A Two-Variable Fragment of English

Ian Pratt-Hartmann
Department of Computer Science,
University of Manchester,
Manchester M13 9PL
DRAFT: comments welcome

Abstract

Controlled languages are regimented fragments of natural language designed to make the processing of natural language, by both humans and computers, more efficient and reliable. This paper defines a controlled language, E2V, whose principal grammatical resources include determiners, relative clauses, reflexives and pronouns. We provide a formal syntax and semantics for E2V, in which anaphoric ambiguities are resolved in a linguistically natural way. We show that the expressive power of E2V is equal to that of the two-variable fragment of first-order logic. It follows that the problem of determining the satisfiability of a set of E2V sentences is NEXPTIME complete. We show that E2V can be extended in various ways without compromising these complexity results; however, relaxing our policy on anaphora resolution renders the satisfiability problem for E2V undecidable. Finally, we argue that our results have a bearing on the broader philosophical issue of the relationship between natural and formal languages.

1 Introduction

Controlled languages are regimented fragments of natural language designed to make the processing of natural language, by both humans and computers, more efficient and reliable. Although work on controlled languages was originally motivated by the need to produce uniform and easily translatable technical documentation, attention has recently turned to their possible application to system requirements specifications (Fantechi *et al.* [6], Fuchs, Schwertl and Torge [8], Fuchs, Schwertl and Schwitter [9], Holt [13], Holt and Klein [14], Macias and Pulman [19], Nelken and Francez [23], Vadera and Meziane [30]). This interest in natural specification languages is motivated by the fact that many design engineers and programmers find formal specification languages—usually, some variety of logic—alien and hard to understand. The hope is that, by selecting a regimented subset of natural language, the precision and clarity offered by formal specification languages can be combined with the familiarity and ease-of-understanding associated with natural language.

In recent years, a new factor has arisen with important implications for the use of controlled languages—the explosion of research into decidable fragments of logic. Known decidable fragments include various prefix classes (see Börger, Grädel and Gurevich [2] for a survey), the guarded fragment (Andréka, van Benthem and Németi [1], Grädel [10]) and, most importantly for the purposes of this paper, the two-variable fragment (Mortimer [21], Grädel and Otto [11]). The relevance of this research to controlled languages is clear: given a controlled language which is mapped to some logic, the question naturally arises as to whether that logic enjoys good computational characteristics; conversely, given a logic whose computational characteristics are well-understood, it would be useful to identify a controlled language which maps to it.

This paper provides a study in how to match a controlled language to a decidable logic with known computational characteristics. The controlled language in question, called E2V, is shown to correspond, in a sense made precise below, to the two-variable fragment of first-order logic. As far as we are aware, this is the first time a fragment of natural language has been so tightly matched to a decidable fragment of first-order logic. The grammar of E2V has deliberately been kept as simple as possible, in order to clarify the logical issues involved; thus, E2V is certainly not being proposed as a practically useful controlled language. However, the techniques developed in this paper easily carry over to various salient extensions of E2V; therefore, we claim, our results are of direct relevance to the development of practically useful controlled languages. In addition, we argue that these results have a bearing on the broader philosophical issue of the relationship between natural and formal languages.

The plan of the paper is as follows. Section 2 introduces the syntax and semantics of E2V; section 3 establishes upper bounds on its expressiveness, while section 4 establishes corresponding lower bounds. Section 5 discusses the broader philosophical significance of this work. The eclectic nature of the research reported here has obliged us to make relatively demanding assumptions about readers' command of logic, linguistics and, to a lesser extent, computing. With this fact in mind, references to textbooks have occasionally been provided.

2 The syntax and semantics of E2V

E2V is a fragment of English coinciding, in a sense to be made precise below, with the two-variable fragment of first-order logic. Its key grammatical resources include determiners, relative clauses, reflexives and pronouns. Examples of E2V sentences are:

(1) Every artist despises every beekeeper

(2) Every artist admires himself

(3) Every beekeeper whom an artist despises admires him

(4) Every artist who employs a carpenter despises every beekeeper who admires him.

The remainder of this section is devoted to a formal specification of the syntax and semantics of E2V. The syntax determines which strings of words count as E2V sentences, and the semantics determines how those sentences are to be interpreted, by mapping them to formulas of first-order logic. The relatively formal nature of the presentation facilitates the proofs of theorems in subsequent sections concerning the expressive power of E2V and the computational complexity of reasoning within it. Generally, the semantics of E2V is unsurprising, in that it interprets E2V sentences in accordance with English speakers' intuitions. In particular, sentences (1)–(4) are mapped to the respective formulas:

(5) $\forall x_1(\text{artist}(x_1) \rightarrow \forall x_2(\text{beekeeper}(x_2) \rightarrow \text{despise}(x_1, x_2)))$.

(6) $\forall x_1(\text{artist}(x_1) \rightarrow \text{admire}(x_1, x_1))$.

(7) $\forall x_1 \forall x_2((\text{beekeeper}(x_1) \wedge (\text{artist}(x_2) \wedge \text{despise}(x_2, x_1))) \rightarrow \text{admire}(x_1, x_2))$

(8) $$\begin{aligned}&\forall x_1((\text{artist}(x_1) \wedge \exists x_2(\text{carpenter}(x_2) \wedge \text{employ}(x_1, x_2))) \rightarrow \\ &\qquad \forall x_3((\text{beekeeper}(x_3) \wedge \text{admire}(x_3, x_1)) \rightarrow \text{despise}(x_1, x_3))).\end{aligned}$$

When reading E2V, some care must be taken to respect its particular conventions regarding scope ambiguities and pronoun resolution; generally, however, E2V sentences mean what you think they mean.

2.1 Syntax

The syntax of E2V has four components: a *grammar*, a *lexicon*, a movement rule and a set of *indexing rules*.

Grammar

The grammar of E2V consists of a set of definite clause grammar (DCG) rules, for example:

IP(A) $\rightarrow$ NP(B/A,I), VP(B,I)
VP(A,I) $\rightarrow$ V(B,I,J), NP(B/A,J),

with the labels IP, NP, etc. indicating categories of phrases in the usual way. The variable expressions A, B and B/A in these rules unify with *semantic values*, which represent the meanings of the phrases in question; and the variables I and J unify with *indices*, which regulate variable bindings in those meanings. Semantic values are explained in detail in section 2.2; for the present, however, think of a semantic value of the form B/A as a function which takes B as input and yields A as output. Thus, the first rule above states that the semantic value of an IP consisting of an NP and a VP is obtained by applying the semantic value of the NP to the semantic value of the VP.

The complete grammar for E2V is given in figure 1. We mention in passing that, in the presentation of E2V here, the issue of pronoun agreement in person,

CP(A,I) → NP(empty,I), C′(A)
C′(A,I) → C(B/A), IP(B)
IP(A) → NP(B/A,I), VP(B,I)
IP(A) → NP(B/A,I), NegP(B,I)
NegP(A,I) → Neg(B/A), VP(B,I)
VP(A,I) → V(B,I,J), NP(B/A,J)
NP(A,I) → RelPro(A,I)
NP(A,I) → Pronoun(A,I)
NP(A,I) → Reflexive(A,I)
NP(A,I) → Det(B/A), N′(B,I)
NP(empty,I) →
N′(A) → N(B,I), CP(B/A,I)
N′(A,I) → N(A,I)

Figure 1: The grammar for E2V

case and gender has been ignored. Such details are easily handled within the framework of DCGs, and need not be discussed further.

Lexicon

The lexicon of E2V also consists of a set of DCG rules, and is divided into two parts: the *closed-class* lexicon and the *open-class* lexicon. The closed-class lexicon gives the meanings of those words in our English fragment concerned with logical form, for example:

Det(A/(B/every(A,B))) → every
Reflexive(A/A,I) → itself (himself, herself)
C(A/(B/(rel(A,B)))) → .

The first of these rules assigns the semantic value A/(B/every(A,B)) to the determiner every. It helps to think of A/(B/every(A,B)) as a function mapping two semantic values, A and B, to the more complex semantic value every(A,B). The second rule assigns the semantic value A/A to any of the reflexive pronouns itself, himself and herself. This semantic value is in effect the identity function, reflecting the fact that the semantic force of a reflexive pronoun is exhausted by its effect on the pattern of indexing in the sentence (discussed below). The third rule assigns the semantic value A/(B/(rel(A,B))) to a (covert) complementizer. Again, it helps to think of A/(B/(rel(A,B))) as a function mapping the semantic values A and B to the more complex semantic value rel(A,B), indicating a relative clause. The complete closed-class lexicon for E2V is given in figure 2.

The open-class lexicon is an indefinitely large set of DCG rules for the terminal categories N and V. These rules determine the semantic values of words of these categories: unary predicates for nouns and binary-predicates for verbs. The open-class lexicon might contain the following entries:

Det(A/(B/every(A,B))) → every
Det(A/(B/every(A,not(B))) → no
Det(A/(B/some(A,B))) → a(n)
Det(A/(B/some(A,B))) → some
Reflexive(A/A,I) → itself (himself, herself)
Pronoun(A/A,I) → it (he, she, him, her)
RelPro(A/A,I) → which (who, whom)
C(A/(B/(rel(A,B)))) →
Neg(A/not(A)) → does not.

Figure 2: The closed-class lexicon for E2V

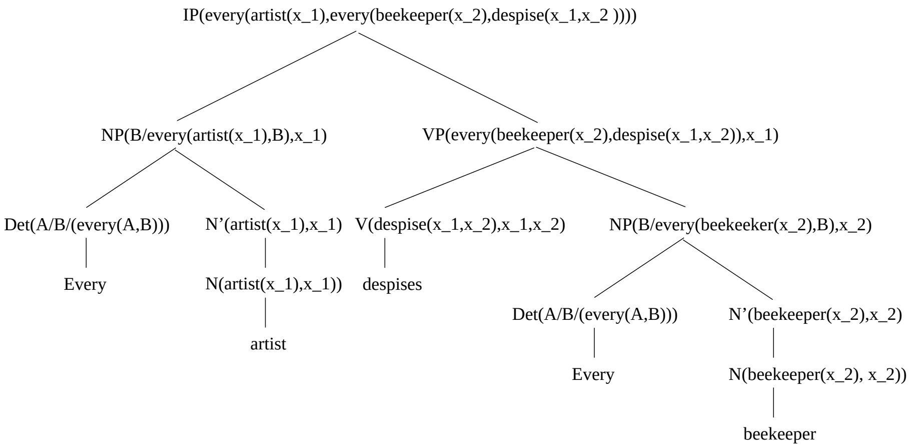

Figure 3: Phrase-structure of sentence (1).

N(artist(I),I) → artist
N(beekeeper(I),I) → beekeeper
...
V(admire(I,J),I,J) → admires
V(despise(I,J),I,J) → despises
....

Notice how, in open-class lexicon entries, the index variables appear as arguments in the semantic values.

Together, the grammar and lexicon generate sentences via successive expansion of nodes under unification of variables in the usual way. Figure 3 illustrates the parsing of sentence (1) using the DCG rules. (The values x_1, x_2 for the index variables are explained below.) The indeterminate nature of the open-class lexicon means that the English fragment we are describing is in reality a *family* of fragments—one for each choice of open-class lexicon. What these fragments have in common is just the overall syntax and the fixed stock of 'logical' words in the closed-class lexicon. This is exactly the situation encountered in logic, where fragments of first-order logic are defined over a variable signature of non-logical constants. For brevity, we call an open-class lexicon a *vocabulary*, and, for a given choice of vocabulary, we speak of the English fragment E2V *over* that vocabulary.

Movement rule

The simplified grammar of E2V allows us to state the usual rule for wh-movement with marginally less technical apparatus than usual. We take one phrase to *dominate* another in a sentence of E2V if the second is reachable from the first by following zero or more downward links in the phrase-structure of the sentence.

Definition 1. A phrase β of category NP *c-commands* a phrase γ of category RelPro in a parsed E2V sentence if the parent of β dominates γ, but β itself does not dominate γ.

The movement rule of E2V is:

> Every phrase β of the form RelPro(A,I) moves to the position immediately below a nearest phrase of form NP(empty,I) which c-commands β; moreover, every node NP(empty,I) is the destination of such a movement.

As usual, we speak of an NP from which a RelPro has been moved as a *trace* NP. Figures 5 and 6 illustrate how this movement rule is applied in the case of sentences (3) and (4).

It is important to uderstand the variables I and A mentioned in the movement rule. The index variable I occurs twice: once in the moved RelPro and once in the NP it is moved to. This is to be understood as requiring that the index variables for these phrases must unify. Given the further unifications of indices forced by the grammar rules, the movement rule thus has the effect of unifying the index variable of a trace NP in a relative clause with the index variable of the NP which that relative clause modifies. By contrast, the variable A, representing the semantic value of the moved RelPro, occurs only once in the movement rule. This means that the movement rule imposes no constraints on A, and, in effect, does not care about the semantic value of the moved RelPro. However, this semantic value has not been wasted, because the grammar rules

> NP(A,I) $\rightarrow$ RelPro(A,I)
> RelPro(A/A,I) $\rightarrow$ which (who, whom)

used in the construction of the relative clause force the semantic value of a trace NP to be the identity function A/A. The combined effect of these rules on the semantic values of relative clauses can be seen in figures 5 and 6.

Indexing rules

In the grammar of E2V, most phrasal categories have exactly one index variable; and in that case, we speak of the value of that variable during parsing as being the *index* of the phrase in question. We insist that every index variable in a phrase-structure tree of an E2V sentence be assigned one of the values x_1, x_2, ... For example, in the parse displayed in figure 3, the NP every artist, the N′ artist and the VP despises every beekeeper all have x_1 as their index. (Two phrases with the same index are said to be *coindexed*.) The assignment of values to index variables must of course conform to the unifications enforced by the DCG rules. However, indices of NPs are additionally required to obey a set of *indexing rules*, which function so as to regulate the use of pronouns and reflexives. Sequences of words corresponding to parse trees where the index variables cannot be assigned values in accordance with the indexing rules fail to qualify as E2V sentences.

Within the transformational tradition, the use of Pronouns and Reflexives is accounted for by *binding theory*; and that is the theory which we adopt for E2V. No particular fealty to one linguistic school is hereby implied: for our purposes, this strategy is simply a convenient way to ensure that our grammar conforms to normal English usage. The indexing rules are divided into two classes: the *natural* indexing rules and an the *artificial* indexing rules. The natural indexing rules are:

I1: Indices of NPs must obey all the usual constraints of binding theory

I2: No two Ns may be coindexed.

Readers unfamiliar with government and binding theory are referred to a a standard text on the subject e.g. Cowper [3], p. 171. Using the standard technical terminology, these rules state that: a Reflexive must be A-bound within its minimal governing category; a Pronoun must be A-free within its minimal governing category; and an R-expression must be A-free. Because of the limited grammar of E2V, the finer points of binding theory may be ignored here. We remark that, in E2V, the minimal governing category of a Pronoun or Reflexive is always the nearest IP dominating it.

Sentences (1)–(3) illustrate the effect of rule **I1**. Consider first sentence (1), containing the two nouns artist and beekeeper. Its phrase-structure is shown in figure 3. Here, the requirement that an R-expression be A-free forces the indices of these nouns to be distinct. The unifications enforced by the grammar rules then imply that the two indices of the verb despises are distinct, and that the whole sentence has the semantic value shown, up to renaming of indices. Consider now sentence (2), containing the single noun artist and the reflexive pronoun himself. Its phrase-structure is shown in figure 4. The requirement that a Reflexive be A-bound in its minimal governing category forces himself and every artist to have the same index. The unifications enforced by the grammar rules then imply that the two indices of the verb admires are identical, and that the whole sentence has the semantic value shown, up to renaming of indices.

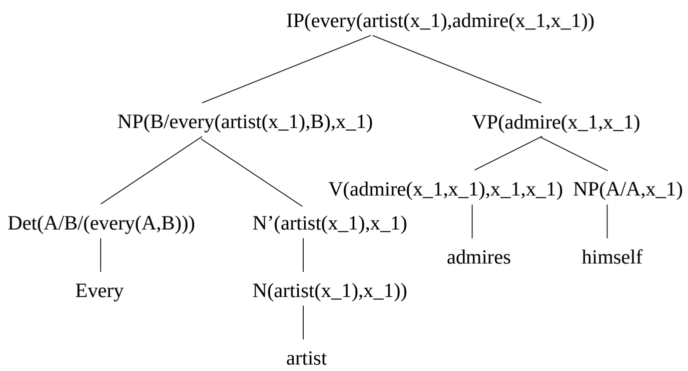

Figure 4: Phrase-structure of sentence (2), illustrating reflexive pronouns.

Consider finally sentence (3), containing the two nouns artist and beekeeper as well as the pronoun him. Its phrase-structure is shown in figure 5. The requirement that a pronoun be A-free in its minimal governing category prevents him from coindexing with beekeeper, but allows it to coindex with artist, resulting in the semantic value shown.

The status of rule **I2** is rather different. Though not part of standard binding theory, it is consistent with it. That is: given an assignment of indices to an E2V sentence obeying rule **I1**, we can always rename indices of some NPs if necessary in such a way that rules **I1** and **I2** are both satisfied. (This is straightforward to verify.) As we shall see in section 2.2, rule **I2** allows us to state the semantics for E2V in a very simple way.

Having dealt with the natural indexing rules, we move on to the artificial indexing rules. These are:

I3: Every pronoun must take an antecedent in the sentence in which it occurs.

I4: Every pronoun must be coindexed with the closest possible NP consistent with rules **I1**–**I3**.

In **I4**, 'closest' means 'closest in the phrase-structure', not 'closest in the lexical order'.

These rules are artificial in that they make no attempt to describe natural English usage. In particular, rule **I3** requires that all anaphora be resolved intrasententially. Clearly, this constitutes a restriction on normal English usage, and is introduced in order to simplify the language we are studying. Rule **I4** is rather more interesting. The effect of this rule can be seen by examining sentence (4), containing the nouns artist, beekeeper and carpenter, and the pronoun him. Its phrase-structure is shown in figure 6. Rules **I1**–**I3** permit the pronoun him to coindex with either artist or carpenter (but not beekeeper), corresponding to a perceived (anaphoric) ambiguity in the English sentence. However, we see from figure 6 that the NP every artist who employs a carpenter is closer to the

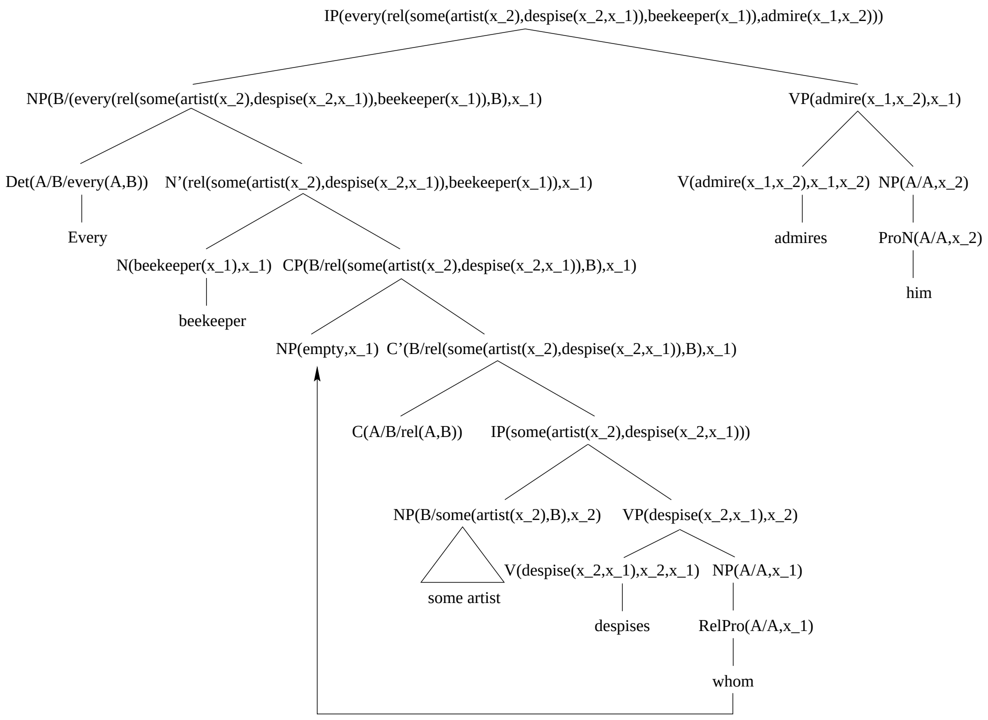

Figure 5: Phrase-structure of sentence (3), illustrating pronouns and movement.

pronoun in the phrase-strcuture than the NP a carpenter is. Hence, rule **I4** requires him to coindex with the former, and results in the semantic value shown. Rule **I4** does not change the set of strings accepted by the fragment E2V; but it does ensure that any accepted string has a unique indexation pattern up to renaming. It also plays a crucial role in restricting the expressive power of E2V to that of the two-variable fragment of first-order logic.

We say that a string of words is an *E2V-sentence* over a given vocabulary if, according to the above syntax, it is the list of terminal nodes below an IP node with no parent.

2.2 Semantics

Having defined the set of E2V sentences over a given vocabulary, we now turn to the translation of these sentences into first-order logic.

We have already seen that the syntax of E2V assigns a semantic value to every E2V sentence. This semantic value is a complex term formed from the primitives occurring in the vocabulary by means of the constructors some(A,B), every(A,B), rel(A,B) and not(A). For example, we see from figures 3–6 that sentences (1)–(4) are assigned the respective semantic values:

(9) every(artist(x_1),every(beekeeper(x_2),despise(x_1,x_2)))

(10) every(artist(x_1),admire(x_1,x_1))

(11) every(rel(some(artist(x_2),despise(x_2,x_1)),beekeeper(x_1)), admire(x_1,x_2))

(12) every(rel(some(carpenter(x_2),employ(x_1,x_2)),
artist(x_1)),
every(rel(admire(x_3,x_1),beekeeper(x_3)), despise(x_1,x_3))).

To complete the semantics for E2V, it suffices to define a function T mapping semantic values of E2V sentences to formulas of first-order logic. The key idea behind this translation is that indices x_1, x_2, ... in semantic values can simultaneously be regarded as variables $x_1, x_2, \ldots$ in formulas. (The different styles of writing indices/variables help to make semantic values and formulas more visually distinct.)

Definition 2. Let A be the semantic value of an E2V-sentence, and let B be any (not necessarily proper) subterm of A. We define the function T from such subterms to formulas of first-order logic as follows:

T1: If B = some(C,D) then $T(\mathrm{B}) = \exists\bar{z}(T(\mathrm{C}) \wedge T(\mathrm{D}))$ where $\bar{z}$ is the tuple of indices which are free variables in $T(\mathrm{C}) \wedge T(\mathrm{D})$ but which do not occur in A outside B.

T2: If B = every(C,D) then $T(\mathrm{B}) = \forall\bar{z}(T(\mathrm{C}) \rightarrow T(\mathrm{D}))$ where $\bar{z}$ is the tuple of indices which are free variables in $T(\mathrm{C}) \rightarrow T(\mathrm{D})$ but which do not occur in A outside B.

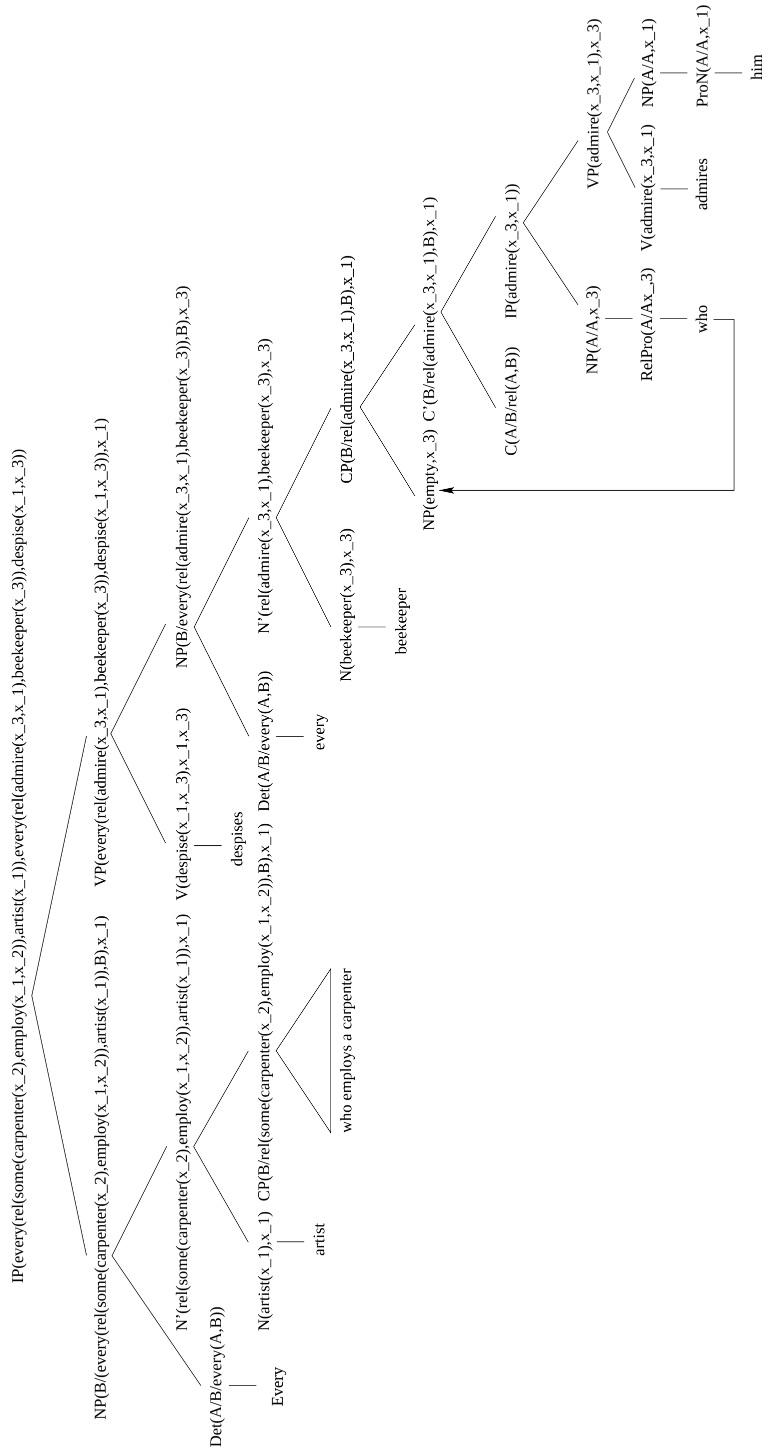

Figure 6: Phrase-structure of sentence (4).

T3: If B = rel(C,D) then T(B) = (T(C) $\wedge$ T(D))

T4: If B = not(C) then T(B) = ($\neg T$(C))

T5: Otherwise, T(B) = B.

We then take the translation of the semantic value A to be simply T(A).

Under these translation rules, the semantic values (9)–(12) of the sentences (1)–(4) translate to the formulas (5)–(8). To see how this translation works in detail, consider first sentence (1) and its semantic value (9). The simple subterms artist(x_1), beekeeper(x_2) and despise(x_1,x_2) translate to the atomic formulas artist(x_1), beekeeper(x_2) and despise(x_1, x_2), respectively, by rule **T5**. The complex subterm

(13) every(beekeeper(x_2),despise(x_1,x_2))

then translates to the formula

(14) $\forall x_2$(beekeeper(x_2) $\rightarrow$ despise(x_1, x_2)),

by rule **T2**, because, of the two free variables in question, namely x_1 and x_2, only the latter satisfies the condition that it does not occur outside the subterm. Finally, the whole expression (9) translates to (5), again by rule **T2**.

We note that, in some applications of rule **T1** (and indeed, with the grammar as presented above, some applications of rule **T2**), the tuple $\bar{z}$ may be empty, in which case, of course, $\exists\bar{v}$ is understood to be absent altogether. Consider sentence (3) and its semantic value (11). Here, the subterm

(15) some(artist(x_3),despise(x_3,x_2)

translates to the (unquantified) formula

(16) (artist(x_2) $\wedge$ despise(x_2, x_1)),

by rule **T1**, because, of the two free variables in question, namely x_1 and x_2, both occur in (11) outside the subterm. Suitable applications of rules **T2** and **T3** generate the final translation (7).

2.3 Further remarks on the syntax and semantics of E2V

Relation to DRT

An alternative and more general approach to the semantics of anaphoric constructions found in E2V is provided by Discourse Representation Theory (DRT) (Kamp and Reyle [17]). Like the semantics presented above, DRT in effect employs a two-stage approach in translating from English to first-order logic. First, English sentences are mapped to so-called Discourse Representation Structures (DRSs)—a formal representation language with nonstandard mechanisms for quantification—and these DRSs can then be translated into formulas of first-order logic in accordance with the standard semantics for the DRS language.

This two-stage approach allows DRT to give an account of the variable-binding encountered in sentence (3) very similar to that given in this paper. An elegant and technical account of DRT-style semantics for a fragment of English similar to E2V can be found in Muskens [22]. (In fact, Muskens uses an ingeneous interpretation of DRSs which effectively collapses the two stages of translation into one; however, this feature of his account does not affect the observations below.)

The main difference between our approach and that of DRT concerns the point at which quantifiers corresponding to indefinite articles are introduced. Our rule **T1** translates the indefinite article as an existential quantifier *provided that the variable which the quantifier would bind is not going to end up appearing outside its scope in the resulting formula.* The strategy of DRT, by contrast, is that the indefinite article does not introduce a quantifier at all: rather, 'left-over' variables corresponding to these determiners are gathered up by quantifiers introduced in the interpretation of DRSs.

For example, consider the sentence

(17) Every artist who admires a beekeeper who berates a carpenter despises himself.

On our approach, sentence (17) has the semantic value

(18)
```
every(rel(some(rel(some(carpenter(x_3),berate(x_2,x_3)),
                      beekeeper(x_2)),
                   admire(x_1,x_2)),
              artist(x_1)),
          despise(x_1,x_1))
```

which then translates to

$$\begin{array}{ll} (19) & \forall x_1((\text{artist}(x_1)\wedge \\ & \quad \exists x_2((\text{beekeeper}(x_2)\wedge \\ & \quad\quad \exists x_3(\text{carpenter}(x_3) \wedge \text{berate}(x_2, x_3))) \wedge \text{admire}(x_1, x_2))) \rightarrow \\ & \quad \text{despise}(x_1, x_1)). \end{array}$$

By contrast, DRT would assign (17) the DRS

$$(20) \quad \boxed{\begin{array}{|l|} \hline x_1 \quad x_2 \quad x_3 \\ \hline \text{artist}(x_1) \\ \text{beekeeper}(x_2) \\ \text{carpenter}(x_3) \\ \text{berates}(x_2,x_3) \\ \text{admires}(x_1,x_2) \\ \hline \end{array} \Rightarrow \boxed{\text{despise}(x_1,\, x_1)}}$$

which then translates via the standard 'weakest-precondition' semantics to

$$\begin{array}{ll} (21) & \forall x_1 \forall x_2 \forall x_3((\text{artist}(x_1) \wedge \text{beekeeper}(x_2)\wedge \\ & \quad\quad \text{carpenter}(x_3) \wedge \text{berate}(x_2, x_3) \wedge \text{admire}(x_1, x_2)) \rightarrow \\ & \quad \text{despise}(x_1, x_1)). \end{array}$$

Formulas (19) and (21) are, of course, logically equivalent. However, there is a difference in the way variables are used: our approach quantifies x_2 and x_3 as early as possible, while DRT does so as late as possible. It turns out that this strategy of early quantification means that we use variables more 'efficiently' than DRT does; and that is why the semantics presented here makes the expressiveness of E2V easier to determine. Thus, we are not (as far as we know) taking issue with DRT; rather, we are simply presenting the semantics of E2V in a form which is more convenient for the issues at hand.

Accessibility

One respect in which our semantics for E2V fails to do justice to English speakers' intuitions concerns pronoun accessibility. Consider again sentence (3), repeated here as (22):

(22) Every beekeeper whom an artist despises admires him.

Recall that we assume all anaphora to be resolved intrasententially, so that the pronoun in this sentence takes the NP an artist as its antecedent. However, the availability of this NP as an antecedent depends on the fact that it is *existentially* quantified. Thus, in the sentence

(23) Every beekeeper whom every artist despises admires him,

the anaphora cannot be resolved intrasententially: the admired individual must have been introduced by some earlier sentence in the discourse. Thus, we might say that the NP every artist in (23) is *inaccessible* to the pronoun which follows it. An adequate grammar for a restricted fragment of English should rule out sentences in which pronouns are required to take inaccessible antecedents. It is thus a defect of the above grammar for E2V that sentence (23) is accepted, and—as the reader may verify—translated into the same formula as sentence (22)!

Accessibility restrictions are discussed in detail within the framework of DRT; and we have nothing to add to their proper treatment in a grammar of English. However, this issue may be safely ignored for the purposes of the present paper, because the argument in sections 3 and 4 shows that no reasonable accessibility restrictions could make any difference to the expressive power of E2V. (This point will become clear at the end of section 4.) Thus, to simplify the proofs in the following sections, we shall take E2V to include sentences such as (23), confident that their eventual exclusion will make no difference to our results.

Quantifier scoping

Another respect in which E2V fails to do justice to English speakers' intuitions is quantifier rescoping. It is generally claimed that the sentence

(24) Every artist despises a beekeeper

is ambiguous between two choices for the scoping of the quantifiers. By contrast, the above semantics unambiguously assigns wide scope to the universal quantifier. We do not consider quantifier rescoping in this paper. Generally, proposals for controlled languages eliminate such ambiguities by stipulating that quantifiers scope in a particular way; and this is the approach we take. Very roughly, the effect of the above grammar rules is that the quantifier introduced by the subject determiner outscopes the quantifier introduced by the object determiner, and that the quantifier introduced by any NP determiner outscopes those introduced in any relative clauses within that NP. This policy seems the most sensible default in those cases where a choice has to be made.

Negation

The primary mechanism in E2V which provides negation is the category Neg, whose sole member is the two-word phrase does not. For example, the sentence

(25) Some artist does not despise every beekeeper

is assigned the semantic value

(26) some(artist(x_1),not(every(beekeeper(x_2),despise(x_1,x_2)))),

which then translates to

(27) $\exists x_1(\text{artist}(x_1) \wedge \neg\forall x_2(\text{beekeeper}(x_2) \rightarrow \text{despise}(x_1, x_2)))$.

A more sensitive account of the semantics of E2V would complicate the rules regarding negation in two related respects. First, the effect of negation on verb-inflection should be taken into account; second, the word does should be assigned category I (or INFL), with the single word not taken to be the representative of the category Neg. These changes can be effected by adopting the following grammar rules for IP (with variables suppressed for readability):

$\text{IP} \rightarrow \text{NP}, \text{I}'$
$\text{I}' \rightarrow \text{I}, \text{NegP}$
$\text{I}' \rightarrow \text{I}, \text{VP}$,

and by subjecting verbs in unnegated sentences to movement into the I position, where they are joined to the inflection.

Negation brings with it some additional complications concerning scoping and the negative polarity determiner any. Again, scoping ambiguities are resolved by fiat. Simplifying somewhat, the above translation rules take the negation in a NegP to outscope quantification within the NegP, but to be outscoped by quantification in the subject governing that NegP, as in sentence (25). Again, this seems to be the most reasonable default.

Negative polarity is ignored altogether in this paper. Thus for example, E2V employs

(28) Some artist does not despise a beekeeper,

rather than the less ambiguous-sounding

(29) Some artist does not despise *any* beekeeper,

to express

(30) $\exists x_1(\text{artist}(x_1) \wedge \neg\exists x_2(\text{beekeeper}(x_2) \wedge \text{despise}(x_1, x_2)))$.

Other negative sentences accepted by E2V are somewhat awkward, for example:

(31) Every artist does not despise a beekeeper,

which translates to

(32) $\forall x_1(\text{artist}(x_1) \rightarrow \neg\exists x_2(\text{beekeeper}(x_2) \wedge \text{despise}(x_1, x_2)))$.

In keeping with the general desire to remove ambiguity from controlled languages, it might be important to consider fragments of English from which sentences such as (28) and (31) are either replaced by sentences involving negative polarity items or excluded altogether. However, as with the issue of pronoun accessibility, so too with that of negation, the restrictions in question can be ignored for the purposes of the present paper. The argument in sections 3 and 4 shows that no reasonable restrictions on the use of negation could make any difference to the expressive power of E2V. (This point will become clear at the end of section 4.) Thus, to simplify the proofs in the following sections, we shall take E2V to include sentences such as (28) and (31), confident that their eventual exclusion or modificatation will make no difference to our results.

Our fragment E2V includes one further mechanism for introducing negation, namely, the determiner no. This addition to the closed class lexicon is a small concession to naturalness of expression. In fact, it could be dropped from E2V without affecting its expressive power.

2.4 Implementation

The foregoing specification of E2V was couched in terms which permit direct computer implementation. In particular, the DCG rules of the grammar and lexicon in figures 1 and 2 map almost literally to Prolog code, with some very minimal extra control structure required to implement the indexing rules **I1**–**I4**. The movement rule can easily be incorporated into this DCG using standard argument-passing techniques, for example, as described in Walker *et al.* [31], pp. 351 ff. Implementation of the translation rules **T1**–**T5** is completely routine. Because only bounded look-ahead is required (to distinguish those N′s which contain relative clauses), the first-order translation of an E2V sentence can certainly be computed in quadratic time. Hence the complexity of translation from E2V is not an issue we shall be concerned with in the sequel.

A program to transform E2V sentences to their semantic values and first-order logic translations has been written for Sicstus Prolog version 3.8.3. This program also incorporates standard DRT accessibility restrictions and enforces correct verb-inflections in negated and unnegated sentences, as discussed in section 2.3. All semantic values and first-order translations of E2V sentences given in this paper are the unedited output of this program.

3 Expressiveness: upper bound

The main result of this section states that, essentially, the translations of E2V-sentences remain inside the two-variable fragment of first-order logic. In the sequel, we denote the set of formulas of first-order logic by $\mathcal{L}$, and the set of formulas of the two-variable fragment by $\mathcal{L}^2$. The following notion captures what we mean by "essentially" in the sentence before last.

Definition 3. If $\phi \in \mathcal{L}$, we say that ϕ is *two-variable compatible* (for short: 2vc) if no proper or improper subformula of ϕ contains more than two free variables.

We have the following result.

Lemma 1. *Every 2vc formula is equivalent to a formula of $\mathcal{L}^2$.*

Proof. We replace successively complex subformulas of ϕ with equivalent formulas in $\mathcal{L}^2$, renaming variables in the rest of ϕ if necessary. The details are routine. □

The property of being a 2vc formula is not closed under the relation of logical equivalence. For example, formulas (19) and (21) are logically equivalent, but only the former is 2vc. Indeed, this example shows why our approach to the semantics for E2V makes for an easier analysis of expressive power than that of DRT.

The result we wish to establish in this section is:

Theorem 1. *If α is an E2V-sentence with semantic value A, then $T(A)$ is a closed 2vc formula.*

To avoid unnecessary circumlocutions, if β is a phrase with semantic value B, then we say that β *translates to* the formula T(B). Our strategy will be to show that, if β is any phrase in α, then β translates to a 2vc formula whose free variables satisfy certain constraints. For clarity, we henceforth display *all* free variables in formulas. Thus, $\phi(x, y)$ has no free variables except for (possibly) x and y, $\phi(x)$ has no free variables except for (possibly) x, and so on.

The following terminology will prove useful:

Definition 4. Let β and γ be phrases in an E2V sentence and Y a category (e.g. Y = N′, IP or VP). Then γ is said to be a *maximal* Y (properly) dominated by β if γ is a Y (properly) dominated by β and no other phrase which is a Y (properly) dominated by β dominates γ. Similarly, β is said to be a *minimal* Y (properly) dominating γ if β is a Y (properly) dominating γ and no other phrase which is a Y (properly) dominating γ is dominated by β.

A phrase β of category X (X = N′, IP or VP) is said to be *pronomial* if a maximal NP dominated by β expands to a Pronoun.

A VP β is said to be *reflexive* if the maximal NP dominated by β expands to a Reflexive.

A VP β is said to be *trace* if the maximal NP dominated by β expands to a RelPro which has been subjected to movement.

The following examples should help to clarify these definitions (t indicates a moved RelPro):

pronomial VP	admires him
pronomial N′	beekeeper whom he despises t
	beekeeper who t admires him
reflexive VP	despises himself
trace VP	admires t

We also need some terminology to deal with so-called *donkey*-sentences. For the purposes of E2V, we may define:

Definition 5. A pronomial E2V sentence—that is one where the object of the main verb is a pronoun—is called a *donkey sentence*, and the pronoun in question is called its *donkey pronoun*. In addition, a phrase β of category X (X = N′, IP or VP) is said to be *donkey* if it is a maximal phrase of category X properly dominated by the root node of a donkey sentence.

The paradigm donkey-sentence is:

(33) Every farmer who owns a donkey beats it.

Within this sentence, the following donkey phrases occur:

donkey N′	farmer who t owns a donkey
donkey IP	t owns a donkey
donkey VP	owns a donkey.

It is clear that a donkey sentence properly contains exactly one donkey N′, IP and VP, and that these must all be maximal phrases of the relevant category properly dominated by the subject NP. We remark that a phrase may be both pronomial and donkey, for example, the italicized N′ in the sentence:

(34) Every *farmer who owns a donkey which likes him* beats it,

which, incidentally, translates to the 2vc formula:

(35) $$\forall x_1\ x_2((\text{farmer}(x_1) \wedge ((\text{donkey}(x_2) \wedge \text{like}(x_2, x_1)) \wedge \text{own}(x_1, x_2))) \rightarrow \text{beat}(x_1, x_2)).$$

Finally, the following notation will be useful:

Definition 6. If β is a phrase of category N′ or VP, we write ind(β) to denote the index of β. If β is a phrase of category IP, but not a sentence, we write ind(β) to denote the index of the NP which is moved out of β by the movement rule, and we refer to ind(β) informally as the index of β.

With this terminology behind us, we prove some auxiliary lemmas about the grammar and translation rules.

Lemma 2. *If β is a non-sentence IP, then ind(β) is also the index of the minimal NP (or N′) dominating β.*

Proof. By the unifications of index variables forced by the movement rule and the grammar rules. □

Lemma 3. *If β is a reflexive VP, then β translates to an atomic formula of the form $d(x, x)$, where $x = ind(\beta)$.*

Proof. By the grammar rule IP(A) $\rightarrow$ NP(B/A,I), VP(B,I), β coindexes with its left sister. By rule **I1**, the left sister of β coindexes with the reflexive in β. Hence the semantic value of β will be a simple term of the form d(x,x). The result then follows from **T5**. □

Lemma 4. *If β is either an N′, a non-sentence IP or a VP, and is pronomial, then β translates to a 2vc formula $\phi(x, y)$ where $x = ind(\beta)$ and y is the index of the pronoun contained in β.*

Proof. If β is a VP, then β has the form shown in figure 7a). Certainly, γ translates to an atomic formula $c(x, y)$, where ind$(\beta) = x$ and y is the index of the pronoun. Then β also translates to $c(x, y)$.

If β is a non-sentence IP, then β has the one of the two possible structures shown in figure 7b) and c). By definition, ind(β) is the index of the trace NP. It is then immediate that β translates to an atomic formula of either of the forms $c(x, y)$ or $c(y, x)$, where $x =$ ind(β) and y is the index of the pronoun contained in β.

If β is an N′, then β has the structure shown in figure 7d), where the IP δ is also pronomial. Let $x =$ ind$(\beta) =$ ind(γ). By lemma 2, ind$(\beta) =$ ind(δ). Certainly, γ translates to some atomic formula $c(x)$ and by the result just obtained δ translates to an atomic formula of one of the forms $d(x, y)$ or $d(y, x)$. Then β translates to $d(x, y) \wedge c(x)$ or $d(y, x) \wedge c(x)$, as required. □

We remark that lemma 4 holds whether or not β is donkey.

Lemma 5. *Let β be either of the phrases depicted in figure 8, with γ pronomial. Then the index of the pronoun in γ is also the index of β.*

Proof. In case a), rule **I4** forces the pronoun in γ to coindex with the minimal NP dominating β. (There must be such an NP, because an NP has been moved out of β.) The result then follows from lemma 2. In case b), the left-sister of β is either a full NP (expanding to a Det and N′) or an NP trace which coindexes with the minimal NP dominating β. Either way, **I4** forces the pronoun in β to coindex with this NP. Again, the unifications of index variables enforced by the grammar rules and (if applicable) the movement rule ensure that β also coindexes with this NP. □

Lemma 6. *Let α be an E2V sentence, β be a non-donkey phrase of category IP or VP properly dominated by α, and γ a maximal N′ properly dominated by β. Then ind(γ) does not occur outside β in the semantic value of α.*

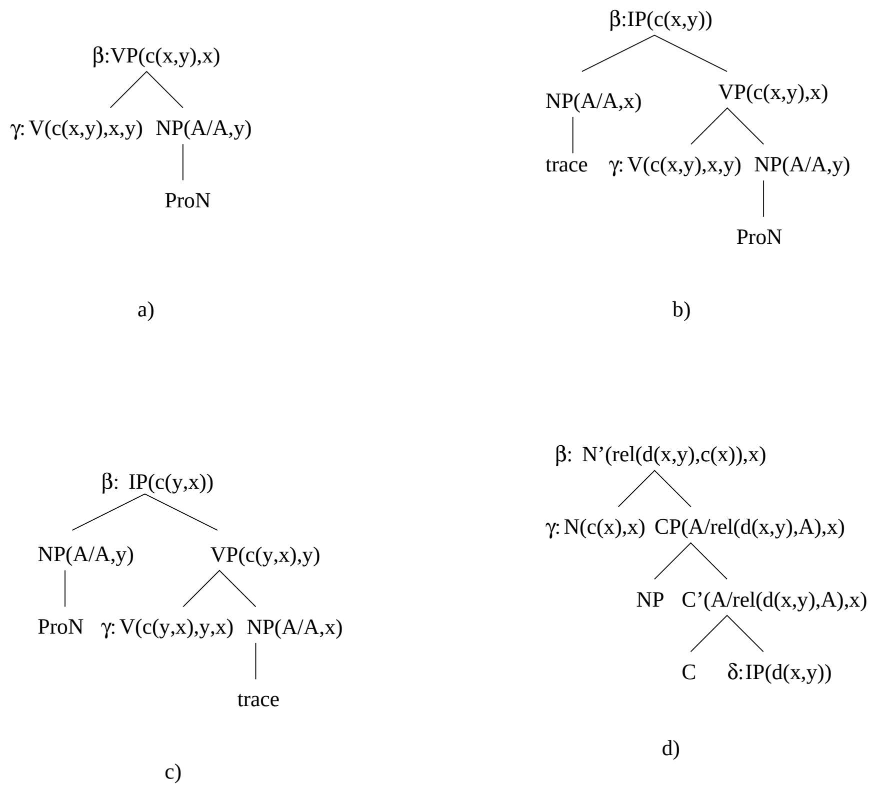

Figure 7: Structures of some pronomial, non-sentence phrases.

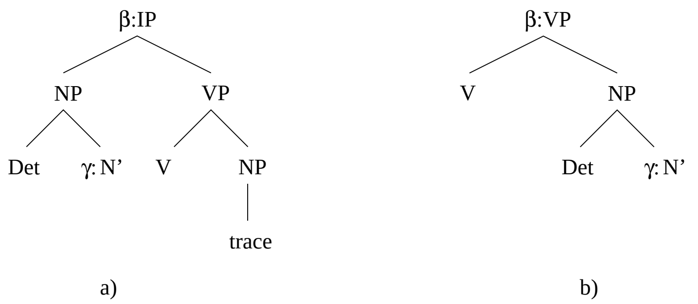

Figure 8: Statement of lemma 5.

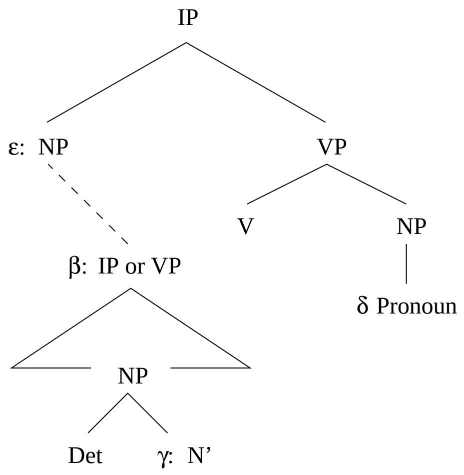

Figure 9: Proof of lemma 6.

Proof. Refer to figure 9. It is easy to check using the grammar rules that any index occurring outside β is the index of some NP occurring outside β, so that if $\mathrm{ind}(\gamma)$ occurs outside β, it does so as the index of some N, Reflexive or Pronoun. Rule **I2** rules out the first possibility, and **I1** rules out the second. It follows that $\mathrm{ind}(\gamma)$ can occur outside β only if it is the index of a pronoun δ. In fact, the minimal NP dominating γ must be the antecedent of δ. Since γ and hence β must precede δ, β must have a minimal dominating NP, say ε. By rules **I2** and **I4**, no full NP (expanding to a Det and N$'$) other than ε can intervene on the path between δ and γ. It is then easy to see that δ must be a donkey pronoun, and that the minimal NP dominating γ is its antecedent. This contradicts the supposition that β is not donkey. □

At this point, we are ready to state the main lemma used in the proof of theorem 1.

Lemma 7. *For every phrase β such that β is either an N$'$, a non-sentence IP or a non-trace VP, we have:*

a) *if β is non-donkey and non-pronomial, then β translates to a 2vc formula $\phi(x)$ where $x = ind(\beta)$;*

b) *if β is donkey, then β translates to a 2vc formula $\phi(x, y)$ where $x = ind(\beta)$ and y is the index of the donkey pronoun.*

Proof. We proceed by induction on the number of N$'$, IP or VP nodes properly dominated by β in the phrase-structure tree for the sentence in question. For simplicity, we ignore NegPs in this proof, which make no difference to the variable binding patterns.

We have three cases to consider, depending on whether β is (i) an N$'$, (ii) an IP or (iii) a VP.

(i) Let β be an N$'$, with ind$(\beta) = x$. Clauses a) and b) of the lemma are then established as follows:

a) Suppose β is non-pronomial and non-donkey. Then β is either a single noun and so translates to, say, $b(x)$, or is of the form depicted in figure 10a). Since β is non-pronomial and non-donkey, so is δ. By lemma 2 ind$(\delta) =$ ind(β), and by inductive hypothesis δ translates to a 2vc formula $\psi(x)$. Since ind$(\gamma) =$ ind(β), γ translates to an atomic formula, say $c(x)$, whence γ translates to $\psi(x) \wedge c(x)$ by **T3** as required.

b) Suppose β is donkey. Then β must be of the form depicted in figure 10a), with δ also donkey. By lemma 2 ind$(\delta) =$ ind(β), and by inductive hypothesis δ translates to a 2vc formula $\psi(x, y)$, where y is the index of the donkey pronoun. Again, γ translates to an atomic formula, say $c(x)$, whence β translates to $\psi(x, y) \wedge c(x)$ by **T3** as required.

(ii) Let β be a non-sentence IP, with ind$(\beta) = x$. Rule **I1** prevents the subject of an IP from being a reflexive; furthermore, an IP with a pronoun subject cannot fall under either of the cases a) or b) of the lemma. Hence, we have two sub-cases to consider, depending on whether the subject or the object of the IP is a trace, as depicted in figure 10b) and c), respectively. Note: in figure 10c)–e), we use the notation Q(S,T) to denote any of some(S,T), every(S,T) and every(S,not(T)), where S and T are semantic values.

We consider first the sub-case of figure 10b). By definition, ind(β) is the index of the trace, which, by the phrase structure rules, is also the index of γ. Thus ind$(\beta) =$ ind$(\gamma) = x$. We establish clauses a) and b) of the lemma as follows:

a) Suppose β is non-pronomial and non-donkey. So is γ, then. By inductive hypothesis, γ translates to a 2vc formula $\psi(x)$. But then so does β.

b) Suppose β is donkey. So is γ then. By inductive hypothesis, γ translates to a 2vc formula $\psi(x, y)$, where y is the index of the donkey pronoun. But then so does β.

Next, we consider the sub-case of figure 10c). By definition, ind(β) is the index of the trace in δ. Let $y =$ ind(γ). Then certainly, δ translates to some atomic formula $d(y, x)$. Furthermore, from its position in the phrase-structure, γ is certainly not donkey. Suppose in addition that it is not pronomial. Then, by inductive hypothesis, γ translates to some 2vc formula $\psi(y)$, which we may write as $\psi(y, x)$. Suppose on the other hand that γ is pronomial. Then by lemma 4, it translates to a 2vc formula $\psi(y, z)$ where z is the index of the pronoun contained in γ. Moreover, by lemma 5, $z =$ ind$(\beta) = x$. Hence γ translates to $\psi(y, x)$. We now establish clauses a) and b) of the lemma as follows:

a) Suppose β is non-pronomial and non-donkey. By lemma 6 then, y does not occur outside β. Depending on whether the determiner in question is a, every or no, by **T1**, **T2** and (possibly) **T4**, β translates to one of $\exists y(\psi(y,x) \wedge d(y,x))$, $\forall y(\psi(y,x) \rightarrow d(y,x))$ or $\forall y(\psi(y,x) \rightarrow \neg d(y,x))$ as required.

b) Suppose β is donkey. By rule **I4**, y must be the index of the donkey pronoun, so that both x and y occur outside β. By similar reasoning to case a), β translates to one of $(\psi(y,x) \wedge d(y,x))$, $(\psi(y,x) \rightarrow d(y,x))$ or $(\psi(y,x) \rightarrow \neg d(y,x))$ as required.

(iii) Let β be a non-trace VP, with $\mathrm{ind}(\beta) = x$. By lemma 3, if β is reflexive, it translates to an atomic formula $b(x,x)$. Moreover, no reflexive VP can be donkey (otherwise the donkey pronoun would have no antecedent). So we may as well assume that β is not reflexive. If β is donkey, then it cannot be pronomial, for then there would be no antecedent for the pronoun in β. In cases a) or b) of the lemma, then, we can take β to have the form depicted in figure 10d). Let $y = \mathrm{ind}(\delta)$, and let γ translate the atomic formula $c(x,y)$. By its position in the phrase-structure, δ is not donkey. If, in addition, δ is not pronomial, by inductive hypothesis, it translates to a 2vc formula $\psi(y)$, which we may write as $\psi(y,x)$. If, on ther other hand, δ is pronomial, by lemma 4, it translates to a 2vc formula $\psi(y,z)$, where z is the index of the pronoun contained in δ. Moreover, by lemma 5, $z = \mathrm{ind}(\beta) = x$. Hence δ translates to $\psi(y,x)$. We now establish clauses a) and b) of the lemma as follows:

a) Suppose β is not pronomial, and not donkey. By lemma 6, y does not occur outside β. Depending on whether the determiner in question is a, every or no, by **T1**, **T2** and (possibly) **T4**, β translates to one of $\exists y(\psi(y,x) \wedge c(x,y))$, $\forall y(\psi(y,x) \rightarrow c(x,y))$ or $\forall y(\psi(y,x) \rightarrow \neg c(x,y))$, as required.

b) Suppose β is donkey. By rule **I4**, y must be the index of the donkey pronoun, so that x and y both occur outside β. By similar reasoning to case a), β translates to one of $(\psi(y,x) \wedge c(x,y))$, $(\psi(y,x) \rightarrow c(x,y))$ or $(\psi(y,x) \rightarrow \neg c(x,y))$, as required.

□

Now we can return to the main theorem of this section.

Proof of theorem 1. Again, for simplicity, we ignore occurrences of NegP in this proof. Let α have the structure shown in figure 10e), then. And let $x = \mathrm{ind}(\gamma) = \mathrm{ind}(\delta)$. Suppose first that α is non-donkey. Then γ and δ are non-pronomial and non-donkey, and δ is certainly non-trace. By lemma 7, γ and δ translate to 2vc formulas $\phi(x)$ and $\psi(x)$ respectively. Since α has no parent, by **T1** and **T2** and (possibly) **T4**, it translates to one of $\exists x(\phi(x) \wedge \psi(x))$, $\forall x(\phi(x) \rightarrow \psi(x))$ or $\forall x(\phi(x) \rightarrow \neg\psi(x))$, as required.

Suppose on the other hand that α is donkey. By lemma 7, γ translates to a 2vc formula $\phi(x,y)$, where y is the index of the donkey pronoun. By lemma 4, δ

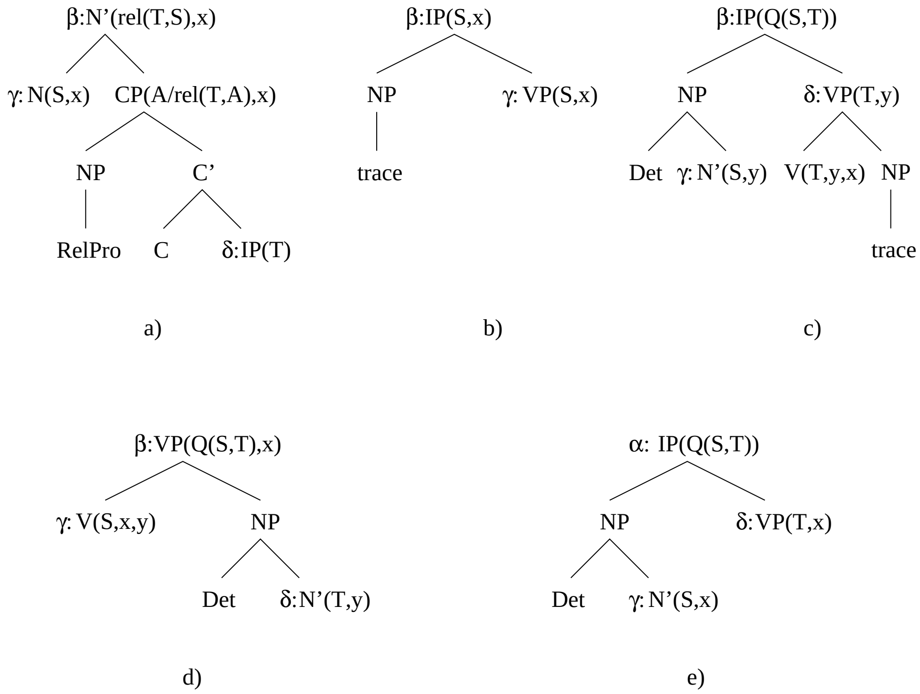

Figure 10: Proofs of lemma 7 and theorem 1. The letters S and T stand for semantic values.

translates to a formula $\psi(x,y)$. Again, since α has no parent, by **T1** and **T2** and (possibly) **T4**, it translates to one of $\exists x \exists y(\phi(x,y) \wedge \psi(x,y))$, $\forall x \forall y(\phi(x,y) \rightarrow \psi(x,y))$ or $\forall x \forall y(\phi(x,y) \rightarrow \neg\psi(x,y))$, as required. □

The following result gives us an upper complexity bound for E2V-satisfiability:

Theorem 2 (Börger, Gurevich and Grädel [2] corollary 8.1.5). *The problem of deciding the satisfiability of a sentence in $\mathcal{L}^2$ is in NEXPTIME.*

Proof strategy. Show that if ϕ is a satisfiable $\mathcal{L}^2$ sentence, then ϕ has a model whose size is at most exponential in the size of ϕ. □

Corollary 1. *The problem of determining the satisfiability of a set E of E2V sentences is in NEXPTIME.*

We conclude this section with some brief remarks on the possibility of extending the syntax of E2V without losing decidability. It should be clear from the preceding analysis that various additions to the syntax of E2V—essential if the fragment is to form a convenient and useable controlled language—would result in no essential increase in expressive power. Additions in this category include proper nouns, intransitive verbs, (intersective) adjectives, the verb to be, conjunctions such as and and or and perhaps some forms of coordination. Other useful additions would, however, increase computational complexity. The most salient of these are the counting determiners such as at least four, exactly three etc. Such expressions will not in general translate into the two-variable fragment, but they will translate into the fragment C^2, which adds counting quantifiers to the two-variable fragment. The satisfiability problem for this language is shown by Pacholski, Szwast and Tendera [24] to be decidable in nondeterministic doubly exponential time.

4 Expressiveness: lower bound

In the previous section, we showed that the English fragment E2V does not take us beyond the expressive power of the two-variable fragment. In this section, we show that, with two minor semantic stipulations, we essentially obtain the whole of the two-variable fragment.

The first semantic stipulation concerns the expression of identity. The identiy relation is not expressible in E2V, and so we need to single out a verb—eqs—which will be mapped to the identity relation. That is, we have the lexicon entry

V(I=J,I,J) $\rightarrow$ eqs.

The second semantic stipulation concerns the expression of the universal property. All noun-phrases in E2V contain common nouns, which automatically restricts quantification in sentence-subjects. It is therefore convenient to single out a noun—thing—which expresses the property possessed by everything. That is, we have the lexicon entry

$\forall x \forall y(a(x,y) \to b(y,x))$	Everything bees everything which ays it
$\forall x(\exists y a(x,y) \to b(x,x))$	Everything which ays something bees itself
$\forall y(\exists x a(x,y) \to b(x,x))$	Everything which something ays bees itself
$\forall x(b(x,x) \to \forall y a(x,y))$	Everything which bees itself ays everything
$\forall x(b(x,x) \to \forall y a(y,x))$	Everything ays everything which bees itself
$\forall x(\exists y a(x,y) \to b(x))$	Everything which ays something eqs a bee
$\forall x(\exists y a(y,x) \to b(x))$	Everything which something ays eqs a bee
$\forall x(b(x) \to \forall y a(x,y))$	Every bee ays everything
$\forall x(b(x) \to \forall y a(y,x))$	Everything ays every bee
$\forall x \forall y(a(x,y) \to \neg b(x,y))$	Nothing bees something which it ays
$\forall x \forall y(a(x,y) \to b(x,y))$	Everything bees everything which it ays
$\forall x \forall y a(x,y)$	Everything ays everything
$\forall x \exists y a(x,y)$	Everything ays something
$\forall x \forall y(\neg b(x,y) \to a(x,y))$	Everything ays everything which it does not bee
$\forall x \forall y((b(x,y) \wedge c(x,y)) \to a(x,y))$	Everything which bees something which it cees ays it

Table 1: Some formulas of $\mathcal{L}^2$ and their E2V-translations

N(I=I,I) $\to$ thing.

For ease of reading, we will perform the usual contractions of every thing to everything, some thing to something etc. We note that both assumptions can be dropped without invalidating corollaries 2 and 3 below.

To understand how E2V can essentially express the whole of the two-variable fragment, we need to establish an appropriate notion of expressive equivalence. To motive ideas, consider the formulas

$$\begin{aligned} \phi &:= \psi \vee (\pi \wedge \chi) \\ \phi' &:= (\psi \vee p) \wedge (p \leftrightarrow (\pi \wedge \chi)), \end{aligned}$$

where p is a new proposition letter not occurring in ϕ. Here, ϕ' has in effect introduced an abbreviation p for a subformula of ϕ, and used that abbreviation in place of the subformula it abbreviates. Thus, ϕ and ϕ', although not logically equivalent, in effect express the same state of affairs, given that p is a newly introduced primitive. Generalizing, we have:

Definition 7. Let $\phi, \phi' \in \mathcal{L}$ over a signature τ. We say that ϕ' is an *definitional equivalent* of ϕ (with respect to τ) if $\phi' \models \phi$ and, for any structure $\mathfrak{A}$ interpreting only τ such that $\mathfrak{A} \models \phi$, there is a unique expansion $\mathfrak{B}$ of $\mathfrak{A}$ such that $\mathfrak{B} \models \phi'$.

Our main result is that, for any closed formula ϕ in the two-variable fragment, a set E of E2V sentences can be found such that E translates to a set of formulas whose conjunction is a definitional equivalent of ϕ.

First, we establish an easy result about the two-variable fragment.

Lemma 8. *There is a function $f : \mathcal{L}^2 \to \mathcal{L}^2$, computable in polynomial time and linear space, such that, for all $\phi \in \mathcal{L}^2$, $f(\phi)$ is a definitional equivalent of*

ϕ consisting of a conjunction of formulas of the forms found in the left-hand column of table 1, where the a, b and c are relation-symbols of the indicated arity.

Proof. We give a recipe for computing $f(\phi)$. By a well-known result, there is a function $f' : \mathcal{L}^2 \to \mathcal{L}^2$, computable in polynomial time and linear space, such that, for all $\phi \in \mathcal{L}^2$, $f'(\phi)$ is a definitional equivalent of ϕ of the form

(36) $\forall x \forall y \chi_0 \wedge \bigwedge_{1 \leq i \leq m} \forall x \exists y \chi_i$

where the χ_i $(0 \leq i \leq m)$ are quantifier-free. So we first obtain the formula $f'(\phi)$.

All atoms in $f'(\phi)$ are of the forms $b(x,y)$, $b(y,x)$, $b(x,x)$, $b(y,y)$, $b(x)$, and $b(y)$. We process $f'(\phi)$ so that all atoms appearing in the χ_i $(0 \leq i \leq m)$ are of the form $b(x,y)$. (If b is the identity relation, the necessary manipulations are trivial, so we may ignore this case in what follows.) We proceed by replacing any atom not in the desired form with an atom $a(x,y)$, where a is a new binary relation, and by adding conjuncts as specified in the following table:

Form of replaced atom	Added conjuncts
$b(y,x)$	$\forall x \forall y(a(x,y) \to b(y,x))$, $\forall x \forall y(b(x,y) \to a(y,x))$
$b(x,x)$	$\forall x(\exists y a(x,y) \to b(x,x))$, $\forall x(b(x,x) \to \forall y a(x,y))$
$b(x)$	$\forall x(\exists y a(x,y) \to b(x))$, $\forall x(b(x) \to \forall y a(x,y))$
$b(y,y)$	$\forall x(\exists y a(y,x) \to b(x,x))$, $\forall x(b(x,x) \to \forall y a(y,x))$
$b(y)$	$\forall x(\exists y a(y,x) \to b(x))$, $\forall x(b(x) \to \forall y a(y,x))$

Let the result of this replacement process be denoted by $f''(\phi)$. It is clear that f'' is computable in polynomial time and linear space, and that $f''(\phi)$ is a definitional equivalent of $f'(\phi)$ and hence of ϕ.

We next replace the smallest non-atomic subformulas appearing in $f''(\phi)$ by atoms of the form $a(x,y)$, where a is a new binary relation symbol; and we add conjuncts according to the following table:

Form of subformula	Added conjuncts
$\neg b(x,y)$	$\forall x \forall y(a(x,y) \to \neg b(x,y))$, $\forall x \forall y(\neg b(x,y) \to a(x,y))$
$b(x,y) \wedge c(x,y)$	$\forall x \forall y((b(x,y) \wedge c(x,y)) \to a(x,y))$, $\forall x \forall y(a(x,y) \to b(x,y))$, $\forall x \forall y(a(x,y) \to c(x,y))$

We then simply take $f(\phi)$ to be the result of applying the above replacement process to exhaustion. It is clear that f is computable in polynomial time and linear space, that $f(\phi)$ is a definitional equivalent of $f''(\phi)$ and hence of ϕ, and moreover that $f(\phi)$ is of the form required by the lemma. □

Theorem 3. *Let ϕ be in $\mathcal{L}^2$. Then we can compute, in polynomial time and linear space, a set E of sentences in E2V, such that E translates to a set of sentences whose conjunction is a definitional equivalent of ϕ.*

Proof. Compute $f(\phi)$ as in lemma 8. The formulas in the left-hand column of table 1 are, modulo some trivial logical manipulation, the translations of the corresponding E2V sentences in the right-hand column. (This fact has been verified using the Prolog implementation mentioned in section 2.4.) Thus, we can read the E2V translations of each conjunct in $f(\phi)$ from table 1. $\square$

At this point, we are in a position to return to remarks made in section 2.3 concerning possible restrictions of (or modifications to) E2V. In particular, we claimed that imposing standard pronoun accessibility restrictions (thus outlawing certain currently legal E2V sentences) would not reduce the fragment's expressive power. That this is so follows immediately from consideration of table 1, since all the sentences occurring there are perfectly acceptable English sentences in which anaphora can be resolved intrasententially, and so could not possibly violate any (reasonable) accessibility restrictions.

Similarly, we claimed in section 2.3 that banning some very awkward negative sentences or insisting on the use of negative polarity determiners would also not affect the expressive power of E2V. But we see that the only faintly objectionable piece of English in table 1 occurs in the row

$\forall x \forall y(a(x,y) \rightarrow \neg b(x,y))$	Nothing bees something which it ays

where, arguably, something should be replaced by anything. But any reasonable modification of E2V along these lines must surely assign the meaning $\forall x \forall y(a(x,y) \rightarrow \neg b(x,y))$ to the sentence Nothing bees anything which it ays, in which case, the proof of theorem 3 would proceed as before. Thus, no reasonable restriction of negative sentences in E2V could lead to a reduction in expressive power.

The following well-known result gives us a lower complexity bound for E2V-satisfiability:

Theorem 4. *The problem of deciding the satisfiability of a formula in $\mathcal{L}^2$ is NEXPTIME-hard.*

Sketch proof. We use a slight modification of the technique used in Börger, Gurevich and Grädel [2], corollary 6.2.13. Read the predicates $X_i(z)$ $(1 \leq i \leq n)$ as "the i-th binary digit of the x-coordinate of square z is 1", and $Y_i(z)$ $(1 \leq i \leq n)$ as "the i-th binary digit of the y-coordinate of square z is 1". We can then fix the interpretation of the predicates $X_i^*(z)$ $(1 \leq i \leq n)$ to be "all binary digits up to (not including) the i-th in the x-coordinate of square z are 1", and similarly for the Y_i^*. Then the formula

$$\bigwedge_{1 \leq i \leq n} X_i(z') \leftrightarrow (X_i(z) \oplus X_i^*(z)),$$

where $\oplus$ indicates exclusive-or, states that the x-coordinate of square z' is 1 greater than the x-coordinate of z. A similar relation can be defined for the y-coordinates.

In this way, it is possible to encode a grid of size 2^n using an $\mathcal{L}^2$-formula whose size is polynomial in n. It is then routine to determine, for a given nondeterministic Turing machine M, a formula ϕ_M whose size is polynomial in n, such that ϕ_M states that there is a run of M halting in no more than 2^n steps. The result then follows. □

This gives us an immediate bound for the complexity of reasoning in E2V:

Corollary 2. *The problem of determining the satisfiability of a set E of E2V sentences is NEXPTIME hard.*

In fact, if complexity (rather than expressive power) is all we are interested in, we could do much better than this. Referring to definition 7, dropping the requirement that the expansion $\mathfrak{B}$ be *unique* still guarantees that ϕ and ϕ' are satisfiable over the same domains. By taking care to move negations inwards in the χ_i in formula (36), we can get away with fewer added conjuncts in producing $f(\phi)$ from $f''(\phi)$ in the proof of lemma 8, and we find that the rows below the horizontal line in table 1 are not needed. Since there is no mention of the word not above this line, we have:

Definition 8. Denote by E2V′ the fragment of English defined in exactly the same way as for E2V, but with all grammar rules involving NegP removed.

Of course, E2V′ still contains the negative determiner no.

Corollary 3. *The problem of determining the satisfiability of a set E of* E2V′ *sentences is NEXPTIME hard.*

We remark that the formula constructed in the proof of theorem 4 does not use equality. It is then routine to show that the assumptions that thing denotes the universal property and eqs expresses the identity relation can both be dropped without invalidating corollary 3.

We end this section with a brief discussion of what happens when the artificial indexing rule **I4** is removed. In this case, additional patterns of indexing are allowed for a given string of words in E2V, resulting in increased expressive power. Consider again sentence (4), repeated here as (37),

(37) Every artist who employs a carpenter despises every beekeeper who admires him.

As we remarked above, rules **I1**–**I3** allow the pronoun to coindex with either artist or carpenter, while rule **I4** forbids the latter possibility. But what if we abolished **I4** and allowed all (intrasentential) anaphoric references conforming to the usual rules of binding theory? For sentence (37), this would result in the possible semantic value:

(38)
every(rel(some(carpenter(x_2),employ(x_1,x_2)),
artist(x_1)),
every(rel(admire(x_3,x_2),beekeeper(x_3)), despise(x_1,x_3))).

Let us apply the standard translation rules **T1**–**T5** to this semantic value. (Our translation rules in no way depend on the adoption of rule **I4**.) The result is the formula

(39) $$\begin{array}{l}\forall x_1\ \forall x_2((\text{artist}(x_1) \wedge (\text{carpenter}(x_2) \wedge \text{employ}(x_1, x_2))) \rightarrow \\ \qquad \forall x_3((\text{beekeeper}(x_3) \wedge \text{admire}(x_3, x_2)) \rightarrow \text{despise}(x_1, x_3))).\end{array}$$

We mention in passing that the Prolog implementation discussed in section 2.4 has been so written that the *first* solution it finds corresponds to an indexing pattern conforming to **I1**–**I4**, and that *subsequent* solutions, obtainable by backtracking in the normal way, are those for other indexation patterns, if any, conforming only to **I1**–**I3**.

Formula (39) is clearly not 2vc. Does this make a difference? Unfortunately, it does. In [26], the author has shown that the satisfiability problem for E2V without rule **I4** is undecidable. Actually, that paper establishes a slightly stronger result: even if all negative items (the determiner no and the rules for NegP) are removed from the grammar, without rule **I4**, the problem of determining whether one set of sentences entails another is also undecidable. The proof proceeds by reduction to the problem of determining subsumption in a variant of the description logic NIKL (Patel-Schneider [25]). The techniques used are similar to those employed in this section, though somewhat long-winded. Here, we simply state without proof:

Theorem 5. *If the indexing rule* **I4** *is removed from E2V, the problem of determining the satisfiability of a set of E2V sentences (with specified indexing patterns) is undecidable.*

We conclude that the artificial indexing rule **I4** really is essential in enforcing decidability.

5 Natural language and formal languages

Let us say that an *argument* in E2V is a finite (possibly empty) set of E2V sentences (the *premises*) paired with a further E2V sentence (the *conclusion*). Informally, an argument is said to be *valid* if the conclusion must be true whenever the premises are true. Assuming that our semantics is a faithful account of the meanings of E2V sentences, we may take an argument to be valid if and only if the translations of the premises entail the translation of the conclusion, in the usual sense of entailment in first-order logic. We take this claim to be uncontentious. Therefore, the validity of arguments in E2V is certainly decidable. The question then arises as to the most efficient practical way of determining the validity of arguments in E2V automatically.

The most obvious strategy is as follows: translate the relevant E2V sentences into first-order logic as specified in section 2; then use a known algorithm for solving satisfiability problems in the two-variable fragment. Of particular interest in regard to the latter is the procedure for determining validity in two-variable fragment *without equality* described in de Nivelle [4]. This procedure takes the form of a resolution ordering strategy, and so enjoys the obvious engineering advantage that it can be implemented as a heuristic in a general-purpose resolution theorem-prover. Furthermore, recent (unpublished) work indicates that the current restriction to the problems not involving equality can be lifted. These observations suggest that the best way to decide the validity of arguments in E2V is to pipe the output of an E2V parser to a resolution theorem-prover equipped with suitable heuristics.

Historically, however, the strategy of translating natural language deduction problems into first-order logic and then using standard logical techniques to solve them has always had its dissenters. Two observations appear to encourage this dissent: (i) that the syntax of first-order logic is unlike that of natural language—particularly in its treatment of quantification—and (ii) that standard theorem-proving techniques are unlike the kinds of reasoning patterns which strike people as most natural. It is then but a small step to the idea that we might obtain a better (i.e. more efficient) method of assessing the validity of arguments by reasoning within a logical calculus whose syntax is closer to that of natural language. This idea is particularly attractive, because, if correct, it would show that the syntax of natural language is well-adapted to the task of making deductive inference in reasonable amounts of time. Moral: treat the syntax of natural language with the respect it is due, and your inference processes will run faster.

The purest manifestation of this school of thought is to give an account of deductive inference in terms of proof-schemata which match directly to patterns of (analysed) natural language sentences. Examples of such natural-language-schematic proof systems are Fitch [7], Hintikka [12] and Suppes [29], as well as the work of the 'traditional' logicians such as Englebretsen [5] and Sommers [28]. However, it is certainly possible to accept the usefulness of translation to a formal language, while maintaining that such a language should be more like natural language than first-order logic. Thus, for example, Purdy [27] presents a language he calls $\mathcal{L}_N$, a variable-free syntax with slightly less expressive power than ordinary first-order logic. (Satisfiability in $\mathcal{L}_N$ remains undecidable, however.) Purdy provides a sound and complete proof procedure for $\mathcal{L}_N$ and a grammar mapping sentences from a fragment of English to formulas in $\mathcal{L}_N$. In addition, he shows how his proof procedure can be used to solve various deductive inference problems in—as he claims—a natural fashion. (An analogous position appears in other papers by Suppes.)

The trouble with all of these systems, however, is that no hard evidence is adduced to support their claimed superiority to the obvious alternative of reasoning with first-order logic translations. The work of Purdy just cited provides a good illustration. It is clear that the very simple natural language fragment he provides does not have the same expressive power as $\mathcal{L}_N$. Indeed, Purdy's English fragment is, with some trivial additions, less expressive than our E2V, and linguistically rather unsatisfactory (for example, relative clauses can only have subject- and never object-traces). It is worth remarking that the example inference problems he gives to illustrate the 'naturalness' of his proof procedure cannot be parsed by his grammar. We conclude that the case for $\mathcal{L}_N$ as an appropriate formalism for solving natural language reasoning problems has not been established. The same goes for all of the alternative schemes mentioned above: vague and unsubstantiated claims about the psychological naturalness of certain reasoning procedures have little merit.

McAllister and Givan [20] present a much more restricted logical language involving a construction similar to the 'window' operator of Humberstone [15, 16], and specifically motivated by the quantification patterns arising in simple natural language sentences. Determining satisfiability in this language is shown to be an NP-complete problem—and indeed to be solvable in polynomial time given certain restrictions. (We remark that the computational complexity of similar, but more expressive, languages is explored in Lutz and Sattler [18].) McAllister and Givan do not present a grammar corresponding to their fragment, though it would not be difficult to write a parser for simple sentences involving nouns (common and proper), transitive verbs, relative clause constructions and the determiners some and every. Whether any linguistically natural fragment of natural language could be given which expresses the whole of McAllister and Givan's formal language is unclear. However, McAllister and Givan's work has the great merit of establishing a precise claim about the computational advantage of restricting attention to their natural-language-inspired formalism.

The complexity results presented above show that no fragment of English translating into McAllister and Givan's formalism could equal the expressive resources of E2V. As McAllister and Givan point out, they seem to have captured a fragment of English from which all anaphora has been removed. Moreover, our results give us reason to believe that *no* analogous natural-language-inspired formalism could confer any computational advantages when it comes to fragments of natural language as expressive as E2V. Section 3 establishes that we can determine the validity of E2V arguments in nondeterministic exponential time by adopting the straighforward strategy of translating the relevant sentences into first-order logic; however, section 4—and specifically table 1—tells us that determining the validity of arguments in E2V is NEXPTIME-hard anyway, so that it is difficult to see how an alternative representation language would confer any computational benefit, at least in terms of (worst case) complexity analysis.

True, complexity analysis is a blunt instrument: and it might be that an alternative formalism—one more in harmony with the structure of natural language—yields a more efficient reasoning procedure for deductive problems couched in natural language. But quite apart from the difficulty of substantiating this claim, practical considerations will inevitably make it difficult for any reasoning procedure developed for the alternative formalism to compete with the impressive array of well-maintained and -documented software already available for theorem-proving in first-order logic. For these reasons, we remain skeptical as to whether formal languages whose syntax is inspired by natural language—or whose syntax just deviates from first-order logic in some other way—really constitute more efficient representation languages for natural-language deduction than does first-order logic. McAllister and Givan substantiated such a claim for some anaphora-free fragments of natural language; but our results suggest that, as soon as even minimal anaphora is introduced, these advantages vanish.

6 Conclusion

This paper has provided a study in how to match a controlled language with a decidable logic whose computational properties are well-understood. The controlled language we chose, E2V, was shown to correspond in expressive power to the two-variable fragment of first-order logic. Two features of this study deserve emphasis. The first is logical rigour: the syntax and semantics of E2V were presented in such a way that results about its expressive power and computational complexity could be established as theorems. Logical rigour is important in the context of controlled languages, because software support for such languages must be shown to be robust and reliable. The second feature is *conservativity*: our presentation of the syntax and semantics of E2V borrowed heavily from accepted linguistic theory (especially in the treatment of anaphora); and our chosen logical representation language was—in contrast to some previous work on natural language deduction—standard first-order logic. Conservativity is important, because of the obvious benefits of relying on well-attested linguistic theories and well-maintained, efficient theorem-proving software.

The ultimate goal of this research is to provide useable tools for natural language system specification. Before that goal is achieved, many questions remain to be answered, not least questions of a psychological nature concerning the practical utility of such tools. However, the work reported here is at least a step towards this goal. At the very least, we have demonstrated that work in formal semantics, mathematical logic and computer science has now reached the stage where relatively expressive controlled languages can be precisely specified and their computational properties rigorously determined.

References

[1] Hajnal Andréka, Johan van Benthem, and István Németi. Modal languages and bounded fragments of predicate logic. *Journal of Philosophical Logic*, 27(3):217–274, 1998.

[2] Egon Börger, Erich Grädel, and Yuri Gurevich. *The Classical Decision Problem*. Perspectives in Mathematical Logic. Springer-Verlag, Berlin, 1997.

[3] Elizabeth A. Cowper. *A Consice Introduction to Syntactic Theory*. University of Chicago Press, Chicago, 1992.

[4] Hans de Nivelle. An overview of resolution decision procedures. In Martina Faller, Stefan Kaufmann, and Marc Pauly, editors, *Proceedings of the 7th CLSI workshop on Logic, Language and Computation*, Stanford, CA, 1999. CSLI Publications.

[5] George Englebretsen. *Three Logicians*. Van Gorcum, Assen, 1981.

[6] A. Fantechi, S. Gnesi, G. Ristori, M. Carenini, M. Vanocchi, and P. Moreschini. Assisting requirement formalization by means of natural language translation. *Formal Methods in System Design*, 4:243–263, 1994.

[7] Frederic B. Fitch. Natural deduction rules for english. *Philosophical Studies*, 24:89–104, 1973.

[8] N. E. Fuchs, U. Schwertel, and S. Torge. Controlled natural language can replace first-order logic. In *14th IEEE International Conference on Automated Software Engineering*, pages 295–298. IEEE Computer Society Press, 1999.

[9] Norbert Fuchs, Uta Schwertl, and Rolf Schwitter. Attempto controlled english–not just another logic specification language. In Pierre Flener, editor, *Logic-based Program Synthesis and Transformation*, Lecture Notes in Computer Science 1559, pages 1–20. Springer Verlag, Berlin, 1999.

[10] Erich Grädel. On the restraining power of guards. *Journal of Symbolic logic*, 1999.

[11] Erich Grädel and Martin Otto. On logics with two variables. *Theoretical Computer Science*, 224(1–2):73–113, 1999.

[12] Jaakko Hintikka. Quantifiers vs quantification theory. *Inquiry*, 5:153–77, 1974.

[13] Alexander Holt. Formal verification with natural language specifications: guidelines, experiments and lessons so far. *South African Computer Journal*, 24:253–257, 1999.

[14] Alexander Holt and Ewan Klein. A semantically-derived subset of English for hardware verification. In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics*, pages 451–456. Association for Computational Linguistics, June 1999.

[15] I. L. Humberstone. Inaccessible worlds. *Notre Dame Journal of Formal Logic*, 24(3):346–352, 1983.

[16] I. L. Humberstone. The modal logic of all and only. *Notre Dame Journal of Formal Logic*, 28(2):177–188, 1987.

[17] Hans Kamp and Uwe Reyle. *From Discourse to Logic: Introduction to Modeltheoretic Semantics of Natural Language, Formal Logic and Discourse Representation Theory.* Kluwer Academic Publishers, London, Boston, Dordrecht, 1993. Studies in Linguistics and Philosophy, Volume 42.

[18] Carsten Lutz and Ulrike Sattler. The complexity of reasoning with boolean modal logics. Technical Report LTCS-00-02, Aachen University of Technology, Research Group for Theoretical Computer Science, Aachen, 2000.

[19] Benjamin Macias and Stephen Pulman. A method for controlling the production of specifications in natural language. *The Computer Journal*, 38(4):310–318, 1995.

[20] David A. McAllester and Robert Givan. Natural language syntax and first-order inference. *Artificial Intelligence*, 56:1–20, 1992.

[21] M. Mortimer. On languages with two variables. *Zeitschrift für mathematische Logik und Grundlagen der Mathematik*, 21:135–140, 1975.

[22] Reinhard Muskens. Combining Montague semantics and discourse representation. *Linguistics and Philosophy*, 19(2):143–186, 1996.

[23] Rani Nelken and Nissim Francez. Translating natural language system specifications into temporal logic via DRT. Technical Report LCL-96-2, Laboratory for Computational Linguistics, Department of Computer Science, Technion, Israel Institute of Technology, 1996.

[24] Pacholski, Szwast, and Tendera. Complexity results for first-order two-variable logic with counting. *SICOMP: SIAM Journal on Computing*, 29, 1999.

[25] Peter Patel-Schneider. Undecidability of subsumption in NIKL. *Artificial Intelligence*, 39:263–272, 1989.

[26] Ian Pratt-Hartmann. On the semantic complexity of some fragments of English. Technical Report UMCS-00-5-1, University of Manchester Department of Computer Science, Manchester, 2000.

[27] William C. Purdy. A logic for natural language. *Notre Dame Journal of Formal Logic*, 32(3):409–425, 1991.

[28] Fred Sommers. *The Logic of Natural Language*. Clarendon Press, Oxford, 1982.

[29] Patrick Suppes. Logical inference in English: a preliminary analysis. *Studia Logica*, 38:375–391, 1979.

[30] Sunil Vadera and Farid Meziane. From English to formal specifications. *The Computer Journal*, 37(9):753–763, 1994.

[31] Adrian Walker, Michael McCord, John F. Sowa, and Walter G. Wilson. *Knowledge Systems and Prolog: A Logical Approach to Expert Systems and Natural Language Processing*. Addison Wesley, Reading, Mass., 1987.