Prolegomena to the Definition of Dynamic Predicate Logic with Local Assignments

Albert Visser

Department of Philosophy, Utrecht University

Heidelberglaan 8, 3584 CS Utrecht, The Netherlands

email: `Albert.Visser@phil.ruu.nl`

October 21, 1997

Abstract

In this paper we subject the possible ways to define versions of DPL with local assignments to a thorough and detailed scrutiny. We hope that our treatment is suggestive of the proper 'abstract' view of dynamic logics that we would like to arrive at eventually.

Key words: Dynamic Predicate Logic, Scope, Natural Language, Meaning, File Semantics, Donkey, Monoid, Monoidal Processing, Partiality, Underflow, Transition Systems

MSC codes: 03B60, 03B65, 68S05

Contents

1 Introduction

1.1 The Case of the Lost Dog

A *A dog comes in. It looks around.*

If we try to analyse such a mini-discourse in predicate logic, we get something like:

A' $\exists x(\mathsf{dog}(x) \wedge \mathsf{comes\text{-}in}(x) \wedge \mathsf{looks\text{-}around}(x))$.

Such an analysis will enable us to evaluate the discourse in a compositional way in a given model. In this case, the value will be either the empty set or the set of all assignments. Now let's continue our discourse. E.g.:

B *A dog comes in. It looks around. It barks.*

The paraphrase of *it barks* will be something like $\mathsf{barks}(x)$ and its evaluation in the model will be the set of assignments which give x a value in $I(\mathsf{barks})$, the set of domain elements that bark, according to the model. There is no way in which we can reasonably combine the meanings of (A) and of it barks to obtain the meaning of (B). The reason is, of course, that in evaluating (A)/(A') we lost our dog. Starting with the sets of assignments that are the outputs of our evaluation process, we cannot extract the dog or dogs relevant for the evaluation. To be sure, we can paraphrase (B) in predicate logic. The paraphrase will be:

B' $\exists x(\mathsf{dog}(x) \wedge \mathsf{comes\text{-}in}(x) \wedge \mathsf{looks\text{-}around}(x) \wedge \mathsf{barks}(x))$.

However, we can only obtain this description from the earlier one by breaking open the syntax. The operation that does this is reflected by nothing in the semantics.

The state of affairs sketched above, is clearly unacceptable. That is best seen by imagining the discourse becoming very long. We definitely do understand the discourse before it is finished —if discourses ever finish in a clear sense. Also our understanding of the whole is obtained by combining our understanding of initial fragments with our interpretations of the newer bits. A good semantics should reflect this in a systematic way. Predicate logic evidently doesn't deliver the goods.[1]

At this point dynamic semantics comes to the rescue. It aims to model the interpretation proces itself and no just to produce the correct truth-conditional outcome of that process. The form of dynamic semantics needed for the case at hand is Dynamic Predicate Logic or DPL. This logic was introduced by Jeroen Groenendijk and Martin Stokhof in their classical paper [5]. The basic move

[1]This only tells us that predicate logic is not suitable as a *discourse semantics*. Since it was never intended as such by its founders nor studied as such in proof theory and model theory, the point is not in any way to be construed as a *criticism* of predicate logic as studied by the logical tradition.

of the new semantics is to take as meanings not sets of assignments, but, more generally, relations between assignments. The meaning of (A) now becomes, roughly, a relation that resets an incoming assignment on x to something that is, according to the model, an incoming, around looking dog.[2] The main point is here that on the output side we have not lost our dog. The output assignments all have incoming, around looking dogs under x.

We may think of the DPL-semantics in the following way. Assignments are hearer states —more precisely: states of the information receiving agent. The meanings of formulas are modifiers of these states. This new way of viewing assignments and meaning gives rise to new philosophical constraints on the formal model. For instance, in their semantics, Groenendijk and Stokhof opted for using total assignments —in consonance with the way the semantics of ordinary predicate logic is commonly set up. It is, however, rather intuitive to ask for finite hearer states. The values given to the variables reflect the previous cognitive activity of the hearer. Only finitely many variables have been considered. The others do, in a sense, not yet exist. Thus specifying the semantics for local assignments, i.e. assignments that are defined only on finitely many variables, yields an obvious and attractive variant of the original system. Only, to give such a variant there are a few details to be filled in .... This paper is about those details.

1.2 What to expect of this Paper

Dear reader, let me first tell you what *not* to expect of this paper.

- ▷ The paper does not contain new theorems of mathematical interest.
- ▷ The paper does not present new applications in either computer science or the semantics of natural language.
- ▷ The paper does yield, perhaps, some philosophical insight, but not in the form of any grand or broad thesis.
- ▷ The paper is not a self-contained presentation of existing materials.

So if the paper does not promise any of these desirable things, then what does it give? The paper is, in a sense, an extended definition of dynamic predicate logics with local assignments. We worry at length over such questions as *what are the signatures involved?* and *what is the proper treatment of undefinedness?*. What gain is to be expected of such protracted definitory activity and why worry so much about precisely this class of systems? The gain, I submitt, is twofold: it brings us closer to a truly modular approach and it stimulates reflection on certain philosophical issues.

[2]We ignore the temporal progression that is clearly present in the example. It is well known how to improve the representation to incorporate this progression.

We want to see dynamic systems as built up in a modular way. Modularity has several advantages.

1. It opens the way to the logical study of such systems. E.g. classical DPL can be viewed as a specialization of Discourse Relation Logic, DRL. DRL is to DPL as propositional logic is to classical predicate logic. However, the connection between DPL and DRL is tighter. For a study of the pair DPL/DRL the reader is referred to [8] and [16].

   Here is another example of how modularity leads to logical questions, taken from the present paper. We address the question of what an elementary action is. The answer to this question becomes relevant as soon as we see that a repertoire of elementary actions is precisely what is needed to specify a system of dynamic predicate logic with local assignments.

2. Modularity promises easy specification of variants. In the end we would like to specify dynamic systems by just setting certain parameters to the right 'values'. The work in this paper is, admittedly, just a very small step on the road to this grand aim. However, what else can one do but taking such small steps?

Building up a definition slowly is one kind of fuel for philosophical reflection. At the background of our efforts, the loftier philosophical issues are lurking darkly. Here is a sample of these issues.

1. *What is the nature of the variable?*
   We all know of Frege's extremely successful *Entmythologisierung* of the variable.[3] The variable was demoted from being a mysterious variable entity to being a syntactic object in a role. I predict that we will come to see Frege's explication as too successful. The variable harbours far more mystery than that. In the context of dynamic semantics we have come to see that there are suddenly more *kinds of variable occurrence*.[4] See [15]. Moreover, it seems that our system has to fulfill certain preconditions to view an occurrence as being of an understandable kind at all. These phenomena already point to the necessity of a more articulate theory of the variable and its occurrences. The aims of dynamic semantics necessitate a more refined view of the variable also along other lines. We will say a bit more on that below.

2. *What is the fundamental operation for combining meanings?*
   In logical linguistics function application is often taken to be the fundamental operation between meanings. A critical discussion of this choice

[3]The reader is referred to Frege's classical paper [4]. See Kit Fine's [3] for a delightful discussion.

[4]In the present paper we will see examples of universal-quantifier-like binding that still does not admitt α-conversion.

is beyond the scope of this paper. Let us just note that function application is in many respects a complicated operation without perspicuous mathematical properties. In dynamic semantics another operation rises to prominence: relation composition, or, more generally: a monoidal operation *the merge*. The merge is the meaning of the silences between the words. Could the merge usurp the place of function application? It is hard to see in general how that could be. E.g. in the systems we study in the present paper we will have in addition to the merge another operation: dynamic implication. Yet I think a further story can be told.

One option, the radical one, is to push a fully monoidal semantics by increasing the semantic memory used. I am a fan of this idea, but I have to admit that it remains to be seen whether it can be executed in a convincing way. See [12] and [13] for a first attempt along these lines.

The other option is to allow other operations that are not definable in terms of the merge, but to stipulate that they are uniquely determined by certain equations involving the merge, i.o.w. to demand a sort of implicit definability. Such a programme will work for dynamic implication[5], but what about generalized quantifiers ...?

3. *What is the proper treatment of partiality and presupposition?*
In linguistics the analysis of semantic presupposition is an important issue. In Computer Science we have the problem of handling undefinedness, underflow and the like. To many researchers these two things are not unconnected: presuppositions are viewed as preconditions for definedness. A basic intuition —already emerging in the Russell-Strawson debate— here is that undefinedness and error are basically different from falsity, In this paper we will run into some problems connected with partiality. We will explore a solution strategy that was invented by Martin van den Berg.

If we grant that reflection on the definition of dynamic systems can be useful, why look at the specific systems studied here? There seems to be a particularly destructive counter argument to the pursuit of the present paper. *We already know that the systems as studied here, are not good modules in the larger study of dynamic semantics.* Let me elaborate. The view of variables that is implemented in ordinary predicate logic, in ordinary DPL and also in the systems studied in the present paper, identifies variables with certain letters or strings with a definite semantical role. The variable x has as meaning the function $(\lambda f{\in}\mathsf{Ass}.f(x))$ of assignments to values in a given domain N. Etcetera. Philosophical considerations in combination with certain technical problems, like the

[5]To be precise, one can show that over a given monoid there can be at most one implication operation (or, equivalently: negation operation) satisfying Marco Hollenberg's axioms (as formulated in [8]). This result is due to Marco Hollenberg and myself. On the other hand, one can easily give examples of monoids that cannot be extended with a dynamic implication satisfying the axioms.

problem of building dynamic systems that support a good notion of information growth and the problem of combining DPL with Frank Veltman's update semantics for *maybe*, necessitated a richer view of the variable. At least three aspects of the variable should be distinghuished.

▷ *The underlying variable*
Here the variable is a file, peg or discourse referent. It is an abstract object lacking specific features. It is just a focus providing 'identity' to a body of information.

▷ *The manifest variable*
Under this heading we study the machinery necessary for identifying different occurrences of the same discourse referent as occurrences of the *same* discourse referent. This machinery may employ labeling, position in a structure, higher level inference ....

▷ *The semantics of the variable*
Here we consider the way in which contents are assigned to the discourse referent. We need not just have one object stored under a discourse referent. These objects may be stacked. We also may have stored other discourse referents under a discourse referent, etc.

Of course, the distinction between the manifest variable and the semantics of the variable may be blurred. Compare this to the pieces of a jig saw puzzle. The forms of the pieces regulate the way they are put together. What we see on the pieces is the content. However, we often employ what we see on the pieces to find neighbouring pieces.

The distinction of underlying file and file name is worked out in [14] and [17]. The solution of the problem of combining DPL with Veltman's Update Semantics is given in [6]. The reader is referred to [11] and [7] for the study of a stacking semantics. See [17] for the idea of iterated storage.

In terms of the three aspects, the present treatment can be characterized as follows. We assume that one discourse referent is associated rigidly with a fixed string, so that we may confuse the referent with the string. We identify occurrences as the same, simply by checking whether the associated strings are equal. The only thing that changes are the values stored 'in' or 'under' the referent. Finally, under any referent we store *at most one object.* (No stacking of objects and the like.)

So if the treatment of variables in the systems studied is thus restricted, then why restrict ourselves to them? I have two answers. First, both from the research strategic and from the didactical point of view, it is often better to first consider a simpler case. To study the richer variety of the variable we cannot escape using higher technology, in this case: category theory. I have good hope that in the richer treatment a number of ideas and ways of questioning of the simpler treatment will still be in force. Secondly, I feel that our picture of dynamics would simply be woefully incomplete if we did not know in some detail how the simplest case works out, even if this simplest case is not, ultimately, the one we will use. Thirdly, I think we will want to understand why precisely we are forced to treat things in a more involved way. Only by having a good grip on the simple systems can we see why they are not adequate for such-and-such a task.

2 A Budget of Signatures and Structures

In this section we present some signatures and structures that we are going to need. We will view a language simply as a free structure over a given set of generators in the, in a sense, most general class of structures around. These most general structures will be e-monoids of a given signature. e-monoids will be introduced in subsection 2.1. The semantic objects that we will be concerned with, will be elements of structures that form a subclass of the e-monoids of the relevant signature. In the case of DPL with total assignments these structures are *discourse relation algebras* or dra's. Discourse relation algebras are introduced in subsection 2.2. We can introduce dra's without speaking of predicate logic: dra's are to DPL as boolean algebras are to predicate logic. The basic concepts for the predicate logical side are introduced in subsection 2.3. These concepts will be put to work in the next section, where we introduce DPL.

2.1 Extended Monoids

In this subsection we introduce our basic class of structures, the e-monoids. Really everything we meet, including the languages, will be an e-monoid.

An *e-signature* Σ is a pair $\langle \mathsf{Func}, \mathsf{Ar}\rangle$, where Func is a finite, possibly empty, set and Ar is a function from Func to the natural numbers, including 0. An *e-monoid* of signature Σ, or, briefly, a Σ-*monoid*, is a structure $\mathcal{E} = \langle \mathcal{M}, J\rangle$, where $\mathcal{M} = \langle M, \mathsf{id}, \bullet\rangle$ is a monoid and J is a function on Func mapping F to an $\mathsf{Ar}(F)$-ary function on M. We wil often notationally confuse F with $J(F)$.

Let P be a, possibly empty, set. $\mathcal{L}_\Sigma(P)$, the (dynamic) language over P of signature Σ, is the free Σ-monoid on generators P. The free monoid is only fixed up to isomorphism. We will employ a specific representative of the equivalence class. It is given as follows. Let $\mathcal{L}_\Sigma(P) = \langle\langle L, \mathsf{id}, \bullet\rangle, J\rangle$

- ▷ The domain, L, of $\mathcal{L}_\Sigma(P)$ is the smallest set such that:
  - ∘ $P \subseteq L$
  - ∘ $\top \in L$
  - ∘ if $t, u \in L$ and $t \neq \top$ and $u \neq \top$, then $t.u \in L$
  - ∘ if $F \in \mathsf{Func}$, $\mathsf{Ar}(F) = n$ and $t_1, \ldots t_n \in L$, then $F(t_1, \ldots, t_n) \in L$
- ▷ $\mathsf{id} = \top$

- ▷ $t \bullet u = \begin{cases} t.u & \text{if } t \neq \top \text{ and } u \neq \top \\ u & \text{if } t = \top \\ v & \text{if } u = \top \end{cases}$
  Note the innocent overlap of the second and third case.

- ▷ Suppose $F \in \mathsf{Func}$, $\mathsf{Ar}(F) = n$ and $t_1, \ldots t_n \in L$, then $J(F)(t_1, \ldots, t_n) = F(t_1, \ldots, t_n)$

Note that we made the usual implicit assumption that things that are not *written* as identical *are* not identical. Our language is not much different from a traditional language, but for the following features. First, '.' does not function as an ordinary binary operation symbol, since it carries no brackets. A good analogy for the dot is the interword space of natural language. The corresponding monoidal operation is a lot like concatenation. Secondly, $\top$ is erased as soon as it becomes a proper part of a larger 'concatenation'. One can think of $\top$ as something like the empty string. We use $\top$, rather than the, in many respects more felicitous, id, to make our formulas read more like traditional formulas of Predicate Logic.

Why are we not using the ordinary notion of a language, but rather this somewhat exotic one? In the first place, I think that the notion of a language simply as a free object of a certain kind is an appealing one, also in the case of classical logic. It is especially attractive to formulate the definition of the semantics in terms of free objects in our context, because by omitting the brackets we lose unique reading. Our formulas become ambiguous. Still, since the 'ambiguity' is reflected by the associativity of the target structure of the interpretation function, our 'non-deterministic' recursion gives us unique values. Secondly, it seems to be rather elegant to think of the language itself as a dynamic object.[6] A third reason for our convention is our adherence to the grand philosophical idea that not function application, as in Montague Grammar, but a monoidal operation, the merge, is the fundamental operation on meanings. In the present way of doing things, the status aparte of the merge is emphasized.

We could very well place a more traditional language 'on top' of our $\mathcal{L}_\Sigma(P)$. We can translate the formulas of this language to those of $\mathcal{L}_\Sigma(P)$ by erasing the brackets for '.' and by erasing the occurrences of $\top$ with flanking dots.

Consider any e-monoid $\mathcal{E}$ of signature Σ. Let f be a mapping from P to $M_\mathcal{E}$. Then f can be extended in a unique way to an e-monoid morphism $[.]_\mathsf{f}$ from $\mathcal{L}_\Sigma(P)$ to $\mathcal{E}$. Let t, u be in $\mathcal{L}_\Sigma(P)$. We write

- ▷ $t =_{\mathcal{E},\mathsf{f}} u$ or $\mathcal{E}, \mathsf{f} \models t = u$ for: $[t]_\mathsf{f} = [u]_\mathsf{f}$

- ▷ $t =_\mathcal{E} u$ or $\mathcal{E} \models t = u$ for: $t =_{\mathcal{E},\mathsf{f}} u$, for all f

The following signature will play an important role.

[6]The objects of our language are reminiscent of DRS's. These are semi-syntactic objects with certain operations on them. In fact it is not that difficult to view DRS's as elements of an appropriate free structure.

▷ $[\bot,\to] := \langle\{\bot,\to\},\{\langle\bot,0\rangle,\langle\to,2\rangle\}\rangle$

We write $(t\to u)$ instead of $\to(t,u)$. We write $\neg t$ for $(t\to\bot)$.

2.2 Dynamic Relation Algebras

Dynamic relation algebras are an important class of e-monoids. They will prominently appear as meaning algebras.

A *dynamic relation algebra*[7] or *dra* on a non-empty set X is a structure $\mathcal{R}_X := \langle\langle \mathsf{Rel}(X), \mathsf{id}_X, ;\rangle, J\rangle$. Here $\mathcal{R}_X$ is a $[\bot,\to]$-monoid. We will also write $\mathcal{R}_X := \langle \mathsf{Rel}(X), \mathsf{id}_X, ;, \bot_X, \to\rangle$. We stipulate:

1. $\mathsf{Rel}(X)$ is the set of binary relations on X, i.e., $\mathsf{Rel}(X) := \wp(X\times X)$
2. id_X is the identity relation. If no confusion is possible we will drop the subscript and just write id.
3. A relation R is a *test* or *condition* if $R\subseteq \mathsf{id}_X$
4. The composition $R;S$ of R and S is defined by: $x(R;S)y :\Leftrightarrow \exists z\ xRzSy$ We distinguish ; from $\circ$ with: $x(R\circ S)y :\Leftrightarrow \exists z\ xSzRy$.)
5. $J(\bot) := \bot_X$. $\bot_X$ is the empty relation on X. Most of the time we will just write $\bot$.
6. $J(\to) :=\to$. Here $\to$ is dynamic implication. It is defined by

$$x(R\to S)y :\Leftrightarrow x=y \text{ and } \forall z(xRz \Rightarrow \exists u\ zSu).$$

   This notion of implication is originally due to Kamp ([10]). In its present form it was introduced by Groenendijk and Stokhof ([5]).

Consider $\mathcal{P}_X := \langle \wp X, X, \emptyset, \cap, \to\rangle$. Here $(Y\to Z) := ((X\setminus Y)\cup Z)$. The function $\mathsf{diag} : \wp X \to \mathsf{Rel}(X)$ with $\mathsf{diag}(Y) := \{\langle y,y\rangle \mid y\in Y\}$ preserves structure going from $\mathcal{P}_X$ to $\mathcal{R}_X$. E.g. $\mathsf{diag}(Y\cap Z) = \mathsf{diag}(Y);\mathsf{diag}(Z)$. The range of diag consists precisely of the conditions.

We write $\mathsf{dom}(R)$ for the domain of R. It is easy to see that

$$\neg R = \mathsf{diag}(X\setminus \mathsf{dom}(R)) \text{ and } \neg\neg R = \mathsf{diag}(\mathsf{dom}(R)).$$

Let $t\in\mathcal{L}_{[\to,\bot]}(P)$, $a\in X$ and $f\in \mathsf{Rel}(X)^P$. We write:

▷ $\mathcal{R}_X, \mathsf{f}, a \models t$ for: $a[t]_{\mathsf{f}}b$, for some b

[7]Our usage of the term *dra* diverges a bit, but for our purposes inessentially, from the usage of [8]. We use $\to$, where Hollenberg uses $\neg$. These operations are interdefinable. The use of $\neg$ often simplifies things. However, I can imagine classes of structures that can be viewed as 'generalizations' of the class of dra's, where the definition of $\to$ from $\neg$ fails, but where the usual reduction of $\neg$ to $\to$ still works.

- ▹ $\mathcal{R}_X \models t$ for: $\forall \mathsf{f}, a\ \exists b\ a[t]_{\mathsf{f}} b$, for some b
- ▹ $\models_{\mathsf{drl}} t$ for: $\forall X\ \mathcal{R}_X \models t$
- ▹ $t =_{\mathsf{dra}} u$ for: $\forall X\ \mathcal{R}_X \models t = u$

Note that $\models_{\mathsf{drl}} t$ iff $(\top \to t) =_{\mathsf{dra}} \top$. The valid principles for $=_{\mathsf{dra}}$ are axiomatized by Marco Hollenberg in [8]. In [16] I show that the valid principles for $=_{\mathsf{dra}}$ are precisely the valid schemes in the dra language for Dynamic Predicate Logic.

We will be interested in subalgebras of dra's. We will call such algebras: sdra's. Obviously, sdra's satisfy the same equations as dra's. Often the domain of such a subalgebra will be singled out by some set of properties $\mathcal{C}$. We will describe such an algebra as $\mathcal{R}_X(\mathcal{C})$. If $\mathcal{C} = \{P_1, \ldots, P_n\}$, we write $\mathcal{R}_X(P_1, \ldots, P_n)$.

2.3 Signatures and Models for Predicate Logic

Our next step, is the introduction of the notions needed to connect the relational algebras of the preceding section with the semantics of predicate logic.

An s-signature Λ is just a signature for Predicate Logic of the usual kind. Λ is a triple $\langle \mathsf{Pred}, \mathsf{Func}, \mathsf{Arg} \rangle$, where Pred is a finite, possibly empty, set of predicate symbols and Func is a finite, possibly empty, set of function symbols. Arg is a function from $\mathsf{Pred} \cup \mathsf{Func}$ to the natural numbers (including 0). We will assume that there is an element $=$ in Pred of arity 2.

A model $\mathcal{N}$ of signature Λ is a pair $\langle N, I \rangle$, Here N is a non-empty set. I assigns to P in Pred with $\mathsf{Ar}(P) = n$ a set of functions from $\{1, \ldots, n\}$ to N. I assigns to F in Func with $\mathsf{Ar}(F) = n$ a set of functions from $\{0, \ldots, n\}$ to N. Here we demand that for each $f : \{1, \ldots, n\} \to N$, there a unique g with $f \subseteq g \in I(F)$. We assume that $I(=) = \{f \in N^{\{1,2\}} \mid f(1) = f(2)\}$.

3 DPL introduced

In this section we introduce the classical system DPL of Jeroen Groenendijk and Martin Stokhof as presented in their classical paper [5]. This system is the paradigm for the systems considered later in this paper.

We define DPL as follows. Let an s-signature Λ be given. We will assume that we have only constants in Func, i.e. only elements of arity 0. We will write $c, c', \ldots$ for the constants and Con for Func. Let Var be a, possibly empty, set of variables. Define:

- ▹ $\mathsf{Ref} := \mathsf{Con} \cup \mathsf{Var}$
- ▹ $\mathsf{Cond} := \{Q(r_1, \ldots, r_n) \mid Q \in \mathsf{Pred},\ \mathsf{Arg}(Q) = n \text{ and } r_1, \ldots, r_n \in \mathsf{Ref}\}$
- ▹ $\mathsf{Reset} := \{\exists v \mid v \in \mathsf{Var}\}$

- ▹ $P_{\Lambda,\mathsf{Var}} := \mathsf{Cond} \cup \mathsf{Reset}$
- ▹ The language $\mathcal{L}^{\mathsf{dpl}}_{\Lambda,\mathsf{Var}}$ of DPL is $\mathcal{L}_{[\bot,\to]}(P_{\Lambda,\mathsf{Var}})$

Let a model $\mathcal{N} = \langle N, I\rangle$ of s-signature Λ be given. Let $\mathsf{Ass} := N^{\mathsf{Var}}$. We define

- ▹ For $f \in \mathsf{Ass}$ and $r \in \mathsf{Ref}$,

$$\tilde{f}(r) := |r|f := \begin{cases} f(r) & \text{if } r \in \mathsf{Var} \\ I(r)(0) & \text{if } r \in \mathsf{Con} \end{cases}$$

Meanings will be relations on Ass. We will make $\mathcal{R}_{\mathsf{Ass}}$ into our meaning algebra. Define:

- ▹ $[Q(r_1,\cdots,r_n)] := \mathsf{diag}(\{f{\in}\mathsf{Ass} \mid (\lambda i{\in}\{1,\ldots n\}\ |r_i|f) \in I(Q)\})$
  If we take $\rho := \lambda i{\in}\{1,\ldots n\}\ r_i$, then we can also write this as:
  $[Q(r_1,\cdots,r_n)] := \mathsf{diag}(\{f{\in}\mathsf{Ass} \mid (\tilde{f}\circ\rho) \in I(Q)\})$
- ▹ $[\exists v]$ is the relation *random-reset* om v. This relation is often written as: $[v :=?]$. It is defined by:

$$f[\exists v]g :\Leftrightarrow f \restriction (\mathsf{Var} \setminus \{v\}) = g \restriction (\mathsf{Var} \setminus \{v\})$$

  I.o.w. $f[\exists v]g$ means that f and g are the same, everywhere except, possibly, on v

We can extend [.] to the full language in a unique way, thus obtaining an e-monoid morphism between $\mathcal{L}^{\mathsf{dpl}}_{\Lambda,\mathsf{Var}}$ and $\mathcal{R}_{\mathsf{Ass}}$. This function is the dynamic interpretation function. Here are some pleasant abbreviations:

- ▹ $\neg(\phi)$ for: $(\phi \to \bot)$
- ▹ $?(\phi)$ for: $(\top \to \phi)$ (or, alternatively, for $\neg(\neg(\phi))$)
- ▹ $[x := c]$ for: $\exists x.x = c$
- ▹ $\forall x\,(\phi)$ for: $(\exists x \to \phi)$

If we compare the way DPL treats scope with the way Predicate Logic treats scope, we see that DPL's way is more like the way Natural Language does it. We remind the reader of Geach's famous Donkey Sentence:

- ▹ *If a farmer owns a donkey, then he beats it.*
  This sentence can be paraphrased by:
  $(\exists x.\mathsf{farmer}(x).\exists y.\mathsf{donkey}(y) \to \mathsf{beats}(x,y))$

There are several unsatisfactory aspects of the way DPL works. The first is the ad-hoc-ness of the way the atomic clauses are defined. We will repair this defect in a later paper devoted to the 'dynamics of argument places'. The second is the use of total assignments. If we want to view an assignment as a *hearer state* (and we do) it is not plausible that all resisters are filled already. We want only values in finitely many registers, corresponding to some finite past of filling, emptying and changing registers. The problem of partializing DPL is the subject of the present paper. The third is the fact that the total 'space' of meanings contains many transitions that are not meaningful in terms of the kind of process we want to think about. Think e.g. of the universal transition. Intuitions about kinds of occurrences of variables are often supported by the relations in the generated fragment, but not by all relations. See [15]. We wil not study this problem in the present paper. See e.g. [14], [15] and [17] for approaches that lead to restricted meaning spaces. The fourth is the fact that resetting is always both loss and gain of information, so that DPL does not support a notion of information growth. This fact creates problems e.g. when one tries to integrate DPL with Frank Veltman's Update Semantics. We will not address this problem in this paper. The reader is referred to [11] (and its companion paper [7]), [14], [6], [17]. A beautiful solution of the problem of integrating DPL with Update Semantics is given in [6]. The solution uses an idea from [14].

4 Local Assignments

4.1 A First Look at Local Assignments

In this subsection I will discuss the options of creating a version of DPL with assigments that are only defined on finitely many variables. I prefer to call such assigments *local* assigments rather than *partial* assignments. The reason for my preference is that the absence of a value for a variable v models *the absence of the variable itself* i.e. the fact that *v has not yet been introduced* or that *v has been thrown away*. The absence of a variable is different from the presence of a variable without an assigned value. Said in a different way, the assignments we consider are perfectly total *on their intended domain*.

The locus classicus for the treatment of DPL with local assignments is Martin van den Berg's PhD. Thesis, [1]. I will follow the main ideas of van den Berg's treatment. We fix a model $\mathcal{N} = \langle N, I \rangle$ and a set of variables Var.

Let $\mathsf{Ass}_{\mathsf{loc}}$ be the set of all partial functions from Var to N with finite domain. Let $V \subseteq \mathsf{Var}$. We write $\mathsf{Ass}_{\mathsf{loc}}(V)$ for the set of all partial functions from V to N with finite domain. Following a standard practice, we identify partial functions to N, with total functions to $N^* := N \cup \{*\}$, putting $f(v) = *$ for $v \notin \mathsf{dom}(f)$. We write $f(v) = g(v)$ for: either f and g are both defined on v and have the same value or both are undefined. So $f(v) = g(v)$ is an ordinary identity if we view f and g as total with the extended range. We will usually take $m, m', n \ldots$

as variables over N, and $a, a', b\ldots$ as variables over N^*. We can view $\mathsf{Ass_{loc}}$ as $\prod^{\mathsf{fin}}_{v\in\mathsf{Var}} N^*$, the subset of $\prod_{v\in\mathsf{Var}} N^*$ of infinite sequences that are $*$ almost everywhere.

It will be pleasant to go back and forth between relations on $\mathsf{Ass_{loc}}$ and relations on N^*. Here are some transformations.

- ▷ A function ρ from Var to relations on N^* is *a presentation* if either, (i) for all $v \in \mathsf{Var}$, $\rho(v) = \bot$ or, (ii) for no v, $\rho(v) = \bot$ and $\rho(v) = \mathsf{id}$, for almost all v.

- ▷ Suppose ρ is a presentation. $\prod_{v\in\mathsf{Var}} \rho$, is the relation R with

$$fRg :\Leftrightarrow \forall v{\in}\mathsf{Var}\ f(v)(\rho(v))g(v).$$

- ▷ Suppose α is a relation on N^*. Then α_v is the relation on $\mathsf{Ass_{loc}}$, specified by:

$$f(\alpha_v)g :\Leftrightarrow f(v)(\alpha)g(v) \text{ and } \forall w{\in}\mathsf{Var}\setminus\{v\}\ f(w) = g(w).$$

  So, in case α is $\bot$, α_v is $\prod_{v\in\mathsf{Var}} \rho$, where ρ is $\bot$ everywhere. In case α is not $\bot$, α_v is $\prod_{v\in\mathsf{Var}} \rho$, where $\rho(v) = \alpha$ and $\rho(w) = \mathsf{id}$, where $w \neq v$.

- ▷ Let R be a binary relation on $\mathsf{Ass_{loc}}$. R^v is the relation on N^* given by:

$$aR^v b :\Leftrightarrow \exists f, g\ f(v) = a \text{ and } g(v) = b \text{ and } fRg.$$

  R^v is the projection of R on 'coordinate' v.

- ▷ Let R be a binary relation on $\mathsf{Ass_{loc}}$. $\tilde{R} := \prod_{v\in\mathsf{Var}} R^v$. It is easily seen that $\tilde{(.)}$ is a *closure operation* w.r.t. $\subseteq$, i.e.

  - ◦ $R \subseteq S \Rightarrow \tilde{R} \subseteq \tilde{S}$,
  - ◦ $R \subseteq \tilde{R}$,
  - ◦ $\tilde{R} = \tilde{\tilde{R}}$

  We will say that R is *independent* if $R = \tilde{R}$.

Let V be any set of variables. We define, for $i, j \in \mathsf{Ass_{loc}}$,

$$i =_V j :\Leftrightarrow i \restriction V = j \restriction V.$$

We put $\approx_V := (=_{\mathsf{Var}\setminus V})$. So $\approx_V$ is the random reset w.r.t. the elements of V. We will write $\approx_v$ for $\approx_{\{v\}}$. Given the above conventions we can inroduce $\approx$ in an alternative way. Let $\approx$ be the universal relation on N^*. Then $\approx_v$ by our conventions will be random reset on v. Finally, if $V = \{v_1, \ldots, v_n\}$, we can put $\approx_V := \approx_{v_1}; \ldots; \approx_{v_n}$. Note that this works independently of order and multiplicity of the v_i's.

We do not want all relations on local assignments in our meaning algebra. We want only to allow relations that are, in some sense, finite. We will restrict ourselves to relations that are only sensitive to the values on a given finite V and that only modify values on V. We will say that R is a V-relation if, for any $v \notin V$, if fRg, then $f(v) = g(v)$ and, for any $a \in N^*$, $f[v := a]Rg[v := a]$. We will say that a relation is *local* if it is a V-relation, for some finite V. loc is the property of being local. $\mathsf{Rel}_{\mathsf{loc}}(\mathsf{Ass}_{\mathsf{loc}})$ will be the set of all local relations on $\mathsf{Ass}_{\mathsf{loc}}$. We will study domains of meanings that are contained in $\mathsf{Rel}_{\mathsf{loc}}(\mathsf{Ass}_{\mathsf{loc}})$. Later we will further modify this choice. It is easy to see that $\mathsf{Rel}_{\mathsf{loc}}(\mathsf{Ass}_{\mathsf{loc}})$ is closed under id, $;$, $\bot$ and $\rightarrow$. Moreover, every $\prod_{v \in \mathsf{Var}} \rho$, for presentation ρ, is in $\mathsf{Rel}_{\mathsf{loc}}(\mathsf{Ass}_{\mathsf{loc}})$. This implies that $\mathsf{Rel}_{\mathsf{loc}}(\mathsf{Ass}_{\mathsf{loc}})$ is closed under $\tilde{(.)}$. Here are some elementary insights about V-relations.

Theorem 4.1 *1. Suppose $R \neq \emptyset$. Let R^V be the relation on $\mathsf{Ass}_{\mathsf{loc}}(V)$ given by: $f(R^V)g :\Leftrightarrow \exists f', g' {\in} \mathsf{Ass}_{\mathsf{loc}}\ f \subseteq f',\ g \subseteq g'$ and $f'Rg'$. Then, the following are equivalent.*

(a) R is a V-relation.

(b) $((=_V; R; =_V) \cap \approx_V) \subseteq R \subseteq \approx_V$.

(c) $fRg \Leftrightarrow (f \upharpoonright V)R^V(g \upharpoonright V)$ and $f \upharpoonright (\mathsf{Var} \setminus V) = g \upharpoonright (\mathsf{Var} \setminus V)$.
(We may say that R is the 'product' of R^V and $\mathsf{id}^{\mathsf{Var} \setminus V}$.)

2. If R is a V-relation and $V \subseteq V'$, then R is a V'-relation.

3. If R is both a V- and a W-relation, then R is also a $(V \cap W)$-relation. It follows that if *R is a V-relation, then there is a smallest V_0 such that R is a V_0-relation. We will write $V(R)$ for this V_0.*

4. $V(\bot) = V(\mathsf{id}) = \emptyset$.

5. $V(R; S) \subseteq V(R) \cup V(S)$ and $V(R \rightarrow S) \subseteq V(R) \cup V(S)$. We do not necessarily have $V(R; S) = V(R) \cup V(S)$ or $V(R \rightarrow S) = V(R) \cup V(S)$.

6. $R \subseteq S$ does not *imply $V(R) \subseteq V(S)$.*

A good first choice to consider as a possible algebra of meanings is the sdra $\mathcal{R}_{\mathsf{Ass}_{\mathsf{loc}}}(\mathsf{loc})$. The sdra's that we consider will be sub sdra's of this one.

4.2 What are the Elementary Actions?

The basic ingredient determining a system with local assignments is the repertoire of *elementary actions*. In this subsection we will try to get a systematic view of these actions. We will see that there are finitely many types of elementary action, 24 to be precise.

We start with the question what the elementary actions w.r.t. a given finite set of variables are. An *elementary action*, or ea, w.r.t. a finite set of variables

V is as a first approximation an action that only modifies the values of elements of V and that operates on all variables independently and that is insensitive to specific values. We explicate this as follows.

Definition 4.2 R is a *elementary action* if:

1. R is a local.
2. R is independent.
3. Let σ be any function from N to N. We extend σ to N^*, by putting $\sigma(*) = *$. If fRg, then $(\sigma \circ f)\, R\, (\sigma \circ g)$.

❑

Here are some trivial, but useful properties of ea's.

Lemma 4.3

1. ea's are closed under id, $\perp$, composition (;) and $\rightarrow$.
2. Equality of domain is a bisimulation between local assignments w.r.t. the transition system of ea's. This means that if $\mathsf{dom}(f) = \mathsf{dom}(f')$ and fRg, then there is a g' with $\mathsf{dom}(g) = \mathsf{dom}(g')$ and $f'Rg'$. (Since equality of domain is an equivalence relation we need only the zig-clause.)

❑

We will say that a relation α on N^* is *functionally invariant* if for any function $\sigma : N \rightarrow N$, extended to N^* by putting $\sigma(*) = *$, we have: if $a(\alpha)b$, then $\sigma(a)(\alpha)\sigma(b)$. If is clear that R is an ea iff R is local, independent and if every R^v is functionally invariant. Thus, the problem of characterizing the ea's reduces to the problem of characterizing the functionally invariant relations α. To solve the problem we introduce an auxiliary category $\mathcal{D}$. The objects of $\mathcal{D}$ are d (for *defined*) and u (for *undefined*). The arrows are $\mathsf{d} := \mathsf{id}_{\mathsf{d}}$, $\mu : \mathsf{d} \rightarrow \mathsf{d}$, $\curvearrowleft : \mathsf{d} \rightarrow \mathsf{u}$, $\mathsf{u} := \mathsf{id}_{\mathsf{u}}$, $\curvearrowright : \mathsf{u} \rightarrow \mathsf{d}$. We stipulate: $\curvearrowleft ; \curvearrowright := \curvearrowright \circ \curvearrowleft := \mu$. A $\mathcal{D}$-set is a set of arrows of $\mathcal{D}$ that does not contain both d and μ. A quick count shows us that there are 24 $\mathcal{D}$-sets. We define a mapping $\langle . \rangle$ from arrows and from $\mathcal{D}$-sets to relations on N^* as follows.

- ▷ $a\langle \mathsf{d} \rangle b \;:\Leftrightarrow\; a = b \in N$,
- ▷ $a\langle \mu \rangle b \;:\Leftrightarrow\; a, b \in N$,
- ▷ $a\langle \curvearrowleft \rangle b \;:\Leftrightarrow\; a \in N,\ b = *$,
- ▷ $a\langle \mathsf{u} \rangle b \;:\Leftrightarrow\; a = b = *$,
- ▷ $a\langle \curvearrowright \rangle b \;:\Leftrightarrow\; a = *,\ b \in N$,
- ▷ $\langle Y \rangle \;=\; \bigcup\{\langle \phi \rangle \mid \phi \in Y\}$.

Here is our characterization theorem.

Theorem 4.4 *Suppose α is functionally invariant. Then, for some $\mathcal{D}$-set Y, $\alpha = \langle Y \rangle$. Moreover if $|N| \geq 2$, then the mapping $\langle . \rangle$ is injective on the $\mathcal{D}$-sets.*

Proof

Consider any transition $\langle a,b\rangle \in \alpha$. If $a = b \in N$, then $\langle a,b\rangle \in \langle \mathsf{d}\rangle \subseteq \alpha$. If $a \in N$, $b \in N$ and $a \neq b$, then $\langle a,b\rangle \in \langle \mu\rangle \subseteq \alpha$. Etcetera. So for every transition $\langle a,b\rangle$ in α we can find a morphism $\phi_{a,b}$ in $\mathcal{D}$, with $\langle a,b\rangle \in \langle \phi_{a,b}\rangle \subseteq \alpha$. Let 's choose $\phi_{a,b}$ in such a way that $\langle \phi_{a,b}\rangle$ is as large as possible. Specifically, in case $\langle \mu\rangle \subseteq \alpha$, we will always prefer μ to d. Now it is easily seen that $\{\phi_{a,b} \mid a(\alpha)b\}$ is a $\mathcal{D}$-set and $\alpha = \langle\{\phi_{a,b} \mid a(\alpha)b\}\rangle$. The second claim is trivial. ❑

We define the following operation on $\mathcal{D}$-sets:

$$Y;Z := \{y;z \mid y \in Y,\ z \in Z,\ y;z \text{ is defined }\} \setminus \{x {\in} \{\mathsf{d}\} \mid \curvearrowleft \in Y \text{ and } \curvearrowright \in Z\}.$$

(The substracted term removes d from our product, in case there would be a d and a μ in the product if we just had the first term. This unnatural clause would disappear under an alternative design choice: put d into a $\mathcal{D}$-set, whenever μ is in. We opted for the other choice to obtain maximally efficient notations for our actions.) It is easy to see that $\langle Y\rangle;\langle Z\rangle = \langle Y;Z\rangle$. If $\langle Y\rangle$ is idempotent, we define:

$$f\langle Y\rangle_V g \;:\Leftrightarrow\; f \approx_V g \text{ and } \forall v{\in}V\ f(v)\langle Y\rangle g(v).$$

Note that, $\langle Y\rangle_v = \langle Y\rangle_{\{v\}}$ and, by idempotency, $\langle Y\rangle_V;\langle Y\rangle_W = \langle Y\rangle_{V\cup W}$.

Here are some sample ea's.

- ▷ $\langle\emptyset\rangle_V = \bot$ (if $V \neq \emptyset$; note that $\langle Y\rangle_\emptyset = \mathsf{id}$, for any idempotent Y),
- ▷ $\langle \mu, \curvearrowleft, \mathsf{u}, \curvearrowright\rangle_V = \approx_V$,
- ▷ $\delta_V := \langle \mu, \mathsf{u}\rangle_V$, so we have:

  $$i(\delta_V)j \;:\Leftrightarrow\; i \approx_V j \text{ and } \mathsf{dom}(i) \cap V = \mathsf{dom}(j) \cap V.$$

  Thus, δ_V resets defined values in V arbitrarily to defined values, but sends undefined to undefined.
- ▷ $\langle \mathsf{d}, \mathsf{u}\rangle_V = \mathsf{id}$,
- ▷ $\iota_V := \langle \mathsf{d}\rangle_V$, so we have: $i(\iota_V)j \;:\Leftrightarrow\; i = j$ and $V \subseteq \mathsf{dom}(i)$.
- ▷ $\preceq_V := \langle \mathsf{d}, \mathsf{u}, \curvearrowright\rangle_V$, so we have: $i \preceq_V j :\Leftrightarrow i \approx_V j$ and $i \subseteq j$. Note that, on our chosen domain of asignments $\mathsf{Ass}_{\mathsf{loc}}$, $i \subseteq j$ iff $i \preceq_V j$, for some finite V.
- ▷ $\preceq^\sharp_V := \langle \mathsf{d}, \curvearrowright\rangle_V$, so we have: $i \preceq^\sharp_V j :\Leftrightarrow i \preceq_V j$ and $V \subseteq \mathsf{dom}(j)$.
- ▷ $\sqsubseteq^\sharp_V := \langle \mu, \curvearrowright\rangle_V$, so we have: $i \sqsubseteq^\sharp_V j :\Leftrightarrow i \approx_V j$ and $\mathsf{dom}(j) = \mathsf{dom}(i) \cup V$. We will use $i \sqsubseteq j$ for : $\mathsf{dom}(i) \subseteq \mathsf{dom}(j)$.

systematic	V:met	B:met	V:obj	B:obj	B:in words
$\langle \mu, \curvearrowleft, \mathsf{u}, \curvearrowright \rangle_x$	$\approx_x$	$\approx_x$	-	η'_x	ra
$\langle \mathsf{d}, \mathsf{u}, \curvearrowright \rangle_x$	$\preceq_x$	-	-	η_x	safe ra
$\langle \mathsf{d}, \curvearrowright \rangle_x$	$\preceq^\sharp_x$	-	-	$\eta_x \wedge x = x$	effective safe ra
$\langle \curvearrowright, \mathsf{u} \rangle_x$	-	-	ϵ_x	ϵ_x	guarded ra
$\langle \curvearrowright \rangle_x$	-	-	$\overset{x}{\curvearrowright}$	$\epsilon_x \wedge x = x$	effective guarded ra
$\langle \mu, \curvearrowright \rangle_x$	$\sqsubseteq^\sharp_x$	-	$\exists x$	-	-

Table 1: Names of kinds of Random Assignment

In table 1, we compare our nomenclature with the one employed by van den Berg (in [1]).[8]

Open Question 4.5 Definition 4.2 could be varied in several ways. E.g. we could omitt the independence clause or we could liberalize the clause on functional invariance by only considering invariance under permutations of N. (This last liberalization would make 'resetting a definite element to a strictly different one' an ea. It would be interesting to have some characterization theorems for sufficiently natural variants. Q

4.3 Properties of Actions

We introduce a number of salient properties of actions and have a look at which ea's have these properties. Further discussion of the virtues of the properties will be deferred till we have introduced repertoires of actions having these properties into our dynamic languages.

4.3.1 Domain Determinism

We define *domain determinism*, or *forward domain determinism*, or fdd, as follows.

$\triangleright$ fRg and $f'\,R\,g'$ and $\mathsf{dom}(f) = \mathsf{dom}(f') \Rightarrow \mathsf{dom}(g) = \mathsf{dom}(g')$

[8] *ra* stands for random assignment. *V:met* is Visser's name of the relation, B:met is van den Berg's name of the same. *V:obj* and *B:obj* are respectively Visser and van den Berg's symbols in dynamic languages denoting these relations. *B:in words* gives the verbal descriptions by van den Berg of the various relations. My own preference would be to use 'random assignment' only if really a definite value is assigned or re-assigned, i.e. in the $\sharp$-cases.

Clearly, domain determinism is preserved by ; and $\rightarrow$. Here are the domain deterministic ea's w.r.t. v:

$$\begin{array}{lll}
\langle\emptyset\rangle_v & \langle\mathsf{u}\rangle_v & \langle\curvearrowright\rangle_v \\
\langle\mathsf{d}\rangle_v & \langle\mathsf{d},\mathsf{u}\rangle_v & \langle\mathsf{d},\curvearrowright\rangle_v \\
\langle\mu\rangle_v & \langle\mu,\mathsf{u}\rangle_v & \langle\mu,\curvearrowright\rangle_v \\
\langle\curvearrowleft\rangle_v & \langle\curvearrowleft,\mathsf{u}\rangle_v & \langle\curvearrowleft,\curvearrowright\rangle_v
\end{array}$$

4.3.2 Backwards Domain Determinism

We define *backwards domain determinism*, or bdd, as follows:

▷ fRg and $f'\,R\,g'$ and $\mathsf{dom}(g) = \mathsf{dom}(g') \Rightarrow \mathsf{dom}(f) = \mathsf{dom}(f')$

Clearly, domain determinism is preserved by ; and $\rightarrow$. Here are the ea's w.r.t. v that are backwards domain deterministic.

$$\begin{array}{lll}
\langle\emptyset\rangle_v & \langle\mathsf{u}\rangle_v & \langle\curvearrowleft\rangle_v \\
\langle\mathsf{d}\rangle_v & \langle\mathsf{d},\mathsf{u}\rangle_v & \langle\mathsf{d},\curvearrowleft\rangle_v \\
\langle\mu\rangle_v & \langle\mu,\mathsf{u}\rangle_v & \langle\mu,\curvearrowleft\rangle_v \\
\langle\curvearrowright\rangle_v & \langle\curvearrowright,\mathsf{u}\rangle_v & \langle\curvearrowright,\curvearrowleft\rangle_v
\end{array}$$

We will call relations that are both back- and forwards deterministic: *doubly domain deterministic* or ddd.

4.3.3 The Zig-property

We define the zig-property, or zig, as follows:

▷ fRg and $f \subseteq f' \Rightarrow \exists g'\; f'\,R\,g'$ and $g{\subseteq}g'$.

This property is sometimes also called the forward property.[9] The zig-property is one form of *monotonicity* of relations. It demands that, for any transition, extension of the incoming assignment leads to an 'extending' transition. Clearly, the zig-property is preserved by ;. We will see that it is not preserved by $\rightarrow$. The zig-property is preserved under $\neg\neg$, i.e. $\mathsf{diag} \circ \mathsf{dom}$. $\langle Y\rangle_v$ has the zig-property iff

1. $u \in Y \Rightarrow (d \in Y$ or $\mu \in Y$ or $\curvearrowleft\, \in Y)$, and
2. $\curvearrowright\, \in Y \Rightarrow \mu \in Y$

Domain deterministic ea's with the zig-property are a particularly important class. Here are the domain deterministic ea's w.r.t. v with the zig-property.

$$\begin{array}{lll}
\langle\emptyset\rangle_v & & \\
\langle\mathsf{d}\rangle_v & \langle\mathsf{d},\mathsf{u}\rangle_v & \\
\langle\mu\rangle_v & \langle\mu,\mathsf{u}\rangle_v & \langle\mu,\curvearrowright\rangle_v \\
\langle\curvearrowleft\rangle_v & \langle\curvearrowleft,\mathsf{u}\rangle_v &
\end{array}$$

[9]Stricly speaking, we should say that R has the zig-property, w.r.t. $\subseteq$, or, equivalently, that $\subseteq$ has the zig-property w.r.t. R.

We will discuss the virtues of these various properties later.

5 Interpretation of Predicate Logical Atomic Formulas

Our next step is to reflect upon the proper definition of $[P(r_1,\ldots,r_n)]$. We define $\tilde{f}(r)$ (or $|r|f$) as before, noting that now $\tilde{f}$ will be a partial function. Let $V := \{r_1,\ldots,r_n\} \cap \mathsf{Var}$, $C := \{r_1,\ldots,r_n\} \cap \mathsf{Con}$. Which relation shall we use for $[P(r_1,\ldots,r_n)]$? In the first place, it should be functionally obtained from $I(P)$, $I(c)$, for $c \in C$, and ρ with $\rho(i) = r_i$. We also want $[P(r_1,\ldots,r_n)]$ to be a V-relation. Consider an incoming f. In case f provides values for all elements of V, our path is clear, the definition should simply mimick the classical one. What if f is undefined for some element of V? I feel that the most natural option is to make the transition fail. This gives the following definition.

- ▷ $f[P(r_1,\ldots,r_n)]_1 g :\Leftrightarrow f = g$ and $\tilde{f} \circ \rho \in I(P)$

There are, however, all kinds of alternative. The most obvious one is this. We could first extend F arbitrarily on the elements of v on which it is undefined and then proceed classically. Thus we are led to the following definition:

- ▷ $f[P(r_1,\ldots,r_n)]_2 g :\Leftrightarrow f \subseteq g$, $\mathsf{dom}(g) = \mathsf{dom}(f) \cup V$ and $\tilde{g} \circ \rho \in I(P)$

We have:

- ▷ $[P(r_1,\ldots,r_n)]_1 \subseteq \iota_V$, $[P(r_1,\ldots,r_n)]_2 \subseteq \preceq^{\sharp}_V$
- ▷ $[P(r_1,\ldots,r_n)]_2 = \preceq^{\sharp}_V; [P(r_1,\ldots,r_n)]_1$

The choice of taking $[P(r_1,\ldots,r_n)] := [P(r_1,\ldots,r_n)]_2$ is considered by Jan Jaspars and Emiel Krahmer in the context of DRT. They call their theory DRT-p. See their paper [9]. The semantics so obtained is surpassingly strange. E.g. it is not hard to see that the following holds (interpreting $[.]$ on the atomic formulas by $[.]_2$).

$$\begin{array}{c}\mathcal{N}, \emptyset \models (x = x \to (y = y.(z = z \to P(x,y,z)))) \text{ iff} \\ \forall x\, \exists y\, \forall z\, P(x,y,z) \text{ is true in } \mathcal{N} \text{ in classical Predicate Logic.}\end{array}$$

An occurrence $\mathfrak{o}$ of a variable v in a formula will function as a quantifier if (i) the incoming assignment is undefined on v, and (ii) if for all earlier occurences $\mathfrak{o}'$ of v there is an implication occurrence containing $\mathfrak{o}'$ but not $\mathfrak{o}$. Thus, we can build a dynamic logic without quantifiers on this idea. Here is a list of curiosities (always interpreting $[.]$ on the atomic formulas by $[.]_2$).

1. We may have $\mathcal{N}, \emptyset \not\models (x = x \to Q(x))$ and $\mathcal{N}, \emptyset \models x = x.(x = x \to Q(x))$. Thus, conjoining an atomic predicate in front can make things more true. So our predicates behave in quite different ways from what we are used to.

2. It is possible that $\mathcal{N}, \emptyset \models R(x)$ and $\mathcal{N}, \{\langle x, n\rangle\} \not\models R(x)$. So truth can be lost when the incoming assignment carries more[10] information.[11]

3. As becomes clear from our first two examples the distinction between free variables and bound variables goes overboard. Except in rare cases, we do not have α-reduction. E.g.

$$[(P(x) \to Q(x))] \neq [(P(y) \to Q(y))].$$

4. The interpretations of our atomic formulas are not conditions any more. The usual equality of $[P(v)]$ and $[\neg\neg P(v)]$ will fail.

All in all, I think that the idea has some attractions —look, everything without visible quantifiers!— but that it is not, ultimately, a good idea.

A third option would be to follow the Jaspars-Krahmer way first by extending the incoming assignment to satisfy the predicate, but afterward resetting it to its original value. Formally this option can be represented as:

$$[P(r_1, \ldots, r_n)]_3 = \neg\neg(\preceq^\sharp_V; [P(r_1, \ldots, r_n)]_1).$$

Under this option we might have: $\mathcal{N}, \emptyset \models P(x); Q(x)$, without there being a single element in $\mathcal{N}$ satisfying both P and Q.

A fourth option is to test whether the atomic formula holds for *all assignments* extending the given one that are total on V. This would be a kind of van Fraassen style interpretation. Formally this option can be represented as:

$$[P(r_1, \ldots, r_n)]_4 = (\preceq^\sharp_V \to [P(r_1, \ldots, r_n)]_1).$$

This option will give the free variables in negated atomic formulas an existential effect. So we might have: $\mathcal{N}, \emptyset \models \neg P(x); \neg Q(x)$, without there being a single element in $\mathcal{N}$ in the intersection of the complements of the interpretations of P and Q.

I strongly feel that the first option is the most natural. So I will choose it: $[P(r_1, \ldots, r_n)] := [P(r_1, \ldots, r_n)]_1$. Note that our choice gives us: $[v = v] = \iota_v$ and $[c = c] = \mathsf{id}$.

We have fixed the interpretations of the atomic predicate logical formulas. We will introduce a notion of *strict local condition* or *strict local test* that in harmony with our choice. R is *strict local condition* or a *strict local test* if, for some finite V, R is a V-relation and $R \subseteq \iota_V$. We introduce a corresponding diagonal function. Let V be a finite set of variables. Let $F \subseteq N^V$. Then:

[10]Perhaps, it would be better to say that in this semantics also *the lack of a value* can be informative.

[11]It might be argued that the first example concerning the difference of $(x = x \to Q(x))$ and $x = x.(x = x \to Q(x))$ is without force, since 'the DRS-construction algorithm will always provide the right formulas'. However, (i) it seems to be putting the cart before the horse to defend a semantics by requiring syntactic preprocessing and (ii) the DRS-construction algorithm has no power whatsoever over the incoming assignment, so the second example retains full force.

- ▷ $f(\mathsf{diag}_{\mathsf{loc}}(V,F))g :\Leftrightarrow f = g,\ V \subseteq \mathsf{dom}(f)$ and $(f \restriction V) \in F$.

The strict local conditions are precisely the elements in the range of $\mathsf{diag}_{\mathsf{loc}}$. Clearly, the $[P(r_1,\ldots,r_n)]_1$ are strict local conditions.[12]

6 What is a Local Dynamic Predicate Logic?

At this point we have everything in place to explain what a local dynamic predicate logic, or ldpl, is. An ldpl is given by an indexed set ℓ of $\mathcal{D}$-sets and a set $\mathcal{C}$ of properties of relations.[13] We demand that the properties in $\mathcal{C}$ are closed under the operations of dra's and contain the strict local tests. ℓ can be viewed as a function from a set of names of elementary action types, EA, to $\mathcal{D}$-sets viewed as elementary action types. We will call ℓ a repertoire of elementary actions. Fix ℓ and $\mathcal{C}$.

We define $\mathsf{DPL}(\ell,\mathcal{C})$ as follows. Let an s-signature Λ be given. We will assume that we have only constants in Func, i.e. only elements of arity 0. We will write $c, c', \ldots$ for the constants and Con for Func. Let Var be a, possibly empty, set of variables. Define:

- ▷ $\mathsf{Ref} := \mathsf{Con} \cup \mathsf{Var}$
- ▷ $\mathsf{Cond} := \{P(r_1,\ldots,r_n) \mid \mathsf{Ar}(P) = n \text{ and } r_1,\ldots,r_n \in \mathsf{Ref}\}$
- ▷ $\mathsf{Reset} := \{a_v \mid a \in \mathsf{EA} \text{ and } v \in \mathsf{Var}\}$. In practice we will not always follow the subscript convention and write e.g. $\exists v$ and $\overset{v}{\curvearrowright}$ for elements of Reset.
- ▷ $P_{\Lambda,\mathsf{Var},\mathsf{EA}} := \mathsf{Cond} \cup \mathsf{Reset}$
- ▷ The language $\mathcal{L} := \mathcal{L}^{\mathsf{dpl}}_{\Lambda,\mathsf{Var},\mathsf{EA}}$ of $\mathsf{DPL}(\ell,\mathcal{C})$ is $\mathcal{L}_{[\bot,\to]}(P_{\Lambda,\mathsf{Var},\mathsf{EA}})$

Let a model $\mathcal{N} = \langle N, I\rangle$ of s-signature Λ be given. We define

- ▷ For $f \in \mathsf{Ass}_{\mathsf{loc}}$ and $r \in \mathsf{Ref}$,

$$\tilde{f}(r) := |r|f := \begin{cases} f(r) & \text{if } r \in \mathsf{Var} \\ I(r)(0) & \text{if } r \in \mathsf{Con} \end{cases}$$

Our meaning algebra will be $\mathcal{R} := \mathcal{R}_{\mathsf{Ass}_{\mathsf{loc}}}(\{\mathsf{loc}\} \cup \mathcal{C})$. So we always demand that our relations are local. Define:

- ▷ $[P(r_1,\cdots,r_n)] := [P(r_1,\cdots,r_n)]_1$

[12]We reserve 'local condition' for local relations that are conditions in the sense of being sub-relations of the diagonal. Note that implications of local relations are always local conditions, but not necessarily strict local conditions.

[13]These properties should be uniformly defined in some way, so that it makes sense to apply them to the relations over different domains.

▷ $[a_v] := \langle \ell(a) \rangle_v$

We can extend [.] to the full language in a unique way, thus obtaining an e-monoid morphism between $\mathcal{L}$ and $\mathcal{R}$. [.] is our interpretation function.

Some further discussion on equivalence of repertoires and sameness of logics would be possible. However, since the matter is not all that urgent, we will not address the issue in this paper.

7 A Repertoire of Doubly Deterministic Actions

Do we want to impose further properties on the semantic domains we want to allow? Such restrictions can be desirable (i) for reasons of modelling and (ii) for metamathematical reasons. As an example of (i) we already argued for restricting ourselves to local relations. The restriction to the zig-property that we will consider in section 9 can be viewed as a restriction that is entailed by viewing undefinedness as 'lack of information'. We refer the reader to [15] for examples to the effect that the very notion of 'kind of variable occurrence' rests on appropriate restrictions of the domain. The metamathematical applications are of a simple kind: to show undefinability of a relation R in the DPL-language it is sufficient to produce a property P of relations that is closed under the DPL-operations and such that R does not have P.

Any property we impose must be a fortiori a property of the ea's and, thus, of the corresponding ea types must satisfy a corresponding induced property. We will consider domains with the following choices of properties: ddd and $\{\mathsf{fdd}, \mathsf{zig}\}$. In both cases the action $\preceq_x^\sharp$ that we encountered in our discussion of the work of Jaspars and Krahmer is excluded. I do not claim that the choices considered are the only reasonable ones. Some distinguished researchers in the area ended up with different ones. Martin van den Berg, for example, in his [1] opts for the ea type ϵ, which is not domain deterministic and does not have the zig-property.

Let's first consider the doubly domain deterministic ea's. Here is the list of its elements.

$$\begin{array}{lll}
\langle \emptyset \rangle_v & \langle \mathsf{u} \rangle_v & \langle \curvearrowright \rangle_v \\
\langle \mathsf{d} \rangle_v & \langle \mathsf{d}, \mathsf{u} \rangle_v & \\
\langle \mu \rangle_v & \langle \mu, \mathsf{u} \rangle_v & \\
\langle \curvearrowleft \rangle_v & & \langle \curvearrowleft, \curvearrowright \rangle_v
\end{array}$$

We consider the following repertoire $\mathfrak{d}$ of types:

$$\begin{array}{ccc}
\curvearrowright & \mapsto & \{\curvearrowright\} \\
\curvearrowleft & \mapsto & \{\curvearrowleft\}
\end{array}$$

We write: $\overset{v}{\curvearrowright}$ instead of $\curvearrowright_v$, and $\overset{v}{\curvearrowleft}$ instead of $\curvearrowleft_v$. Let's call the logic we build on this repertoire $\mathsf{d}\text{-}\mathsf{DPL} := \mathsf{DPL}(\mathfrak{d}, \mathsf{ddd})$. d stands either for determinism or for

Dekker.[14] Here is a list of the definable and undefinable doubly deterministic ea's of our language.

- ▷ $\langle\emptyset\rangle_v = \bot = [\bot] = [\overset{v}{\curvearrowright}.\overset{v}{\curvearrowright}] = [v = v.\overset{v}{\curvearrowright}] = [\overset{v}{\curvearrowleft}.v = v] = [\overset{v}{\curvearrowleft}.\overset{v}{\curvearrowleft}]$.
- ▷ $\langle \mathsf{u}\rangle_v = [\neg(v = v)]$.
- ▷ $\langle\curvearrowright\rangle_v = [\overset{v}{\curvearrowright}]$.
- ▷ $\langle \mathsf{d}\rangle_v = \iota_v = [v = v]$.
- ▷ $\langle \mathsf{d}, \mathsf{u}\rangle_v = \mathsf{id} = [\top]$.
- ▷ $\langle\mu\rangle_v = [\overset{v}{\curvearrowleft}.\overset{v}{\curvearrowright}]$.
- ▷ $\langle\mu, \mathsf{u}\rangle_v$ is not definable.
- ▷ $\langle\curvearrowleft\rangle_v = [\overset{v}{\curvearrowleft}]$.
- ▷ $\langle\curvearrowleft, \curvearrowright\rangle_v$ is not definable.

$\overset{v}{\curvearrowright}$ and $\overset{v}{\curvearrowleft}$ function as a sort of brackets creating an environment in which v has a certain value. The system is 'fussy': we jump to 'error', i.e. $\emptyset$, as soon as we want to use a variable that is not declared or as soon as we try to introduce a variable into an environment where it is already present.

In effect, our $\overset{x}{\curvearrowright}$ is the existential quantifier as employed by Paul Dekker in [2]. Only Paul Dekker obtains the effect in a somewhat curious way. He *defines* the quantifier as $\preceq^{\sharp}_{x}$ and then puts *extra conditions* on definedness in the semantic clauses. That is not allowed in the compositional game as I perceive it. Admittedly, *our* definition as it stands still conflates undefinedness and falsity. We will remedy this in section 8.

Here are some examples.

- ▷ $\overset{x}{\curvearrowleft}.P(x)$
  This formula always gives 'error', since $P(x)$ asks for a value of x. But any such value has just been thrown away by $\overset{x}{\curvearrowleft}$.
- ▷ $\overset{x}{\curvearrowright}.P(x).\overset{x}{\curvearrowleft}.\overset{x}{\curvearrowright}.Q(x)$
  This formula behaves pretty much like $\exists x.P(x).\exists x.Q(x)$ in ordinary DPL. Only it is obligatory that the incoming assignment does not have a value on x.
- ▷ $(\overset{x}{\curvearrowright} \to P(x))$
  This formmula behaves in many respects like $\forall x(P(x))$. However, if the

[14]Our $\curvearrowright$ is very much like Paul Dekker's existential quantifier in DPLE.

incoming assignment has a value for x, this formula will become true, since there is no $\overset{x}{\curvearrowright}$-transition. Thus,

$$[(\overset{x}{\curvearrowright} \to x = x)] = \mathsf{id} \text{ and } [(\overset{x}{\curvearrowright} \to \neg(x = x))] = \langle \mathsf{d} \rangle_x .$$

Why do we not get 'error', when no value of x is provided? Evidently, this is because the failing transition occurs negatively. Below we will discuss a strategy to avoid this phenomenon. See section 8. Note that we do not have α-conversion: $[(\overset{x}{\curvearrowright} \to P(x))] \neq [(\overset{y}{\curvearrowright} \to P(y))]$.

▷ $(\overset{x}{\curvearrowleft}.\overset{x}{\curvearrowright} \to P(x))$
This formula behaves a lot like $\forall x(P(x))$. This time, if the incoming assignment has no value for x, the formula becomes true. Thus,

$$[(\overset{x}{\curvearrowleft}.\overset{x}{\curvearrowright} \to x = x)] = \mathsf{id} \text{ and } [(\overset{x}{\curvearrowleft}.\overset{x}{\curvearrowright} \to \neg(x = x))] = \langle \mathsf{u} \rangle_x .$$

Because of the non-monotonic character of the actions in our repertoire the semantics we are considering now still has some remarkable features, like the failure of α-conversion. Still it seems to support a notion of free variable.[15]

8 DPL with Local Assignments and Contexts

8.1 Error versus Falsity

Can we have a version of DPL with local assignments with the zig property? It does not seem possible. The reason is very simple. $[\neg(x = x)] = \langle \mathsf{u} \rangle_x$, and $\langle \mathsf{u} \rangle_x$ does not have the zig-property. The semantics of $\neg(x = x)$ rests firmly on choices we have already made. So what can we do?

The fact that $[\neg(x = x)] = \langle \mathsf{u} \rangle_x$ is not very satisfactory anyway. Why not? Well, $x = x$ is 'not defined' on an incoming assignment f that does not provide a value for x. This 'undefinedness' expresses itself by there not being an $[x = x]$-transition from f. Thus $x = x$, becomes a test for there being of a value stored in x. Now one would expect $\neg(x = x)$ also to be 'undefined' on incoming assignments not carrying a value for x. However, the 'test' we specified as the semantics of $x = x$ 'moves to a negative place' if we evaluate $\neg(x = x)$. Thus, contrary to our expectation, on assigments not defined on x, $\neg(x = x)$ will be counted true, i.e. it will pass on f.

The basic idea for the solution is to treat the fact that a transition does not occur because of phenomena like a variable not carrying a value not in the same manner as falsity. We will distinguish 'error' from 'falsity'. We take care that the test for definedness associated with $x = x$ does *not* 'shift to a negative place'

[15] This remark is deliberately vague. A closer analysis of kinds of free and bound variable occurrence, would call for merging the discussion of this paper with ideas from [15] and [17]. Such a discussion would carry us far beyond the scope of the present paper.

when we evaluate $\neg(x = x)$. We will treat 'x having a value' as (analogous to) a *presupposition*. "John is not a bachelor" carries the presupposition of John being both male and adult. This is also precisely the presupposition of "John is not a bachelor". So presuppositions are not shifted to a negative place if the sentence is negated.

The technique that we will present in this section is a minor modification of the approach of Martin van den Berg, developed in his [1].

8.2 The Pair Semantics

To modify our approach to distinguish error from falsity, our new semantic objects will be pairs of local relations $\langle R^+, R\rangle$, where (i) R^+ is an ea and (ii) $R \subseteq R^+$. Here R^+ has the role of a 'context' or 'presupposition'. R^+ is the companion of R in which we have *abstracted away from all contentual information*. As we will see the R^+'s transform independently of the R's, but not vice versa. We will call the class of pairs so defined: $\mathsf{p\text{-}REL}(\mathsf{Ass_{loc}})$. ($\mathsf{p}$ for *pair* or for *presupposition*.) We define the following operations on $\mathsf{p\text{-}REL}(\mathsf{Ass_{loc}})$.

- ▷ $\mathsf{id} := \langle \mathsf{id}, \mathsf{id}\rangle$,
- ▷ $\bot := \langle \mathsf{id}, \emptyset\rangle$,
- ▷ $\langle R^+, R\rangle ; \langle S^+, S\rangle := \langle R^+; S^+ \, , \, R; S\rangle$,
- ▷ $\langle R^+, R\rangle \to \langle S^+, S\rangle := \langle \neg\neg(R^+; S^+) \, , \, \neg\neg(R^+; S^+); (R; \neg\neg S^+ \to S)\rangle$.[16]

It is immediate that $\mathsf{p\text{-}REL}(\mathsf{Ass_{loc}})$ is closed under our operations. Remember that $\neg\neg(U) = (\mathsf{id} \to U) = (\mathsf{diag} \circ \mathsf{dom})(U)$ and, thus,

$$\begin{aligned}\neg\neg(R^+; S^+); (R; \neg\neg S^+ \to S) &= \neg\neg(R^+; S^+) \cap (R; \neg\neg S^+ \to S)\\ &= \mathsf{diag}\,(\mathsf{dom}(R^+; S^+) \cap \mathsf{dom}(R; \neg\neg S^+ \to S)).\end{aligned}$$

Our semantic e-monoid will be a structure of the form

$$\mathcal{S} := \mathcal{S}_{\mathsf{Ass_{loc}}}(\mathcal{C}) := \langle \mathsf{p\text{-}REL}(\mathsf{Ass_{loc}}), \mathsf{id}, ;, \bot, \to\rangle(\mathsf{loc}, \mathcal{C}).$$

Here $\mathcal{C}$ is a set of properties of pairs. We demand that each of these properties is closed under the operations on the pairs and contains the strict local tests. A strict local test is in our new set-up a pair of the form $\langle \iota_V, S\rangle$, where S is a strict local test on V in the ordinary sense. If P is a property of relations, then we will say that $\langle R^+, R\rangle$ has property P iff both R^+ and R have P. $\mathcal{S}$ is not a dra, not even modulo isomorphism. $\mathsf{DPL_p}(\ell, \mathcal{C})$ is defined like $\mathsf{DPL}(\ell, \mathcal{C})$. Except that:

- ▷ We replace the meaning algebra $\mathcal{R}$ by $\mathcal{S}$.

[16] We employ the convention that ; binds stronger than $\to$.

▷ $[a_v] := \langle\langle\ell(a)\rangle_v, \langle\ell(a)\rangle_v\rangle$.

▷ Let $V := \{r_1, \ldots, r_n\} \cap \mathsf{Var}$. Then

$$[P(r_1, \ldots, r_n)] := \langle\iota_V, [P(r_1, \ldots, r_n)]_1\rangle.$$

Note that:

▷ $\neg\langle R^+, R\rangle := (\langle R^+, R\rangle \to \bot) = \langle\neg\neg R^+ \,,\, \neg\neg R^+; \neg R\rangle$.

Example 8.1 We give an example to the effect that $\mathcal{S}_{\mathsf{Ass}_{\mathsf{loc}}}$ is not a dra. Consider $[\neg(x = x); x = x]$. In a logic based on a dra this will be $[\bot]$. However, we will get:

$$\begin{array}{rcl} [\neg(x = x); x = x] & = & \neg\langle\iota_x, \iota_x\rangle; \langle\iota_x, \iota_x\rangle \\ & = & \langle\iota_x \,,\, \iota_x; \neg\iota_x\rangle; \langle\iota_x, \iota_x\rangle \\ & = & \langle\iota_x; \iota_x \,,\, \bot; \iota_x\rangle \\ & = & \langle\iota_x, \bot\rangle \end{array}$$

Thus $[\neg(x = x); x = x] \neq [\bot]$. Hence $\mathcal{S}_{\mathsf{Ass}_{\mathsf{loc}}}$ is not a dra. ❑

Example 8.2 Why did we define our implication as

$$\langle\neg\neg(R^+; S^+) \,,\, \neg\neg(R^+; S^+); (R; \neg\neg S^+ \to S)\rangle$$

and not simply as the more obvious

$$\langle\neg\neg(R^+; S^+) \,,\, \neg\neg(R^+; S^+); (R \to S)\rangle?$$

Consider $(\epsilon_x \to x = x)$. Here:

▷ $[\epsilon_x] = \langle\langle\mathsf{u}, \curvearrowright\rangle_x, \langle\mathsf{u}, \curvearrowright\rangle_x\rangle$,

▷ $[x = x] = \langle\iota_x, \iota_x\rangle$.

A quick computation shows:

$$[(\epsilon_x \to x = x)] = \langle\langle\mathsf{u}\rangle_x, \langle\mathsf{u}\rangle_x\rangle.$$

However, if we compute $[(\epsilon_x \to x = x)]$ using the alternative definition we obtain: $\langle\langle\mathsf{u}\rangle_x, \bot\rangle$ as value. The reason is that, under the alternative definition, we also take the ϵ_x-transition from f to f, in case $f(x) = *$ into consideration. But there is no $(x = x)$-transition from such an f. We want to rule out the possibility of an $[\epsilon_x]$-transition that leads to error. The presence of $\neg\neg S^+$ in $(R; \neg\neg S^+ \to S)$ does precisely this: it assures us that for the evaluation of implication, we consider only transitions on which S is defined. ❑

We could have started with negation and have defined implication, as witnessed by the following computation.

$$\begin{array}{rcl}
\neg(\langle R^+,R\rangle;\neg\langle S^+,S\rangle) & = & \neg(\langle R^+,R\rangle;\langle\neg\neg S^+\,,\,\neg\neg S^+;\neg S\rangle)\\
& = & \neg(\langle R^+;\neg\neg S^+\,,\,R;\neg\neg S^+;\neg S\rangle)\\
& = & \langle\neg\neg(R^+;\neg\neg S^+)\,,\\
& & \quad\neg\neg(R^+;\neg\neg S^+);\neg(R;\neg\neg S^+;\neg S)\rangle\\
& = & \langle\neg\neg(R^+;S^+)\,,\,\neg\neg(R^+;S^+);\neg(R;\neg\neg S^+;\neg S)\rangle\\
& = & \langle\neg\neg(R^+;S^+)\,,\,\neg\neg(R^+;S^+);(R;\neg\neg S^+\to S)\rangle
\end{array}$$

If we restrict ourselves to domain deterministic $\langle R^+,R\rangle$, then we can simplify the definition of implication.

Theorem 8.3 *Suppose both* $\langle R^+,R\rangle$ *and* $\langle S^+,S\rangle$ *are domain deterministic. Then* $(\langle R^+,R\rangle\to\langle S^+,S\rangle)=\langle\neg\neg(R^+;S^+)\,,\,\neg\neg(R^+;S^+);(R\to S)\rangle$.

Proof

Suppose $f(\neg\neg(R^+;S^+))f\,R\,g$. It is clearly sufficient to show: $g(\neg\neg S^+)g$. We have, for some h and i, $f\,R^+\,h\,S^+i$. Since fRg, we also have fR^+g. By domain determinism, $\mathsf{dom}(h)=\mathsf{dom}(g)$. From hS^+i, we may conclude, by lemma 4.3(2), that for some j, gS^+j. ❑

Open Question 8.4 It is my feeling that the choice of the definition of 'implication' as given in this subsection, is the optimal one. However, both some more experimentation with alternative definitions and a tighter philosophical justification of the present one would be very wellcome. ❑

Open Question 8.5 The pair semantics provides all kinds of possibilities to constrain our algebra of meanings. The second components can be demanded to 'fit' the first components in all kinds of ways. The gain of such restrictions of the space of meanings, is greater control and more direct insight in what the meanings of the objects of our language look like.

There is a possibility to improve upon the present set-up, by not only putting 'being defined', 'being undefined' into our contexts, but also 'being constrained' and 'being unconstrained'.

Here is one way such a development *could* go. As a first step, note that we could replace our first components by more abstract functions to $\mathcal{D}$-sets. As a second step we could enrich $\mathcal{D}$ to a new category $\mathcal{D}^+$, to incorporate the ideas on constraining variables of [15]. To do this we enrich the set of arrows from d to d by adding γ ('constrain'), $\gamma\mu$ ('constrain and then reset'), $\mu\gamma$ ('reset and then constrain') and $\mu\gamma\mu$ ('constrain, reset, constrain'). In table 2, we give the composition of the arrows on d. Now contexts will be functions from the

$\circ$	d	μ	γ	$\gamma\mu$	$\mu\gamma$	$\gamma\mu\gamma$
d	d	μ	γ	$\gamma\mu$	$\mu\gamma$	$\gamma\mu\gamma$
μ	μ	μ	$\mu\gamma$	μ	$\mu\gamma$	$\mu\gamma$
γ	γ	$\gamma\mu$	γ	$\gamma\mu$	$\gamma\mu\gamma$	$\gamma\mu\gamma$
$\gamma\mu$	$\gamma\mu$	$\gamma\mu$	$\gamma\mu\gamma$	$\gamma\mu$	$\gamma\mu\gamma$	$\gamma\mu\gamma$
$\mu\gamma$	$\mu\gamma$	μ	$\mu\gamma$	μ	$\mu\gamma$	$\mu\gamma$
$\gamma\mu\gamma$	$\gamma\mu\gamma$	$\gamma\mu$	$\gamma\mu\gamma$	$\gamma\mu$	$\gamma\mu\gamma$	$\gamma\mu\gamma$

Table 2: Multiplication table of the morphisms on d

variables to $\mathcal{D}^+$-sets that are almost everywhere $\{\mathsf{d}, \mathsf{u}\}$. Our meaning objects are pairs $\langle f, R\rangle$ (satsifying some further constraints), where f is a context and where R is a local relation. For example, we would give $[x = x]$ as meaning a pair $\langle f, R\rangle$, where R is the usual test for identity and where f is a function assigning to each variable, except x, the value $\{\mathsf{d}, \mathsf{u}\}$ and to x the value $\{\gamma\}$.

The question is to develop a full theory of relations in context taking in account both partiality and constrainedness along the lines of both the present paper and [15]. $\square$

Open Question 8.6 Our treatment of partiality using the pairs, suggests to view the first component as providing the *presupposition* of the 'statement'. Can one connect this treatment with more standard treatements of presupposition? $\square$

Open Question 8.7 Can one axiomatize the valid equations of algebras $\mathcal{S}$ of pairs? $\square$

8.3 A Translation

Consider a fixed language $\mathcal{L}$. Suppose we give it both the interpretation to relations $[.]_{\mathsf{rel}}$ and the corresponding interpretation to pairs, say $[.]_{\mathsf{pair}}$. We will 'translate' $\mathcal{L}$ to an e-monoid $\mathcal{L}^\circ$ of pairs of elements of $\mathcal{L}$. $\mathcal{L}^\circ$ is specified as follows.

- ▷ The objects of $\mathcal{L}^\circ$ are pairs of elements of $\mathcal{L}$,
- ▷ $\top := \langle\top, \top\rangle$, $\bot := \langle\top, \bot\rangle$,
- ▷ $(\langle\phi^+, \phi\rangle \bullet \langle\psi^+, \psi\rangle := \langle\phi^+ \bullet \psi^+, \phi \bullet \psi\rangle$,
- ▷ $(\langle\phi^+, \phi\rangle \to \langle\psi^+, \psi\rangle) :=$
  $\langle\neg\neg(\phi^+ \bullet \psi^+), \neg\neg(\phi^+ \bullet \psi^+) \bullet (\phi \bullet \neg\neg(\psi^+) \to \psi)\rangle$.

$(.)^\circ$ maps the generators of $\mathcal{L}$ to elements of $\mathcal{L}^\circ$ as follows.

▷ $(P(r_1,\ldots,r_n))^\circ := \langle (r_1 = r_1. \ \ldots \ .r_n = r_n)\, ,\, P(r_1,\ldots,r_n)\rangle$,

▷ $(a_v)^\circ := \langle a_v, a_v\rangle$

We can extend $(.)^\circ$ in a unique way to an e-monoid morphism on $\mathcal{L}$. We have:

$$[.]_{\mathsf{pair}} = ([.]_{\mathsf{rel}} \times [.]_{\mathsf{rel}}) \circ (.)^\circ.$$

I.o.w, if $\phi^\circ = \langle \phi_1, \phi_2\rangle$, then $[\phi^\circ]_{\mathsf{pair}} = \langle [\phi_1]_{\mathsf{rel}}, [\phi_2]_{\mathsf{rel}}\rangle$.

8.4 An Alternative Representation

There is an alternative representation of our pairs of relations, that makes our new semantics fully relational. Let's introduce a new variable $\sharp$. We assume $\sharp \notin \mathsf{Var}$. Let $\mathsf{Var}^\sharp := \mathsf{Var} \cup \{\sharp\}$. Now $\mathsf{Ass}^\sharp_{\mathsf{loc}}$ is the set of functions f with finite domain, such that $\sharp \in \mathsf{dom}(f) \subseteq \mathsf{Var}^\sharp$, $f(v) \in N$, for $v \in \mathsf{Var}$, and $f(\sharp) \in \{0,1\}$. We now map a pair of relations on $\mathsf{Ass}_{\mathsf{loc}}$, $\langle R_0, R_1\rangle$, to a relation $R^\sharp$ on $\mathsf{Ass}^\sharp_{\mathsf{loc}}$ as follows:

▷ $fR^\sharp g :\Leftrightarrow f(\sharp) = g(\sharp)$ and $(f \restriction \mathsf{Var})R_{1-f(\sharp)}(g \restriction \mathsf{Var})$.

It is easily seen that:

$$f\, R_i\, g \ \Leftrightarrow\ (f \cup \{\langle \sharp, 1-i\rangle\})\, R^\sharp\, (g \cup \{\langle \sharp, 1-i\rangle\}).$$

Define

▷ $f \lhd g :\Leftrightarrow (f \restriction \mathsf{Var}) = (g \restriction \mathsf{Var})$ and $g(\sharp) = 1$.

▷ $f \perp\!\!\!\perp g :\Leftrightarrow f = g$ and $f(\sharp) = 1$.

▷ We define the notion of a local relation as before (remembering that our functions are always defined on $\sharp$).

▷ A relation R is $\lhd$-zig if fRg and $f \lhd f'$ implies that, for some g', $f'Rg'$ and $g \lhd g'$. (In fact this g' is unique, since $\lhd$ is functional.)

▷ A relation R is $\sharp$-monotonic ($\sharp$-mon), if fRg implies $f(\sharp) \leq g(\sharp)$.

Note that id, $\perp$, $\lhd$ and $\perp\!\!\!\perp$ are local, $\lhd$-zig and $\sharp$-monotonic. Moreover, loc, $\lhd$-zig and $\sharp$-mon are closed under ;. loc and $\sharp$-mon are closed under $\rightarrow$. Now consider the following sdra: $\mathcal{U} := \mathcal{R}_{\mathsf{Ass}^\sharp_{\mathsf{loc}}}(\mathsf{loc}, \sharp\text{-}\mathsf{mon})$. Let U be its domain. We define:

▷ $R \Rightarrow S \ :=\ \neg\neg(\lhd; R; S); (R; \neg\neg(\lhd; S) \rightarrow S)$.

We have:

Lemma 8.8 $W := U \cap (\lhd\text{-}\mathsf{zig})$ is closed under $\Rightarrow$ ❑

Proof

Suppose $f(R \Rightarrow S)g$ and $f \lhd f'$. Clearly $f = g$. We show: $f'(R \Rightarrow S)f'$. Since $f(\neg\neg(\lhd;R;S))f$, there are g', h', i', such that $f \lhd g'\, R\, h'\, S\, i'$. By the functionality of $\lhd$, we find that $g' = f'$. Hence, $f' \lhd f'\, R\, h'\, S\, i'$. Ergo, $f'(\neg\neg(\lhd;R;S))f'$. By $\sharp$-monotonicity, we find that for any j with $j(\sharp) = 1$:

$$j((R;\neg\neg(\lhd;S) \to S)j.$$

Hence we have: $f'(\neg\neg(\lhd;R;S);(R;\neg\neg(\lhd;S) \to S))f'$. ❑

Consider the following e-monoid $\mathcal{W}$ of signature $[\bot, \to]$. $\mathcal{W} := \langle\langle W, \mathsf{id}, ;\rangle, J\rangle$, where $J(\bot) := \perp\!\!\!\perp$ and $J(\to) := \Rightarrow$. Note that $\mathcal{W}$ is not an sdra. We interpret a language $\mathcal{L}$ in $\mathcal{W}$ by putting:

- ▹ Suppose $V = \{r_1, \ldots, r_n\} \cap, \mathsf{Var}$, then

$$\begin{array}{lll} f[P(r_1,\ldots,r_n)]g & :\Leftrightarrow & f = g,\ V \subseteq \mathsf{dom}(f) \text{ and} \\ & & (f(\sharp) = 1 \text{ or } (\lambda i{\in}\{1,\ldots,n\}\ |r_i|f) \in I(P)). \end{array}$$

- ▹ $f[a_v]g :\Leftrightarrow f(\sharp) = g(\sharp)$ and $(f \restriction \mathsf{Var})\,\langle\ell(a)\rangle_v\,(g \restriction \mathsf{Var})$.

Let us call the mapping we described earlier in this subsection, from $\langle R_0, R_1\rangle$ to $R^\sharp$: Φ. Let us name our semantical interpretation function $[.]_\sharp$. Then we have: $[.]_\sharp = \Phi \circ [.]_{\mathsf{pair}}$. We omitt the tedious verification.

9 Return of the Zig-Property

This section describes our final proposal. It is the version of dynamic predicate logic with local assigments that I consider as the best and most natural option (given the self-imposed limitations of our project).[17]

In this section, $\langle R^+, R\rangle$ will always be in $\mathsf{p\text{-}REL}(\mathsf{Ass}_{\mathsf{loc}})$. Let Z be the set of all pairs $\langle R^+, R\rangle$, where R^+ and R are local, domain deterministic relations, that have the zig-property and where *the pair* $\langle R^+, R\rangle$ has the following property.

- ▹ Suppose $f \subseteq f'$, $f'\,R\,g'$, fR^+h. Then there is a g with fRg and $g \subseteq g'$.

We call this property the rzig-property, or rzig, since it is a kind of reverse zig-property.

The following theorem assures us that Z is closed under the relevant operations and the strict local tests. Assuming this result we see that Z is the domain of $\mathcal{Z} := \mathcal{S}_{\mathsf{Ass}_{\mathsf{loc}}}(\mathsf{loc}, \mathsf{fdd}, \mathsf{zig}, \mathsf{rzig})$.

[17] Let me remind you, however, that prominent researchers like Paul Dekker and Martin van den Berg have chosen otherwise.

Theorem 9.1 *Z is closed under the following operations: id, $\bot$, $\langle \alpha_x, \alpha_x \rangle$ for $\alpha_x \in (\mathsf{fdd} \cap \mathsf{zig})$, ;, $\to$. Moreover all strict local tests are in Z.*

We give the proof in appendix A. Here are the ea's that are both domain deterministic and that have the zig-property.

$$\begin{array}{lll}
\langle \emptyset \rangle_v & & \\
\langle \mathsf{d} \rangle_v & \langle \mathsf{d}, \mathsf{u} \rangle_v & \\
\langle \mu \rangle_v & \langle \mu, \mathsf{u} \rangle_v & \langle \mu, \curvearrowright \rangle_v \\
\langle \curvearrowleft \rangle_v & \langle \curvearrowleft, \mathsf{u} \rangle_v &
\end{array}$$

Here is our repertoire, say $\mathfrak{z}$.

$$\begin{array}{lll}
\exists & \mapsto & \{\mu, \curvearrowright\} \\
\mathsf{E} & \mapsto & \{\curvearrowleft, \mathsf{u}\}
\end{array}$$

We write $\exists v$ instead of $\exists_v$ and $v\mathsf{E}$ instead of E_v. Our logic will be $\mathsf{z\text{-}DPL} := \mathsf{DPL_p}(\mathfrak{z}, \mathsf{fdd}, \mathsf{zig}, \mathsf{rzig})$. We give a list of the domain deterministic ea's with the zig-property that are and are not definable in our logic. Let's write $\Delta(\alpha_v)$ for $\langle \alpha_v, \alpha_v \rangle$.

- ▷ $\Delta(\emptyset_v) = [\bot]$.
- ▷ $\Delta(\langle \mathsf{d} \rangle_v) = [v = v]$.
- ▷ $\Delta(\langle \mathsf{d}, \mathsf{u} \rangle_v) = [\top]$.
- ▷ $\Delta(\langle \mu \rangle_v) = [v = v . \exists v]$.
- ▷ $\Delta(\langle \mu, \mathsf{u} \rangle_v)$ is not definable.
- ▷ $\Delta(\langle \mu, \curvearrowright \rangle_v) = [\exists v]$.
- ▷ $\Delta(\langle \curvearrowleft \rangle_v) = [v = v . v\mathsf{E}]$.
- ▷ $\Delta(\langle \curvearrowleft, \mathsf{u} \rangle_v) = [v\mathsf{E}]$

We define $\forall x(A)$ by $(\exists x \to A)$. We present some examples.

- ▷ $\forall x(P(x))$
  We have: $[\forall x(P(x))] = \langle \mathsf{id}, R \rangle$, where $R = \bot$, if classically $\forall x(P(x))$ is false, and $R = \mathsf{id}$, if classically $\forall x(P(x))$ is true.
- ▷ $\exists x . P(x) . x\mathsf{E}$
  We have: $[\exists x . P(x) . x\mathsf{E}] = \langle \langle \curvearrowleft, \mathsf{u} \rangle_x, R \rangle$, where $R = \bot$, if $\exists x(P(x))$ is false in classical predicate logic in the given model, and $R = \langle \curvearrowleft, \mathsf{u} \rangle_x$, if classically $\exists x(P(x))$ is true.
- ▷ $x\mathsf{E} . x = x$
  Evidently, $[x\mathsf{E} . x = x] = \langle \bot, \bot \rangle$.

Open Question 9.2 Our first example suggests that for variables bound in the appropriate way z-DPL allows α-conversion. I did not attempt to formulate this idea precisely. So I pose it as a problem for the reader to give a precise formulation of the intuitive insight and prove it. See also footnote 15. ❑

10 Conclusion

In this paper we studied dynamic predicate logics with local assignments. Our aim was a detailed understanding of their definition.

We introduced a definitional format to specify such logics. The notion of elementary action turned out to play a central role. We gave a characterization of this notion. There is some room for experimentation with less constrained classes of actions to play the role of our elementary actions. The ordinary dynamic predicate logics with local relations behave not completely in the desired way, since they conflate error and falsity. Following the lead of Martin van den Berg, we studied a strategy of remedying this situation. This strategy is interesting in its own right because of the analogy with the use of presuppositions in natural language.

We ended the paper by introducing a dynamic predicate logic, z-DPL, 'with presuppositions' in which all relations have the zig-property. Philosophically speaking the presence of the zig-property means that we stick to the intuition that *undefined* means lack of information. In the definition of z-DPL all ideas of the previous sections were brought into play.

We hope to have given the reader a good feeling both for the options one has and for the pitfalls one meets, in defining a dynamic predicate logic with local assignments. We also hope to have convinced the reader that such logics are respectable mathematical objects worthy of further study.

Acknowledgements

I thank Kees Vermeulen for reading carefully through the penultimate version of this manuscript. I am grateful to Willem Groeneveld, Karst Koymans, Freek Wiedijk and Kees Vermeulen for discussing notational issues. Finally, I thank professor Carsetti for being an editor with an almost angelic patience.

A Z is closed under Various Operations

Lemma A.1 Let $\langle R^+, R\rangle$ be domain deterministic. Suppose that fR^+iS^+j and fRg. Then, for some k, we have gS^+h. ❑

Proof

Since fRg, we have fR^+g. From the fact that fR^+i, we find, by domain determinism, $\mathsf{dom}(g) = \mathsf{dom}(i)$. But then, by lemma 4.3(2) and the fact that iS^+j, there is a k with gS^+k. ❑

Lemma A.2 1. $\mathsf{id}, \bot \in Z$,

2. strict local tests are in Z,

3. if α_x is domain deterministic and has the zig-property, then $\langle \alpha_x, \alpha_x \rangle \in Z$. ❑

Proof

We only do the proof of 3. Suppose $f'(\alpha_x)g'$, $f \subseteq f'$ and $f(\alpha_x)h$. We are looking for a g with $f(\alpha_x)g \subseteq g'$. In case $f(x) \in N$, we have, a fortiori, $f(x) = f'(x)$. We can choose $g := g' \restriction \mathsf{dom}(f)$. Suppose $f(x) = *$. If $h(x) = *$, we have $h \subseteq g'$. So we can choose $g := h$. Suppose $h(x) = n \in N$. By the zig-property, there is an h' with $h \subseteq h'$ and $f'Rh'$. By domain determinism, we find that $\mathsf{dom}(g') = \mathsf{dom}(h')$. Ergo $g'(x) = m$, for some $m \in N$. Since $f(\alpha_x)h$, $f(x) = *$, $h(x) = n \in N$, we find $f(\alpha_x)h[x := m] \subseteq g'$. So we may take $g := h[x := m]$. ❑

Lemma A.3 Z is closed under composition. ❑

Proof

It is clear that zig and fdd are preserved under composition. We verify the fact that rzig is preserved under composition. Consider $\langle R^+, R \rangle$, $\langle S^+, S \rangle$ in Z. Suppose $f'Rg'Sh'$, $f \subseteq f'$ and fR^+iS^+j. Since we have $f'Rg'$, $f \subseteq f'$ and fR^+i, we can find, by the rzig-property for $\langle R^+, R \rangle$, a g such that $fRg \subseteq g'$. We have fRg and fR^+iS^+j. By lemma A.1, it follows that there is a k with gS^+k. We have the following constellation: $g \subseteq g'$, $g'Sh'$, gS^+k. So, by the rzig-property for $\langle S^+, S \rangle$, we can find an h such that gSh and $h \subseteq h'$. Collecting the previous observations, we conclude: $fRgSh \subseteq h'$. ❑

Lemma A.4 Z is closed under implication. ❑

Proof

We verify preservation of the zig-property under implication. Consider $\langle R^+, R\rangle$, $\langle S^+, S\rangle$ in Z. Since the elements of Z are domain deterministic, we may employ implication between pairs in its the simplified form.

The verification that the first component of the composition has the zig-property is immediate by the fact that the zig-property is preserved by ; and $\neg\neg$.

We turn to the second component. Suppose $f(\neg\neg(R^+;S^+);(R \to S))f$ and $f \subseteq f'$. We have to show: $f'(\neg\neg(R^+;S^+);(R \to S))f'$. We already found $f'(\neg\neg(R^+;S^+)f'$. So it is sufficient to show: $f'(R \to S)f'$. Suppose $f'Rg'$. We have fR^+i, for some i. Hence, by rzig, there is a g with $fRg \subseteq g'$. Since, $f(R \to S)f$, we find an h with gSh. So $g \subseteq g'$ and gSh. Hence, by the zig-property for S, there is an h' with $g'Sh'$. We may conclude: $f'(R \to S)f'$.

We verify the rzig-property for implication. Suppose

$$f'(\neg\neg(R^+;S^+);(R \to S))f',\ f \subseteq f' \text{ and } f(\neg\neg(R^+;S^+))f.$$

It is sufficient to show that $f(R \to S)f$. Suppose fRg. We have to find an h, with gSh. By the zig-property, we can find a g' such that $f'Rg'$ and $g \subseteq g'$. Since, $f'(R \to S)f'$, there is an h' such that $g'Sh'$. On the other hand we have, for some i, j, fR^+iS^+j and fRg. By lemma A.1, it folows that there is a k with gS^+k. Now we have the following constellation: $g \subseteq g'Sh'$ and gS^+k. By applying the rzig-property for $\langle S^+, S\rangle$, we find an h with gSh. ❑

References

[1] M.H. van den Berg. *The Internal Structure of Discourse.* PhD thesis, ILLC Dissertation Series 1996-3, March 1996.

[2] P. Dekker. *Transsentential Meditations, ups and downs in dynamic semantics.* PhD thesis, University of Amsterdam, ILLC, May 1993.

[3] K. Fine. *Reasoning with arbitrary objects.* Basil Blackwell, Oxford and New York, 1985.

[4] G. Frege. Logische Mängel in der Mathematik. In H. Hermes, F. Kambartel, and F. Kaulbach, editors, *Gottlob Frege. Nachgelassene Schriften*, pages 171–181. Felix Meiner Verlag, Hamburg, 1983.

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

[6] J. Groenendijk, M. Stokhof, and F. Veltman. Coreference and modality. In S. Lappin, editor, *Handbook of Contemporary Semantic Theory*, pages 179–213. Blackwell, Oxford, 1996.

[7] M. Hollenberg and C. Vermeulen. Counting variables in a dynamic setting. *Journal of Logic and Computation*, 6(5):725–744, 1996.

[8] M.J. Hollenberg. An equational axiomatisation of dynamic negation and relational composition. *Journal of Language, Logic and Information*, 6(4):381–401, 1997.

[9] J. Jaspars and E. Krahmer. A programme of modal unification of dynamic theories. In P. Dekker and M. Stokhof, editors, *Proceedings of the 10th Amsterdam Colloquium, part II*, pages 425–444. ILLC, University of Amsterdam, Amsterdam, 1995.

[10] H. Kamp. A theory of truth and semantic representation. In J. Groenendijk et al., editor, *Truth, Interpretation and Information*, pages 1–41. Foris, Dordrecht, 1981.

[11] C.F.M. Vermeulen. Sequence semantics for dynamic predicate logic. *Journal of Logic, Language and Information*, 2:217–254, 1993.

[12] C.F.M. Vermeulen. Incremental semantics for propositional texts. *Notre Dame Journal of Formal Logic*, 35:243–271, 1994.

[13] C.F.M. Vermeulen. Update semantics for propositional texts. In M. Masuch and L. Pólós, editors, *Applied Logic: How, What and Why?*, Dordrecht, 1994. Kluwer. Proceedings of the Applied Logic Conference, Logic@Work, 1992.

[14] C.F.M. Vermeulen. Merging without mystery, variables in dynamic semantics. *Journal of Philosophical Logic*, 24:405–450, 1995.

[15] A. Visser. Contexts for dynamic predicate logic. Logic Group Preprint Series 143, Department of Philosophy, Utrecht University, Heidelberglaan 8, 3584 CS Utrecht, October 1995.

[16] A. Visser. Dynamic Relation Logic is the logic of DPL-relations. *Journal of Language, Logic and Information*, 6(4):441–452, 1997.

[17] A. Visser and C. Vermeulen. Dynamic bracketing and discourse representation. *Notre Dame Journal of Formal Logic*, 37:321–365, 1996.