \quad \Rightarrow \quad \text{dog}'(x) \wedge \text{loves}'(x,x)$
c. $\text{John}_{\text{John}'}\ \text{loves himself}_{\text{John}'} \quad \Rightarrow \quad \text{loves}'(\text{John}', \text{John}')$

3.2 Discourse (inter-sentential) anaphora

One benefit of the DPL system over previous approaches is that it clearly enumerates the circumstances under which pronouns may co-refer with quantifiers in previous sentences. In the standard approach, pronouns translate as variables, and a formula containing unbound variables represents the set of assignments that verify the formula. For instance the meaning of the sentence *it*$_i$ *brayed* is the set of all assignments mapping i to an individual that brayed:

$$\{g \mid g(i) \in [\![\text{brayed}]\!]\}$$

In DPL, on the other hand, sentences denote sets of *pairs* of assignments, which represent input-output contexts. The same sentence in DPL thus denotes the set of all pairs of assignments $\langle g, h\rangle$ such that $g(i)$ brayed and h is the assignment after evaluating this sentence. Since there are no defined dynamic elements in this sentence, h will be the same as g, yielding:

$$\{\langle g, g\rangle \mid g(i) \in [\![\text{brayed}]\!]\}$$

Sentences with dynamic elements may change the output assignment, for instance to add a referent. Thus, the DPL meaning for the sentence "a donkey$_i$ brayed"—which contains a dynamic existential quantifier—is as follows:

$$\{\langle g,h\rangle \mid \exists k : k[i]g \wedge k(i) \in [\![\text{donkey}]\!] \wedge \langle k,h\rangle \in [\![i \text{ brayed}]\!]\}$$

This is the set of all pairs of assignments $\langle g,h\rangle$ such that there is a third assignment k differing from the input function g at most in its value for i such that $k(i)$ is a donkey and the pair $\langle k,h\rangle$ is in the denotation of *i brayed.* The dynamic existential quantifier has the effect of adding an individual (the one that brayed) to the input assignment g and using the result as the output assignment h.

Next, DPL treats discourse sequencing by using the output assignment of one sentence as the input assignment of the subsequent sentence. Therefore, a short discourse such as "*A donkey brayed. It was hungry.*" will use the output assignment of the first sentence, containing a referent for the braying donkey, as the input assignment for the second sentence, allowing the pronoun *it* to refer back to this donkey.

In the CNF system, on the other hand, there are no explicit quantifiers in the interpretations of sentences. Instead, Skolem variables are implicitly existentially quantified over whole discourses, and universal variables are implicitly universally quantified within the scope of the Skolem variables, as described in Section 1.2. Thus, the same discourse would simply yield the following CNF formulas:

(33) a. $\text{a}_{\dot{x}}\ \text{donkey} \wedge \text{brayed} \quad \Rightarrow \quad \text{donkey}'(\dot{x}) \wedge \text{brayed}'(\dot{x})$
b. $\text{it}_{\dot{x}}\ \text{was hungry} \quad \Rightarrow \quad \text{hungry}'(\dot{x})$

The Skolem Variables rule (19) specifies that a DP headed by *a*, in a positive context, has as index term a Skolem function applied to an argument list. In the case of (33a), the argument list is empty, since there are no outscoping universal variables. Next, the Anaphora rule (17) requires a pronoun to share an index with its antecedent, ensuring that *it* has the same index as *a donkey.* Sentence interpretations are implicitly conjoined, and so the final CNF formula (intension) is

$$\text{donkey}'(\dot{x}) \wedge \text{brayed}'(\dot{x}) \wedge \text{hungry}'(\dot{x}).$$

The extension of this formula is determined by existentially quantifying over the zero-arity Skolem function $\dot{x}$, yielding the meaning "there was a donkey that brayed and was hungry."

DPL defines universal quantifiers such as *every donkey* such that they do not change the assignment for subsequent sentences, correctly ruling out sequences such as the following (although see section 4.2 below for a discussion of exceptions to this rule):

(34) Every donkey_i brayed. $^*\text{It}_i$ was hungry.

As things stand, the CNF system allows such sequences. The sentence in (34) has the following translation in CNF:

(35) a. $\text{every}_x\ \overline{\text{donkey}} \vee \text{brayed} \quad \Rightarrow \quad \neg\text{donkey}'(x) \vee \text{brayed}'(x)$
b. $\text{it}_x\ \text{was hungry} \quad \Rightarrow \quad \text{hungry}'(x)$

Even worse, this yields the following meaning for the discourse, consisting of two CNF clauses:

(36) $[\neg \text{donkey}'(x) \vee \text{brayed}'(x)] \wedge \text{hungry}'(x)$.

In words, "every donkey brayed, and everything was hungry." Obviously, this is not the correct interpretation even for this aberrant English discourse.

A fairly natural way to refine the system is suggested by the following observation. The expression (36) is logically equivalent to the following:

(37) $[\neg \text{donkey}'(x) \vee \text{brayed}'(x)] \wedge \text{hungry}'(y)$.

In other words, there is no connection conveyed between being a (braying) donkey and being hungry; the only contribution that the universal quantifier *every donkey* makes towards the interpretation of the pronoun *it* is to make it a universal variable, rather than a Skolem term. To put it another way, the co-indexation is vacuous, in the sense that the interpretation of (34) is logically equivalent to the interpretation of the same discourse *without* the anaphora:[9]

(38) Every donkey$_x$ brayed. It$_y$ was hungry.

We define:

(39) An anaphor α is **vacuously bound** in a discourse ϕ if $[\![\phi]\!]$ is logically equivalent to $[\![\phi']\!]$, where ϕ' is identical to ϕ except that the index of α is replaced with a new universal variable, not appearing elsewhere in the discourse.

We add the following constraint on anaphora.

(40) **Prohibition against Vacuous Binding**: A pronoun may not be vacuously bound.

The purpose of co-indexing two items is to indicate a certain pattern of co-reference between them. It makes sense to disallow such co-indexing when the two items in question are not referentially linked whatsoever. See Büring (2005, Ch. 6) for discussion of similar issues in standard binding theory.

The prohibition in (40) also rules out certain illicit cases where the quantifier and pronoun appear in the same sentence. For instance, consider the following sentence and its LF representation:

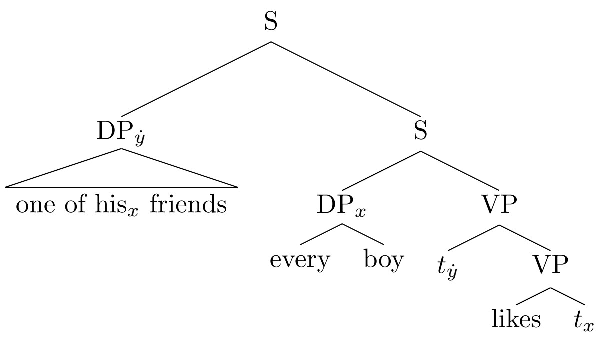
(41) One of his$_x$ friends likes every$_x$ boy.

[9]Note that the following are *not* logically equivalent:

(i) a. $\exists x \,.\, [A(x) \wedge B(x)]$
b. $[\exists x \,.\, A(x)] \wedge [\exists y \,.\, B(y)]$

Expression (ia) entails (ib), but not vice versa. Hence, when a pronoun translates as a Skolem term, it is not vacuously bound, even when its antecedent occurs in a separate CNF clause.

(42)



This sequence is not ruled out by any syntactic binding condition (Chomsky 1981), since the coindexed *his* and *every boy* do not c-command one another. Furthermore, it does not run afoul of any prohibition against crossover, since the two items are in their surface order. However, such an indexing is ruled out by (40). The meaning derived would be something like the following:

(43) $\text{friend-of}'(\dot{y}, x) \wedge [\neg\text{boy}'(x) \vee \text{likes}'(\dot{y}, x)]$

In other words, there is a someone $\dot{y}$ who is friends with everyone, and $\dot{y}$ likes every boy. This is obviously not the correct interpretation for (41). Notice, however, that the anaphor *his* is vacuously bound: the variable x in the first clause—contributed by the index on the pronoun *his*—could be replaced by a fresh universal variable without changing the meaning of the discourse. Hence (40) rules out *every boy* as a possible antecedent for *his* in this construction.

3.3 Donkey anaphora

Another benefit to the DPL system is that it allows dynamic elements to change the assignment function in the middle of computing the denotation of a sentence. This comes in handy in interpreting so-called donkey anaphora, pronouns which seem to refer back to existential quantifiers not in a position to bind them in standard predicate logic. The DPL definition of universal quantification allows this by using the output assignment *as changed by its restriction* as the input assignment to its nuclear scope.[10] So, for instance, the classic donkey-anaphora example *every farmer who owns a donkey beats it* (Geach 1962) is evaluated in such a way that the input assignment for the nuclear scope *beats it* contains a suitable referent for the pronoun *it*—a referent "added" to the assignment during the computation of the restriction *farmer who owns a donkey.* As seen in Section 2.7.3 above, the CNF system handles this type of example without any further machinery.

[10]This is a simplification. DPL actually quantifies over all possible output assignments of the restriction and uses each as a potential input assignment for the nuclear scope.

3.4 Negation, Disjunction, and Universal Quantification

As seen above, DPL was designed to allow existential quantifiers to effectively extend their scope between two sentences sequenced in a discourse and between the restriction and nuclear scope of a quantifier. In addition, the DPL definition of conjunctive *and* renders it synonymous with discourse sequencing. Based on examples such as the following, however, the DPL definitions for negation, disjunction, and universally quantified sentences were designed to block existentials from scoping beyond a certain point:

(44) a. John doesn't own a car$_i$. *It$_i$'s in his driveway.
b. Every farmer who owns a donkey$_i$ beats it$_i$. *It$_i$ is very stubborn.
c. *John owns a car$_i$ or it$_i$'s in his driveway.

In the CNF approach, the Prohibition against Vacuous Binding defined in (40) above correctly rules out the first two cases. The analysis for (44a) goes as follows:

(45) a. $\text{not } \overline{\text{a}_{\dot{x}} \text{ car} \wedge \text{John}_{\text{John}'} \text{ owns } t_{\dot{x}}} \wedge \text{It}_{\dot{x}} \text{ is in his driveway}$
b. $[\neg\text{car}'(x) \vee \neg\text{owns}'(\text{John}', x)] \wedge \text{in-driveway}'(x)$

This final formula runs afoul of the Prohibition against Vacuous Binding, since the pronoun *it* is vacuously bound: if its interpretation x is replaced with a new variable y, the result is logically equivalent to (45b):

(46) $[\neg\text{car}'(x) \vee \neg\text{owns}'(\text{John}', x)] \wedge \text{in-driveway}'(y)$

The analysis for (44b) is a little more complex, but yields a very similar result:

(47) a. $\text{every}_x\ \overline{\text{farmer} \wedge \text{who a}_{\dot{y}} \text{ donkey} \wedge \text{owns}} \vee \text{beats it}_{\dot{y}}$
$\wedge\ \text{it}_{\dot{y}} \text{ is stubborn}$
b. $[\neg\text{farmer}'(x) \vee \neg\text{donkey}'(y) \vee \neg\text{owns}'(x, y) \vee \text{beats}'(x, y)]$
$\wedge\ \text{stubborn}'(y)$

Once again, the last instance of *it* is vacuously bound, since the y in stubborn$'(y)$ could be replaced with a new variable without altering the truth conditions.

The Prohibition against Vacous Binding does not rule out (44c). However, examples of this form are not uniformly infelicitous. For instance, consider the following disjunctions, which sound fine despite exhibiting a scoping that is illegal under the rules of DPL:

(48) a. John bought a car$_i$, or perhaps he stole it$_i$
b. John bought a car$_i$, or Mary bought it$_i$
c. A dog$_i$ ate our dinner, or at least it$_i$ ate most of our dinner.

And, in fact, the original example (44c) improves if you make it a more plausible disjunction, such as the following:

(49) I see John owns a Ferrari$_j$, or maybe a rich friend parked it$_j$ in his driveway.

Under the CNF account, (44c) and (48a–c) all involve pronouns whose antecedents bear Skolem indices, hence they are all predicted to be good.

So, what is the difference between cases like (49) that sound good and those such as (44c) that do not? Although we leave this as an open question, we suggest that cases like (44c) may be infelicitous for reasons of discourse coherence. The sentence in (44c) involves the disjunction of two seemingly unrelated statements: one about John owning a car, and the second about a car being in his driveway. To the extent that the sentences can be related by supplying more context, such as in (49), the example improves.

3.5 The Limits of DPL

The strict rules of DPL quickly run into problems in cases quite similar to those given above, as pointed out by Groenendijk and Stokhof (1991) themselves:

(50) a. Either there isn't a bathroom$_i$ here or it$_i$'s in a funny place.
b. Every player chooses a token$_i$. It$_i$ goes on square one.

The simplest formulation of DPL predicts these sentences to sound odd, since they involve scoping out of negation, disjunction, and/or a universally quantified sentence; but they are actually entirely acceptable. Groenendijk and Stokhof (1991) propose extensions to accommodate examples like these. By contrast, the CNF system makes the correct predictions without modification. For instance, consider the following analysis for (50a):

(51) a. There isn't $\overline{\text{a}_{\dot{x}} \text{ bathroom} \wedge \text{here}}$ or $\vee$ it$_{\dot{x}}$ is in a funny place
b. $\neg\text{bathroom}'(x) \vee \neg\text{here}'(x) \vee \text{in-funny-place}'(x)$

Since the existential *a bathroom* is negated, it translates as a universal variable. The resulting interpretation is the same as the interpretation of "every/any bathroom here is in a funny place."

Similarly, (50b) has the following analysis under the CNF system:

(52) a. [every$_x$ $\overline{\text{player}}$ $\vee$ [a$_{\dot{y}}$ token $\wedge$ t_x chooses $t_{\dot{y}}$]] $\wedge$ it$_{\dot{y}}$ goes on square one
b. $[\neg\text{player}'(x) \vee [\text{token}'(\dot{y}_x) \wedge \text{chooses}'(x, \dot{y}_x)]] \wedge \text{goes-on-sq-one}'(\dot{y}_x)$

Remember for the last clause here that $\dot{y}_x$ actually represents the application of the function $\dot{y}$ to the argument x, and x is implicitly universally bound. Thus goes-on-sq-one$'(\dot{y}_x)$ means that for all x, $\dot{y}(x)$ goes on square one.

The grouping, and in particular the fact that the second sentence introduces a separate CNF clause, makes for a certain subtlety. The first clause establishes for all players p that $\dot{y}(p)$ is a token. However, for individuals x that are not players, the complete discourse (52b) is satisfied as long as $\dot{y}(x)$ is on square one, regardless whether $\dot{y}(x)$ is a token or not. Let us call anything that is not a player's token an "extraneous item." The discourse is satisfied by a situation in which there are extraneous items on square one, but it is also satisfied if there are *no* extraneous items on square one, as long as all the players' tokens are there. (As long as there exists any player p_0, we may choose $\dot{y}(x) = \dot{y}(p_0)$

for all non-players x.) Hence the discourse *asserts* only that all players' tokens are on square one, though it is not contradicted if there happen also to be extraneous items on square one. This does coincide with the meaning of the English sentence.

3.6 Cataphora

Cataphora is another phenomenon where the predictions of DPL and the CNF approach diverge.[11] DPL admits of no cataphoric binding (besides perhaps accidental co-reference), since dynamic items such as quantifiers only affect the assignments of material after them. However, cataphoric bound pronouns are not uncommon:

(53) a. It_1 will (probably never) be used, but an officer in each county has been equipped with a tranquilizer gun_1.
b. Everyone in its_1 way tends to flee a tiger_1.

The case in (53a) is arguably a cataphoric case of standard quantifier-pronoun binding. The case in (53b) involves a cataphoric donkey pronoun.

In contrast to DPL, the CNF system treats cataphora the same as anaphora. For example, here is the CNF analysis of (53a):

(54) a. It_1 will (probably never) be used, but an officer in each county has been equipped with a tranquilizer gun_1.
b. $\text{not } \overline{\text{it}_{\dot{y}} \text{ will be used}}, \text{but} \wedge [\text{each}_c\ \overline{\text{county}} \vee$
$[\text{an}_{\dot{x}}\ [\text{officer} \wedge \text{in } t_c] \wedge \text{a}_{\dot{y}} \text{ tranq. gun} \wedge t_{\dot{x}} \text{ has been equipped with } t_{\dot{y}}]]$
c. $\neg\text{be-used}(\dot{y}_c) \wedge [\neg\text{county}(c) \vee [\text{officer}(\dot{x}_c) \wedge \text{in}(\dot{x}_c, c) \wedge \text{tranq-gun}(\dot{y}_c) \wedge$
$\text{equipped-with}(\dot{x}_c, \dot{y}_c)]]$

Classic Bach-Peters sentences (Bach 1970, Karttunen 1969, 1971) such as (55a) provide particularly complex examples of cataphora. Again, the CNF system as it stands predicts them to be good and assigns the correct interpretation, as shown in (55). For simplicity, we treat the definite determiner as a simple existential, putting the uniqueness presupposition aside.

(55) a. every pilot_1 who shot at it_2 hit the MIG_2 that chased him_1
b. $\text{every}_x\ \overline{\text{pilot} \wedge \text{who } t_x \text{ shot at it}_{\dot{y}}} \vee [\text{the}_{\dot{y}}\ \text{MIG} \wedge \text{that } t_{\dot{y}} \text{ chased}$
$\text{him}_x \ \wedge\ [t_x \text{ hit } t_{\dot{y}}]]$
c. $\neg\text{pilot}(x) \vee \neg\text{shot-at}(x, \dot{y}_x) \vee [\text{MIG}(\dot{y}_x) \wedge \text{chased}(\dot{y}_x, x) \wedge \text{hit}(x, \dot{y}_x)]$

3.7 Summary

To summarize, the CNF account handles both standard anaphora and donkey anaphora. A proposed prohibition against vacuous binding accounts for the ill-formedness of certain cases of intrasentential anaphora that are not ruled out by

[11]Thanks to David Beaver for pointing this out.

the standard binding conditions or constraints on crossover. The vacuous binding prohibition also eliminates some but not all more complex cases combining negation, disjunction and quantification; and the predictions are in good accordance with human judgments. Moreover, DPL and the CNF account differ in their predictions about the acceptability of cataphora; the facts generally accord with the predictions of the latter. On the whole, the CNF account compares favorably with DPL both on empirical coverage and simplicity.

4 Additional Issues

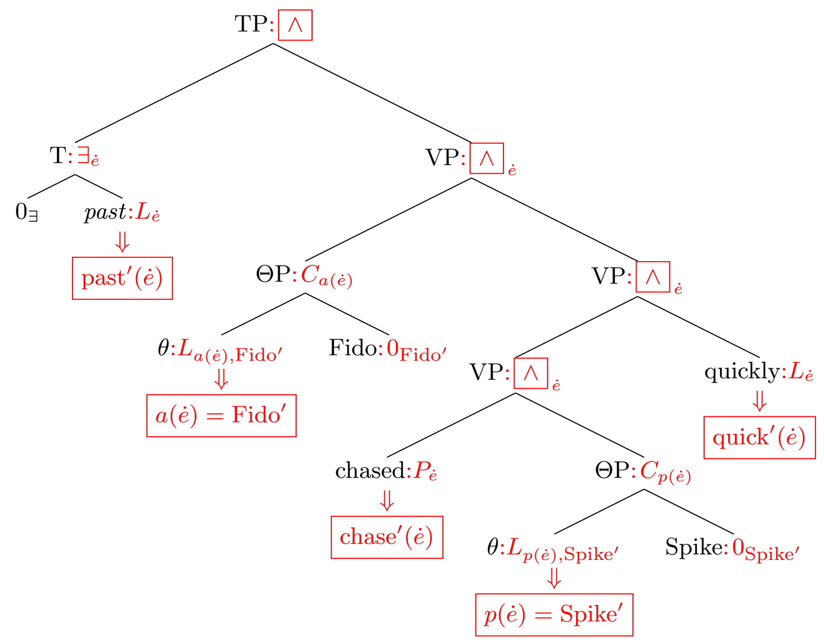
In this section, we first present a CNF version of a 'neo-Davidsonian' analysis of LF. Next, we use this to help capture certain cases of telescoping anaphora (Roberts 1987, 1989). Last, we examine a case where CNF predicts anaphoric relations not actually possible in English.

4.1 Event variables

Many researchers following Davidson (1967) have argued that the traditional account in which transitive verbs are interpreted as two-place predicates does not accommodate verbal modifiers. The neo-Davidsonian account, in which a verb denotes a subclass of situation, is more adequate, though somewhat more verbose (see Higginbotham 1985, 2000, Dowty 1988, Parsons 1990, 2000). We sketch a neo-Davidsonian version of the CNF system, for two reasons. First, it necessitates some modifications to the system; and second, event variables play a role in our account of certain cases of telescoping, discussed below.

Let us consider a simple example, "Fido chased Spike quickly." We assume the following syntactic structure.

(56)


TP: $\boxed{\wedge}$
T: $\exists_{\dot e}$
$0_\exists$
past: $L_{\dot e}$
$\boxed{\text{past}'(\dot e)}$
VP: $\boxed{\wedge}_{\dot e}$
ΘP: $C_{a(\dot e)}$
θ: $L_{a(\dot e),\text{Fido}'}$
$\boxed{a(\dot e) = \text{Fido}'}$
Fido: $0_{\text{Fido}'}$
VP: $\boxed{\wedge}_{\dot e}$
VP: $\boxed{\wedge}_{\dot e}$
quickly: $L_{\dot e}$
$\boxed{\text{quick}'(\dot e)}$
chased: $P_{\dot e}$
$\boxed{\text{chase}'(\dot e)}$
ΘP: $C_{p(\dot e)}$
θ: $L_{p(\dot e),\text{Spike}'}$
$\boxed{p(\dot e) = \text{Spike}'}$
Spike: $0_{\text{Spike}'}$

Notice the use of event variables and the new thematic-argument phrases (ΘP). These ΘPs provide a place to attach the semantics of theta-role assignment—$a(\dot e)$ for agent and $p(\dot e)$ for patient—and they will also play a role in our analysis of telescoping below. These changes require additions to our interpretation rules, as we outline here.

Q-Scope Variant. For consistency with the Q-Restriction rule (12), the tense element T is treated as consisting of an empty determiner-like element with semantic type 0, and an empty literal that introduces the time predicate.

A modified version of the Q-Scope rule (13) is needed to combine T with VP. Specifically, Q-Scope as it currently stands requires the non-$\exists$ child (VP) to have no index. This is appropriate when the presence of the index indicates an unsaturated argument, but in the Davidsonian approach, a VP is analogous to an NP in the sense that the index does not represent an unsaturated argument but rather the object denoted by the phrase—in our example, $\dot e$ represents the chasing event denoted by the VP.

Here is the new variant of the Q-Scope rule. The child α is restricted to being a (semantic) quantifier that has not undergone QR—i.e., it is limited to being a tense element.

(57) $$\left[\!\!\left[\begin{array}{c} \gamma : \wedge \\ \alpha : \exists_u \qquad \beta : \tau_u \end{array} \right]\!\!\right] = [\![\alpha]\!] \wedge [\![\beta]\!] \qquad \left[\!\!\left[\begin{array}{c} \gamma : \overline{\wedge} \\ \alpha : \overline{\exists}_u \qquad \beta : \overline{\tau}_u \end{array} \right]\!\!\right] = [\![\alpha]\!] \vee [\![\beta]\!]$$

Theta-role Assignment. An abstract head is also introduced for theta-role assignment. The nodes labeled θ in (56) are abstract words that translate as equality. They are combined with their DP arguments (*Fido, Spike*) using Application (10).

We therefore require a new rule to combine the resulting ΘP with VP. Like the Modification rule, the Theta-role Assignment rule applies when neither child is of type 0 or quantificational ($\forall$, $\exists$). Modification applies when one child modifies the other, and Theta-role Assignment applies when one child (β) is a syntactic argument of the other (α). The function θ in the rule represents the theta role assigned to the argument.

(58) $$\left[\!\!\left[\begin{array}{c} \gamma : \wedge_u \\ \alpha : \sigma_u \quad \beta : \tau_{\theta(u)} \end{array} \right]\!\!\right] = [\![\alpha]\!] \wedge [\![\beta]\!] \qquad \left[\!\!\left[\begin{array}{c} \gamma : \overline{\wedge}_u \\ \alpha : \overline{\sigma}_u \quad \beta : \overline{\tau}_{\theta(u)} \end{array} \right]\!\!\right] = [\![\alpha]\!] \vee [\![\beta]\!]$$

With these added rules, the CNF system yields the meaning indicated above in (56): there is a past chasing event $\dot{e}$ whose agent is Fido and whose patient is Spike.

4.2 Telescoping

Roberts (1987, 1989) describes a phenomenon she calls **telescoping**, wherein a universal quantifier in one sentence seems to bind a pronoun in a subsequent sentence:

(59) [Each candidate]$_i$ approached the stage. He$_i$ shook the dean's hand and returned to his$_i$ seat.

The CNF account as developed so far predicts this discourse to be ruled out, on the grounds that it violates the Prohibition against Vacuous Binding:

(60) $[\neg\text{candidate}'(x) \vee \text{approached-stage}'(x)] \wedge \text{shook-deans-hand}'(x) \wedge \text{returned-to-seat}'(x)$

Before suggesting a solution to this problem, we turn to another type of sentence discussed by Roberts (1989), where an existential below a universal appears to scope across two sentences. We saw an example of this in Section 3.5 above:

(61) Every player chooses [a token]$_i$. It$_i$ goes on the first square.

As shown above, this kind of example is captured straightforwardly in the CNF system. We repeat the analysis here:

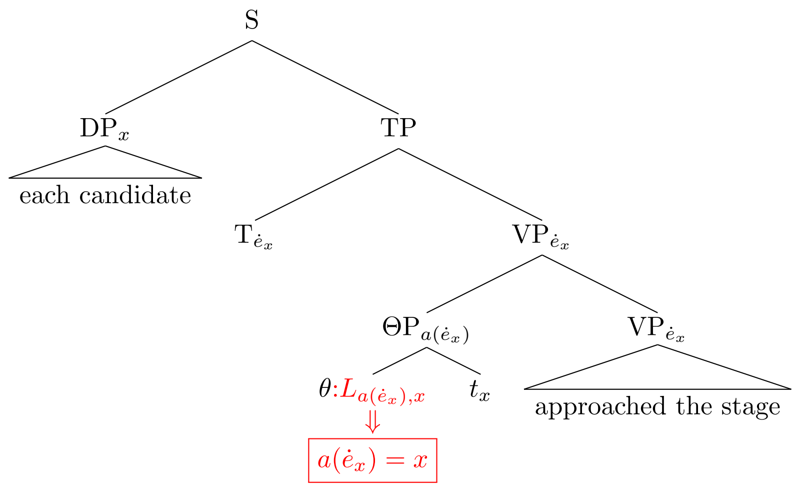
(62) a. $[\text{every}_x\ \overline{\text{player}} \vee [\text{a}_{\dot{y}}\ \text{token} \wedge t_x\ \text{chooses}\ t_{\dot{y}}]] \wedge\ \text{it}_{\dot{y}}\ \text{goes on square one}$
b. $[\neg\text{player}'(x) \vee [\text{token}'(\dot{y}_x) \wedge \text{chooses}'(x, \dot{y}_x)]] \wedge \text{goes-on-sq-one}'(\dot{y}_x)$

The Skolem term $\dot{y}_x$ is contributed by the quantified DP *a token.* The pronoun *it* is coindexed with *a token* and therefore also contributes the term $\dot{y}_x$ to the interpretation of the second sentence. In this case, the binding is not vacuous, and no violation results.

So, the CNF account easily handles one of Roberts' telescoping cases, but what about the other? Is there any way to capture the binding in (59) with a Skolem term, similar to that in (62b)? The event variables in the Davidsonian analysis of section 4.1 provide one possibility.

Let us consider the Davidsonian analysis of the first sentence of (59). For clarity, we show only the indices, not the full semantic types (with the exception of the θ node):

(63)


S
DP_x
each candidate
TP
$\mathrm{T}_{\dot{e}_x}$
$\mathrm{VP}_{\dot{e}_x}$
$\Theta\mathrm{P}_{a(\dot{e}_x)}$
$\theta{:}L_{a(\dot{e}_x),x}$
t_x
$\mathrm{VP}_{\dot{e}_x}$
approached the stage
$\Downarrow$
$\boxed{a(\dot{e}_x) = x}$

The translation of (63) is:

(64) $\neg\text{candidate}'(x) \vee [a(\dot{e}_x) = x \wedge \text{approach-stage}'(\dot{e}_x)]$

Note that, under the Davidsonian analysis, the predicate approach-stage$'$ is true of events in which someone approaches the stage. The Skolem variable $\dot{e}_x$ represents the stage-approaching event for each candidate x.

The innovation we require to enable telescoping in this case is to assume that not only the subject trace, but also the ΘP, are potential antecedents for pronouns in subsequent sentences. Consider the continuation "he shook the dean's hand and returned to his seat." Choosing the trace as antecedent of "he" violates Vacuous Binding, but choosing the ΘP does not. If we choose ΘP as antecedent, the pronoun's index is the Skolem term $a(\dot{e}_x)$, analogous to the Skolem term $\dot{y}_x$ in (62b).

A subtlety does arise. If we take the second clause ("he shook the dean's hand") to introduce an unrelated event variable $\dot{s}$, we get the interpretation:

(65) $[\neg\text{candidate}'(x) \vee [a(\dot{e}_x) = x \wedge \text{approach-stage}'(\dot{e}_x)]]$
$\wedge\, a(\dot{s}) = a(\dot{e}_x) \wedge \text{shake-hand}'(\dot{s})$

Because $\dot{s}$ names a single event, $a(\dot{s})$ is a single individual a_0 (the agent of the event). Under this interpretation, the second clause states that the agent of $\dot{e}_x$

equals a_0, for all x, entailing that all stage-approaching events have the same agent a_0.

It does not seem necessary to rule this interpretation out semantically, inasmuch as it flaunts Gricean principles (Grice 1975): it is deceptive to say "every candidate" instead of "the only candidate" in a situation where there is only one candidate.[12] If one entertains the possibility that the speaker is flaunting Gricean principles in this manner, the interpretation does seem to be accessible: every candidate approached the stage, and he (the one and only candidate) shook the dean's hand etc.

However, we would like also to *allow* an interpretation in which the event variable of the second clause is a function of the event of the first clause—what we might call an *anaphoric tense* (cf. Partee 1973). Specifically, let "$s(\dot{e})$" denote the event that is the narrative successor of event $\dot{e}$. We would like to allow the T node in the second sentence to bear the index $s(\dot{e})$, instead of a freshly-allocated Skolem term—and indeed, assigning the index $s(\dot{e})$ to the T node *expresses* the narrative succession discourse relation.

This provides the correct interpretation. The event variable for the second sentence, "he shook the dean's hand," is $s(\dot{e}_x)$, and the pronoun "he" receives the index $a(\dot{e}_x)$ by taking the subject ΘP of the first sentence as antecedent. The CNF for the two sentences together comes out as:

(66) $$[\neg\text{candidate}'(x) \vee [a(\dot{e}_x) = x \wedge \text{approach-stage}'(\dot{e}_x)]] \\ \wedge\, a(s(\dot{e}_x)) = a(\dot{e}_x) \wedge \text{shake-hand}'(s(\dot{e}_x))$$

The modification needed to permit anaphoric tenses actually concerns the index-assignment rules of section 2.5, rather than the rules of interpretation. The Skolem Variables rule (19) needs to be revised to permit Skolem terms to be assigned to T nodes (and not only to DP nodes), while also permitting a T node optionally to receive the index $s(\dot{e})$, where $\dot{e}$ is the event variable of the previous sentence. A more formal statement would be premature at this point, in the absence of a systematic study of the referential possibilities for T nodes.

We do note one issue that bears on the statement of the Skolem Variables rule. Consider a discourse such as (67):

(67) a. Each player$_1$ approaches the board. b. He$_1$ chooses a token.

Suppose *each player* receives the index x. The DP *a token* in the second sentence is assigned a Skolem variable, $\dot{y}$, and since *a token* is not outscoped by *each player,* it would appear that $\dot{y}$ has no arguments. In the resulting reading, there is only one token; every player chooses the same token. Empirically, this is clearly incorrect.

However, under the Davidsonian analysis, the event variable of the first sentence is $\dot{e}_x$, and the event variable of the second sentence, if anaphoric, is $s(\dot{e}_x)$. If *a token* scopes below TP in the second sentence, then the x in the event

[12]This may also be viewed as a consequence of Heim's (1991) 'Maximize Presupposition' principle.

variable $s(\dot{e}_x)$ is an outscoping universal variable, and the index for *a token* is $\dot{y}_x$, correctly yielding a reading in which each player chooses a different token.

When we defined the Skolem Variables rule, we noted that its statement was slightly nonstandard in that the argument list of a Skolem function consisted of all universal variables *occurring in* the index terms of outscoping quantifiers. Under the standard definition the argument list would include only universal variables that *are* index terms of outscoping quantifiers. In all previous cases, the two definitions have been equivalent, but for (67), they diverge, and the standard definition would yield the wrong reading.

Finally, we would like to make some comments about the postulated narrative successor function $s(\cdot)$. We would like to emphasize that it is not merely an expedient way of converting a universal variable to something that behaves like a Skolem function. Empirically, when the discourse relation is something other than narrative succession, telescoping is usually unavailable. Consider the following sentence and several potential next sentences (after Fodor and Sag 1982):

(68) Each of the students was accused of cheating on the exam.

(69) a. He was reprimanded by the dean.
b. #He has a Ph.D. in astrophysics.
c. #He was failing the course and wanted a better grade.

The continuation in (69a) is a valid case of telescoping, but (69b) sounds odd. Wang et al. (2006) discuss such cases at length and determine that the discourse relation between two sentences is crucial in determining whether they will support telescoping. A Narration relation holds between (68) and (69a), but a Background or Explanation relation holds between (68) and (69c). Possibly, the Background and Explanation discourse relations fail to identify a unique successor situation, hence fail to provide an event like $s(\dot{e}_x)$ in (66) that enables the telescoping interpretation.

To complicate matters, Roberts (1989) points out that (69b) sounds fine as successor to a different sentence, to which it bears a different discourse relation:

(70) Each candidate for the space mission meets all our requirements. He has a Ph.D. in astrophysics and extensive prior flight experience.

In this case, the discourse relation is Elaboration, which, like Narration, appears to license telescoping.

Perhaps what Narration and Elaboration have in common is that they involve a single large event divided into subevents. Unfortunately, it is unclear whether that genuinely sets them apart from other discourse relations, and whether it provides a compelling generalization of the successor-function mechanism that correctly discriminates between cases that permit telescoping and those that do not. We leave this as an open question.

4.3 Additional constraints and open questions

Cases remain in which the CNF system is overly permissive, allowing anaphoric relations that are empirically unavailable. It appears that two separate constraints are required. We provide tentative formulations, but we also flag significant questions that remain.

Prohibition on vacuous Skolem binding. When a Skolem variable appears in a disjunction, but only in one disjunct, one cannot refer back to it using a pronoun. Consider the following discourse:

(71) a. Either Mary got a question$_1$ wrong or I won my bet.
b. $[\text{question}'(\dot{q}) \wedge \text{got-wrong}'(\text{Mary}', \dot{q})] \vee [\text{bet}'(\dot{b}) \wedge \text{won}'(\text{I}', \dot{b})]$

(72) a. *It$_1$ was (probably) question 14.
b. $\dot{q} = \text{question-14}'$

The sentence (72) attempts to refer back to the question that Mary got wrong, but it just ends up sounding odd. Note, though, that (71) does not in fact assert that such a question exists, giving a natural explanation for why (72) is unacceptable.

More technically, define a *vacuous Skolem function* to be one that is undefined everywhere. That is, a one-place function $\dot{y}$ is vacuous if $\dot{y}(x) = \perp$ for all x, and a zero-place function $\dot{y}$ is vacuous if $\dot{y} = \perp$. There exist models and assignments for (71b) in which the Skolem function $\dot{q}$ is vacuous. In particular, in models in which I won my bet, (71b) is satisfied even if $\dot{q}$ is vacuous. Under these conditions, we define the Skolem variable $\dot{q}$ to be *potentially vacuous.*

We propose to exclude the anaphora in (72) by extending the prohibition on vacuous binding to exclude binding in which the index term of the antecedent is a potentially vacuous Skolem variable.[13] Most of the cases we have seen so far will not run afoul of this new prohibition, since they introduce Skolem variables in literals which are then connected to later clauses via conjunction. Due to the nature of conjunction, previous clauses must be true and hence the Skolem terms in them must be non-vacuous.

The prohibition does not always prevent Skolems in disjunctions from acting as antecedents. For instance, contrast (71) with (50a), repeated here as (73), where the Skolem variable appears in both preceding disjuncts:

(73) a. Either there isn't a bathroom$_1$ here or it$_1$'s in a funny place
b. $\neg\text{bathroom}'(\dot{b}) \vee \neg\text{here}'(\dot{b}) \vee \text{in-funny-place}'(\dot{b})$

In (73b), the first English clause ("there isn't a bathroom here") translates as the disjunction $\neg\text{bathroom}'(\dot{b}) \vee \neg\text{here}'(\dot{b})$, but in this case, the disjunction is

[13] Notice that a follow-up statement that acknowledges this potential vacuity actually sounds much better:

(i) ?If I lost my bet, it$_1$ was probably question 14.

not satisfied under any model if $\dot{b}$ is vacuous: if $\dot{b}$ is vacuous, then the value of $\neg\text{bathroom}'(\dot{b}) \vee \neg\text{here}'(\dot{b})$ is undefined. Since $\dot{b}$ cannot be vacuous, it is suitable as an antecedent index—that is, the pronoun *it* in the second English clause ("it's in a funny place") can refer back to the bathroom $\dot{b}$.[14]

In all the examples we have considered so far, Skolem variables arise from existential determiners. It is also possible to generate Skolem variables from universal determiners, by placing them in a negative context. In such cases, attempts to refer back to the Skolem variable fail spectacularly:

(74) a. **Every player who holds every suit$_1$ eventually gets it$_1$.

b. $\neg\text{player}'(p) \vee [\text{suit}'(\dot{s}_p) \wedge \neg\text{hold}'(\dot{s}_p)] \vee \text{gets}'(p, \dot{s}_p)$.

(75) a. Every girl that answered every$_1$ question was rewarded.
**It$_1$ wasn't even a trick question, usually.

b. $[\neg\text{girl}'(x) \vee [\text{question}'(\dot{y}_x) \wedge \neg\text{answered}'(x, \dot{y}_x)] \vee \text{was-rewarded}(x)]$
$\wedge\ \neg\text{trick-question}'(\dot{y}_x)$

The profound unacceptability of (74) and (75) suggests that the proposed prohibition against vacuous Skolem binding is not the whole story. We will turn next to a second prohibition, against a universal quantifier in a negative context serving as an antecedent for a pronoun. If negative contexts include the restrictions of universals, then (74) and (75) violate both prohibitions, giving a plausible reason why they should sound so bad.

Negative universals cannot be antecedents. An *every* phrase in the scope of a negative appears to be a bad antecedent, independently of anything else. Consider the following contrast:

(76) a. Some runner$_1$ must lose.
He$_1$ must be satisfied with doing his best.

b. $\text{runner}'(\dot{r}) \wedge \text{loses}'(\dot{r}) \wedge \text{must-be-satisfied}'(\dot{r})$

(77) a. Not every runner$_1$ wins.
*He$_1$ must be satisfied with doing his best.

b. $\text{runner}'(\dot{r}) \wedge \neg\text{wins}'(\dot{r}) \wedge \text{must-be-satisfied}'(\dot{r})$

Example (77) violates neither the prohibition on vacuous binding nor the prohibition on binding by vacuous Skolem variables. The only obvious difference between (76) and (77) is the explicit "not" in (77). The presence of "not" appears to trigger a violation even if we construct a doubly negative (hence semantically positive) context:

[14]Similarly, example (49) above continues to be admitted. The context "I see John owns a Ferrari" cannot be satisfied if the Skolem variable for "a Ferrari" is vacuous; hence binding in the following phrase "or a rich friend parked it in his driveway" is possible. In fact, this whole sentence also licenses binding of "a Ferrari" in a following sentence, such as "It's blocking my way out!"

(78) a. Every player who is missing a suit$_1$ must wait till he gets it$_1$
b. $\neg\text{player}'(p) \vee \neg\text{suit}'(s) \vee \neg\text{missing}'(p,s) \vee \text{wait-for}'(p,s)$

(79) a. *Every player who does not hold every suit$_1$ must wait till he gets it$_1$
b. $\neg\text{player}'(p) \vee \neg\text{suit}'(s) \vee \text{holds}'(p,s) \vee \text{wait-for}'(p,s)$

Examples (76)–(77) contain Skolem variables in the translation and examples (78)–(79) contain universals, so the prohibition evidently does not depend on the type of variable. In fact, on the assumption that not winning is the same as losing, (76) and (77) are logically equivalent, and assuming that not missing a suit is the same as holding it, (78) and (79) are logically equivalent. Presumably, then, whatever distinguishes the good and bad examples is syntactic. A possibility that suggests itself is that *every* in the scope of *not* is syntactically plural. Namely, changing the pronoun from "it" to "them" makes (79) good:[15]

(80) Every player who does not hold every suit$_1$ must wait till he gets them$_1$.

However, it is questionable whether (79b) is precisely the correct interpretation for (80). The pronoun *them* in (80) appears to refer explicitly to the quantifier's restriction set, the set of suits. Unfortunately, exploring the question properly would require a general treatment of pluralities within the CNF approach, and would take us beyond the scope of this paper.

We do append one observation. Anaphoric reference to the set reading of a universal quantifier appears to be an obviative strategy, in the sense that the set reading is available only when the normal distributive reading is unavailable. In (80), the distributive reading violates the prohibition against taking "not every" as antecedent. There are also cases in which the distributive reading violates the prohibition against vacuous binding, and the set reading is available as an alternative:

(81) a. Every girl$_1$ in our class works hard. *She$_1$ always turns in assignments on time.
b. Every girl$_1$ in our class works hard. They$_1$ always turn in assignments on time.

When the distributive reading *is* available, it appears generally to preclude the set reading:

(82) a. Every boy$_1$ in our class does his$_1$ work carefully.
b. *Every boy$_1$ in our class does their$_1$ work carefully.

[15]A similar change improves (74):

(i) Every player who holds every suit eventually must use them all.

5 Conclusion

We have shown how a simple homomorphism can map LF structures into an expression in conjunctive normal form and how such expressions can serve as a full semantic metalanguage. Furthermore, we have shown how the predictions of such a CNF system accord with human judgments in most cases better than those of competing systems, such as dynamic predicate logic, over a range of phenomena including donkey anaphora and telescoping.

We have left some significant issues to future work. In particular, we have not provided a method for interpreting inherently second-order phenomena, such as pluralities or generalized quantifiers (Barwise and Cooper 1981), using CNF. We also left some open questions in our discussion of telescoping (section 4.2) and in the additional constraints of section 4.3.

On balance, though, using CNF as metalanguage yields a simple system capable of explaining a large range of empirical phenomena. Furthermore, the homomorphism from LF trees to CNF allows for simple bi-directional translation to and from CNF, facilitating the connection between natural-language parsers and logical inference systems.

References

Abney, S.: 2012, Reversibility of interpretation by direct translation to CNF. Unpublished manuscript.

Bach, E.: 1970, Problominalization, *Linguistic Inquiry* **1**, 121–122.

Barwise, J. and Cooper, R.: 1981, Generalized quantifiers and natural language, *Linguistics and philosophy* **4**(2), 159–219.

Büring, D.: 2005, *Binding theory*, Cambridge Univ Pr.

Chomsky, N.: 1981, Lectures on government and binding.

Davidson, D.: 1967, The logical form of action sentences, *Essays on actions and events* **5**, 105–148.

Dowty, D.: 1988, *On the Semantic Content of the Notion of Thematic Role*, Vol. 2, Kluwer, Dordrecht, The Netherlands, pp. 69–129.

Fodor, J. and Sag, I.: 1982, Referential and quantificational indefinites, *Linguistics and philosophy* **5**(3), 355–398.

Geach, P. T.: 1962, *Reference and generality: An examination of some medieval and modern theories*, Cornell University Press.

Gödel, K.: 1939, Consistency-proof for the generalized continuum-hypothesis, *Proceedings of the National Academy of Sciences of the United States of America* **25**(4), 220.

Grice, H. P.: 1975, Logic and conversation, *in* P. Cole and J. Morgan (eds), *Syntax and Semantics 3: Speech Acts*, New York: Academic Press, pp. 41–58.

Groenendijk, J. and Stokhof, M.: 1991, Dynamic predicate logic, *Linguistics and philosophy* **14**(1), 39–100.

Heim, I.: 1991, Artikel und definitheit, *Semantik: ein internationales Handbuch der Zeitgenössischen forschung* pp. 487–535.

Herbrand, J.: 1930, *Recherches sur la théorie de la démonstration*, number 33 in *Travaux de la Societé des Sciences et des Lettres de Varsovie, Class III, Sciences Mathématiques et Physiques*, Nakł. Tow. Naukowego Warszawskiego.

Higginbotham, J.: 1985, On semantics, *Linguistic inquiry* **16**(4), 547–593.

Higginbotham, J.: 2000, *On Events in Linguistic Semantics*, Oxford: Oxford University Press, pp. 49–79.

Kamp, H.: 1981, A theory of truth and semantic representation, *Formal Semantics* pp. 189–222.

Kamp, H. and Reyle, U.: 1993, *From discourse to logic: Introduction to modeltheoretic semantics of natural language, formal logic and discourse representation theory*, Vol. 42, Kluwer Academic Dordrecht,, The Netherlands.

Karttunen, L.: 1969, Pronouns and variables, *Fifth Regional Meeting of the Chicago Linguistic Society*, pp. 108–115.

Karttunen, L.: 1971, Definite descriptions with crossing coreference: A study of the bach-peters paradox, *Foundations of Language* **7**(2), 157–182.

McCune, W.: 2003, Mace4 reference manual and guide, *Technical Report Tech. Memo ANL/MCS-TM-264*, Mathematics and Computer Science Division, Argonne National Laboratory, Argonne, IL.

Montague, R.: 1970a, *English as a formal language*, Edizioni di Communità, Milan, pp. 189–224.

Montague, R.: 1970b, Universal grammar, *Theoria* **36**(3), 373–398.

Parsons, T.: 1990, *Events in the Semantics of English*, MIT press, Cambridge, MA.

Parsons, T.: 2000, *Underlying states and time travel*, Oxford: Oxford University Press, pp. 81–93.

Partee, B.: 1973, Some structural analogies between tenses and pronouns in english, *The Journal of Philosophy* pp. 601–609.

Roberts, C.: 1987, *Modal subordination, anaphora, and distributivity*, PhD thesis, University of Massachusetts at Amherst.

Roberts, C.: 1989, Modal subordination and pronominal anaphora in discourse, *Linguistics and philosophy* **12**(6), 683–721.

Robinson, J.: 1965, A machine-oriented logic based on the resolution principle, *Journal of the ACM (JACM)* **12**(1), 23–41.

Russell, S. and Norvig, P.: 1995, *Artificial Intelligence: A Modern Approach*, Prentice Hall series in artificial intelligence, Prentice Hall.

Skolem, T.: 1934, Über die nicht-charakterisierbarkeit der zahlenreihe mittels endlich oder abzählbar unendlich vieler aussagen mit ausschliesslich zahlenvariablen, *Fundamenta mathematicae* **23**, 150–161.

Wang, L., McCready, E. and Asher, N.: 2006, Information dependency in quantificational subordination, *Where semantics meets pragmatics* pp. 267–306.

Whitehead, A. N. and Russell, B.: 1910, *Principia Mathematica*, Vol. 1, Cambridge University Press.