

A Translation from Logic to English with Dynamic Semantics^{*}

Elizabeth Coppock and David Baxter

Cycorp, Inc., Austin TX 78731, USA

Abstract. We present a procedure for translating standard predicate logic into English. The procedure generates both referring expressions and non-referring expressions, including both referential and bound variable anaphora. Non-referring expressions correspond to *short-term discourse referents* [1], which present a special set of challenges for a natural language generation system: (i) they have limited ‘lifespans’ and (ii) the determiner with which they are introduced (*every, some, any, no*) is sensitive to the logical context. Our system addresses these challenges using dynamically updated information states.

Keywords. natural language generation, dynamic semantics, predicate logic, quantification, anaphora

1 Introduction

Things change as discourse progresses. One type of change that occurs is that entities become part of the discourse context when they are first mentioned. This affects the form that referring expressions take; the cat in (1) is introduced with an indefinite, and subsequent references to it take the form of pronouns [2–9].

(1) I saw a cat and kept it. It ran away.

Another type of change that occurs during discourse is that quantifiers may begin or cease to take scope. This occurs in the following example [10, 11]:

(2) Everybody found a cat and kept it. It ran away.

On the reading of (2) in which different people found (potentially) different cats, the *it* in the second sentence cannot corefer with the *it* in the first sentence. This is because, on that reading, the discourse referent introduced by *a cat* is associated with a quantifier whose scope does not extend past the first sentence.

Dynamic semantics [10, 11, 5, 12–14], as opposed to truth-conditional semantics, e.g. [15], takes the dynamic nature of texts very seriously: “Very generally, the dynamic view on meaning comes to this: the meaning of a sentence is the

^{*} Thanks to Lucas Champollion, Elias Ponvert, David Beaver, and two anonymous reviewers for thoughtful suggestions. This work was partially supported under DARPA’s Rapid Knowledge Formation program.

change an utterance of it brings about, and the meanings of non-sentential expressions consist in their contributions to this change” [14], p. 182. But as this quotation suggests, dynamic semantics has so far taken the perspective of interpretation rather than generation of natural language.

Dynamically updated information states are useful for speakers as well as hearers. Like hearers, speakers must keep track of what discourse referents have already been mentioned; a speaker who does not do so cannot determine when to use a pronoun. Speakers must also keep track of what quantifiers and logical operators take scope over the current expression, so that they know when it is no longer possible to use a pronoun, as in (2), and in order to decide between determiners like *every*, *some*, *any*, and *no*, as we will discuss further below.

In this paper, we combine natural language generation with dynamic semantics. Specifically, the problem we aim to solve is to define a translation from standard predicate logic to English. The input may be any formula of predicate logic, and the output is a string of English text. We are not concerned with generating every possible English translation (cf. logical *completeness*), but the translations that are generated should be faithful and natural representations of the input formula (cf. logical *soundness*). “Faithfulness” can be evaluated as follows: if a formula ϕ entails another formula ψ , and $G(\alpha)$ is the natural language translation of α , then native speakers should have the intuition that $G(\phi)$ implies $G(\psi)$ (cf. [16]). At the same time, we aim to keep track of the changes that occur as the discourse progresses. Our solution is a logic-to-English translation procedure that uses dynamically updated information states.

2 Problems with non-referential ‘discourse referents’

Because the possible inputs to the logic-to-English translation problem includes quantificational formulas, it is necessary to manage both short-term and long-term *discourse referents* in Karttunen’s sense [1]: “the appearance of an indefinite noun phrase [or any noun phrase, for our purposes] establishes a ‘discourse referent’ just in case it justifies the occurrence of a coreferential pronoun or a definite noun phrase later in the text.” This definition allows the study of coreference to proceed “independently of any general theory of extralinguistic reference” (p. 367). On this definition, discourse referents can be established by expressions that genuinely refer as well as non-referential expressions such as *nobody*. Discourse referents that do not correspond to proper referents come with a special set of challenges, motivating a dynamic approach to natural language generation: they have limited ‘lifespans’ (§2.1) and they are sensitive to the presence of operators with limited scope (§2.2).

2.1 Lifespan limitations

Non-referential discourse referents have a limited ‘lifespan’ [1]. This is exemplified in (2), repeated here:

- (3) Everybody found a cat and kept it. It ran away. [11]

As mentioned above, on the narrow-scope existential reading of (3), the *it* in the second sentence cannot corefer with the *it* in the first sentence. The discourse referent introduced by *a cat* on that reading is what Karttunen [1] calls a *short-term referent*, whose lifespan ends with the end of the first sentence. From a generation perspective, the challenge is to avoid generating pronouns anaphoric to discourse referents that are no longer active.¹

A related challenge is to capture the difference between *a* and *every* with respect to the lifespan of the discourse referents they introduce [10]:

(4) If a_i farmer owns a_j /**every* $_j$ donkey, then he_i loves it_j .

A discourse referent introduced by *a* in the protasis of a conditional may extend to the apodosis, but one introduced by *every* is limited to the protasis.

2.2 Scope sensitivity

English has several determiner series for expressing quantification: *every** (*everyone, everything, everywhere, every man*, etc.); *some** (*someone, something, somewhere, some man*, etc.); *any** (*anyone, anything, anywhere, any man*, etc.); and *no** (*noone, nothing, nowhere, no man*, etc.). The choice among these depends on the logical context, i.e., the logical operators that bear scope over the current expression.

Universal quantification as in (5) is generally realized with *every**, as in (6).

(5) $\forall x[\mathbf{loves}(\mathbf{Mary}, x)]$

(6) Mary loves everything.

However, in the scope of negation, when the universal quantifier outscopes the negative operator, as in (7), the appropriate translation uses either *no**, as in (8), or *any**, as in (9).

(7) $\forall x[\neg \mathbf{loves}(\mathbf{Mary}, x)]$

(8) Mary loves nothing.

(9) Mary doesn't love anything.

In (8), the determiner also expresses negation; in (9), negation is expressed on the verb, and the variable is expressed as a negative polarity item (NPI).² The

¹ There is a class of examples that are superficially similar to (3), in which the lifespan of the relevant referent extends beyond the first sentence: *Every chess set comes with a spare pawn. It is taped to the outside of the box.* See [17] and references cited therein.

² Giannakidou [18, 19] points out that NPIs are not restricted to negative polarity environments, and argues that the relevant property is *nonveridicality*. If Giannakidou's analysis is correct, we should think of "NPI" as an acronym for "nonveridical polarity item" rather than "negative polarity item".

choice among these expression types thus depends not only on the presence of negation, but also whether negation has already been expressed.

As is well known, although universal quantification is generally expressed with *every**, as in (6), it can be expressed with *some** in the antecedent of a conditional:

$$(10) \quad \forall x[\mathbf{loves}(\mathbf{Doug}, x) \rightarrow \mathbf{loves}(\mathbf{Mary}, x)]$$

(11) If Doug loves something, then Mary loves it.

Thus, whether or not the expression is currently in the antecedent of a conditional is another factor that influences how it should be expressed. As with negation, being in the antecedent of a conditional is a transient state from the point of view of a speaker generating a sentence over time. The state ends with the antecedent; afterwards, universal quantification is expressed with *every** again, as in the apodosis of (13):

$$(12) \quad \forall x \forall y[\mathbf{loves}(\mathbf{Doug}, x) \rightarrow \mathbf{loves}(y, x)]$$

(13) If Doug loves something then everyone loves it.

The choice among these expressions also depends on the relative scoping of the logical operators. If the negation and the universal quantifier in (7) were reversed, as in (14), then (8) and (9) would not be appropriate. Rather, *every** should be used, as in (15).

$$(14) \quad \neg \forall x[\mathbf{loves}(\mathbf{Mary}, x)]$$

(15) Mary doesn't love everything.

In summary, a speaker or natural language generation device must maintain a model of which short-term referents are available, and what logical operators have scope over the current expression and in what order. These are transient states, changing as the discourse proceeds.

3 Previous work

3.1 Natural language generation

Natural language generation can be subdivided into the ‘strategic’ component (what to say) and the ‘tactical’ component (how to say it) [20]. The present work falls into the ‘tactical’ category, as the semantic content to be expressed is given. Tactical generation systems have been developed for a wide range of grammatical formalisms including Unification Categorical Grammar [21]; LFG [22, 23]; HPSG [24, 25] and TAG [26]. Seminal progress in this area was made by the development of semantic head-driven generation, a technique that can be applied to multiple grammatical formalisms [27, 28, 24, 29, 30]. To the extent that the corresponding grammatical formalisms deal with quantification and scope, so

do the corresponding generation systems. [24] deals specifically with quantifiers, addressing a problem that arises when combining “HPSG textbook semantics, with quantifier storage and contextual background conditions” with semantic head-driven generation; to solve it, they propose to “adopt recent HPSG proposals to put quantifier store and contextual background inside semantic heads” (p. 1393). However, unlike the present system, these existing tactical generation systems do not involve dynamically updated information states.

Work on generating referring expressions ([31–43], i.a.) has made steps towards a translation from logic to natural language with dynamic semantics, insofar as the systems developed in this area are sensitive to how recently a given discourse referent has been mentioned. However, these systems do not deal with short-term discourse referents, because they are designed for the task of providing an appropriate expression for referring to a given object in a domain. For example, the input might be a situation in which there is more than one book, and the book in question is on a unique table. An appropriate output for this situation would be *the book on the table*. In one respect the current task is easier: It is purely ‘tactical’ and non-‘strategic’, as the content to be expressed is given. In another respect it is harder: In order to handle the full range of predicate logic expressions, it is necessary to generate both genuinely referring expressions like *the book on the table* and non-referential expressions such as *no book*.

The papers by Shaw and McKeown [36], and Varges and Deemter [40] do address the generation of “quantificational” expressions such as *those women who have fewer than two children*, or *the people who work for exactly 2 employers*; but these expressions refer to groups, so they are actually referential, despite also being quantificational. These systems also do not take quantified logical formulas as input.

Finally, there has been some work on the problem of article choice in machine translation (see [44] and references cited therein). This work focuses on issues of definiteness (e.g. choosing between *the* and *a*), and does not address issues related to quantification and scope; these models are not based on a logical semantic representation.

3.2 Coreference

Theoretical work on coreference [7, 4, 6, 8, 9] is, as Beaver [9] points out, “dynamic, in that the core of these models is an account of the impact of an utterance on the information state of conversational participants,” whether the information state is couched in terms of ‘accessibility’ [3], ‘cognitive status’ [4], ‘attention’ and ‘salience’ [6], or ‘focus’ [7, 8]. Beaver’s theory of coreference [9], which combines Centering Theory [7, 8] with Optimality Theory [45], is a noteworthy member of this class for being reversible; it can be used to identify the antecedent of a pronoun or to determine whether a pronoun should be used in a given instance. However, work in this area does not deal with lifespan limitations or the choice between *some*, *every*, *any*, and *no*.

3.3 Dynamic semantics

Dynamic approaches to meaning [10, 5, 12–14] are specifically designed to account for the kind of phenomena described in §2. In principle, a theory of semantics is independent of any parsing or generation algorithm, so we might be expected to take an existing semantic theory “off the shelf.” But because dynamic semantic theories encode procedural notions like updates, they are not trivially reversible from parsing to generation. The Discourse Representation Structures (DRSs) of Discourse Representation Theory (DRT; [5]) – built up during the *interpretation* of a sentence – can be translated into predicate logic; one might begin by applying the reverse transformation, from predicate logic to DRSs [46]. But a complete DRS is not a good input to a generation algorithm, because the representation of the discourse must unfold as the discourse progresses.

Dynamic Montague Grammar [47], Compositional DRT [13], and the version of DRT developed by Van Leusen and Muskens [48] are declaratively stated, so it might be reasonable to consider one of them as a starting point for a generation system using dynamic semantics. But it is not standard predicate logic into which these theories translate natural language. For instance, many of the formulas that are valid in predicate logic are not valid in Dynamic Predicate Logic, the logic into which Dynamic Montague Grammar translates English, including $\phi \rightarrow \psi \equiv \neg\phi \vee \psi$; $\phi \equiv \neg\neg\phi$; $\phi \wedge \psi \equiv \psi \wedge \phi$; and $\exists x\phi \equiv \exists y[y/x]\phi$.

4 Our solution

We now present a procedure for translating predicate logic into English, which deals with the full range of discourse referents, including those with limited lifespans and scope sensitivity. The system we describe is the natural language generation system for Cyc[®] [49], a large-scale commonsense knowledge base and reasoning engine. Cyc is based on CycL, a logic that subsumes first order logic [50],⁴ and the system we describe translates from CycL to English. Here, we concentrate on the first order portion of CycL, making limited use of Cyc-specific ontological distinctions, in order to maximize the applicability of our model.

Our procedure uses two forms of dynamically updated context: the *discourse context*, which lists the discourse referents that have been mentioned, and the *operator context*, which stores information that is stripped away from the input formula. The discourse context is discussed in the following subsection; the operator context will be discussed in §4.2.

4.1 Discourse context

Definition. A discourse context D is a set of *discourse referents* [1]. Like Heim’s “file cards” [10, 11] and Karttunen’s discourse referents [1], discourse referents

³ Cyc is a registered trademark of Cycorp, Inc.

⁴ The majority of the assertions in the Cyc Knowledge Base are first-order; the majority of the remainder can be made first-order through a transformation [50].

here need not genuinely refer, and two discourse referents may in principle be associated with the same “real world” referent. Each discourse referent r is associated with a logical expression α , which can be either a closed term, composed entirely of constants (either atomic, e.g. **Mary**, or non-atomic, e.g. **Mother(Mary)** ‘Mary’s mother’) or a variable.

Insofar as the logical expression associated with these discourse referents can be either a variable or a closed term, these discourse referents are similar to Muskens’s ‘registers’ [13], and unlike the elements of DRT ‘universes’. In standard DRT [5], a proper name like *Mary* introduces a discourse condition of the form of **Mary**(x), signifying that x is named *Mary*. This is shown in the Discourse Representation Structure for the sentence *Mary owns a dog* in (a) below; a hypothetical alternative style in which the constant **Mary** is treated as a discourse referent is given in (b).

$$(16) \quad \text{a.} \quad \begin{array}{|c|} \hline x, y \\ \hline \mathbf{Mary}(x) \\ \mathbf{dog}(y) \\ \mathbf{owns}(x, y) \\ \hline \end{array} \quad \text{b.} \quad \begin{array}{|c|} \hline \mathbf{Mary}, y \\ \hline \mathbf{dog}(y) \\ \mathbf{owns}(\mathbf{Mary}, y) \\ \hline \end{array}$$

The DRT practice of treating all discourse referents as variables (as in (a)) complicates the procedure for converting a DRS into predicate logic. The first step in that procedure is to form the *matrix*, in which the conditions are conjoined with $\&$, signifying conjunction. The matrix of the DRS in (16) is as follows:

$$(17) \quad \mathbf{Mary}(x) \ \& \ \mathbf{dog}(y) \ \& \ \mathbf{owns}(x, y)$$

Next, the discourse referents are translated as existentially bound variables, yielding the formula:

$$(18) \quad \exists xy[\mathbf{Mary}(x) \ \& \ \mathbf{dog}(y) \ \& \ \mathbf{owns}(x, y)]$$

As Kamp and Reyle point out, proper names generally correspond to logical constants rather than predicates in logical representations. If **Mary** is not a predicate but an individual constant, then **Mary**(x) is not an admissible formula in predicate logic. Therefore the DRS condition **Mary**(x) must be translated $x = \mathbf{Mary}$, and the translation of (16) is:

$$(19) \quad \exists xy[x = \mathbf{Mary} \ \& \ \mathbf{dog}(y) \ \& \ \mathbf{owns}(x, y)]$$

In addition to complicating the translation to predicate logic, treating **Mary** as a predicate signifying ‘being named Mary’ leads to difficulties of interpretation. Muskens summarizes the problem, and Kamp and Reyle’s solution, thus (p. 9):

In a situation where two girls a and b are both called ‘Zebedea’ we cannot use ‘Zebedea loves a stockbroker’ to express that either a or b loves a stockbroker; the sentence can only be used with unique reference to some Zebedea. For this reason Kamp and Reyle propose to adopt the device of *external anchoring*. An external anchor is a finite function from discourse referents to the objects they are meant to denote and in Kamp and Reyle’s proposal these anchors may appear in the discourse representation.

As Muskens points out [13], allowing proper names to be translated with constants eliminates the need for this external anchoring device. From a generation perspective, this design choice is quite natural; it would be a waste to convert constants in the input formula to variables before listing them among the discourse referents.

Discourse referents are also associated with *index features*: person, number and gender [51, 52]. Index features are computed on the basis of morphosyntactic information if it is available, or encyclopedic knowledge in the Cyc Knowledge Base (KB) otherwise. For example, the individual **Mary** is asserted to be a human female in the KB, so corresponding discourse referents will have a feminine gender feature. These index features determine the form that pronouns take.⁵

Side effects. We recursively define a generation function $G(\alpha)$, where α can be any expression of the logic, which depends not only on α , but also on a global discourse D and a global operator context O . We subdivide the definition of $G(\alpha)$ into cases based on the logical expression type of α . A fundamental case is when α is an *atomic formula*, as in (20).

(20) **loves**(**Mary**, **Mary**)

The most appropriate method for atomic formulas uses the *reverse lexical entry* for the predicate – a mapping from a logical predicate to a partial specification of a sentence in natural language. For example, the reverse lexical entry for the predicate **loves** specifies that the main verb should be a form of the word *love*, the subject should be the translation of the first argument, and the object should be the translation of the second argument.⁶ The reverse lexical entry for **loves** specifies a template from which a syntax tree is built. A syntax tree is like an HPSG sign [51], with “phonological”, semantic, and syntactic information, including daughters for phrases. In our case, we are computing textual output, the value of the so-called PHON feature is an orthographic string.

A simplified rendition of the phrase that is ultimately generated for (20) is in Fig. 4.1. The tree is traversed left to right, depth first, and may be expanded during the traversal. Each time a node is visited, the value of the PHON feature is computed. We say that a subexpression of the logical formula has been *realized* if the PHON value of the phrase it corresponds to has been set. The PHON value of the mother is the concatenation of the PHON values of the daughters. The value of G is the PHON value of the top-level phrase.

What makes the semantics dynamic is that the computation of G can have *side effects* in the form of updates to D and O . The discourse context D is updated when a discourse referent is realized, as part of the algorithm for generating a constant or a variable. If α is an individual-denoting constant, and is

⁵ Incidentally, these features also determine subject-verb agreement, as pronoun and subject-verb agreement generally coincide, modulo some complications [52].

⁶ From an efficiency perspective, the use of reverse lexical entries amounts to offline computation of chains in semantic head-driven generation [26, 27]. We do not make use of a reversible grammar, however.

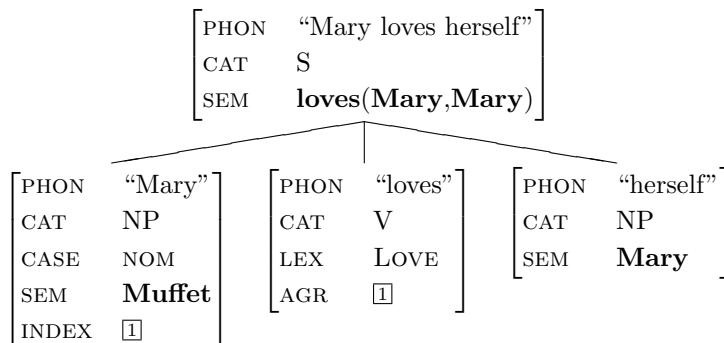


Fig. 1. Phrase generated for the input [loves(Mary, Mary)] (simplified)

not listed among the discourse referents, then $G(\alpha)$ returns a proper name, *and the discourse context is updated*. For example, the expression **Mary** in (20) is added to D after its first instance in the input formula is realized.

If α is listed in D , then an anaphoric expression is used. The algorithm for generating an anaphor is: If a pronoun would be ambiguous, then use a definite description or name; otherwise, use a pronoun. There is a great deal of sophistication and subtlety this treatment could acquire [2–4, 6–9], but this is not our focus here. A (reflexive) pronoun is appropriate for the second instance of **Mary** in (20), so the formula is rendered, “Mary likes herself.”

Like other discourse referents, variables are added to D after being mentioned, so as to be made eligible for subsequent rendering with anaphora. Thus, after *everything* is generated in (21), the variable x is added to the set of discourse referents, so it can be realized with a pronoun on its second mention:

- (21) Everything likes itself.
 $\forall x[\mathbf{likes}(x, x)]$

Lifespan limitations. Constants added to D during the execution of an instance of G may remain in D . However, it is necessary to remove variables from D when their ‘life’ has ended. Thus, if the argument to G is a quantificational sentence binding a variable ξ , then the following line must be executed before G exits:

$$D \leftarrow D - R(\xi)$$

where $R(\xi)$ stands for the discourse referent associated with ξ . Suppose the input formula were:

- (22) $\forall x\neg[\mathbf{loves}(\mathbf{Doug}, x)] \wedge \forall x[\mathbf{loves}(\mathbf{Mary}, x)]$

This would be rendered correctly as: “Doug loves nothing and Mary loves everything” rather than “Doug loves nothing and Mary loves it.” The latter would be generated if x were not removed from the discourse context.

Removal of variables from D is the key to accounting for the limited lifespan of the discourse referent introduced by *a cat* in (3) (“Everybody found a cat and kept it. It ran away.”). For the sake of discussion, let us treat **found** and **kept** as binary predicates relating the subject and the object, rather than invoking Davidsonian event representations [53], as we normally would in Cyc. Then a possible representation for the first sentence would be:

$$(23) \quad \forall x \exists y [\mathbf{found}(x, y) \wedge \mathbf{kept}(x, y)]$$

At the end of the sentence, both x and y are removed from the discourse context, making it impossible for there to be subsequent anaphoric references to them.⁷ If a pronoun is generated in the following sentence, it will have to be associated with some other discourse referent.

4.2 Operator context

$G(\alpha)$ depends not only on a discourse context D , but also on an operator context O . As we describe further below, logical operators and negations are stripped away from the input formula, leaving a ‘clausal skeleton’, and the information stripped away is stored in the operator context.

Definition. We define an *operator context* O as a tuple $\langle V, S, n \rangle$ where V is a set of variable type entries, S is a stack of logical symbols, and n is an integer representing the number of negations remaining to be expressed.

A variable type entry $v \in V$ is a tuple $\langle \alpha, \theta, \tau \rangle$, where α is a variable over individuals, θ is a quantifier symbol, and τ is a type. Types are Cyc *collections*, such as **Donkey**, the collection of all donkeys. Cyc collections are like sets except that they are (in principle) intensionally defined and constitute natural classes [54].⁸ Associating a type with a variable is conceptually similar to identifying the type as the ‘restriction’ of the quantifier, even though formally, quantifiers in first order logic do not specify restrictions, unlike generalized quantifiers.

The stack of symbols S contains the variables and logical operators with scope over the current position in the formula. We will represent stacks as tuples $\langle \alpha_1 \dots \alpha_n \rangle$, where α_n is at the top of the stack. The wider the scope of the operator, the deeper on the stack it is. For example, when generating the formula $\forall x \neg [\mathbf{loves}(\mathbf{Mary}, x)]$, at the point when **Mary** is being generated, $S = \langle x, \neg \rangle$.

⁷ Some of Karttunen’s observations in [1] suggest that discourse referents corresponding to existentials with maximally wide scope should not be removed from the discourse context. For example, consider the sentence “I have a proof of this theorem but it won’t fit in this margin.” One is tempted to analyze the first sentence using an existential quantifier; but then how can there be subsequent anaphoric reference to it? In our framework, this would be handled using Skolemization.

⁸ The formula $\mathbf{isa}(x, \mathbf{Donkey})$ can be converted to a corresponding unary predication $\mathbf{donkey}(x)$ without any change in meaning; collections are simply more convenient than unary predicates for the purposes of constructing concept hierarchies and expressing argument type constraints on predicates, as we do in Cyc.

We use the variable x rather than the quantifier \forall to indicate the scope of the associated universal quantifier, because the variable uniquely identifies the quantifier in question, while there may be many universal quantifiers in a formula in principle. The quantifier associated with the variable is computable from the variable's type entry in V .

Finally, n is used to keep track of unexpressed negations. As we will see in the next section, negations can be removed from formulas over the course of constructing the clausal skeleton, and this counter helps to ensure that every negation is expressed exactly once. (Note that the value of n is not derivable from S , as any negation on S may be either expressed or unexpressed.)

Clausal skeletons. In the generation of some formulas, parts of the input formula are stripped away, leaving a *clausal skeleton*. For example, the clausal skeleton of (24) is (25).

$$(24) \quad \forall x \forall y [[\mathbf{isa}(x, \mathbf{Man}) \wedge \mathbf{isa}(y, \mathbf{Donkey}) \wedge \mathbf{owns}(x, y)] \rightarrow \mathbf{loves}(x, y)]$$

$$(25) \quad \mathbf{owns}(x, y) \rightarrow \mathbf{loves}(x, y)$$

The two **isa** statements in the antecedent of the conditional in (24) are *variable typing clauses*. The binary predicate **isa** relates an individual to a collection, and signifies that the individual is an instance of the collection. Variable typing clauses are removed, along with the universal quantifiers (as mentioned in [55]).

For any input formula of the form $\forall \xi \alpha$, where ξ is a variable and α is a sentence, a simplified version of the algorithm for constructing clausal skeletons σ and new variable type entries v is as follows (where **Thing** is the most general collection, and \sim stands for “is of the form” or “matches”):

$$(26) \quad \begin{array}{l} - \text{ If } \alpha \sim [\psi \rightarrow \phi]: \\ \quad \bullet \text{ If } \psi \sim [\mathbf{isa}(\xi, \gamma)]: \\ \quad \quad \sigma = \phi \\ \quad \quad v = \langle \xi, \forall, \gamma \rangle \\ \quad \bullet \text{ Else if } \psi \sim [\delta_1 \wedge \dots \wedge \delta_n] \text{ where } \delta_i \sim [\mathbf{isa}(\xi, \gamma)]: \\ \quad \quad \sigma = [\delta_1 \wedge \dots \wedge \delta_{i-1} \wedge \delta_{i+1} \wedge \dots \wedge \delta_n] \rightarrow \phi \\ \quad \quad v = \langle \xi, \forall, \gamma \rangle \\ \quad \bullet \text{ Else:} \\ \quad \quad \sigma = \alpha \\ \quad \quad v = \langle \xi, \forall, \mathbf{Thing} \rangle \\ - \text{ Else:} \\ \quad \sigma = \alpha \\ \quad v = \langle \xi, \forall, \mathbf{Thing} \rangle \end{array}$$

For a formula of the form $\exists \xi \alpha$, there are two cases:

$$(27) \quad \begin{array}{l} - \text{ If } \alpha \sim [\delta_1 \wedge \dots \wedge \delta_n] \text{ where } \delta_i \sim [\mathbf{isa}(\xi, \gamma)]: \\ \quad \sigma = [\delta_1 \wedge \dots \wedge \delta_{i-1} \wedge \delta_{i+1} \wedge \dots \wedge \delta_n] \\ \quad v = \langle \xi, \exists, \gamma \rangle \end{array}$$

- Else:
 - $\sigma = \alpha$
 - $v = \langle \xi, \exists, \mathbf{Thing} \rangle$

The variable type entry v is added to the set V of variable type entries in the operator context O . After the operator context is updated, $G(\sigma)$ is computed; in other words, the clausal skeleton is realized. Then, importantly, the operator context is restored to its previous state. The information stored in the operator context surfaces when the variable is expressed, as described below.

The clausal skeleton is isomorphic, clause for clause, to the resulting English sentence. The realization of (24) has two clauses, just as its clausal skeleton (25) has:

- (28) If a farmer owns a donkey, then he loves it.

When the antecedent of the input formula consists entirely of a variable typing clause, all that remains in the clausal skeleton is the consequent. An input formula such as (29) will be realized as in (30), a single clause.

- (29) $\mathbf{isa}(x, \mathbf{Man}) \rightarrow \mathbf{loves}(x, \mathbf{Mary})$

- (30) Every man loves Mary.

Thus, all of the content that is expressed below the clause level (in quantified noun phrases) comes from the operator context, and every clause in the English translation is part of the clausal skeleton.⁹

Negation stripping. When α is of the form $\neg\phi$, the clausal skeleton of α is ϕ . No variable type entries are produced in this case, of course, but the counter representing the number of unexpressed negations, n , is incremented. Stripping the negation in this way makes it possible for negations to be expressed sub-clausally, using a member of the *no*-series (*nobody*, *nothing*, etc.). As we discuss in §4.3, we use clausal negation as a back-up strategy when expressing negation sub-clausally fails; if $n > 0$ after ϕ is realized, either the verb is negated or negation is expressed periphrastically with, for example, “It is false that...”.

Updating the operator stack. The main purpose of the operator stack is to determine when NPIs are licensed. The generation procedure $G(\alpha)$ updates the operator stack whenever α is a formula whose operator is in the set $\{\forall, \exists, \neg, \rightarrow\}$.

⁹ If the variable typing clauses were not removed, the variables would be registered with type **Thing** and the output would contain more clauses: “If something is a farmer and some other thing is a donkey then the thing loves the other thing.” If the variables were not registered in the operator context at all, then the universal quantifier would not be stripped from the formula, and the output would be as follows, with explicit variables: “For every x , for every y , if x is a farmer and y is a donkey and x owns y , then x loves y .”

If α is a quantificational formula such as $\forall x\phi$, then the variable (x) is pushed onto the operator stack, and popped off of it at the completion of $G(\alpha)$. If α is a negative formula, then a \neg symbol is pushed onto the operator stack and popped off of it again at the completion of $G(\alpha)$. If α is an implication, then the symbol \rightarrow is pushed onto the stack and the stack is popped at the completion of the generation of the *antecedent*, because implications only license NPis in the antecedent.

Determiner selection algorithm. The definition of $G(\alpha)$ where α is a variable (*variable realization*) involves the operator context as well as the discourse context. Variables can be realized as pronouns, definite descriptions, or with *every*, *any*, *some*, or *no*, depending on the operator and discourse contexts. As mentioned in §4.1, variables are realized as pronouns when they are listed in D and a pronoun would not be ambiguous. If a pronoun is not appropriate for realizing a variable, a lexical noun phrase containing a determiner and a noun is used. The noun is the realization of the variable’s type, the τ such that $\langle \alpha, \theta, \tau \rangle \in V$. τ represents a collection, e.g. **Man** (the collection of all men), and can be realized as, for example, *man*. There are several types of determiner that may accompany the noun: DEFINITE (*the*), NO (*no*), UNIVERSAL (*every*), INDEFINITE (*some*), NPI (*any*).¹⁰

The first step in the algorithm for choosing a determiner type is to compute whether or not a variable could be expressed as an NPI. Both universally and existentially bound variables can be realized as NPis but the two quantifier types differ with respect to the scope that they must have relative to an NPI licenser. In order to qualify for being realized as an NPI, an instance of a variable must be bound by either a universally quantified variable outscoping an NPI licenser, or an existentially bound quantifier outscoped by one. For example, (7), repeated below is rendered as in (32).

(31) $\forall x[\neg\text{loves}(\text{Mary}, x)]$

(32) Mary doesn’t love anything.

In contrast, when the universal quantifier is inside the scope of negation, the output should be *Mary doesn’t love everything*.

It is sometimes assumed that *any* NPs correspond to existentially bound variables in the scope of an NPI licenser (e.g. [56]). Since negation of an existential is equivalent to universal quantification over a negation, one might imagine an approach in which universals outscoping negations are “canonicalized” to negatives of existentials, to maintain the generalization that *any* corresponds to an existential quantifier (cf. [16] regarding the problem of logical form equivalence). So (31) would be canonicalized to (33):

¹⁰ Another determiner type is WH (*what*, *which*). *Wh*-determiners are used for unbound variables in formulas to be generated with interrogative force. We ignore questions here for the sake of simplicity.

$$(33) \quad \neg \exists x[\mathbf{loves}(\mathbf{Mary}, x)]$$

There are at least two problems with this approach. First, it would not work with implications. NPIs are licensed in the antecedent of a conditional, and there is no way of “canonicalizing” a universal quantification over an implication to a corresponding formula with the existential quantifier in the scope of the implication operator, because the variable bound by the universal quantifier may appear on both sides of the implication. Secondly, this would not deal well with multiple universal quantifiers. When multiple universal quantifiers outscope negation as in (34), both may correspond to an *any* phrase as in (35).

$$(34) \quad \forall x \forall y[\neg \mathbf{loves}(x, y)]$$

(35) It is not the case that anyone loves anyone (else).

Only one of the quantifiers in (34) can be “swapped” with a negation in order to produce an existential quantifier, yet there are two *any* phrases. One of them must correspond to a universal quantifier.

With this in mind, let us summarize the algorithm for determining whether or not an NPI is licensed as follows: First, look up the quantifier of the variable in question; then if the quantifier is \forall , the question is whether the variable is deeper on the stack than an NPI licenser; if the quantifier is \exists , the question is whether the NPI licenser is deeper than it. Call the NPI licenser π ; if the variable has no NPI licenser, then π is null.¹¹ Given π , a variable type entry $\langle \alpha, \theta, \tau \rangle$ for variable ξ , and an unexpressed negation counter n , the determiner is chosen via the following algorithm:

- (36) – If $R(\chi) \in D$, return DEFINITE.
 – If π is non-null:
 • If $\pi = \neg$ and $n > 0$, then return NO and decrement n by one.
 • Otherwise,
 * If this will be the only realization of the variable, return NPI.¹²
 * Otherwise, return INDEFINITE.
 – If $\theta = \forall$, return UNIVERSAL.
 – Otherwise, return INDEFINITE.

Now consider example (24) again, repeated here as (37).

$$(37) \quad \forall x \forall y[[\mathbf{isa}(x, \mathbf{Man}) \wedge \mathbf{isa}(y, \mathbf{Donkey}) \wedge \mathbf{owns}(x, y)] \rightarrow \mathbf{loves}(x, y)]$$

¹¹ David Beaver (p.c.) points out that an alternative algorithm would involve a theorem prover to determine whether or not the variable is in downward-entailing environment. This would be interesting and feasible for us to try, because the Cyc paraphrase system has full access to the Cyc inference engine. The more “syntactic” algorithm we present here is almost certainly more computationally efficient, however, as inference can be quite expensive.

¹² This element of the algorithm is purely stylistic.

Right before the first realization of x , $\langle x, \mathbf{Man}, \forall \rangle \in V$ (i.e. x is universally quantified, and has type **Man**), $S = \langle x, \rightarrow \rangle$ (i.e., the current expression is inside the antecedent of a conditional), $n = 0$ (there are no unexpressed negations), and x is not in D (so x has not previously been realized). Therefore $G(x)$ will be realized with as either *any man* or with an indefinite (*a* or *some*). After x is realized, it will be in D , so it will be realized as a masculine pronoun or as *the man*. If instead, S were merely $\langle x \rangle$ (so the current expression is not in the scope of an NPI licenser), then π would be null, and the appropriate determiner type would be **EVERY**.

Another case where NPI-type determiners are chosen is in the scope of negation, when there is no negation to be expressed, i.e., when $\pi = \neg$ and $n = 0$. Such a situation arises when negation is expressed on the verb, as in (9), *Mary doesn't love anything*. When $n > 0$ (there are unexpressed negations) and $\pi = \neg$ (the NPI licenser for the variable is negation), the variable can be used to express negation, as in (8), *Mary loves nothing*.

We are now in a position return to the contrast between indefinite determiners and *every* with respect to the lifespan of the discourse referents they introduce. Recall example (4), repeated here:

(38) If a_i farmer owns a_j /**every* $_j$ donkey, then he_i loves it $_j$.

We have just seen how the acceptable variant of (38), with an indefinite determiner in the antecedent, is generated for an input like (24). The only way for *every* to be generated in the antecedent of a conditional is for the universally quantified variable to be outscoped by \rightarrow , as in the following example:

(39) $\forall x[\mathbf{isa}(x, \mathbf{Farmer}) \wedge \forall y[\mathbf{isa}(x, \mathbf{Donkey}) \rightarrow \mathbf{owns}(x, y)]] \rightarrow \mathbf{loves}(x, \mathbf{Mary})$

The formula in (39) would be rendered as follows:

(40) If a farmer owns every donkey, then he loves Mary.

When it is time to realize the variable y for the first time, $S = \langle x, \rightarrow, y \rangle$, so the computation of π for y will yield a null value. The determiner selection algorithm will therefore correctly choose *every*. But in this case, the scope of the quantifier will have ended by the time the consequent of the conditional is reached. Thus, discourse referents introduced by *every* in the protasis of a conditional will never extend to the apodosis.

4.3 Two potential problems and their solutions

There are (at least) two dangers that the system we have described may appear vulnerable to, but they can be avoided, as we will show in this section. In both cases, the solution has to do with a preference to generate one type of output over another, an idea reminiscent of Optimality Theory [45].

Spurious negative polarity items? A possible objection to the algorithm given in (36) is that it risks generating (41) for (34).

(41) *Anyone doesn't love anyone.

Suppose that verbal negation is generated before the subject is generated, so that there are no remaining negations to express by the time it is time to generate the subject. Then n will be equal to zero, and π will be equal to \neg , so the determiner selection algorithm will return NPI.

The ungrammaticality of (41) suggests that NPI licensing might be at least partly syntactic (see [56] p. 178, and references cited there). The relevant constraint would seem to be that an NPI such as *anyone* must be in the syntactic scope (e.g. *c*-command domain) of an NPI licensor as well as its semantic scope (i.e., it must be in the *direct scope* of the licensor [57, 56]).

However, there are cases in which NPIs appear outside the direct scope of their licensor [58]:

(42) A doctor who knew anything about acupuncture was not available.

Here, the NPI *anything* is outside the syntactic scope of the licensor (*not*). This casts doubt on the 'direct scope' approach.

Furthermore, an output filter – that NPIs must be in the direct scope of their licensor – is less useful for the purposes of generation than an algorithm for choosing the most appropriate determiner.

An alternative possible explanation for the ungrammaticality of (41) is that English requires negation to be expressed as early in the sentence as possible. It is well-known that negative quantifiers such as *nobody* express negation; evidence comes from the fact that they give rise to multiple negation readings in the context of other negations (in standard English), unlike *any* [59]:

(43) I didn't see nobody. [multiple negation reading]

(44) I didn't see anybody. [single negation reading]

Further evidence that *nobody*, etc., expresses inherent negation, in contrast to *anybody*, etc., is that *nobody* occur in a sentence without the presence of another negation marker, unlike *anybody*, as in *I saw nobody* [59].

If English requires negation to be expressed as early in the sentence as possible, then a variable that can be realized with *no* will have to be. This effectively rules out (41). Furthermore, it leaves open the possibility of generating (42) – it would be possible to define “as early as possible” in such a way as to rule out expressing negation any earlier than the matrix verb in this configuration, since it would not be possible to express the same meaning by replacing *anything* with *nothing* and removing the negation from the verb; the negation cannot take scope outside the relative clause.

Spurious positive polarity items? Another possible objection to the algorithm given in (36) is that it risks generating *some*, a positive polarity item, in the scope of negation, as in:

(45) Mary doesn't love someone.

While this sentence is grammatical, it is not equivalent to *Mary doesn't love anyone*, and should not be used to express that idea. Doing so is a risk because we treat *some* as an NPI-type determiner. However, in the system as implemented, we generate *Mary loves noone* rather than *Mary doesn't love anyone*, because negation is always expressed subclausally (e.g. with *no*) if possible. Therefore, this case does not arise (although perhaps, admittedly, it should).

There are cases in which our system generates a *some* phrase in the scope of negation, such as the following:

(46) It is never the case that Mary loves someone.

In contrast to (45), (46) is equivalent to the sentence obtained by replacing *some* with *any* – as Ladusaw pointed out [60], *some* can scope under extraclausal negation. Therefore the *some* under negation in (46) is innocuous.

5 Summary and outlook

In summary, we present an algorithm for translating predicate logic to English that uses dynamically updated information states. Our solution relies on two components of context: the discourse context and the operator context. The discourse context is used for referents that have already been mentioned, and the operator context stores information stripped away from the clausal skeleton of the input formula, and determines how variables are expressed. The result produces appropriate quantificational and anaphoric expressions, and enforces lifespan limitations on discourse referents.

The system we have presented here is specific to English. In the future, we would like to identify the parameters that would have to be changed to generate correct outputs for languages other than English. Negative concord [61, 59] presents a particularly interesting example. We believe that negative concord might be modelled effectively by eliminating the *n* component of the operator context, which keeps track of unexpressed negations. In addition, we would like to expand the algorithm to generate a wider range of positive and negative polarity items [62], and do so under the appropriate circumstances. We believe that the natural language generation perspective – given a semantic formula, producing a natural language translation – may shed new light on these phenomena and on the associated theoretical issues in semantics.

References

1. Karttunen, L.: Discourse referents. In McCawley, J.D., ed.: *Syntax and Semantics 7: Notes from the Linguistic Underground*. Academic Press, New York (1976) 363–385

2. Chafe, W.L.: Givenness, contrastiveness, definiteness, subjects, topics and point of view. In Li, C.N., ed.: *Subject and topic*, New York, Academic Press (1976) 25–55
3. Ariel, M.: *Accessing NP antecedents*. Routledge, London (1990)
4. Gundel, J.K., Hedberg, N., Zacharski, R.: Cognitive status and the form of referring expressions in discourse. *Language* **69** (1993) 274–307
5. Kamp, H., Reyle, U.: *From Discourse to Logic*. Kluwer Academic Publishers, Dordrecht (1993)
6. Brennan, S.: Centering attention in discourse. *Language and Cognitive Processes* **10** (1995) 137–167
7. Grosz, B.J., Joshi, A.K., Weinstein, S.: Providing a unified account of definite noun phrases in discourse. In: *Proceedings of the 21st Annual Meeting of the Association for Computational Linguistics*, Cambridge, MA (1983) 44–49
8. Grosz, B.J., Joshi, A.K., Weinstein, S.: Centering: A framework for modeling the local coherence of discourse. *Computational Linguistics* **21** (1995) 203–226
9. Beaver, D.L.: The optimization of discourse anaphora. *Linguistics and Philosophy* **27** (2004) 3–56
10. Heim, I.: *The Semantics of Definite and Indefinite Noun Phrases*. PhD thesis, MIT (1982)
11. Heim, I.: File change semantics and the familiarity theory of definiteness. In Baurle, R., Schwarze, C., Von Stechow, A., eds.: *Meaning, Use, and the Interpretation of Language*. Walter de Gruyter, Berlin (1983) 164–189
12. Groenendijk, J., Stokhof, M.: Dynamic predicate logic. *Linguistics and Philosophy* **14** (1991) 39–100
13. Muskens, R.: Combining Montague semantics and discourse representation. *Linguistics and Philosophy* **19** (1996) 143–186
14. Groenendijk, J., Stokhof, M., Veltman, F.: *Coreference and modality* (1996)
15. Montague, R.: English as a formal language. In Thomason, R.H., ed.: *Formal Philosophy: Selected Papers of Richard Montague*. Yale University Press, New Haven (1974) 188–221
16. Shieber, S.M.: The problem of logical-form equivalence. *Computational Linguistics* **19** (1993) 179–190
17. Poesio, M., Zucchi, A.: On telescoping. In: *Proceedings of SALT II*. (1992) 347–366
18. Giannakidou, A.: Polarity Sensitivity as (Non)veridical Dependency. John Benjamins, Amsterdam (1998)
19. Giannakidou, A.: Licensing and sensitivity in polarity items: from downward entailment to (non)veridicality. In Andronis, M., Pycha, A., Yoshimura, K., eds.: *CLS 38: Papers from the 38th Annual Meeting of the Chicago Linguistic Society, Parasession on Polarity and Negation*, Chicago Linguistic Society (2002)
20. Thompson, H.: Strategy and tactics: A model for language production. In: *Papers from the 13th Regional Meeting of the Chicago Linguistic Society*, Chicago, IL (1977)
21. Calder, J., Reape, M., Zeevat, H.: An algorithm for generation in unification categorial grammar. In: *Proceedings of the 4th Conference of the European Chapter of the Association for Computational Linguistics*, Manchester, UK (1989) 233–240
22. Wedekind, J., Kaplan, R.M.: Ambiguity-preserving generation with LFG- and PATR-style grammars. *Computational Linguistics* **22** (1996) 555–558
23. Wedekind, J.: Semantic-driven generation with LFG- and PATR-style grammars. *Computational Linguistics* **25** (1999) 277–281
24. Wilcock, G., Matsumoto, Y.: Head-driven generation with HPSG. In: *Proceedings of COLING-ACL '98: Workshop on Usage of WordNet in Natural Language Processing Systems*. (1998) 1393–1397

25. Carroll, J., Flickinger, D., Copestake, A., Poznanski, V.: An efficient chart generator for (semi-)lexicalist grammars. In: Proceedings of the 7th European Workshop on Natural Language Generation, Toulouse, France (1990)
26. Becker, T.: Fully lexicalized head-driven syntactic generation. In: Proceedings of the Ninth International Workshop on Natural Language Generation, Association for Computational Linguistics (1998) 208–217
27. Shieber, S.M.: Semantic head-driven generation. *Computational Linguistics* **16** (1990) 30–42
28. Van Noord, G.: An overview of head-driven bottom-up generation. In Dale, R., Mellish, C., Zock, M., eds.: *Current Research in Natural Language Generation*. Academic Press (1994) 141–165
29. Kikui, G.: Feature structure based semantic head driven generation. In: Proceedings of COLING-92, Nantes (1992) 32–38
30. Tuells, T., Rigau, G., Rodriguez, H. In: *Offline Compilation of Chains for Head-Driven Generation with Constraint-Based Grammars*. Springer, Berlin (2008) 267–286
31. Horacek, H.: An algorithm for generating referential descriptions with flexible interfaces. In: Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics. (1988) 206–213
32. Dale, R.: Cooking up referring expressions. In: Proceedings of the 27th Annual Meeting of the Association for Computational Linguistics. (1989)
33. Reiter, E., Dale, R.: A fast algorithm for the generation of referring expressions. In: Proceedings of the 14th International Conference on Computational Linguistics, Nantes (1992) 232–238
34. Dale, R., Reiter, E.: Computational interpretations of the Gricean maxims in the generation of referring expressions. *Cognitive Science* **19** (1994) 233–263
35. Copestake, A., Flickinger, D., Malouf, R., Riehemann, S., Sag, I.: Translation using minimal recursion semantics. In: Proceedings of the Sixth International Conference on Theoretical and Methodological Issues in Machine Translation, Leuven, Belgium (1995)
36. Shaw, J., McKeown, K.: Generating referring quantified expressions. In: Proceedings of the first international conference on natural language generation, Mitzpe Ramon, Israel (2000) 100–107
37. Krahmer, E., Van Erk, S., Verleg, A.: Graph-based generation of referring expressions. *Computational Linguistics* (2003)
38. Van Deemter, K.: Generating referring expressions: Boolean extensions of the incremental algorithm. *Computational Linguistics* **28** (2002) 37–52
39. Siddharthan, A., Copestake, A.: Generating referring expressions in open domains. In: Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics, Barcelona, Spain (2004) 408–415
40. Varges, S., Deemter, K.V.: Generating referring expressions containing quantifiers. In: Proceedings of the 6th International Workshop on Computational Semantics. (2005) 1–13
41. Kelleher, J.D., Kruijff, G.J.M.: Incremental generation of spatial referring expressions in situated dialog. In: Proceedings of COLING/ACL-06. (2006)
42. Paraboni, I., Van Deemter, K., Masthoff, J.: Generating referring expressions: Making referents easy to identify. *Computational Linguistics* **33** (2007) 229–254
43. Areces, C., Koller, A., Striegnitz, K.: Referring expressions as formulas of description logic. In White, M., Nakatsu, C., McDonald, D., eds.: Proceedings of the Fifth International Natural Language Generation Conference, Salt Fork, Ohio, Association for Computational Linguistics (2008) 42–49

44. Minnen, G., Copestake, A.: Memory-based learning for article generation. In: Proceedings of the Fourth Workshop on Computational Natural Language Learning, Lisbon, Portugal (2000)
45. Prince, A., Smolensky, P.: Optimality Theory: constraint interaction in generative grammar: Technical report 2. Technical report, Rutgers University Center for Cognitive Science (1993)
46. Muskens, R.: Tense and the logic of change. In Egli, U., Pause, P.E., Schwarze, C., Von Stechow, A., Wienold, G., eds.: *Lexical Knowledge in the Organization of Language*, Amsterdam, Benjamins (1995) 147–183
47. Groenendijk, J., Stokhof, M.: Dynamic Montague grammar. In: Proceedings of the Second Symposium on Logic and Language, Budapest (1990) 3–48
48. Van Leusen, N., Muskens, R.: Construction by description in discourse representation. In Peregrin, J., ed.: *Meaning: The Dynamic Turn*, Oxford, Elsevier (2003) 33–65
49. Lenat, D.: Cyc: A large-scale investment in knowledge infrastructure. *Communications of the ACM* **38** (1995)
50. Ramachandran, D., Reagan, P., Goolsbey, K.: First-orderized ResearchCyc: Expressivity and efficiency in a common-sense ontology. In: Papers from the AAAI Workshop on Contexts and Ontologies: Theory, Practice and Applications, Pittsburgh, PA (2005)
51. Pollard, C., Sag, I.A.: *Head-Driven Phrase Structure Grammar*. University of Chicago Press, Chicago (1994)
52. Wechsler, S., Zlatić, L.: *The Many Faces of Agreement*. Center for the Study of Language and Information, Stanford (2003)
53. Davidson, D.: The logical form of action sentences. In Rescher, N., ed.: *The Logic of Decision and Action*. University of Pittsburgh Press, Pittsburgh (1967) 81–95
54. Lenat, D.B., Guha, R.V.: *Building Large Knowledge-Based Systems*. Addison-Wesley, Reading, MA (1990)
55. Baxter, D., Shepard, B., Siegel, N., Gottesman, B., Schneider, D.: Interactive natural language explanations of cyc inferences. In: Proceedings of AAAI 2005: International Symposium on Explanation-aware Computing, Washington, D.C. (2005)
56. De Swart, H.: Licensing of negative polarity items under inverse scope. *Lingua* **105** (1998) 175–200
57. Klima, E.: Negation in English. In Fodor, J.A., Katz, B., eds.: *The Structure of Language*. Prentice-Hall, Englewood Cliffs, N.J. (1964) 246–323
58. Linebarger, M.C.: *The grammar of negative polarity*. PhD thesis, MIT (1981)
59. Giannakidou, A.: Negative ... concord? *Natural Language and Linguistic Theory* **18** (2000) 457–523
60. Ladusaw, W.A.: *Polarity Sensitivity as Inherent Scope Relations*. Garland, New York (1980)
61. Laka Mugarza, M.I.: *Negation in Syntax: On the Nature of Functional Categories and Projections*. PhD thesis, MIT (1990)
62. Szabolcsi, A.: Positive polarity – negative polarity. *Natural Language and Linguistic Theory* **22** (2004) 409–452