Procedural dynamic semantics, verb-phrase ellipsis, and presupposition

D.J.N. van Eijck, N. Francez

Computer Science/Department of Software Technology

Report CS-R9311 February 1993

Procedural Dynamic Semantics, Verb-Phrase Ellipsis, and Presupposition

Jan van Eijck[1,2] & Nissim Francez[3]

[1] *CWI, P.O. Box 4079, 1009 AB Amsterdam, The Netherlands*

[2] *OTS, Trans 10, 3512 JK Utrecht, The Netherlands*

[3] *Department of Computer Science, The Technion, Haifa, Israel*

Abstract

In this paper, we study Verb-phrase Ellipsis (VPE) and show its relationship to presupposition and its failure. In doing so, we use a novel representation of VPE by means of a dynamic semantics approach, using an extended version of Dynamic Predicate Logic which includes procedures, scope rules and functions. We focus on the representation of the strict/sloppy ambiguity in connection with ellipsis constructs, which comes "for free" in a procedural context. We also provide independent justification for our representation of VPE in terms of procedure definitions and their invocations. The relationship of VPE and presupposition also sheds some new light on the arguments about the question whether VPE is syntactic or semantic in nature, as well as on the relative order of meaning determination and presupposition determination.

1991 CR Categories: I.2.7.

Keywords and Phrases: Dynamic semantics, Verb-phrase ellipsis, Strict/Sloppy ambiguity, Presupposition, Procedures, Static and Dynamic binding.

1 Introduction

In this paper, the relationship between Verb-phrase Ellipsis (henceforth abbreviated to VPE) and presupposition—and in particular presupposition failure—is studied. As far as we know, no such relationship has been considered before in the literature. Note though, that the idea to look at VPE as procedure invocation is not new (it can be found in [7]). We show that the famous ambiguity in VPE, namely the distinction between *strict* reading vs. *sloppy* reading, is related to presuppositions and their failure. We formalize our theory using an extension of *Dynamic Predicate Logic* (*DPL*) [8], as well as its axiomatization via *quantified dynamic logic* (*qdl*), both as presented in [4, 5]. For that purpose, both formalisms have to be extended. The extension we propose includes the following constructs:

Report CS-R9311
ISSN 0169-118X
CWI
P.O. Box 4079, 1009 AB Amsterdam, The Netherlands

- Procedures with simple parameters (without recursion)
- Scope rules for free variables in procedures (dynamic vs. static scoping)

The full formal development of these extensions appears in a companion paper [6]. Here, we rely on the intuitive understanding of these concepts as they are used in programming languages. The examples provide a level of detail sufficient for understanding the arguments made.

We would like to stress one methodological issue, not directly related to presuppositions and their relation to VPE. Incorporating procedures into a dynamic semantics account of VPE has a two-fold advantage:

1. It provides a succinct representation of VPE which naturally accounts for the famous ambiguity of strict/sloppy readings of certain VPEs, and moreover, it does so in terms of scope-rules that are well understood in programming languages semantics. The binding mechanisms involved turn out to have an independent justification in a broader context, rather than being an ad-hoc construction for VPE representation only.
2. It provides new grounds for a possible answer to a question often raised in the literature, where it tends to generate a heated argument:

   > Is VPE resolution a *syntactic* process, depending on syntactic reconstruction of unrealized syntactic material, or is it a *semantic* process, *directly* interpreting the ellipsis based on some semantic representation of the realized syntactic material?

   Two recent papers, taking opposite views on this issue, are [2] for the semantic view and [10] for the syntactic view.

   A question often asked by the proponents of the syntactic view, raised as an argument against the semantic view, is the following:

   > How can a semantic representation be assigned to an unrealized syntactic construct?

   In the debate, this question is a rhetorical one, of course. On the syntactic view, this can simply not be done. According to this syntactic view, it is mandatory first to reconstruct the unrealized syntactic material by a process of 'copy and paste', and then to apply ordinary semantic interpretation to the reconstructed material.

   Well, procedures and their invocations (in programming languages) are an example *par excellence* of a construct exhibiting exactly the puzzling characteristics being questioned. A procedure is a *one definition - multiple use* construct. The procedure definition is assigned some meaning by the semantics of the programming language. Then, *every* single invocation is *directly interpreted*: it inherits the meaning of the procedure, possibly adapted in accordance with parametrization and scope considerations. The view that reconstruction always takes place before interpretation of a procedure, would correspond to a *macro expansion* view of procedure invocation. This view prevailed in the early days of programming languages, but has yielded long ago to the perspective of direct interpretation with its many possibilities for richer semantic contents (e.g., recursion).

   Thus, our development can be seen as a re-enforcement of the semantic view of VPE. On the other hand, we concede to the defenders of the syntactic view that there is a key difference between procedure invocation in programming and VPE in natural language. While in programming it is always clear which procedure is being invoked, an elliptical verb phrase in natural language refers to a 'procedure' that has to be reconstructed from the previous context.

In Section 2 we explain the way in which VPE can be modelled by means of procedures (in DPL) and their invocations, and the way in which the strict/sloppy ambiguity manifests itself in the dynamic vs. static binding distinction. In Section 3 we consider the relationship between VPE and presupposition, and the way this relationship is captured by our representation.

2 A Procedural Representation of VPE

One of the most basic observations regarding VPE is the ambiguity between *strict readings* and *sloppy readings*, in the presence of an anaphoric pronoun. Consider example 1

1 John loves his wife and Bill does too.

Most of the examples of VPE consist of a conjunctive coordination, where the full conjunct (usually the first one) is the *source* clause, while the ellided conjunct (usually the second one) is the *target* clause. Under the strict reading, the interpretation of the target clause is that Bill loves John's wife, while under the sloppy reading the interpretation is that Bill loves his own wife.

As already mentioned, a question which is often raised in the literature is, what is the source of this ambiguity? This question is also related to the question whether VPE is a syntactic or a semantic phenomenon. Below, we indicate how our representation improves on the solution to these issues too, as a side effect of correctly accounting for presuppositions in VPE.

In the first accounts of VPE, where VPE is viewed as mainly a syntactic phenomenon, the ambiguity was attributed to the source clause [14]. Thus, for the example above, two different logical forms were assigned to John loves his wife, each yielding a different target clause interpretation when completing the target clause accordingly:

$$\lambda x : \ \mathit{love}\ (x,\ \mathit{the\text{-}wife\text{-}of}\,(j))(j)$$
$$\lambda x : \ \mathit{love}\ (x,\ \mathit{the\text{-}wife\text{-}of}\,(x))(j)$$

Note that in the assumed LF language, we have represented the possessive by means of a term with a function symbol. In our extended DPL language, to be used in the sequel, we will use ι assignments (commands to assign an individual that uniquely satisfies a given property to a variable) for this purpose. Note that we represent NL verb phrases or common nouns R with corresponding symbols R in the representation language.

In more recent approaches, where the semantic nature of VPE is stressed, the ambiguity is *not* attributed to the source clause; rather, the interpretation process itself yields multiple interpretations for the target clause. A notable example is the VPE resolution via *higher-order* *unification* [3], by which two different solutions for an equation (derived from the source clause) are obtained, each inducing a different interpretation of the target clause. In both cases, there is an intimate relationship between the strict/sloppy dichotomy and the referring anaphora/bound variable anaphora dichotomy.

In the approach advocated in this paper, the ambiguity is pushed even further. We generate a *unique* representation of the target clause in the form of a procedure in (the extended) *DPL*, this representation itself having multiple interpretations, according to the policy of binding free (global) variables in procedure bodies. What is known (in the study of programming languages) as *static binding*, whereby free variables are bound at the level of the definition of the procedure, will yield the strict reading; on the other hand, *dynamic binding*, whereby free variables are bound at the level of *procedure invocation*, yields the sloppy interpretation. Of course, we now have to explain why in certain cases only one of these readings is available. Our account is in line with the idea that the strict/sloppy identity should be analyzed in terms of the difference between binding in a global or local environment; see the concluding remarks of [7].

In passing, we note that interpreting the procedure as even more global (to both the defining environment and to the invoking environment) generates the *deictic* interpretation of example 1, accounting for the (well-known) fact that the pronoun may refer deictically to the *same* referent in both clauses.

We note here, that in the realm of programming languages, the procedure construct has two major characteristics:

Abstraction: The procedure encapsulates the details of representation.

Single definition - multiple use: The procedure is defined once only, but activated (possibly parametrized) as often as needed.

It is especially this second characteristic that suggests representing VPE by means of procedures and their invocations. Ellipsis can be viewed as a way of referring more than once to a construct, though in contrast to what is the case in programming languages, surface structure contains no *explicit* definition of the construct. The definition of an appropriate procedure has to be derived from the first use of the procedure, so to speak, during interpretation, and included in the semantic representation.

By having the ambiguity (when present) associated with the (unique) representation of the target clause, we achieve a modelling of the semantic representation in a way much closer to the surface structure of the sentence, which we view as an advantage over previous accounts. This improvement is in line with the whole of the modern approach to natural language semantics, generically referred to here as *dynamic semantics*, by which semantic methods that proved themselves in defining programming languages are successfully adapted to natural language. Our approach extends the scope of programming language constructs which induce representations for natural language constructs, by including procedures, scope rules and binding rules among the former.

Suppose we want to construct a representation for example 1 above. Without entering here into the details of a *systematic* translation from NL to DPL (see [1, 11] for that), we describe the resulting representations only.

First, from the source clause the following procedure definition[1] is obtained by abstraction[2]:

$$p(x): \iota z: \ \textit{wife-of}\,(y,\ z);\ \textit{love}\ (x,\ z)$$

The procedure has one formal parameter, x, representing the subject. The procedure has also one *global (i.e., free)* variable y, the binding of which depends on the binding semantics chosen, as explained below. This global variable represents the pronoun in the source sentence. Finally, the procedure has also a *local* (i.e., bound) variable z, for which definiteness is represented by means of the ι operator [4], a representation internalizing the functional connection, our representation of possessives. This local variable is also the representation of the object of the clause.

Next, we get the following representation for the source clause.

$$\textbf{new}\ y;\ y := j;\ p(j)\ \textbf{end}$$

In this representation, a new scope unit (commonly called a *block* in programming languages) is created, declaring a variable y local to that unit. Then the constant j (denoting John) is assigned to y, signaling its exposure to anaphoric references, much as a discourse marker does in DRT [9]. Finally, the procedure $p(x)$ is invoked with an actual parameter j. The parameter-passing semantics assumed here is the simplest one, directly binding the formal parameter (x in this case) to the value of the actual parameter (j in this case). We do not bother with the full power of parametrization present in programming languages. Next, a representation for the target clause is constructed, very much like that source clause, with the sole change of the constants involved.

$$\textbf{new}\ y;\ y := b;\ p(b)\ \textbf{end}$$

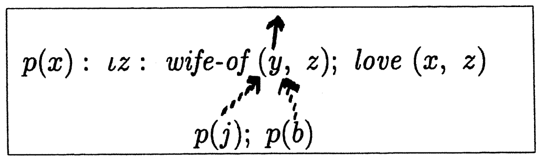
Here the constant b is assumed to denote Bill. This similarity of representation between the two clauses is also separately motivated by the need to form *cascaded VPE* (an example appears in the next section).

We now turn to a recovery of the strict/sloppy ambiguity of this example. We distinguish between two cases, inducing two different evaluations of procedure invocations in general, and of $p(b)$ (the example target clause representation) in particular. For the sake of the exposition, let us denote by y_0, y_1 and y_2 the external variable y (existing outside the representation of the whole VPE), the y of the source clause and the y of the target clause, respectively.

Static binding: In this case, the global variable y of the procedure body is taken as y_1, as the binding takes place at the point of the *definition* of the procedure, and therefore y is still referring to j, the value of y_1 obtained by the assignment in the representation of the source clause. The second assignment, that of b to y_2 (within the target clause representation) remains unsensed, so to speak, not having an effect on the outcome of the procedure invocation. Thus we get the strict reading.

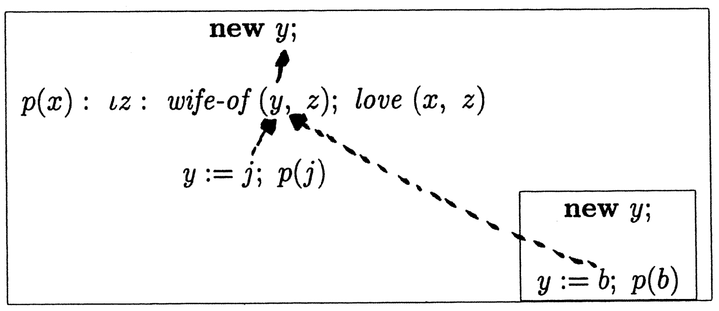
[1]Note that we switch here to a ι-representation for functionality, for which an error-state semantics, needed for calculating presuppositions, is provided in [4].

[2]The actual mechanism of abstraction involves the source clause, the target clause(s) and parallelism among their components, very much like the higher-order unification mechanism used to solve equations derived from the clauses in [3]. For the use of parallelism in explaining empirical facts about VPE see [12].



$p(x):\ \iota z:\ \textit{wife-of}\,(y,\ z);\ \textit{love}\ (x,\ z)$
$p(j);\ p(b)$

Figure 1: The external binding -deictic reading



new y;
$p(x):\ \iota z:\ \textit{wife-of}\,(y,\ z);\ \textit{love}\ (x,\ z)$
$y := j;\ p(j)$
new y;
$y := b;\ p(b)$

Figure 2: The static binding - strict reading

Dynamic binding: Here, the global y of the procedure body is rebound to y_2, as the binding takes place at the point of invocation, referring now to b, the value of y_2 obtained via the assignment to y immediately prior to the procedure invocation. We thus get the sloppy reading.

We may summarize the discussion by the graphic representation of Figures 1 through 3.

In Figure 1 we see the external binding of the free variable y; independently of whether binding is static or dynamic this yields a deictic reading, interpreting 'his' in the source and target clause as the value of y_0, the current denotation of y. Note that since no "local" anaphoric references for this pronoun are intended, no local scope unit is created (and no assignments to y occur prior to procedure invocation; cf. next two figures).

In Figure 2 we have the static binding, where in both procedure invocations y is bound in the same way. By a calculation (relying on the formal semantics not given here), the invocation $p(b)$ gets a meaning equivalent to that of

$$\iota z:\ \textit{wife-of}\,(j,\ z);\ \textit{love}\ (b,\ z)$$

Finally, in Figure 3 the dynamic binding is assumed, binding the invocation $p(b)$ to the innermost y. Here the meaning of the invocation $p(b)$ turns out to be equivalent to

$$\iota z:\ \textit{wife-of}\,(b,\ z);\ \textit{love}\ (b,\ z)$$

as expected.

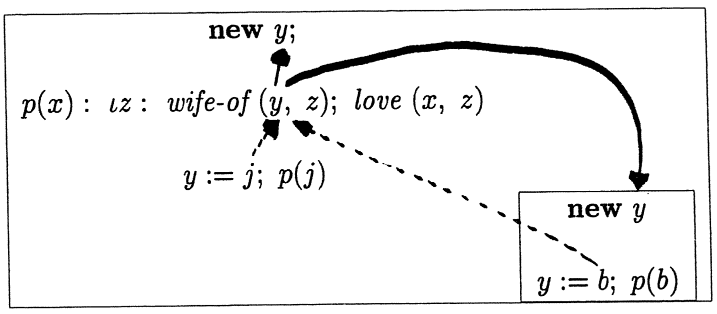


Figure 3: The dynamic binding - sloppy reading

Note that in the simpler cases where no free pronoun is used in a VPE, our representation produces procedures without global (i.e., free) variables in their body. This co predicts the unambiguity in the interpretation of such simple VPEs. Thus, consider

2 John loves Mary and Bill does too

The resulting abstracted procedure is

$$p(x) :\ love\ (x,\ m)$$

The representation of the two clauses is, as before,

$$\textbf{new}\ y;\ y := j;\ p(j)\ \textbf{end}$$

and

$$\textbf{new}\ y;\ y := b;\ p(b)\ \textbf{end}$$

respectively. However, since the body of this procedure has no free occurrence of y, there is no difference in meaning resulting via a difference in the binding semantics. These redundant assignments can be optimized away (similar to simplification obtained in Montague semantics by applying λ-reductions).

Note that the wholle representation here is oversimplified in a certain way, for the sake of not complicating the discussion by orthogonal issues. The problem is the locality of the *object* variable (to the procedure body), rendering this object unaccessible to anaphoric reference outside the procedure. In reality, such anaphoric references are possible, very similarly to subject accessibility. Thus, consider

3 John loves his wife. Bill does too. She is beatiful.

Clearly, 'she' refers here anaphorically to the object of the second sentence in the sequence above. It seems that in the preferred reading of the whole sequence, the use of 'she' is disambiguating, imposing a strict interpretation of the VPE. The less preferred interpretation is the sloppy reading of the VPE, where 'she' refers to the object of the second sentence. We believe that the sloppy reading where 'she' refers to John's wife is unavailable.

$$\boxed{\begin{array}{l} \qquad\qquad\qquad\qquad \mathbf{new}\ y;\ v; \\ p(x,\ u) : \iota z :\ \textit{wife-of}\,(y,\ z);\ \textit{love}\ (x,\ z);\ u := z; \\ \qquad\qquad\qquad y := j;\ p(j;\ v) \\ \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \boxed{\begin{array}{c} \mathbf{new}\ y,\ v \\ y := b;\ p(b,\ v) \end{array}} \end{array}}$$

Figure 4: Object representation

$$\boxed{\begin{array}{l} \qquad\qquad\qquad \mathbf{new}\ y; \\ p(x) : \iota z :\ \textit{wife-of}\,(y,\ z);\ \textit{love}\ (x,\ z) \\ \qquad\qquad\qquad y := j;\ p(j) \\ \qquad\qquad\qquad\qquad\qquad\qquad \boxed{\begin{array}{c} \mathbf{new}\ y \\ \neg;\ (\ \eta u :\ \textit{man}(u);\ y := u;\ \neg p(u)) \end{array}} \end{array}}$$

Figure 5: Quantified NP representation

To accomodate the accessibility of the object in the procedural interpretation, a certain extension is needed. We augment the procedure with another formal parameter, of a **result** type, to which the local object is assigned, and hence accessible outside the procedure. The full representation has the form as in Figure 4.

The value of v is the object. Since the second assignment to v overides the first one, this explains the ruling out of the anaphoric reference of 'she' to John's wife in the sloppy case.

For simplicity, we remain for the rest of the paper with the original, simplified representation.

There are several situations, mentioned in the literature, in which one of the two interpretations of VPE is *blocked*, resulting in an enforced disambiguation. In the next section, we identify *presupposition failure* as one source of blocking VPE ambiguity (not considered hitherto in the literature). Most of the reasons for blocking mentioned are either syntactic, or semantic and related to licensing anaphoric reference, e.g., to quantified NPs, or to negation. We show how our procedural representation handles those situations. For example:

4 Every man loves his wife and John does too

Here only the sloppy reading is available, due to the impossibility of an anaphoric reference to the quantified NP. The representation of the source clause for the quantified case is as indicated in Figure 5.

Dynamic negation is not present in conventional programming notation. Here, we adopt its common destructive use in dynamic semantics: the value of u, and hence of y, *does not exist* outside the scope of the negation. Hence, any attempt to impose a strict reading would result in attempting to access the value of an uninitialized variable, thereby ruled out.

Another example is

5 John despises himself and Bill does too

Here too only a sloppy reading is available, due to the fact that the source clause has a reflexive pronoun instead of an ordinary one. In the interpretation process, the recognition of a reflexive pronoun results in a different procedure being the result of the abstraction. We get

$$p(x):\ love\ (x,\ x)$$

This procedure also has no free occurrence of a variable in its body. Thus, in the target clause representation

$$\textbf{new}\ y;\ y := b;\ p(b)\ \textbf{end}$$

the assignment to y has no effect, and the meaning obtained is equivalent to

$$love\ (b,\ b)$$

the expected sloppy one.

A different kind of blocking is presented in [13, 12], where the discourse structure, and in particular the *parallelism* induced by it, blocks one of the readings (or more, in case of the greater ambiguity arising in discourse-related VPE).

6 John likes his hat. Fred likes it too, and Susan does too

Here only the strict reading is available for the third clause, because of the chained ellipsis, whereby the second clause (which has no sloppy reading) enforces the strictness on the third clause (which would be ambiguous if following the first clause directly). The details here are similar the the previous example where overiding of assignments took place.

A genuine limitation of our representation is its inability to handle a certain kind of *plural* pronouns. Consider

7 John loves his wife. Bill does too. They are faithful husbands

It would be impossible to generate a reference to 'they' since nowhere are *both* subjects *coaccessible.* A more significant extension is needed to cope with this problem, left for a different opportunity.

As a final note about procedural interpretation, we would like to point out another advantage it has, independently of VPE representation. Consider the sentence

8 *Mary avoids herself

This sentence is considered unacceptable because 'avoid' is assumed to require non-coreferring arguments (at least for a simple context like the example). This would be reflected in our representation as a restriction (derived from lexical information) on procedural abstractions derived from 'avoid'. They have to have the form

$$p(x): \ avoid\ (x,\ y)$$

with the condition $x \neq y$, a condition that prohibits procedure invocations violating it. This condition and the blocking imposed by it are commonly known as the *no aliasing* condition for procedures in programming languages. It is often assumed to obtain simpler program verification. Thus, we see again a connection to a familiar phenomenon, independently justified. By the way, it is very tempting to regard this condition as a (lexical) presupposition of 'avoid' (and its procedural representation), rather than part of its meaning.

3 VPE and Presupposition

In this section we present an informal account of the relationship between VPE and presupposition (and its failure) as we view it. We briefly recapitulate some of the main background issues of presupposition theory, as needed for presenting our views. The basic fact about them can be introduced with the following example.

9 John loves his wife

As is well-known, in order to assign a truth value to 9, the following *presupposition* needs to hold:

10 John has a (unique) wife

In case 10 does not hold, 9 is said to be *undefined*, or to exhibit a truth-value gap. One way to set a grounds for a theory of presuppositions is to abandon two-valued logics in favoyr of three-valued logics, or partial-logics. A more recent approach [4] uses a dynamic semantics with *error states* for that purpose. We view here the treatment of presupposition as a semantic problem, in contrast to other views that consider it to belong to the pragmatics.

There are two basic issues studied in presupposition theories, as described below. Both of them will turn to need modifications to correctly apply to VPE.

Projection: The projection problem is the problem of the determination of the presupposition of a compound sentence, given the presuppositions of its components, as well as their meanings. Thus, a successful solution to the projection problem supposed to be formulated in terms of a compositional presupposition theory. Note that no information (either linguistic or other) that does not arise from the components may be used in computing the projected presupposition.

Cancellation: Here the main issue is to account for *cancellation* of presupposition depending on *contextual* information. By context here we shall mean other sentences from the discourse from which the analyzed sentence is drawn.

3.1 Presupposition Projection and VPE

When trying to calculate the presupposition of VPE sentences like 1, one immediately faces a problem: in attempting to apply whatever rule is available for projecting conjunctions, the second conjunct (the target clause) *does not have an independent presupposition*, similarly to not having an independent meaning. Two conclusions follow.

1. The determination of presupposition can not be done solely by projection. At least for VPE, a process of presupposition *inducing* (of that of the target clause by that of the source clause) takes place. Presupposition of the target clause depends on the meaning of the source clause.
2. The processes of meaning determination and presupposition determination can not be ordered, as suggested in [15] (which does not consider VPE), but have to be interleaved. As seen from the previous point, presupposition of the target clause may depend on the meaning of the source clause. However, as will be shown here, the opposite also holds: the meaning of the target clause may depend on presuppositions of the source clause.

We envisage the following way to incorporate presupposition (and its failure) in our procedural DPL representation language. A presupposition is assigned to every definition of a procedure. This procedural presupposition may be expressed by referring also to the formal parameters of the procedure, as well as to its free variables, in addition to its local variables. A consequence of this provision is explained below. Then, every specific procedure invocation is assigned its own presupposition, derived from that assigned to the procedure, possibly adapted for actual parameters and scope rules. Thus, the relationship between a procedure and its invocations are parallel both for meaning determination and for presupposition determination.

Let us first consider the situation in which no presupposition failure is involved. Consider again the representation of example 1 above.

The presupposition associated with the procedure $p(x)$ is

$$\exists! z : \ \textit{wife-of}\,(y,\ z)$$

Note that, as already stated before, the presupposition also has a free variable, y. We want this variable to be the same as that in the procedure body. We thus impose the following rule:

> A presupposition associated with a procedure is always evaluated in the same binding as the procedure itself.

In the determination of the presupposition of the invocation $p(j)$, due to the assumption that y is bound to y_1 with value j, we get the presupposition

$$\exists! z : \ \textit{wife-of}\,(j,\ z)$$

i.e., that John has a unique wife, which is what one expects.

Let us assume that this presupposition indeed holds.

Next, the interpretation and presupposition of the target clause are constructed. Under static binding, the global variable y is still bound to y_1 and has a value j. Thus, we get the *same* presupposition for the target clause as for the sour This is a characteristic of the strict reading.

Now, consider the dynamic binding. Here, assuming that y is rebound to y_2 with value b, the presupposition turns out to be

$$\exists ! z : \ \textit{wife-of}\,(b,\ z)$$

this time assuming that *Bill* has a unique wife. This shift of presupposition characterizes sloppy readings of VPE. -1z In both cases, first the presupposition of the target clause gets determined. Only then can projection take place and can the presupposition of the whole be computed. Under the strict reading, we get just the shared presupposition of the two components. Under the sloppy reading, we get (assuming a natural conjoining rule for conjunction of the two presuppositions: both John and Bill are assumed to have unique wives.

Next, we turn to presupposition failure. Suppose that the contextual information implies that Bill is a bachelor. In this case, the presupposition under the sloppy reading would fail. We stipulate that under such circumstances, the sloppy reading is blocked, and the ambiguity resolved in favour of the strict reading. Thus, the meaning may depend on presupposition failure. A question, which at this point we leave as unresolved, is what happens in case the presupposition of the source clause fails (i.e., John is a bachelor). Under the *OTAT* assumption[3] [4], namely the propagation of error states, we would expect the interpretation of the whole sentence to abort in an error state. This should be the case even if the target clause is computed as the sloppy reading, and possibly having a non-failing presupposition. A forteriori, this is the expected result for the strict reading, in which both presuppositions of the clauses fail.

The interplay between meaning determination and presupposition determination exhibited by these examples casts a serious doubt on attempts at the pragmatic handling of the latter. We see that the semantic machinery is necessary for that purpose. Still, this does not preclude that pragmatic factors may be involved too.

3.2 Presupposition Failure and Meaning Postulates

Consider another example discussed frequently in the literature:

11 Mary corrected her mother's mistake before she did

Usually, the discussion focuses on the strict/sloppy interpretation of the pronoun 'her', where the pronoun 'she' is assumed to anaphorically refer to 'her mother'. Why is the reference of 'she' to 'Mary' ruled out? The answer is, that the sentential adverb 'before' is assumed to be non-reflexive. In other words, in a sentence of the form $p\ before\ q$, if time t_1 is associated with p and time t_2 is associated with q, $t_1 < t_2$ is assumed. On a reading in which 'she' anaphorically refers to 'Mary', we would get a meaning equivalent to that of

12 Mary corrected her mother's mistake before Mary corrected her mother's mistake

[3]Acronym for *Once a Thief, Always a Thief.*

and assuming that the two occurrences of 'mistake' refer to the same mistake, this would lead to a self-contradictory proposition under the non-reflexivity assumption.

In a Montegovian set-up, this assumption would be brought about by a *meaning postulate*, excluding models with reflexive interpretations of 'before'. However, it is also possible to regard this assumption as a presupposition of sentences of the above form[4]. Adopting such a point of view allows for a more uniform explanations of blocking of readings, in this case for VPE. It allows also for considering *arbitrary* models, correctly predicting the interpretation in "bad" ones as an error state, reflecting presupposition failure.

4 Conclusions

In this paper, we have inspected Verb-phrase Ellipsis and investigated its relationship to presupposition and its failure. We proposed a novel representation of VPE by means of a dynamic semantics approach, using an extended version of Dynamic Predicate Logic which includes procedures, scope rules and functions. Thus, we show that rather strong relationships between natural languages and programming languages may be pointed out, bringing to intuitively appealing ways of semantical representation. We have focused on the representation strict/sloppy ambiguity of VPE, showing that it may be viewed as resulting from the well-known distinction between static binding and dynamic binding of free variables in procedure bodies, a phenomenon often encountered in computer science. We also have provided independent justification for this representation. The main justification is in its ability to accommodate direct interpretation of constructs with non-fully-realized syntactic material., Thereby, we believe we have strengthened the case of the semantic conception of VPE interpretation and resolution.

Regarding presupposition failure, we showed that it causes disambiguation of VPEs with otherwise strict/sloppy ambiguity. Therefore, we view the relative order of meaning determination and presupposition determination as non-fixed, and we argue for a set-up where those two activities are interleaved.

We hope that this paper adds to the understanding of both VPE and presupposition, and that it shows the advantage of dynamic semantics over more traditional LF-based representations.

Acknowledgements

This work was initiated during a visit of the second author to CWI, sponsored by a grant from project NF 102/62-356 ('Structural and Semantic Parallels in Natural Languages and Programming Languages'), funded by the Netherlands Organisation for the advancement of research (N.W.O.).

At the Technion, work of the second author was partly supported by a grant from the joint fund from the Technion - IIT and Haifa university, a grant from the Technion vice-president for research (Formal semantics for natural language) and by the fund for the promotion of research in the Technion.

[4]Note that in such sentences, this presupposition is *not* projected from the component sentences p and q, but it enters the picture as a lexical presupposition of before.

References

[1] O. Bouchez, O. Istace, and J. van Eijck. A strategy for dynamic interpretation: a fragment and an implementation. Proceedings EACL 93, Utrecht, 1993.

[2] Mary Dalrymple. Against reconstruction in ellipsis. Technical Report TR, Xerox PARC, Palo Alto, CA, 1991.

[3] Mary Dalrymple, Stuart M. Shieber, and Fernando C.N. Pereira. Ellipsis and higher-order unification. *Linguistics and philosophy*, 14, 1991.

[4] J. van Eijck. Presupposition failure — a comedy of errors. Manuscript, CWI, Amsterdam, 1992.

[5] J. van Eijck. Axiomatizing dynamic predicate logic with quantified dynamic logic. In J. van Eijck and A. Visser, editors, *Logic and Information Flow.* Kluwer, Dordrecht, to appear.

[6] J. van Eijck and N. Francez. Dynamic logic and error state semantics — draft. Manuscript, CWI, Amsterdam, 1992.

[7] C. Gardent. Dynamic semantics and vp-ellipsis. In J. van Eijck, editor, *Logics in AI / European Workshop JELIA '90 / Amsterdam, The Netherlands, September 1990 / Proceedings*, Lecture Notes in Artificial Intelligence 478, pages 251–266. Springer Verlag, 1991.

[8] J. Groenendijk and M. Stokhof. Dynamic predicate logic. *Linguistics and Philosophy*, 14:39–100, 1991.

[9] Hans Kamp. A theory of truth and semantic representation. In Jeroen Groenendijk and Martin Stokhof, editors, *Formal methods in the study of language*, pages 277 – 322. Mathematical Center, 1981.

[10] Shalom Lappin. The syntactic basis of ellipsis resolution. Technical Report TR, IBM T.J. Watson research center, Yorktown Heights, N.Y., 1992.

[11] R. Muskens. Anaphora and the logic of change. In J. van Eijck, editor, *Logics in AI / European Workshop JELIA '90 / Amsterdam, The Netherlands, September 1990 / Proceedings*, Lecture Notes in Artificial Intelligence 478, pages 412–427. Springer Verlag, 1991.

[12] Hub Prust. *On discourse structuring, VP anaphora and gapping.* PhD thesis, University of Amsterdam, 1992.

[13] Hub Prust, Remko Scha, and Martin van der Berg. A formal grammar tackling verb-phrase anaphora. Technical Report CL-91-03, Institute for Language, logic and information(ILLC), U. of Amsterdam, 1991.

[14] Ivan Sag. *Deletion and logical form.* PhD thesis, Massachusetts Institute of Technology, Cambridge, Ma., 1976.

[15] Rob A. van der Sandt and Bart Geurts. Presupposition, anaphora and lexical contents. Technical Report IWBS Report 185, IBM scientific center, Stuttgart, August 1991.