Merging without Mystery
or: Variables in Dynamic Semantics

C.F.M. Vermeulen
O.T.S. (Research Institute for Language and Speech)
Trans 10, 3512 JK Utrecht
the Netherlands*

December 1991
revised January 1993

Abstract

In this paper we discuss the treatment of variables in dynamic semantics. Referent systems are introduced as a flexible mechanism for working with variables. In a referent system we carefully distinguish the variables themselves both from the machinery by which we manipulate them—their names—and from the information that we store in them—their values. It is shown that the referent systems provide a natural basis for dynamic semantics. The semantics with referent systems is compared with the familiar formalisms in dynamic semantics, *DRT* and *DPL*.

1 Introduction to Dynamic Semantics

1.1 Dynamics

It is probably fair to say that the start of dynamic semantics was made by Kamp [15] and Heim [14], in *Discourse Representation Theory* (*DRT*) and *File Change Semantics* (*FCS*) respectively. They both develop a formal semantics of anaphora that deviates from the narrow path of traditional, static logic. Crucial for the way in which Kamp and Heim interpret anaphora is the attention in the formalism not only for the *results* of the interpretation process, but also for aspects of the process itself.

*The idea that something like referent systems are needed for the proper treatment of variables in dynamic semantics arose in discussion with Albert Visser. Jan van Eijck, Marcus Kracht, Tim Fernando and Patrick Blackburn were kind enough to comment on an earlier version of this paper.

The idea underlying both *DRT* and *FCS* is that, when we have to interpret an anaphoric expression, we have to establish a connection with an antecedent. In both formalisms the potential antecedents can be found in a representation of the result of the interpretation process so far. Interpreting an anaphor then amounts to establishing the link with the correct antecedent. In the next subsection we will give an outline of their account in a particularly elegant formulation due to Zeevat.
In subsequent developments in dynamic semantics, such as the development of *Dynamic Predicate Logic* (*DPL*) by Groenendijk and Stokhof [11], the procedural aspect of semantics has become the main point of attention. Groenendijk and Stokhof give a formulation of the ideas of Kamp and Heim in terms of *relations* on assignments: each sentence has as its meaning a relation between an assignment that is provided by the context (for example by the interpretation of previous text) and the assignments that can be obtained from the input assignment by interpreting the sentence. In this approach an interpretation can set a variable at a certain value in its output state. This output state is the input state for the interpretation of the next sentence. Thus the interpretation of one sentence can pass on a value to the interpretation of the next sentence. This is the way in which *DPL* represents the act of linking antecedent and anaphor.
The success and the elegance of Groenendijk and Stokhof's formulation and also the development of another interesting formalism with a procedural flavour, namely Veltman's [23] *Update Semantics* (*US*),[1] has led to an interest in the logic of procedures per se. For if the basic objects in our semantic universe are going to be *actions* (either represented as relations (*DPL*) or as functions (*US*)), then we should try to discover which are the general principles governing the behaviour of actions. This interest lead to various logical investigations centered around *DPL*, such as the Hoare Logic approach of van Eijck ([6], [7]), modal logical approaches, especially using two-dimensional modal logic (Blackburn and Venema [4]), and relation algebra approaches (de Rijke [20]). Similar considerations have driven the development of *Arrow Logic* by van Benthem [3]. Thus interest in the logic of actions is one important development in dynamic semantics.
However, there is another important starting point for investigations in dynamics, namely that found in Zeevat [29]. Zeevat gives a particularly elegant formulation of *DRT*. The crux of his formulation is the use of semantic objects that are context-content pairs. The context component allows us to understand the interaction with the context, the content component collects the information content. The interaction between context and content is captured by *the merger*, an operation on the pairs that tells us how to compose meanings. Thus all the dynamic effects of *DRT* get an elegant representation in an algebraic framework, without representing meanings as procedures.

[1]Veltman's system is not designed for the treatment of anaphora. It gives a dynamic treatment of modalities. The details of his system need not concern us here.

This formulation of *DRT* has led to a new look on the old idea that the implementation of the context-content distinction is very useful for formal semantic in general. It enables us to distinguish different kinds of semantic contributions and to study their interaction in an elegant way. Actually we already find this idea in Montague's *Universal Grammar* [18], and later in the work of Kaplan [17] and Stalnaker [22]. But Zeevat's presentation has led to a new outlook on it and has inspired Visser [28] to develop general techniques for a semantics with context-content pairs.
In this paper we are going to extend Zeevat's treatment in a manner that permits both the ideas underlying *DRT* and the ideas underlying *DPL* to be captured. We will present machinery that allows us to describe the interaction of variables in dynamic semantics in an elegant way, introducing the notion of a *referent system* (section 2). The motivating ideas behind our definition of referent systems lead to a natural notion of the merger of referent systems. Some abstract properties of the algebra of referent systems are presented in section 3. Then we show how they can be used in dynamic semantics (section 4). We will discuss the relation of our semantics with both *DRT* and *DPL* (section 5). It will be argued that the use of referent systems is philosophically preferable to the treatment of variables in both systems. We will also argue that a clear implementation of the distinction between the different roles that variables play in semantics — as part of the context component on the one hand, but also as a carrier of information content on the other hand — will make the system more robust: it will allow us to make adjustments in one of these areas without disturbing the balance in the other compartment. Finally there is some general discussion on the use of variables in the semantics of anaphora (section 6).
But before we do all this we need to look at Zeevat's proposal in some detail.

1.2 Contexts and Contents

In Zeevat's formulation of *DRT* we find a discourse representation language in which discourse fragments are represented as pairs.[2] These pairs are the *discourse representation structures* (*DRSs*). The first component of a *DRS* consists of a number of *discourse markers.* They are the topics that have been introduced by the corresponding discourse fragment. In the other component conditions on these discourse markers are stored. Thus for a piece of discourse such as

A dog barked. It was lonely.

we can expect a representation such as

$$(\{r\},\ \{\ dog(r),\ bark(r),\ lonely(r)\ \}).$$

[2]For more details, also on the relation with Kamp's formulation, cf [29].

In the first component of the representation we find the discourse markers introduced in the example. In fact there is just one such marker, the dog. In the second component we see the conditions on this marker that the piece of discourse expresses.
In the semantics these representations are interpreted as pairs: the set of discourse markers is simply copied and in the second component of the semantic objects we store all assignments of values to these markers that satisfy the conditions in the second component of the *DRS*. So the interpretation of the *DRS* given above is:

$$(\{r\},\ \{\ f:\ \mathit{MARKERS} \rightarrow D\ :\ f(r) \in \mathbf{dog} \cap \mathbf{bark} \cap \mathbf{lonely}\ \}).$$

(Here D is the domain of objects in (our model of) the real world. *MARKERS* is the set of discourse markers.) The second component represents the actual testing of the information content of the *DRS* in the real world. The role of the first component becomes clear as soon as we see how the interpretation of a large text is composed from the interpretations of its parts. The *DRS*-interpretation we see above can be described as the result of such a composition process.

$$\begin{array}{lcl} (\{r\},\ \{\ \mathit{dog}(r),\ \mathit{bark}(r)\ \}) & \rightsquigarrow & (\{r\},\ \{\ f:\ f(r) \in \mathbf{dog} \cap \mathbf{bark}\ \}) \\ (\emptyset,\ \{\ \mathit{lonely}(r)\ \}) & \rightsquigarrow & (\emptyset,\ \{\ f:\ f(r) \in \mathbf{lonely}\ \}) \\ (\{r\}, \{\ \mathit{dog}(r), \mathit{bark}(r), \mathit{lonely}(r)\ \}) & \rightsquigarrow & \\ & & (\{r\},\ \{f:\ f(r) \in \mathbf{dog} \cap \mathbf{bark} \cap \mathbf{lonely}\}) \end{array}$$

We see that in the process the condition in the second *DRS*, *lonely*, is linked to the marker r that was introduced in the first *DRS*. This process of linking is steered by the first components of the semantic objects. This is what the first component is for. The general recipe for the composition operation, that Zeevat [29] calls the *merger*, is:

$$(V,\ F) \bullet (W,\ G) \quad = \quad (V \cup W,\ F \cap G).$$

By taking simple set union as the operation on the marker sets, the link between the marker that is lonely and the marker that is a barking dog, is established. By adjusting the operation on the first component we can steer the linking process in the formalism to our liking.
As an example, consider the following alternative for the definition of the merger:

$$(V,\ F) \bullet (W,\ G) \quad = \quad (V \oplus W,\ F^\star \cap G^\star).$$

(Here $\oplus$ stands for disjoint union. * is the operation on the function sets that follows the renaming instructions involved in the disjoint union. Suppose for example that $x \in V$ and also $x \in W$; then we make these two variables distinct when we take the disjoint union. But then we have to do the same thing in the domain of the functions in F and G. This gives us $F^\star$ and $G^\star$.)

Here the idea is implemented that two markers introduced in distinct parts of discourse can never be identical. We see that, whenever a marker occurs in both V and W, the markers are made distinct before we take the union of the sets.
Thus we can implement different linking strategies by varying the operation on the first component. It is this operation that steers the merging process: once the merger is defined on the first component, there is only one sensible choice left for the way in which the second components should interact. The first component *controls* the process of meaning composition.
Note that in our example both definitions will have the same effect: in both cases the anaphoric link is established. But it is not hard to think of (admittedly artificial) operations that would not establish the required link, even in this simple example. Take for example:

$$(V,\ F) \bullet (W,\ G) \quad = \quad (\,(V \times \{1\}) \cup (W \times \{2\}),\ F^1 \cap G^2).$$

(Here the role of $v \in V$ is taken over by $(v, 1)$. This is also done in the domain of the functions in F, which is indicated by the super script 1. Similar for the superscript 2.)
If we apply this procedure in our example, we get:

$$(\{r\},\ \{\, f :\ f(r) \in \mathbf{dog} \cap \mathbf{bark}\}) \ \bullet\ (\emptyset,\ \{\, f :\ f(r) \in \mathbf{lonely}\}) \ =$$
$$(\{(r,1)\},\ \{\, f :\ f(r,1) \in \mathbf{dog} \cap \mathbf{bark}\ \ \&\ \ f(r) \in \mathbf{lonely}\,\})$$

Now the condition **lonely** has not been linked to $(r, 1)$.

We see here that it is indeed the first component that steers the linking process. Once the first components have decided what goes where, the sets of assignments have to follow these instructions. Thus the two tasks that are involved in the interpretation of the anaphor are nicely separated: the first components establish the anaphoric links between different pieces of discourse and the second components compute the joint information content (and truth conditions) according to these links.

2 Referent Systems

2.1 Variables

What Zeevat's formulation makes clear is that if we wish to be serious about the semantics of anaphora, we need to distinguish carefully between control features and information content. Here *control* amounts to linking the variables in the appropriate way, so we need to conduct a more serious investigation into the notion of variable and the role it plays in semantics.
Logic traditionally uses the Fregean notion of variable. In Frege's opinion it is not correct to think of variables as having a denotation. To say that a variable

has a reference of some sort, would merely provoke the question what it is that a variable denotes. Probably the answer would be something like: variables denote arbitrary objects. Attempts to make sense of this answer would, according to Frege[3], lead us to problems that we do not even want to think about: certainly he showed that this was not necessary in standard predicate logic.
However, Frege's objections against other concepts of variables may have been too hasty. For example, in the work of Fine [9] it is shown that it is possible to have a sensible theory of variables as arbitrary objects, something that Frege strongly rejects. Moreover in computer science, for example in denotational semantics, a notion of variable is used that is very different from the Fregean notion of variable, but nevertheless makes perfect sense. In the computer science notion of variable a distinction is made between the syntactic variable or variable name, and the real variable, which is usually thought of as some location (in the memory of a computer) where information can be stored. Thus it becomes possible to distinguish three things for each variable: its name, the variable itself and the information that is stored in the variable.
It can be suspected that some such notion of variable will be particularly useful for us, who are trying to distinguish between the control features that have to be represented in dynamic semantics, and the information content. With this notion of variable we can think of variables as storage facilities and their names as the mechanism by which we manipulate the variables and thereby the information that we have stored in them.
In what follows we will give a formal treatment of such a notion of variable. We will then use the new techniques to do dynamic semantics. We will see how in the new formulation the referent systems are used to describe the control features essential for the explanation of anaphora.

2.2 Referent Systems

In the formal definition of the referent systems we will use a fixed stock of names, $NOM = \{x_1, x_2, \ldots\}$. We will usually write x, y, z, v, w for elements of NOM. We also need some set theoretic representation for the variables, or referents as we shall be calling them.[4] Remember that referents in our set up are just locations in memory that we happen to have reserved to store a particular piece of information. We do not want to include in our model any assumptions about the nature or the structure of memory. We regard memory as an unstructured substance that has no properties that are of interest to us: memory gets all its interesting properties by our actions on it. One such action is the declaration of a variable, whereby we reserve some arbitrary part of memory for the storage of information. This action actually *creates* the variable: before we performed

[3]In [10]: Logische Mängel in der Mathematik.

[4]In fact we will use the words 'variable', 'referent' and 'discourse marker' interchangeably. Note, however, that when we talk about the Fregean notion of variable, we are *not* talking about referents.

the action there was nothing. After the declaration there is a variable. We will assume that at each point it is possible to perform such a creation.
An advantage of this way of looking at variables over the use of a fixed stock of variables, declared in advance, is that we will now always have just a finite amount of these storage facilities. Since there is, in principle, no limit to the amount of variables one can use, working with a fixed stock would imply working with an infinite amount of variables from the start. In our approach we will always just have a finite set of variables in our model, the ones we actually use. Also note that the use of a fixed stock introduces a moment of choice in the way of dealing with variables: whenever we need a variable, we have to decide which one to use. This is a kind of complication that we choose to avoid, both for reasons of technical convenience an because this issue does not fit our intuitions concerning variables. The complication can be avoided as described above, by creating new variables 'on the spot'.
The lack of structure and properties of memory is inherited by the variables. They are simply arbitrary parts of memory. This is another reason why it would be unfortunate if the variables in our model would all come from a fixed set. This would inevitably give them properties that real variables do not have.
Hence, if we want to find a set theoretic representation of some set of referents, it seems that no set is good enough. Or, if we look on the bright side, any set is equally good. When we are at a point where we have created some referents and we want to represent this situation set theoretically, then there is no *natural* choice for the set of referents: any set of the right size will do equally well (or equally bad). Therefore we can use any set as a set of referents, as long as we keep in mind that nothing forced us to choose this set instead of another one (with the same cardinality).
This leads to the following definition of *referent systems.*

Definition 2.1 *A triple (I, R, E) is a referent system iff:*

1. *R is a finite set, the referents;*
2. *I is a partial injection from NOM to R, the import function;*
3. *E is a partial injection from R to NOM, the export function.*

   (From now on we will use postfix notation for function application and function composition. We will omit brackets whenever this is convenient.)

At the core of a referent system (I, R, E) we find the referents R. To this we have added an import function and an export function. They allow us to manipulate the referents. Recall that referent systems will be used later to model the way we manipulate variables in dynamic semantics. In section 2.5 we will see examples of this, her we just give teh general picture. The idea is that the import function tells us which referents are picked up from the context and under which name. So if $xI = r$, then this means that the referent r is

not introduced in this referent system: instead it is picked up from the context under the name x. Hence r will be the referent that was called x already.
The export function E tells us which referents in R can be picked up in what is to follow and under which name. Hence, if $rE = x$, then x is the current name of this referent. This means that the next referent system can import r under the name x. In such a situation the first referent system exports what the next referent system wants to import: a transaction can take place. Thereby a referent can interact with the context to its left via the import function and with the context to its right via the export function. Thus we create a model in which the (chronological) order of our actions on variables is represented.
Every referent system comes with a dual. It can be obtained, as it were, by reading the original referent system from right to left. So the dual of (I, R, E) is (E^{-1}, R, I^{-1}). From this duality we can learn that E^{-1} will have to behave like a proper import function or, dually, that I^{-1} has to behave like a proper export function. We will use this duality of referent systems whenever we can.
Remember that the choice of the set R is arbitrary. R is just a finite set of the right cardinality, but any other set of the same cardinality would have done equally well for a set theoretic representation of the same referent system. Therefore we should not use set theoretic identity as the identity criterion for referent systems. We will introduce a notion of isomorphism for our representations of referent systems that will provide us with the sort of identity that makes sense for referent systems.

Definition 2.2 *Let two referent systems (I, R, E) and (I', R', E') be given. A homomorphism $\Phi : (I, R, E) \to (I', R', E')$ is an injection $R \to R'$ such that:*

1. *$dom(I) \subseteq dom(I')$ & $xI\Phi = xI'$ (for all x);*
2. *$dom(E) \subseteq dom(\Phi E')$ & $rE = r\Phi E'$ (for all r).*
   (Note that this is not *the dual of the previous clause.)*

By Φ we can recognise the referents of (I, R, E) in (I', R', E') in such a way that the names that a referent has in (I, R, E) are also present in (I', R', E'). Note that the conditions (1) and (2) can be rewritten as:

$I\Phi \subseteq I'$

and

$E \subseteq \Phi E'$.

In a picture:

$$\begin{tikzcd}
& R \arrow[dr, "E"] \arrow[dd, "\Phi"] & \\
NOM \arrow[ur, "I"] \arrow[dr, "I'"'] & & NOM \\
& R' \arrow[ur, "E'"'] &
\end{tikzcd}$$

Now we can make the following observation: referent systems and homomorphisms form a category. Hereby the familiar notion of isomorphism from category theory becomes available:[5] two referent systems (I, R, E) and (I', R', E') are isomorphic iff there are homomorphisms $\Phi : (I, R, E) \to (I', R', E')$ and $\Psi : (I', R', E') \to (I, R, E)$ such that $\Phi\Psi = id_{(I,R,E)}$ and $\Psi\Phi = id_{(I',R',E')}$. The following proposition gives us easy way of recognising isomorphic referent systems.

Proposition 2.3 *There is a bijection Φ between R and R' such that $I\Phi = I'$ and $E = \Phi E'$ iff (I, R, E) and (I', R', E') are isomorphic.*

Proof:
Clearly such a bijection Φ induces a homomorphism $(I, R, E) \to (I', R', E')$ and Φ^{-1} induces a homomorphism $(I', R', E') \to (I, R, E)$ such that $\Phi\Phi^{-1} = id_{(I,R,E)}$ and $\Phi^{-1}\Phi = id_{(I',R',E')}$.
On the other hand if Φ and Ψ are homomorphisms such that $\Phi\Psi = id_{(I,R,E)}$ and $\Psi\Phi = id_{(I',R',E')}$, then we know from the definition of homomorphism that

1. that Φ is a bijection $R \to R'$ (with inverse Ψ);
2. $I\Phi \subseteq I'$;
3. $I'\Psi \subseteq I$;

From (2) we get $dom(I) \subseteq dom(I')$. From (3) we get $dom(I') \subseteq dom(I)$. This means $I\Phi = I'$. Similarly we can prove $E = \Phi E'$.$\Box$

The proposition shows that the notion of isomorphism that we have defined is indeed the one we were looking for: two triples are isomorphic if the only difference between them is the choice of the set of referents. As we have explained, two such triples represent the same referent system. Therefore we will no longer distinguish them. At this point our only use for this notion of homomorphism of referent systems is that it allows us to make the notion of equivalence of referent systems precise. The reader who feels uncomfortable with these notions can rest assured: in the rest of the paper he can simple use proposition 2.3 as a

[5]Recall that we use postfix notation.

definition of the equivalence of referent systems. But we suspect that at some point we will have some other use for it.[6]

2.3 The merger

In the previous subsection we have explained what the import and export function are for. The export function tells which referents are passed on by the system and under which name, the import function tells which referents are needed by the system by giving their names. This is how the communication between variables in different parts of a text will be modelled (cf. section 2.5 for examples). With this explanation in mind, we define the merger of two referent systems as follows: (We use a dot "$\bullet$" for the merger because that reminds us of the way sentences in discourse are merged into a text.)

Definition 2.4 *Let two referent systems (I, R, E) and (I', R', E') be given. We define the merger of the two referent systems, $(I'', R'', E'') = (I, R, E) \bullet (I', R', E')$, as follows:*

1. $R'' = (R \oplus R')/\sim$

   where $\oplus$ stands for disjoint union and $\sim$ is the smallest equivalence relation such that for $r \in R$, $r' \in R'$ we have:

   $r \sim r'$ *iff* $rEI' = r'$.

   (We will write r for $r \in R$ as well as for the image of r in $R \oplus R'$ and also for the equivalence class of r in $(R \oplus R')/\sim$ if no confusion can arise.)

2. $xI'' = xI$ *if* xI *is defined;*

   $xI'' = xI'$ *if* xI' *is defined and* xI *is not defined and for no* $r \in R : r \sim xI'$*;*

   xI'' *is undefined otherwise.*

3. E'' *is defined dually, i.e.*

   $rE'' = rE'$ *if* $r \in R'$ *and* rE' *is defined;*

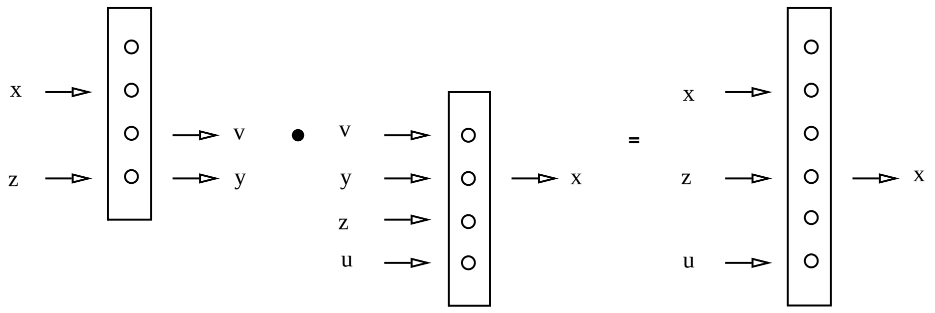
   $rE'' = rE$ *if* $r \in R$ *and* rE *is defined and for no* $r' \in R'$ $r \sim r'$ *and for no* $r' \in R'$ $r'E' = rE$*;*

   rE'' *is undefined otherwise.*

[6]We expect this for the following reason: in dynamic semantics we should not only be concerned with the question whether some sentence is a part of some text, but also which part it is exactly. This kind of distinction can easily be made in category theory. Therefore it seems that at some point the use of categories in text semantics will be inevitable. Also see Visser [28] for more details on the use of categories in dynamic semantics.

Before we discuss this definition, we will show in a picture how the merger works. In the picture the sets of referents are represented as vertical blocks. Their import names, if any, are found on the left hand side, their export names on the right hand side. We have tried to make this example such that you can see for each of the import clauses of (2) how they work. By duality a good example for the export clauses can then be obtained by turning the page upside down.

Example:



Note that it is the import of z by the first referent system that is preserved in the merger of the two systems. Also note that the variable called v has now become invisible: it has become a local variable.
Now if we consider the definition, we see that clause (1) contains no surprises: two referents get identified iff a transaction can take place, i.e. iff $rEI' = r'$. In other words, two referents are identified iff they have the same name at the time of a merger. The new import and export functions are defined in clauses (2) and (3). To understand these definitions it is important to keep in mind that the left to right order is to correspond to the chronological order. Of course the merger of the two referent systems still exports all the referents the second referent system exports (the first part of clause 3). And if no transaction has taken place, then the export behaviour of the first referent system should be taken over by the merger as well. For, there will still be the possibility to export the referents that the first referent system provides—unless the second referent system already provides them.
A difficulty arises when the two referent systems both supply a referent with the same name, i.e., for some $r \in R$, $r' \in R'$ $rE = r'E'$. This will only cause problems, of course, if the second referent system does not import the referent r. For when this referent will be imported by the second referent system, the merger of the two systems will only be able to export the referent r'. But if $rE = r'E'$ and for no $r'' \in R$: $rEI' = r''$, then there will be a serious competition between the two systems for the export of a referent with name rE. It seems natural to assume that in this case the second referent system will

win the competition since it will be 'closer' to the candidates that may want to import a referent called x.

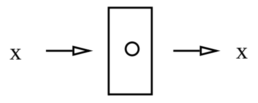
Hence $rE'' = rE'$ whenever rE' is defined (first option in 3), while we can only have $rE'' = rE$ if not $rE \in \mathit{range}(E')$. We have this priority rule for export functions, because the order of the referent systems is to reflect the chronological order. Hence, if some referent with name x is imported later on, then the importing referent system will not wait for the first referent system to make a transaction: it will just import the referent that the second system — that is closer — has to offer.

As we have said, I'' is defined dually. This is the first time that we can see that duality can save a lot of work. But because it is the first time we have given the details anyhow. By duality we know that I''^{-1} will be the export function of $(E'^{-1}, R, I'^{-1}) \bullet (E^{-1}, R, I^{-1})$. This means that it will behave like I and like I' (if there are no transactions) and that in case of a clash (i.e. $x \in \mathit{dom}(I) \cap \mathit{dom}(I')$) I will have priority over I'. As one can see, this is what is (2) gives us.

Note that we are only interested in referent systems up to isomorphism: we do not care about the particular set R that is used to represent the referents. Therefore it should be checked that the definition of the merger preserves isomorphism of referent systems. Given the characterisation in proposition 2.3 this is not difficult.

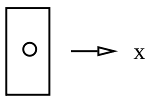
2.4 Referent systems in semantics

We want to use the referent systems for dynamic semantics: with each formula we will associate a referent system that tells us all about the variables that occur in the formula. We will do this in detail later, but we can already give some examples. A formula $\phi(x)$ with a free variable x will give rise to a referent system with a referent, r say, such that $xI = r$ and $rE = x$. The referent and its name are simply passed on. The referent system of $P(x)$, for example, will be: $(\{\langle x, r\rangle\}, \{r\}, \{\langle r, x\rangle\})$ or, in a picture:[7]

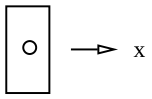


Quantified variables are exported, but they do not have to be imported. So $\exists x$ gets the referent system

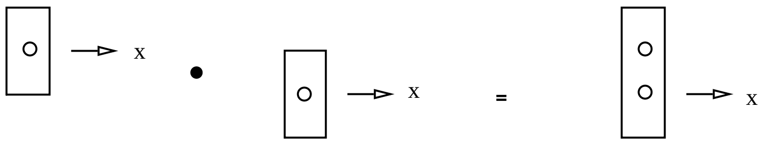
[7]In pictures we prefer to represent referents as (featureless) os. In the text we will still use letters r, r' etc.



Thereby $\exists x$ only allows for interaction with the context to the right, while $P(x)$ can interact both ways. By merging the two referent systems in the right order we get the referent system for $\exists x.P(x)$:



The priority rule for the export function will be used to handle repeated quantification: the formula $\exists x.P(x).\exists xQ(x)$ gets the referent system:



This is what we predicted: the first referent is still there, but it no longer has a name.

3 Properties of Referent Systems

The purpose of this section is to study the formal properties of referent systems. First we will define some special referent systems. They are special for two reasons: first because they are probably the easiest ones you can think of and second because they are in a way the basic referent systems.

Definition 3.1 (Special referent systems) *For each finite set $V \in NOM$ we define:*

1. $Z = (\emptyset, \emptyset, \emptyset)$;
2. $O[V] = (\emptyset, V, \emptyset)$;
3. $I[V] = (id_V, V, \emptyset)$;

4. $E[V] = (\emptyset, V, id_V)$;

5. $T[V] = (id_V, V, id_V)$.

When $V = \{v\}$, we will omit the brackets.

Here we have chosen the names as (representations of) referents. Of course this is not essential (compare section 2.1): it is just a very convenient choice. Note also that we use capitals I, E here, that are also used for import and export functions. This is because here I and E also stand for 'import' and 'export'. Furthermore, T stands for transport and Z for zero.
We can use these special referent systems to show how calculations with referent systems work. Some easy results (and their duals) are in the following proposition.

Proposition 3.2 *Let $V, W \subseteq NOM$ be given.*

▷ *(The clauses on the left hand side are the special case where $V = W$.)*

(i)	$E[V] \bullet I[V] = O[V]$	$E[W] \bullet I[V] = E[W \backslash V] \bullet O[W \cap V] \bullet I[V \backslash W]$
(ii)	$T[V] \bullet T[V] = T[V]$	$T[W] \bullet T[V] = T[W \cup V]$
(iii)	$E[V] \bullet E[V] = O[V] \bullet E[V]$	$E[W] \bullet E[V] = O[W \cap V] \bullet E[W \cup V]$
(iv)	$I[V] \bullet I[V] = I[V] \bullet O[V]$	$I[V] \bullet I[W] = T[V \cup W] \bullet O[V \cap W]$
(v)	$E[V] \bullet T[V] = E[V]$	$E[W] \bullet T[V] = E[W] \bullet T[V \backslash W]$
(vi)	$T[V] \bullet I[V] = I[V]$	$T[V] \bullet I[W] = T[V \backslash W] \bullet I[W]$

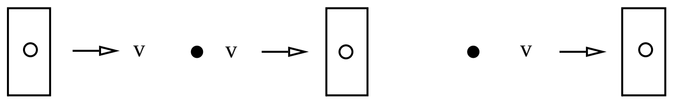
In $T[V] \bullet E[V]$, $I[V] \bullet T[V]$ and $I[V] \bullet E[V]$ no transactions are possible.

▷ *(Non-associativity)*

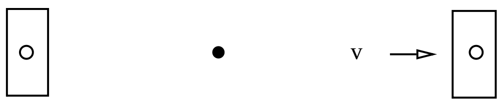
$(E[V] \bullet E[V]) \bullet I[V] \neq E[V] \bullet (E[V] \bullet I[V])$ $(V \neq \emptyset)$;

$E[V] \bullet (I[V] \bullet I[V]) \neq (E[V] \bullet I[V]) \bullet I[V]$ $(V \neq \emptyset)$.

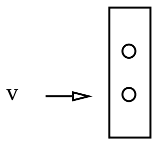
Proof:
Omitted. We will just give a picture of an example of a situation where the merger is not associative. We take $V = \{v\}$.



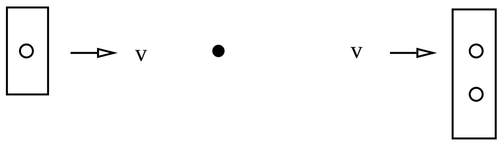
Either this reduces first to:



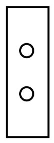
and then to:



or it reduces first to:



and then to:



□

The results of the proposition are easy, so we have omitted the proofs. On the other hand, they are representative for the way referent systems work, so we will discuss them in a little more detail. Note that the statements in the first column are just instances of the statements in the second column. In (i) we see that we can create local variables explicitly: first we export a variable, then it is imported. Note that the statements in (i) and (ii) are self-dual. In (iii) we find another way to create local variables: we already have variables called V and then we declare new variables called V. Now, by our priority rules, the old variables lose their names and become local. Clause (iv) is the dual of (iii).

The second item shows that the merger is not an associative operation. From a technical point of view this is an unpleasant property.[8] It will complicate the calculations with referent systems. It should be noted, however, that the example of non-associativity that we have here is rather subtle: it involves both a transaction between referent systems and a clash of names. Therefore we can still hope that there are natural classes of referent systems in which this does not occur and in which the merger is an associative operation. Let us consider a few of those natural classes.

Definition 3.3

1. *A referent system (I, R, E) is partially persistent (PP) iff $IE \subseteq id_{NOM}$;*
2. *A referent system (I, R, E) is extending iff $dom(I) \subseteq range(E)$;*
3. *A referent system (I, R, E) is faithful iff $dom(I) = range(E)$.*

These classes of referent systems are all defined by a condition on the way they handle the names of referents. This is a natural kind of condition to consider for referent systems, since this is what referent systems were introduced for. For example, a restriction on the number of referents in R would make much less sense. The first condition says that if a referent r is imported with a name, x say (i.e. $xI = r$), then, if the referent is also exported, its export name is still x (if $rE\downarrow$, then $rE = x$). Note that a referent can lose its name in a PP system, it is only a change of name that is not allowed.
The extending referent systems are such that the names that are imported are also exported. It is possible that there are also other export names or that some old name now stands for another referent, but the old names still have a reference.
Faithful referent systems have a fixed set of names. Again the reference of the names may change, but not the presence of the names.
These three classes of referent systems have the following closure property:

Proposition 3.4

1. *The PP referent systems are closed under merger.*
2. *The extending referent systems are closed under merger.*
3. *The faithful referent systems are closed under merger.*

□

[8]There are also methodological reasons for preferring an associative operation in text semantics. We will not go into that here, but refer to [25], [28].

It is easy to see that the counterexample against associativity that we have, is built up from partially persistent referent systems. Therefore the merger is not an associative operation on partially persistent referent systems. With some effort it can be shown that merger is associative within the other two classes. We will not prove this now, because it is an easy consequence of a more general result:

Proposition 3.5 *Let referent systems σ, τ and ρ be given. Then:*

$$(\sigma \bullet \tau) \bullet \rho \ = \ \sigma \bullet (\tau \bullet \rho) \text{ iff}$$
$$range(E_\sigma) \cap dom(I_\tau) \cap dom(I_\rho) \subseteq range(E_\tau) \cup dom(I_\sigma) \text{ and}$$
$$dom(I_\rho) \cap range(E_\tau) \cap range(E_\sigma) \subseteq range(E_\rho) \cup dom(I_\tau).$$

This result is proved and discussed in the appendix (A.1). It gives rise to the following corollary:

Corollary 3.6

1. *The merger is not associative in the class of partially persistent referent systems;*

2. *The merger is associative in the class of extending referent systems;*

3. *The merger is associative in the class of faithful referent systems.*

□

The referent systems that will actually occur in our semantics are all partially persistent and extending. So we will get an associative merger in our semantics. We conclude this section with an observation:

Proposition 3.7

▷ $Z = I[\emptyset] = E[\emptyset] = T[\emptyset] = O[\emptyset]$.

▷ *The class of partially persistent referent systems can be generated from referent systems of the form Z, $I[v]$, $E[v]$ and $T[v]$.*

▷ *The class of partially persistent and extending referent systems can be generated from referent systems of the form Z, $O[v]$, $E[v]$ and $T[v]$.*

□

This proposition shows that every referent system can be seen as the result of some very basic actions, such as importing or exporting one referent.

4 Dynamic Semantics with Referent Systems

In this section we show how we can use referent systems for dynamic semantics. In fact we will use referent systems in the semantics of $\mathcal{L}_{DPL}$, our language for dynamic predicate logic.

Definition 4.1 ($\mathcal{L}_{DPL}$) *There is an enumerable set of variable names, x, y, z, x_1, $x_2, \ldots$ and a set of relation symbols P, Q, R, $P', \ldots$. The set of DPL formulas, $\mathcal{L}_{DPL}$, is the smallest set containing:*

1. $\bot$;
2. $\exists x$ *for each variable* x;
3. $P(x_1, \ldots, x_n)$ *for each n-ary predicate symbol P and variables* $x_1, \ldots, x_n$;
4. $\phi.\psi$ *whenever* ϕ, $\psi \in \mathcal{L}_{DPL}$;
5. $(\phi \to \psi)$ *whenever* ϕ, $\psi \in \mathcal{L}_{DPL}$.

In this language we can define abbreviations such as $\forall x(\phi) \equiv (\exists x \to \phi)$ and $\neg(\phi) \equiv (\phi \to \bot)$. The referent systems will be used to interpret the variables of this language: we will identify the syntactic variables of $\mathcal{L}_{DPL}$ with the names in NOM.
Note that $\mathcal{L}_{DPL}$ differs slightly from the language that is usually used in DPL. Usually it is insisted that quantifiers such as $\exists x$ do not occur in isolation, but are always accompanied by some formula, ϕ say. We find it appropriate to have $\exists x$ as a (sub)formula since it has a clearly distinguishable semantic contribution: it represents the act of creating a new referent with name x. But those who find this deviation from standard practice disturbing may replace all occurrences of $\exists x$ in our formulas by $(\exists x\ \top)$ (where $\top$ is a harmless tautology) to obtain formulas in the usual format with the same interpretation.
We will call the interpretations of $\mathcal{L}_{DPL}$ *discourse structures*. They will consist of a referent system and a set of assignments of values to the referents in the system. We use assignments to referents, not to their names, because the information that we find in $\mathcal{L}_{DPL}$ is not information about the syntactic variables but about the 'real' variables that have the syntactic variables as names.
The formal definition is as follows. (We assume that the domain of interpretation of our model, D, is given. The notation $f|_X$ is used for the restriction of the function f to domain X.)

Definition 4.2 (Discourse Structures (DSs))

1. *A discourse structure δ is a pair $(\sigma_\delta, F_\delta)$, where $\sigma_\delta = (I_\delta, R_\delta, E_\delta)$ is a referent system and F_δ is a set of assignments from R_δ to D;*

2. *A homomorphism of discourse structures* $\Phi : \delta \to \varepsilon$ *consists of a homomorphism of referent systems* $\Phi_1 : \sigma_\delta \to \sigma_\varepsilon$ *and a function* $\Phi_2 : F_\varepsilon \to F_\delta$ *such that for all* $f \in F_\varepsilon : \quad f\Phi_2 = \Phi_1 \circ f$.

   (Thereby $\forall f \in F_\varepsilon \exists g \in F_\delta : \quad g = \Phi_1 \circ f$.*)*

   We say: Φ_2 *gives the restriction of* f *to* σ_δ *according to* Φ_1.

3. *If* δ *and* ε *are discourse structures, then the merger* $\gamma = \delta \bullet \varepsilon$ *where* $\sigma_\gamma = \sigma_\delta \bullet \sigma_\varepsilon$ *and* $F_\gamma = \{f : R_{\delta\bullet\varepsilon} \to D : f|_{R_\delta} \in F_\delta$ *and* $f|_{R_\varepsilon} \in F_\varepsilon\}$.

$$\begin{tikzcd}
R_\delta \arrow[rr, "\Phi"] \arrow[dr, "f\Phi_2"'] & & R_\varepsilon \arrow[dl, "f"] \\
& D &
\end{tikzcd}$$

Again we are only interested in the referent systems up to isomorphism. The notion of homomorphism of discourse structures shows how we can extend this notion to discourse structures. For the homomorphisms of discourse structures Φ we introduce a notation convention: we will simply use Φ instead of Φ_i if no confusion can arise.
In the discourse structures we have sets of total functions: they are total on the set of referents of the structure. If we merge two discourse structures, we glue these total assignment together, if possible. The result is a total assignment on the new referent set.
The definition of homomorphism, as we have given it, requires some explanation. The idea is that there is a homomorphism from δ to ε, iff ε contains more information than δ. Therefore it is natural that we require that ε has 'more' referents and that ε has better access to the referents. This is guaranteed by the presence of a homomorphism of referent systems. It is also natural that the more informative *DS* should allow less assignments. This is what the mapping Φ_2 takes care of: it guarantees that no new (configurations of) values are allowed.

Now we can interpret $\mathcal{L}_{DPL}$. In fact we have already seen most of the crucial examples in section 2, where we defined referent systems. Here we will extend the apparatus with the construction of the referent system for implications. Once this is done we can straightforwardly add the second component of the interpretation, the set of assignments.

Definition 4.3

1. *For two referent systems σ, τ we define $(\sigma \rightarrow \tau)$ as follows:*

   $(\sigma \rightarrow \tau) = (I, R, E)$,

   where $I = I_{\sigma \bullet \tau}$, $R = range(I)$ and $E = I^{-1}$.

2. *For two DSs δ, ε we define $(\delta \rightarrow \varepsilon)$ as follows:*

   $(\delta \rightarrow \varepsilon) \; = \; ((\sigma_\delta \rightarrow \sigma_\varepsilon), \;\; F)$

   where $f \in F$ iff $f : R_{(\sigma_\delta \rightarrow \sigma_\varepsilon)} \rightarrow D : \; \forall g \in F_\delta :$

   $f|_{dom(g)} \leq g \rightarrow \exists h \in F_{\delta \bullet \varepsilon} : \quad f \leq h \;\; \& \;\; g \leq h.$

We will discuss this definition shortly, but first we give the interpretation of $\mathcal{L}_{DPL}$. (Here the notation X^Y is used for all the functions from Y to X and $\mathbf{P}$ is the extension of P.)

Definition 4.4 *Each formula of $\mathcal{L}_{DPL}$ is interpreted as a discourse structure according to the following clauses:*

- ▷ $[\![\bot]\!] = \; (Z, \emptyset)$;
- ▷ $[\![\exists x]\!] = \; (E[x], D^{\{r\}})$;
- ▷ $[\![P(x_1, \ldots, x_n)]\!] = (T[x_1, \ldots, x_n], \{f \in D^{\{x_1, \ldots, x_n\}} : \; \langle x_1 f, \ldots, x_n f\rangle \in \mathbf{P}\}$;
- ▷ $[\![\phi . \psi]\!] = [\![\phi]\!] \bullet [\![\psi]\!]$;
- ▷ $[\![(\phi \rightarrow \psi)]\!] = [\![\phi]\!] \rightarrow [\![\psi]\!]$.

A first remark about this definition is that all the referent systems that we find in it are partially persistent and extending. From this it follows that the merger is an associative operation on $\mathcal{L}_{DPL}$ interpretations.
We can see that the sets of assignments for simple formulas are just the sets that were to be expected: the formula $\exists x$ does not restrict the values of the assignments and an atomic formula, $P(x)$ say, gives rise to the obvious restriction that the value on the referent of x should have property P. For the conjunction of formulas, $\phi.\psi$, we have to glue together the assignments of $[\![\phi]\!]$ and $[\![\psi]\!]$. This will only be possible for some assignments. For example if we try to glue together assignments from $[\![P(x)]\!]$ with assignments from $[\![\neg P(x)]\!]$, we will see that this cannot be done: any assignment from $[\![P(x)]\!]$ will assign to the referent called x a value in $\mathbf{P}$, while the assignments in $[\![\neg P(x)]\!]$ will assign values outside $\mathbf{P}$ to the referent called x. By the definition of the merger, these referents called x are to be identified in $[\![P(x).\neg P(x)]\!]$. Hence no assignments will survive the glueing procedure. As a result we get: $[\![P(x).\neg(P(x))]\!] = (T[x], \; \emptyset)$.
The clause for implication requires some explanation. We see that in the referent system of an implication, $(\phi \rightarrow \psi)$, we find the referents that are imported in $\phi.\psi$. So we get, for example:

for $(P(x) \to Q(x))$, the set $\{x\}$;

for $(\exists x.P(x) \to Q(x))$, the empty set;

for $(P(x) \to \exists x.Q(x))$, $\{x\}$, the first referent called x.

In general we get the referents that correspond to the free variables in $(\phi \to \psi)$ (with the DPL notion of binding and freeness in mind). These 'free referents' are simply transported through the referent system: they have the same im- and export name.
The assignments that are allowed in $(\phi \to \psi)$ are the assignments of which any assignment that satisfies ϕ can be extended to an assignment that satisfies $\phi.\psi$. This is just the usual construction from DRT and DPL. There are some subtleties involved that have to do with the domains of the assignments that we have to consider.

> First we have to restrict a function f that is defined on $R_{\sigma_\delta \to \sigma_\varepsilon}$ to R_δ. This is necessary for implications like $(P(x) \to Q(y))$, where there are free referents that do not occur in the antecedent.
>
> Then any extension of the resulting assignment that satisfies ϕ should be extendible to an assignment that satisfies $\phi.\psi$.
>
> But there is a further requirement on this assignment: it has to agree with the original f on the free variables of ψ. That is why we also have to demand $f \leq h$ in the definition.

This can be illustrated with the formula $(\exists x.P(x) \to Q(x,y))$: f will be defined on $\{y\}$. First this f will be restricted to the empty function. Now every extension of the empty function that assigns to x a value in $\mathbf{P}$, needs an extension h such that $(xh, yh) \in \mathbf{Q}$. But we are only interested in extensions h that assign to y the value yf.

The definition of $[\![(\phi \to \psi)]\!]$ can be motivated further: we can compare it with the notion of valid inference that we will develop for discourse structures. We will see that the implication that we have defined is just the implication that goes with the notion of valid inference for DSs.
In the definition of valid inference we will touch upon the issue of partiality. So this is a good opportunity to make some general remarks about partiality and referent systems. Since a DS represents the information expressed by some discourse fragment, it will, in general, only give partial information about the underlying model. For not every discourse fragment gives complete information about the whole world. This is the first kind of partiality we have to distinguish. We have seen already that this does not give rise to the use of partial functions in our DSs: all assignments in a DS are defined on all the referents in the DS. The partiality is expressed in other ways: first by the limited (finite)

number of referents that we have in one DS. This reflects the fact that we only have information about a limited number of objects. Then there is the fact that we work with a set of assignments, not with one assignment. By allowing ourselves to consider several values for one referent we reflect the partiality of our information about this referent: we do not always know exactly which entity the referent is a stand in for, we just know some conditions that limit the range of possibilities.

This kinds of partiality is important, of course. But it is not—or should not be—typical for dynamic semantics.

However, there also is a kind of partiality that is directly related to the dynamics. It has to do with the context components of the DSs. In a DS, $((I, R, E),\ F)$ say, not only F gives information about the referents in R. I and E also reveal some relevant facts. If a referent is exported, $rE = x$ say, then this means that more information about r can be given in what is to follow. If a DS imports some referent, $xI = r$ say, then this means that this DS will depend on the context—some other DS—to get the referent r.

This is important information about r. In particular if a referent is *imported* this reveals a strong kind of partiality of our information: in the context component of the DS it is not really known where r comes from. The context component is not saturated. This kind of partiality, *partiality-as-context-dependency*, is typical of dynamic semantics. It corresponds more or less to the distinction of free and bound variables in a more traditional approach.

With this kind of partiality in mind we define valid inference as follows:

Definition 4.5

1. *Let two referent systems σ and σ' be given. Then:*

   $\sigma \models \sigma'$ *iff* $dom(I_{\sigma'}) \subseteq range(E_\sigma)$

2. *Let two DSs δ and δ' be given. Then:* $\delta \models \delta'$ *iff*

   $\sigma_\delta \models \sigma_{\delta'}$

   and

   $\forall f \in F_\delta :\ \ f \in F_\delta\ \exists f' \in F_{\delta'} :\ \ I_{\delta'} f'\ \subseteq\ E^{-1} f.$

3. *For two formulas ϕ and ψ we define valid inference as follows:*

   $\phi \models \psi$ *iff* $[\![\phi]\!] \models [\![\psi]\!]$.

4. *A formula ϕ is valid iff* $(Z,\ \{\emptyset\}) \models [\![\phi]\!]$.

(Recall that $Z = (\emptyset, \emptyset, \emptyset)$.)

In clause (1) $\sigma \models \sigma'$ can be read as: σ provides a suitable context for σ'. In the other clauses one can read the $\models$ sign as 'supports'. Here we are talking about truth conditional and contextual support at the same time.

Of course we want $\delta \models \delta$ to express that the information in δ' follows from the information in δ. But, as we have seen, δ' does not only give information about $R_{\delta'}$, it also requires information about the origin of the referents in $range(I_{\delta'})$. This means that if $\delta \models \delta'$, then these referents have to be supplied by δ. In other words, we want $\sigma_\delta \models \sigma_{\delta'}$. But we also want all the assignments in F_δ to have a corresponding assignment in $F_{\delta'}$ to guarantee that δ' allows more values for the referents than δ. For otherwise there is more information about those referents in δ' than in δ. This is what clause (2) says.
Now we can look at the definition of $[\![(\phi \rightarrow \psi)]\!]$ again. Our observations about partiality do not apply here. A formula $(\phi \rightarrow \psi)$ can occur in a context in which the imported referents of ψ have been introduced properly, even if this is not done by the formula ϕ.
Think for example of

$$\exists x.(\exists y.P(y) \rightarrow Q(x,y)).$$

We have argued that $[\![\exists y.P(y)]\!] \models [\![Q(x,y)]\!]$ cannot hold, because the referent of x is not supplied by $[\![\exists y.P(y)]\!]$. Similarly, the formula $(\exists y.P(y) \rightarrow Q(x,y))$ cannot be valid. But this does not mean that the formula $(\exists y.P(y) \rightarrow Q(x,y))$ does not have a meaning. It just means that the context component of its meaning will have to import a referent called x.
In this situation the best relation between inference and implication that we could hope for is:

$$[\![\phi]\!] \models [\![\psi]\!] \text{ iff } F_{[\![(\phi \rightarrow \psi)]\!]} \neq \emptyset \quad \& \quad I_{[\![(\phi \rightarrow \psi)]\!]} = I_{[\![\phi]\!]}.$$

This says that truthconditionally—i.e. in terms of assignment sets—both situations are equivalent, but for valid inference we have the extra requirement that the context component of ϕ supports the context component of ψ.
It can be checked that this is indeed what we get. This gives another justification of our definition of implication: it is just the internalisation of the notion of validity for *DSs*.
Note that the concern with the partiality of the context is not standard in *DPL* or *DRT*. Above we have argued that this kind of partiality is typically a concern of a dynamic semantics, but still it is usually ignored in presentations of *DRT* and *DPL*. We will see in the next section that our interpretation of implications corresponds to the familiar notions in *DRT* and *DPL*. This will, at the same time, make clear how our notion of valid inference relates to what we find in those formalisms: they agree up to partiality.

5 DS, DRS and DPL

In this section we compare the *DS* semantics, which uses referent systems, with the relational *DPL* semantics and the representational *DRT* semantics. *DPL*

likes to see its variables as just syntactic entities, in the Fregean sense. On the other hand the notion of discourse marker that we find in *DRT* seems to be closer to the notion of a storage facility than to the strictly syntactic notion of variable.
Therefore we would expect to find that we get something very much like *DRT* if we ignore the names of the referents in our referent systems, while ignoring the referents and concentrating on the names instead should give us something very close to *DPL*. We will see that this is indeed what we find.

5.1 DRT

Technically the comparison with *DPL* will be straightforward, since the language we use is the *DPL* language. *DRS*s cannot be compared with *DPL* formulas as easily: *DRSs* are just a different kind of things. They live on an intermediate level between syntax and semantics: in a *DRS* we find both the discourse markers, which also belong to the semantic domain, and the conditions on these markers that live on the level of syntax. Hence for the comparison it is convenient to introduce a similar intermediate level in between the *DPL* formulas and *DS* interpretations. We will call the intermediaries *DPSs*, *Dynamic Predicate Structures.* The *DPSs* can be defined inductively as follows:[9]

Definition 5.1 *We define DPSs and DPS-conditions by a simultaneous induction:*

1. *an atomic DPL-formula is a DPS-condition;*
2. *if δ and δ' are DPSs, then $(\delta \rightarrow \delta')$ is a DPS-condition;*
3. *for each referent system σ and set of DPS-conditions C, (σ, C) is a DPS.*

Now we can give for each *DPL* formula a *DPS* that represents it.

Definition 5.2 *For each DPL formula ϕ, we define $[\phi] = (\sigma_\phi, C_\phi)$, the DPS of ϕ, as follows:*

$$\begin{array}{ll}
[\bot] & = (Z, \emptyset) \\
[\exists x] & = (E[x], D^{\{x\}}); \\
[P(x_1, \ldots, x_n)] & = (T[\{x_1, \ldots, x_n\}], \{P(x_1, \ldots, x_n)\}); \\
[(\phi \rightarrow \psi)] & = ((\sigma_\phi \rightarrow \sigma\psi), \{([\phi] \rightarrow [\psi])\}); \\
[\phi.\psi] & = (\sigma_\phi \bullet \sigma_\psi,\ C_\phi \bullet C_\psi).
\end{array}$$

Here $C_\phi \bullet C_\psi$ indicates that the names of the variables as they occur in C_ϕ and C_ψ should follow the identifications and dis-identifications of $\sigma_\phi \bullet \sigma_\psi$. For example, $[\exists x.P(x)] = (E[x], \{P(x)\})$ and $[\exists x.Q(x)] = (E[x], \{Q(x)\})$. But if we merge the referent systems the two referents x will not be identified. We have to make this visible in the conditions:

[9] This definition closely follows the definition of a *DRS* in Zeevat [29].

$$[\exists x.P(x).\exists x.Q(x)] \;=\; (O[x'] \bullet E[x], \{P(x'), Q(x)\}).$$

The interpretation of the *DPSs* as *DSs* is obvious: we simply replace the set of conditions C by the set of assignments (on the referents) that satisfy these conditions.
Now we have a representational level in our interpretation with referent systems and we are able to compare the referent systems approach with *DRT*.

$$\begin{tikzcd}
 & \text{DPL} \arrow[d] & \\
 & \text{DPS} \arrow[dl] \arrow[dr, "\text{o}"] & \\
\text{DS} \arrow[dr, "\text{o}"] & & \text{DRS} \arrow[dl] \\
 & |\text{DRS}| &
\end{tikzcd}
\qquad
\begin{tikzcd}
 & \varphi \arrow[d] & \\
 & [\varphi] \arrow[dl] \arrow[dr, "\text{o}"] & \\
[[\varphi]] \arrow[dr, "\text{o}"] & & \text{DRS}_{\varphi} \arrow[dl] \\
 & |\text{DRS}_{\varphi}| &
\end{tikzcd}$$

($|DRS|$ stands for the class of DRS-interpretations as discussed in section 1.2 and defined in [29].)
The diagram gives the picture that one should keep in mind. Most of the links in the diagram have not yet been defined properly. But their formal definition can easily be constructed from what has been said so far. The $\circ$-links, for example, rely on the following operation:

Definition 5.3 *For each referent system $\sigma = (I, R, E)$ we define the reduction of σ to a set as follows: $set((I, R, E)) \;=\; \{r \in R : \; r \in dom(E) \;\;\&\;\; r \notin range(I)\}$.*

Now if we want to make *DRSs* out of *DPSs* or *DRS*-interpretations out of *DSs*, we just have to apply this set function to the first component of the *DPS* or *DS*. Our claim is, of course, that the diagram commutes. We will make this claim precise in the following proposition, but we will omit the proofs. They are tedious but straightforward.

Proposition 5.4 *For each DPL-formula ϕ we claim that:*
$|[\phi]| = [\![\phi]\!]$ *DPSs are intermediate between DPL and DSs;*
$|[\phi]^\circ| \;=\; |[\phi]|^\circ$ *the DRS-interpretation of a reduced DPS is the reduced DS.*

We give an example of all the links in the diagram:

$$\begin{tikzcd}
 & \exists\, x.\mathrm{DOG}(x),\mathrm{BARK}(x) \arrow[d] & \\
 & (E[x], \{\mathrm{DOG}(x), \mathrm{BARK}(x)\}) \arrow[dl] \arrow[dr, "o"] & \\
(E[x], \{f: \mathrm{dom}(f)=\{x\} \text{ and } xf \,\varepsilon\, \mathrm{DOG} \text{ and } xf \,\varepsilon\, \mathrm{BARK}\}) \arrow[dr, "o"] & & (\{x\}, \{\mathrm{DOG}(x),\mathrm{BARK}(x)\}) \arrow[dl] \\
 & (\{x\}, \{f: \mathrm{dom}(f)=\{x\} \text{ and } xf \,\varepsilon\, \mathrm{DOG} \text{ and } xf \,\varepsilon\, \mathrm{BARK}\}) &
\end{tikzcd}$$

Our conclusion from this comparison with *DRT* is that *DSs* are 'just' *DRSs* with import and export functions. We have a set of referents and a set of conditions on these referents, just as in *DRT*. But we also have import and export functions that make our manipulations of these referents explicit.
These manipulations are crucial for the semantics of anaphora. They handle the question which referents are to be identified and which referents are to be kept distinct. But precisely this point is left implicit in *DRT*. There it is usually assumed that we have automatically chosen the referents in a way that solves all problems. For example, it is usually assumed that the sets of new discourse markers U and U' are disjoint whenever we merge two *DRS*s (U, C) and (U', C'). This is not unreasonable if discourse markers are indeed to be compared with our referents. If they are indeed just featureless storage facilities that we have *created* during the interpretation, then—by definition—they cannot be the same. But if referents are indeed featureless, then it is unclear how we can work with them at all: if they have no features we cannot recognise them. For example, we cannot see from two markers whether they have to be identified because of some anaphoric link unless these markers have a feature that shows this. The import and export function add such features, names, and thereby make it possible to be explicit about the interaction of the referents.
So although these details are usually ignored in formalisations of *DRT*, we see that this is not necessary. In our machinery we have not only the featureless storage facilities, but also labels for them, that allow us to do something with them.
Apart from allowing us to be explicit about the manipulation of referents, there are also some other pleasant consequences of our set up. Since we have the

referents we inherit all the advantages of DRT: an intuitively appealing story about how we store antecedents in memory, a situation where at each point we will only have a finite number of these antecedents, etc. Furthermore, now that we have an explicit representation of the syntactic variable *and* the semantic variable, we are in a position where we can distinguish properties of variables as syntactic objects and properties of variables as storage facilities. For example, while it is by definition impossible to create the same referent twice, we can use the same words as anaphors over and over again. Therefore it is reasonable to allow re-using element of NOM, syntactic variables, although the idea of re-using a semantic variable makes no sense.
So we have obtained a more flexible machinery in which we can implement the DRT-ideas about anaphora in a more felicitous way.

5.2 DPL

The comparison with the relational DPL semantics is technically more straightforward. We have an interpretation of the DPL language as relations and another one as DSs. We will show that the relational interpretation can be obtained from the DS interpretation. We will do this in two steps: first we construct a relational interpretation in terms of referent systems from the DS interpretation. Then we will see that these new relations are closely related to the relational interpretation that we know already. So the picture is as follows: (DPR is the class of *Dynamic Predicate Relations*, that we will define shortly; $|DPL|$ the usual relational interpretation; $|DPL|_p$ the extension thereof to partial functions.)

$$\begin{tikzcd}
\text{DPL} \arrow[r] & \text{DS} \arrow[r, leftrightarrow] & \text{DPR} \arrow[d] \\
 & & |\text{DPL}|_p \arrow[d] \\
 & & |\text{DPL}|
\end{tikzcd}$$

At the DPR level we find pairs (σ, f) consisting of a referent system σ and an assignment f on the referents of that referent system. We create relations on these objects from DSs as follows:

Definition 5.5

1. *For each DS* $\varepsilon = (\sigma_\varepsilon,\ F_\varepsilon)$ *we define the binary relation* $[\varepsilon]$ *as follows:*[10]

   $(\sigma, g)[\varepsilon](\sigma', h)$ *iff* $dom(I_{\sigma_\varepsilon}) \subseteq range(E_{\sigma_\varepsilon})$ & $\sigma' = \sigma \bullet \sigma_\varepsilon$ & $h = g \oplus f$ *for some* $f \in F_\varepsilon$.[11]

2. *For each formula* ϕ *we define the relation* $[\phi]$ *as follows:*

   $[\phi]\ =\ [[\![\phi]\!]]$.

The relation $[\phi]$ works with assignments that are defined on referents. The usual relational interpretation is defined on assignments that are defined on (all) variable names. We will obtain this kind of relation in two steps: first we switch from referents to names and then we switch from partial to total functions. The switch to names is made as follows:

Definition 5.6

1. *For each binary relation* R *on* DPR*-pairs we define the following relation on partial assignments* $f:\ NOM \supset\!\!\!\rightarrow D$.

   $f|R|_p g$ *iff* $\exists(\sigma, f'), (\sigma', g'):\ (\sigma, f')R(\sigma', g')$ & $f = E_\sigma^{-1}f'$ & $g = E_{\sigma'}^{-1}g'$.

2. *For each* DPL *formula* ϕ *we define a relation on partial assignments as follows:*

   $|\phi|_p\ =\ |[\phi]|_p$.

Definition 5.6 gives us an interpretation of formulas ϕ as relations on partial functions $NOM \to D$. From this relational interpretation we obtain a relation on total assignments $NOM \to D$ by restricting $|\phi|_p$ to total functions. This gives us the usual relational interpretation. We will prove this shortly, but first we try to gain some insight in the relations $[\phi]$.

Lemma 1 *Let* (σ, g), (τ, h) *be suitable pairs. Then:*

$$\begin{array}{lll}
(\sigma,g)[\bot](\tau,h) & \textit{iff} & g \neq g; \\
(\sigma,g)[P(x)](\tau,h) & \textit{iff} & \sigma = \tau \ \ \&\ \ g = h \ \ \&\ \ xI_\tau h \in \mathbf{P}; \\
(\sigma,g)[\exists x](\tau,h) & \textit{iff} & \tau = \sigma \bullet E[x] \ \ \&\ \ h = g \cup \{(xI_\tau, d)\} \textit{ for some } d \in D; \\
(\sigma,g)[\phi.\psi](\tau,h) & \textit{iff} & \tau = \sigma \bullet (\sigma_\phi \bullet \sigma_\psi) \ \ \&\ \ \exists f \in F_{\phi.\psi}:\ \ g \oplus f = h; \\
(\sigma,g)[(\phi \to \psi)](\tau,h) & \textit{iff} & \tau = \sigma \bullet \sigma_{(\phi\to\psi)} \ \ \&\ \ h = g \ \ \&\ \ \forall(\rho,k):\ (\sigma,g)[\phi](\rho,k) \\
 & & \exists(\mu,l):(\rho,k)[\psi](\mu,l).
\end{array}$$

[10]Be careful: in this subsection we use the same brackets as in the previous subsection, but with a different meaning!

[11]Here $\oplus$ glues g and f together.

We omit the proof of the lemma.
Now we are ready for the following proposition (proof in the appendix):

Proposition 5.7 *Let g and h be partial assignments $NOM \supset\!\!\!\rightarrow D$. Then:*

$$\begin{array}{lll}
g|\bot|_p h & \textit{iff} & g \neq g; \\
g|P(x)|_p h & \textit{iff} & x \in dom(g) \;\;\&\;\; g = h \;\;\&\;\; xh \in \mathbf{P}; \\
g|\exists x|_p h & \textit{iff} & x \in dom(h) \;\;\&\;\; (y \neq x \;\;\Rightarrow\;\; xg = xh); \\
g|\phi.\psi|_p h & \textit{iff} & \exists k: \;\; g|\phi|_p k \;\;\&\;\; k|\psi|_p h; \\
g|(\phi \rightarrow \psi)|_p h & \textit{iff} & g = h \;\;\&\;\; \forall k: \;\; g|\phi|_p k \;\;\Rightarrow\;\; \exists l: \;\; k|\psi|_p l.
\end{array}$$

The proposition says that the usual clauses for the relational interpretation of DPL define the relational interpretation with partial functions. From this it follows that the restriction of $|.|_p$ to total assignments really is the usual DPL interpretation.
The proof of the proposition will be given in some detail in the appendix, partly because there are some interesting constructions involved, partly also because we want to make up for the lack of detail in our proofs so far. From the proposition it is clear that the restriction of $|.|_p$ to total functions is indeed the usual relational DPL interpretation.
This means that we indeed get DPL by forgetting the referents themselves and retaining only their names. This confirms that DPL sticks to the Fregean notion of variable.
In DPL the basic objects in the semantics are the assignments of values to variable names. These objects are used to model the way in which we store antecedents in memory when we interpret texts with anaphors. Of course it would be better, for an intuitively acceptable explanation of this process, if finite assignments were used instead of total, infinite assignments (cf. Fernando [8] for discussion). But this requires but a small adaptation of the standard formulation of DPL-semantics.
Another interesting point is that DPL-variables can be re-used. This is what we can expect with a syntactic notion of variable: just as pronouns (and other anaphoric expressions) in natural language, a variable name can be used over and over again with different meanings in different contexts. But since DPL does not distinguish the variable name from the 'real' variable, the re-use of a variable name can have nasty side effects. For whenever we give a new use to a variable name x, with $\exists x$, we are forced to forget the information that was previously attached to that name. In the DSs, where we can also re-use variable names, such a re-use will only result in the creation of a local variable, a variable that no longer has a name. Thereby we no longer have access to that variable, but we will not be forced to throw this variable itself away and we save the (truthconditional) information that was stored in it. In DPL, however, we do not have such an option. Whenever we re-use a variable we automatically lose the information that was attached to it. This causes the *non-eliminativity* problem, as discussed in [12] and [24].

It has been suggested that this problem for *DPL* can easily be prevented by using different variables all the time, but this is an option that really does not go very well with a syntactic notion of variable. Syntactic elements, lexical items, typically *can* be used over and over again. It seems unelegant to put a semantically motivated restriction on the syntax, especially since the whole problem can be prevented by using the machinery developed here. Again we see that having explicit representations of both the syntactic and the semantic variable makes the machinery more flexible and allows us to give a natural representation of all the phenomena involved.

We conclude that the distinction between variables and variable names allows us to formulate the crucial ideas about the dynamic semantics of anaphora. It makes it possible to show explicitly how the link between an anaphor and its antecedent is established. Furthermore, our semantics with referent systems has enabled us to keep two kinds of tasks separate: the task of manipulating the variables can be described separately from the task of computing the (truthconditional) result of these manipulations.
This distinction also has the advantage that it allows us to recognise the fact that variables and variable names are different kinds of things with different kinds of behaviour. This way we can do justice both to our intuitions about syntax and to our intuitions about semantics.

6 Discussion

In this paper we have developed techniques for passing on information that is stored in variables. The techniques are of general interest for all situations where information is manipulated, but were designed with a special application in mind: anaphora. We have shown that the machinery of the referent systems can compare with the two major alternatives in this field: *DRT* and *DPL*.
Still some questions about the precise relation between our formal machinery and the situation in natural language remain that we would like to go into in some more detail. These questions are of a rather general nature and apply not only to our semantics for anaphora with referent systems, but to any formal treatment of anaphora. Still we feel we have to say something about these questions, since we have noticed that confusion about these general points has led to misjudgements of our intentions and our results.
First there is one kind of approach to semantics that centers all its questions around the practical problem of translating natural language into a formal language. Although the idea of translation of natural language has provoked many interesting developments in formal semantics, we do not feel that it is correct to judge all developments in semantics from this perspective. Thus the question which formalism is better for semantics, for us, is not the same as the question which formalism allows the smoothest translations. There can be improvement

in the semantic representation without much improvement in terms of translation. Perhaps the system that we have developed is not much of an improvement from the translational point of view, but we have argued that it *does* improve our representation of the situation in natural language, indeed we would argue that it gives clearer insight into the mechanisms operating in natural language. To make the point plain, we can consider the issue of reoccurrence of variables. Arguably the translational perspective has led to confusion here. Some problems in dynamic semantics, such as the eliminativity problem (cf. [24], [12]), are connected with the possibility of using a variable name more than once. Now, it has been suggested that these problems do not require much attention since these situations can easily be avoided by a suitable translation procedure: simply choose new variables as often as you can in your translation (cf. [5]). This is of course a remark made from a typically translational perspective on semantics. However, from the point of view of semantic representation in general it does not make much sense. We all know that in natural language anaphoric expressions, such as pronouns, can be used over and over again without any trouble. The same lexical item simply gets a different meaning depending on the (linguistic) context. If this situation is widely spread and unproblematic in natural language, but very problematic in our formal representation, then the formal system does not provide a good explanation of the natural language situation. The translation-with-new-variables approaches finesse this whole issue. Arguably a clear picture of this issue is what is needed to get a firm grasp on what the semantics of pronouns really is.
So our main goal in this paper has not been an improvement of the possibilities for translation.[12] Instead it has been our aim to have a formal situation where the communication of information between different parts of formal language mimics in most of the important respects the situation in natural language. We think that our proposal is a clear improvement from this perspective. The distinction between the variable itself as storage facility that can be created at any point and the variable *name* as the syntactic machinery that we use to manipulate those storage facilities makes sense from the point of view of natural language interpretation. It avoids the situation where in the formalism a fixed infinite store of abstract variables is provided in advance and gives a better representation of our intuitions concerning the way in which the interpretation of anaphora in natural language involves the creation of new discourse markers. Furthermore, it allows us to distinguish the syntactic issue of reoccurrence of variable names from the semantic issue of re-using the same storage facility. Although re-using a syntactic item should always be allowed, if we want to represent the situation in natural language, the re-use of a storage facility is connected with the forgetting of information, something which is much less likely to happen in the interpretation of ordinary natural language sentences

[12]Although the use of referent systems certainly does not seem to make translation more difficult.

(cf. Vermeulen [24]).
Of course lots of points remain where our formal system deviates from the system we use in natural language. For example, all the connections between anaphors and antecedents are represented in our system by identity of variable names. But in the natural language an anaphoric expression is typically *not* syntactically identical with its antecedent.
Such a deviation from real life is potentially a bad thing. But in this case it is not so clear that there is actually something wrong with our system. In the formal representation we distinguish radically between on the other hand the things that are said about some discourse marker, the information content which is typically represented by some predicate, and on the other hand the machinery by which we manipulate this information, the referent names in *NOM*. This radical distinction is not made in natural language. It seems that there the same words serve both purposes. If we say, for example:

A man and a woman came in. The man was wearing a black coat.

Then the word *man* does not only serve to give information, but can also be used when we want to name the referent associated with it. This is what we do when we say *the man*. So in natural language we sometimes do two things with one word. This does not mean that the distinction between these two aspects of the meaning of one word that we have made is a disadvantage of the system. The insight that words have this double role is an important one and we do not have to be ashamed that we make it in our formalism.
However, there is a follow up to the criticism that to me seems quite correct. Although there is no harm in separating formally things that are not always explicitly separated in natural language, it is desirable that the aspects that we have separated are represented correctly. We think that it is fair to say that the contribution to the information content via predicates is sufficiently close to real life for our purposes, although it might require improvement in other situations. But our representation of the linking machinery with variables is admittedly too simple. In natural language there are many clues that can be used to link up with an antecedent. Consider for example the following alternatives for the above example:

A man and a woman came in. He was wearing a black coat.

A man and a woman came in. The former was wearing a black coat.

We see that already in this simple example there are at least three ways of establishing the anaphoric link. But in the formal system we only have one clue for each antecedent, its current name.
Forunately there is ample room for improvement in that area within our set up. We could, for example, have instead of one current name for each referent, a set of names. This would amount to replacing the import and export functions with functions of type: $R \to \wp(NOM)$.

This is but a simple adjustment of the definition of a referent system, representing the situation where there are several ways to refer to a variable, just as in natural language. But already this simple change allows us to do very wild things in the definition of the merger. We could, for example, identify a referent with the most suitable antecedent at this point. A first attempt to model this strategy is given by a condition such as:

> In merging (I, R, E) and (I', R', E') we identify $r' \in R'$ with $r \in R$ under the following condition:
>
> $$r \equiv r' \;\Leftrightarrow\; max\{\, s \in R :\ E(s) \cap I(r')\} \;=\; E(r) \cap I(r') \;\;\&\;\; E(r) \cap I(r') \neq \emptyset.$$

(We ignore for the moment all sorts of details, such as the problem of non-unique maxima.)
A definition of this sort could begin to account for the unstability of anaphoric links.[13] For if a referent system is preceded by another referent system, then the choice of the preferred antecedent might thereby be influenced. In natural language we sometimes have a similar situation. Take the (extended) example:

> A man and a woman came in. He liked her.
>
> John was looking at the door. A man and a woman came in. He liked her. But he hated him.

So there is room for criticism in our overly simple representation of the linking mechanism involved in anaphora, but there is also room for improvement. But the advantage of our set up is that this kind of improvement in the context component need not have any side effects in the content component. This is so because we have neatly seperated the two ways in which variables contribute to dynamic interpretations.
Still we should not be too optimistic here: it is clear that even when we improve our representation of the syntactic clues that we use in natural language, we still are far from an ideal representation of what happens in the interpretation of anaphora. It is well known that apart from syntax also semantic and pragmatic phenomena play a role in the choice of an antecedent. For example, in

> Mary threw a brick at the window. It broke.

it is hard to imagine a syntactic clue that would cause us to choose the window instead of the brick as an antecedent. (By the way, here the link is also clearly instable. Imagine the example being preceded by : *It is not true that bricks don't break.*) Clearly what is involved is our knowledge of the world and of the meaning of the words used in the fragment.

[13]Technically this instability will be difficult to capture.

To some extent these influences could be described systematically and to that extent they can be incorporated in our framework. Now that we have distinguished in our semantics the different problems involved in interpreting anaphora—the linking problem on the one hand and the computation of information content on the other—one can easily imagine that such systematic influences on the linking process could be represented in the referent system without making it necessary to re-think the definition of the other component of our meaning objects.[14] Taking the context-content distinction serious in the semantics is the big step. After that improvements of either one of the components is just a matter of fine tuning.

A Appendix

A.1 Associativity

In this appendix we pick up the question of the associativity of the merger. We will prove the following

Proposition A.1 (Proposition 3.5)

$$\begin{aligned}(\sigma \bullet \tau) \bullet \rho \;&=\; \sigma \bullet (\tau \bullet \rho) \textit{ iff}\\ dom(E_\sigma^{-1}) \cap dom(I_\tau) \cap dom(I_\rho) \;&\subseteq\; dom(E_\tau^{-1}) \cup dom(I_\sigma) \textit{ and}\\ dom(I_\rho) \cap dom(E_\tau^{-1}) \cap dom(E_\sigma^{-1}) \;&\subseteq\; dom(E_\rho^{-1}) \cup dom(I_\tau).\end{aligned}$$

(We have written $dom(E_\sigma^{-1})$ instead of $range(E_\sigma)$ to make it easier to see that the two conditions are dual.)
We will see that these conditions say that there is no "clash" of names that can be prevented by a transaction under one of the bracketings.
This is exactly what *does* happen in the counterexample against associativity that we have seen:

$(E[v] \bullet E[v]) \bullet I[v]$:there is an export-clash;
$E[v] \bullet (E[v] \bullet I[v])$:clash prevented by the transaction between $E[v]$ and $I[v]$.

In such a situation the import-export behaviour will not be independent of the bracketing.

Now we can see that the second condition of the proposition captures this situation. The intersection on the lefthand side of the condition says that a transaction between τ and ρ is possible and that an export-clash between τ and σ

[14] Work by Zeevat [30] in this direction, in particular in unification formalisms, is in this spirit.

is possible. The union on the right hand side is to prevent the trouble that we have seen in the example. (In the example the rhs-union is empty.)
If something in the intersection is also in $dom(I_\tau)$, then this means that there is not only a transaction between τ and ρ, but also between σ and τ. This means that there is no clash that can be prevented. This is what happens for example in $E[v] \bullet T[v] \bullet I[v]$.
If something in the intersection is also in $dom(E_\rho^{-1})$, the clash is not really prevented, it is merely delayed. This happens for example in $E[v] \bullet E[v] \bullet T[v]$.
The first condition can be illustrated by looking at the dual of our example, i.e. $E[v] \bullet I[v] \bullet I[v]$. (Remember that to find the dual, we have to read from right to left and consider import as export and vv.)
We see that the conditions of the proposition can be understood by looking at these basic examples.

Proof:
$\Leftarrow$:
Assume that both conditions hold. Also assume that we have chosen the symbols for the referents (r, r', etc.) in such a way that no confusion can arise if we simply speak of a referent without mentioning its referent system (i.e. we will say r and xI simpliciter instead of $r \in R$, $xI \in R'$, etc).
Notation:

$$\begin{aligned} I_1 &= I_{(\sigma \bullet \tau) \bullet \rho}, \\ I_2 &= I_{\sigma \bullet (\tau \bullet \rho)}. \end{aligned}$$

We will check that $I_1 = I_2$. Then $E_1 = E_2$ follows by duality (note that the two clauses in the proposition are dual). Strictly speaking we have to check two things: first that $dom(I_1) = dom(I_2)$ and second that for all x in the domain $xI_1 = xI_2$. We will concentrate on the second task and perform the first one implicitly.
We consider the definition of xI_1:

$$\begin{aligned} xI_1 &= xI_{\sigma \bullet \tau} && \text{if } x \in dom(I_{\sigma \bullet \tau}) \\ &= xI_\rho && \text{if } x \in dom(I_\rho) \backslash (dom(I_{\sigma \bullet \tau}) \cup dom(E_{\sigma \bullet \tau}^{-1})) \end{aligned}$$

i.e.

$$\begin{aligned} xI_1 &= xI_\sigma && \text{if } x \in dom(I_\sigma) \\ &= xI_\tau && \text{if } x \in dom(I_\tau) \backslash (dom(I_\sigma) \cup dom(E_\sigma^{-1})) \\ &= xI_\rho && \text{if } x \in dom(I_\rho) \backslash (dom(I_\sigma) \cup (dom(I_\tau) \backslash dom(E_\sigma^{-1})) \\ & && \cup\, dom(E_\tau^{-1}) \cup (dom(E_\sigma^{-1}) \backslash dom(I_\tau))) \end{aligned}$$

The definition of I_2 gives us:

$$\begin{aligned} xI_2 &= xI_\sigma && \text{if } x \in dom(I_\sigma) \\ &= xI_{\tau \bullet \rho} && \text{if } x \in dom(I_{\tau \bullet \rho}) \backslash (dom(I_\sigma) \cup dom(E_\sigma^{-1})) \end{aligned}$$

i.e.

$$\begin{aligned} xI_2 &= xI_\sigma && \text{if } x \in dom(I_\sigma) \\ &= xI_\tau && \text{if } x \in dom(I_\tau) \backslash (dom(I_\sigma) \cup dom(E_\sigma^{-1})) \\ &= xI_\rho && \text{if } x \in dom(I_\rho) \backslash (dom(I_\sigma) \cup dom(I_\tau) \cup dom(E_\tau^{-1}) \cup dom(E_\sigma^{-1})). \end{aligned}$$

We see that the only difference between xI_1 and xI_2 can occur if the third clause of the definitions is activated. But in the proposition we see that by the first condition we have that for $x \in dom(I_\rho)$:

$$x \in dom(E_\sigma^{-1}) \cup dom(I_\tau) \quad \Rightarrow \quad x \in dom(E_\tau^{-1}) \cup dom(I_\sigma).$$

Hence for $x \in dom(I_\rho)$:

$x \notin dom(I_\sigma) \cup dom(I_\tau) \cup dom(E_\tau^{-1}) \cup dom(E_\sigma^{-1})$ iff

$x \notin dom(I_\sigma) \cup (dom(I_\tau) \backslash dom(E_\sigma^{-1})) \cup (dom(E_\sigma^{-1}) \cap dom(I_\tau)) \cup dom(E_\tau^{-1}) \cup dom(E_\sigma^{-1})$ iff

$x \notin dom(I_\sigma) \cup (dom(I_\tau) \backslash dom(E_\sigma^{-1})) \cup dom(E_\tau^{-1}) \cup dom(E_\sigma^{-1})$.

If we apply the same trick again by splitting up $dom(E_\sigma^{-1})$ into $(dom(E_\sigma^{-1}) \backslash dom(I_\tau))$ and $dom(E_\sigma^{-1}) \cap dom(I_\tau)$, we find:

$x \notin dom(I_\sigma) \cup (dom(I_\tau) \backslash dom(E_\sigma^{-1})) \cup dom(E_\tau^{-1}) \cup (dom(E_\sigma^{-1}) \backslash dom(I_\tau))$ iff

$x \notin dom(I_\sigma) \cup dom(I_\tau) \cup dom(E_\tau^{-1}) \cup dom(E_\sigma^{-1})$.

We conclude that: $I_1 = I_2$.

$\Rightarrow$:
Suppose that the first condition of the proposition does not hold. (The case where the second clause fails is dual.) Then we have $x \in dom(E_\sigma^{-1}) \cap dom(I_\tau) \cap dom(I_\rho)$ but $x \notin dom(E_\tau^{-1}) \cup dom(I_\sigma)$ for some $x \in NOM$. Now if we look at the import functions I_1 and I_2, we see that in both cases the first two clauses do not apply. Hence we have to consider the third clause of the definitions. We see that since $x \in dom(E_\sigma^{-1}) \cap dom(I_\tau) \cap dom(I_\rho)$, also $x \in dom(I_\tau)$ and therefore $x \in dom(I_\sigma) \cup dom(I_\tau) \cup dom(E_\tau^{-1}) \cup dom(E_\sigma^{-1})$. This means that xI_2 is undefined.
But if $x \in dom(E_\sigma^{-1}) \cap dom(I_\tau) \cap dom(I_\rho)$, then $x \notin dom(I_\sigma) \cup (dom(I_\tau) \backslash dom(E_\sigma^{-1})) \cup dom(E_\tau^{-1}) \cup (dom(E_\sigma^{-1}) \backslash dom(I_\tau))$. Therefore xI_1 is defined and in fact $xI_1 = xI_\rho$.
So we find that xI_1 is defined, while xI_2 is undefined. Hence $(\sigma \bullet \tau) \bullet \rho \neq \sigma \bullet (\tau \bullet \rho)$.

This completes the proof of the proposition.$\Box$

A.2 DPL and DS

In this section of the appendix we prove the following proposition from section 5.

Proposition A.2 (Proposition 5.7) *Let g and h be partial assignments* $NOM \supset \rightarrow D$. *Then:*

$g\|\bot\|_p h$	*iff*	$g \neq g$;
$g\|P(x)\|_p h$	*iff*	$x \in dom(g)$ & $g = h$ & $xh \in \mathbf{P}$;
$g\|\exists x\|_p h$	*iff*	$x \in dom(h)$ & $(y \neq x \Rightarrow xg = xh)$;
$g\|\phi.\psi\|_p h$	*iff*	$\exists k:\ g\|\phi\|_p k$ & $k\|\psi\|_p h$;
$g\|(\phi \rightarrow \psi)\|_p h$	*iff*	$g = h$ & $\forall k:\ g\|\phi\|_p k \Rightarrow \exists l:\ k\|\psi\|_p l$.

We use the following two results:

Corollary A.3 *Let three referent systems* ρ, σ *and* τ *be given such that* σ *and* τ *are partially persistent and extending. Then* $\rho \bullet (\sigma \bullet \tau) = (\rho \bullet \sigma) \bullet \tau$.

Proof:
Follows immediately from the associativity result that is proved in the appendix.□

Lemma 2 (Extension lemma) *Let* (ρ, k), (σ, g), (τ, h) *be given:*

▷ *if* ρ *is partially persistent and extending, then*
$(\sigma, g)[\phi](\tau, h) \Rightarrow (\sigma, g) \bullet (\rho, k)[\phi](\tau, h) \bullet (\rho, k)$;

▷ *if* σ *is partially persistent and extending, then*
$(\sigma, g)[\phi](\tau, h) \Rightarrow (\rho, k) \bullet (\sigma, g)[\phi](\rho, k) \bullet (\tau, h)$.

Proof
Direct. (1) requires partial persistence and extending to keep the variables linked to the same referents. In (2) the condition is used to make sure that the merger is associative (see corollary).□

Now we can prove the proposition:

Proof:
The proof is a lengthy induction in which a construction of suitable partially persistent and extending referent systems is implicitly defined.

1. The case where ϕ is $\bot$ is obvious;

2. $g|P(x_1, \ldots, x_n)|_p h$ iff
   there are (σ, g'), (τ, h') such that $(\sigma, g')[P(x_1, \ldots, x_n)](\tau, h')$ & $g = E_\sigma^{-1} g'$ & $h = E_\tau^{-1} h'$ iff
   there are (σ, g'), (τ, h') such that $(\sigma, g') = (\tau, h')$ & $range(g') \supseteq \{x_1, \ldots, x_n\}$ &
   $(x_1 g', \ldots, x_n g') \in \mathbf{P}$ & $g = E_\sigma^{-1} g'$ & $h = E_\tau^{-1} h'$ iff

$g = h$ & $(x_1 g, \ldots, x_n g) \in \mathbf{P}$.

For the last 'iff' we need a construction. The following will do: take $\sigma = T[dom(g')]$, $g' = g$, $\tau = \sigma$ and $h' = g'$.

3. $g|\exists x|_p h$ iff

   there are (σ, g'), (τ, h') such that $(\sigma, g')[\exists x](\tau, h')$ and $g = E_\sigma^{-1} g'$ & $h = E_\tau^{-1} h'$ iff

   $\sigma \bullet E[x] = \tau$ & $h' = g' \oplus (x, d)$ for some $d \in D$ & $g = E_\sigma^{-1} g'$ & $h = E_\tau^{-1} h'$ iff

   $y \neq x \to yg = yh$ & $x \in dom(h)$.

   For the last 'iff' we need a construction. Take σ as above and take $\tau = \sigma \bullet E[x]$. Then $g' = g$ and $h' = h \oplus (x, d)$ will do.

4. $g|\phi.\psi|_p h$ iff

   there are (σ, g'), (τ, h') such that $(\sigma, g')[\phi.\psi](\tau, h')$ and $g = E_\sigma^{-1} g'$ & $h = E_\tau^{-1} h'$ iff

   there are (σ, g'), (τ, h') such that $\tau = \sigma \bullet \sigma_{\phi.\psi}$ & $h' = g \oplus (f \oplus f')$ for some $f \in F_\phi$ and $f' \in F_\psi$ & $g = E_\sigma^{-1} g'$ & $h = E_\tau^{-1} h'$ iff (by corollary)

   there are (σ, g') and (τ, h') such that $\tau = (\sigma \bullet \sigma_\phi) \bullet \sigma_\psi$ & $h' = (g \oplus f) \oplus f'$ &

   $g = E_\sigma^{-1} g'$ & $h = E_\tau^{-1} h'$ iff (take $\rho = \sigma \bullet \sigma_\phi$ and $k' = g' \oplus f$)

   there is (ρ, k') such that $(\sigma, g')[\phi](\rho, k')$ and $(\rho, k')[\psi](\tau, h')$ & $g = E_\sigma^{-1} g'$ &

   $h = E_\tau^{-1} h'$ iff

   there is a k such that $g|\phi|_p k|\psi|_p h$.

   The last 'iff' requires a construction. The extension lemma tells us that the constructions that we find for ϕ and ψ by induction hypothesis can be combined to one for $\phi.\psi$. (Check that the construction gives partially persistent and extending referent systems!)

5. $g|(\phi \to \psi)|_p h$ iff

   there are (σ, g'), (τ, h') such that $(\sigma, g')[(\phi \to \psi)](\tau, h')$ and $g = E_\sigma^{-1} g'$ & $h = E_\tau^{-1} h'$ iff

   there are (σ, g'), (τ, h') such that $\sigma = \tau$ & $g' = h'$ &

   $\forall(\rho, k') : \ (\sigma, g')[\phi](\rho, k') \ \exists(\mu, l') : \ (\rho, k')[\psi](\mu, l')$.

   Now the induction hypothesis, the extension lemma and the construction procedure tell us the this holds iff:

   for all k such that $g|\phi|_p k$ there is a l such that $k|\psi|_p l$.

□

References

[1] van Benthem, J.: 1989, 'Semantic parallels in natural language and computation', in *Logic Colloquium '87*, H. Ebbinghaus *etal.* (eds.), North Holland

[2] van Benthem, J.: 1991, 'General Dynamics', *Theoretical Linguistics*, pp. 159–201

[3] van Benthem, J.: 1991, *Language in Action*, North Holland

[4] Blackburn, P and Y. Venema: 1993, *Dynamic squares*, in Logic Group Preprint Series, no. 92, Utrecht University, (Heildelberglaan 8, 3584 CS Utrecht, Holland)

[5] Dekker, P.: 1992, 'An update semantics for dynamic predicate logic', in *Proceedings of the* 8^{th} *Amsterdam Colloquium*, P. Dekker, M. Stokhof (eds.), University of Amsterdam (Nieuwe Doelenstraat 15, Amsterdam, Holland)

[6] van Eijck, D.J.N.: 1991, *The dynamics of description*, Report CS-R9143, CWI (Kruislaan 413, Amsterdam, Holland)

[7] van Eijck, D.J.N. and F.-J. de Vries: 1991, *Dynamic interpretation and Hoare deduction*, Report CS-R9115, CWI (Kruislaan 413, Amsterdam, Holland)

[8] Fernando, T.: 1992, 'Transition Systems', in *Logics in AI — European workshop JELIA 1992*, Pearce D. and G. Wagner *eds.*, LNCS 633, Springer, Berlin

[9] Fine, K.: 1985, *Reasoning with Arbitrary Objects*, Aristotelian Society Series, 3, Blackwell, Oxford

[10] Frege, G.: 'Logische Mängel in der Mathematik', in *Nachgelassene Schriften*, Hermes, H. *et al* (eds.), 1969, University of Chicago Press, Chicago

[11] Groenendijk, J. and Stokhof, M.: 1991, 'Dynamic Predicate Logic', *Linguistics and Philosophy 14*, pp. 39–100

[12] Groenendijk, J. and Stokhof, M.: 1991, 'Two theories of dynamic semantics', in *Logics in AI — European Workshop JELIA '90*, J. van Eijck (ed.), pp. 55–64, LNCS 478, Springer, Berlin

[13] Harel, D: 1984, 'Dynamic Logic', in *Handbook of Philosophical Logic*, Gabbay, D & H. Günthner (eds.), Kluwer ,Dordrecht

[14] Heim, I: 1982, *The semantics of definite and indefinite noun phrases*, dissertation, University of Massachusets, Amherst

[15] Kamp, H.: 1981, 'A Theory of Truth and Semantic Representation', in *Formal Methods in the Study of Language*, Groenendijk *et al.* (eds.), CWI (Kruislaan 413, Amsterdam, Holland)

[16] Kamp, H. and U. Reyle: 1993, *From Discourse to Logic*, Kluwer, Dordrecht

[17] Kaplan D.: 1979, 'On the logic of demonstratives', in *Contemporary Perspectives in the Philosophy of Language* French, Uehling and Wettstein (eds.), pp. 401–412, University of Minnesota Press, Minneapolis

[18] Montague, R.: 1970, 'Universal Grammar', in *Theoria 36*

[19] Pratt, V.: 1991, 'Action Logic and Pure Induction', in *Logics in AI — European Workshop JELIA '90*, J. van Eijck (ed.), pp. 97–120, LNCS 478, Springer, Berlin

[20] de Rijke, M.: 1992, *A system of dynamic modal logic*, ILLC-prepublication LP-92-08, University of Amsterdam (Plantage Muidergracht 24, Amsterdam, Holland)

[21] Stalnaker, R.: 1978, 'Pragmatics', in *Semantics of natural language* Davidson, P. and G. Harman (eds.), Kluwer, Dordrecht

[22] Stalnaker, R.: 1978, 'Assertion', in *Syntax and Semantics 9: Pragmatics* P. Cole (ed.), Academic Press, New York

[23] Veltman, F.: 1990, *Defaults in update semantics*, in *Conditionals, Defaults and Belief Revision*, H. Kamp (ed.), Edinburgh, Dyana deliverable R.2.5.A

[24] Vermeulen, C.F.M.: 1991, 'Sequence semantics for dynamic predicate logic' *Journal of Logic Language and Information 2*

[25] Vermeulen, C.F.M.: 1992, *Incremental semantics for propositional texts* in Logic Group Preprint Series, no. 85, Utrecht University, (to appear in: *Notre Dame Journal of Formal Logic*)

[26] Visser, A.: 1992, *Actions under Presupposition* in Logic Group Preprint Series, no.76, Utrecht University (Heidelberglaan 8, 3584 CS Utrecht, Holland)

[27] Visser, A.: 1992, *Lazy and Quarrelsome Brackets* in Logic Group Preprint Series, Utrecht University (Heidelberglaan 8, 3584 CS Utrecht, Holland)

[28] Visser, A.: 199?, *Meanings in Time*, manuscript (Heidelberglaan 8, 3584 CS Utrecht, Holland)

[29] Zeevat, H.: 1989, 'A Compositional Approach to Discourse Representation Theory', *Linguistics and Philosophy 12*

[30] Zeevat, H.: 1991, *Aspects of discourse semantics and unification grammar*, PhD. thesis, University of Amsterdam (Nieuwe Doelenstraat 15, Amsterdam, Holland)