Dynamic Modal Predicate Logic
preliminary version

Jan van Eijck[1,2] & Giovanna Cepparello[1,3]

[1] CWI, Kruislaan 413, 1098 SJ Amsterdam, The Netherlands

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

[3] Scuola Normale Superiore, Piazza dei Cavalieri 7, 56100 Pisa, Italy

June 15, 1992

Abstract

This paper presents a combination of dynamic predicate logic and update logic in the form of a dynamic version of modal predicate logic (DMPL). It is shown that this logic is eliminative at a global level and distributive at a local level. The logic is axiomatized in Quantified Dynamic Modal Logic. The axiomatization gives rise to a translation function from quantified dynamic modal logic to S5 modal predicate logic. This translation function can be used to compute the conditions for success of DMPL programs, i.e. to derive the static truth conditions of DMPL meaning representations.

1 Introduction

Our goal in this paper is to develop a representation language suitable to represent the meanings of natural language texts involving unbound anaphora and epistemic operators.[1] Consider the following example texts.

(1) *A man walked out. Maybe he was angry.*

(2) *If a man walks out, then maybe he is angry.*

The difficulty with these examples is that they saddle us with a combination of two problems. In the first place, there is the problem of getting the pronominal binding right (in example (1) a link across a sentence boundary, in example (2) a 'donkey pronoun' link). In the second place, there is the problem of getting a correct interpretation of *maybe* as an epistemic operator. The first problem points in the direction of a dynamic version of predicate logic as a representation language [5]. The second problem suggests the introduction of another kind of dynamic logic. To get a correct interpretation of *maybe* as an epistemic operator something along the lines of Veltman's update logic [13] seems needed.

The question now becomes: how can we combine dynamic predicate logic and update logic? Several possible approaches present themselves. To compare these we will first quickly review dynamic predicate logic and update logic, focussing on the notions of eliminability and distributivity (the terminology is borrowed from [6]). The review will make it possible to compare our proposal for a dynamic modal predicate logic with two earlier proposals from the recent literature.

2 Dynamic Predicate Logic

For a quick sketch of standard dynamic predicate logic, let a set V of variables, a set C of constants (lumped together these are the terms of the language) and a set of relation symbols with information

[1]The second author was sponsored by a travel grant from project NF 102/62-356 ('Structural and Semantic Parallels in Natural Languages and Programming Languages'), funded by the Netherlands Organization for the Advancement of Research (N.W.O.).

about their arities be given. In defining the syntax of dynamic predicate logic we write ; for sequential composition (corresponding to $\wedge$ in standard predicate logic), $\Rightarrow$ for dynamic implication, and $\eta x : \pi$ for indefinite assignment (corresponding to existential quantification in standard predicate logic).

The language DPL is the smallest set such that the following hold.

1. If t_1, t_2 are DPL terms, then $t_1 = t_2 \in$ DPL.
2. If R is an n-place relation symbol and $t_1, \ldots, t_n$ are DPL terms, then $Rt_1 \cdots t_n \in$ DPL,
3. If $\pi \in$ DPL, then $\neg\pi \in$ DPL.
4. If $\pi_1 \in$ DPL and $\pi_2 \in$ DPL, then $(\pi_1;\ \pi_2) \in$ DPL and $(\pi_1 \Rightarrow \pi_2) \in$ DPL.
5. If $\pi \in$ DPL and x is a variable, then $\eta x : \pi \in$ DPL.

With reference to a model $\mathcal{M} = \langle M, F \rangle$, where M is a domain of individuals and F an interpretation function for the constants and relation symbols of the language, the semantics for DPL can be given as a function from sets of states to sets of states, where states are functions from V to M.

To state this formally, let $\mathcal{M} = \langle M, F \rangle$ and the set of variables V be given. Then the state set for $\mathcal{M}$ is the set of functions M^V. If $s \in M^V$ and $d \in M$, then $s(x|d)$ is the state which is like s except for the possible difference that x is mapped to d. If $S \subseteq M^V$ and $x \in V$ then $S[x]$ is the set $\{s(x|d) \mid s \in S, d \in M\}$.

For any formula A of the form $Rx_1 \cdots x_n$ or $t_1 = t_2$ the relation $\mathcal{M} \models A\ [s]$ (A is true in $\mathcal{M}$ under state or assignment s) is defined in the usual way.

The interpretation function $[\![\cdot]\!]_{\mathcal{M}} : \mathcal{P}(M^V) \to \mathcal{P}(M^V)$ is defined as follows.

1. $[\![Rt_1 \cdots t_n]\!]_{\mathcal{M}}(S) = \{s \in S \mid \mathcal{M} \models Rt_1 \cdots t_n\ [s]\}$.
2. $[\![t_1 = t_2]\!]_{\mathcal{M}}(S) = \{s \in S \mid \mathcal{M} \models t_1 = t_2\ [s]\}$.
3. $[\![\pi_1;\ \pi_2]\!]_{\mathcal{M}}(S) = [\![\pi_2]\!]_{\mathcal{M}}([\![\pi_1]\!]_{\mathcal{M}}(S))$.
4. $[\![\neg\pi]\!]_{\mathcal{M}}(S) = \{s \in S \mid [\![\pi]\!]_{\mathcal{M}}(\{s\}) = \emptyset\}$.
5. $[\![\eta x : \pi]\!]_{\mathcal{M}}(S) = [\![\pi]\!]_{\mathcal{M}}(S[x])$.

Note that negation is computed pointwise on the states. Dynamic implication between programs can be defined in terms of negation and sequential composition. $\pi_1 \Rightarrow \pi_2$ abbreviates $\neg(\pi_1;\ \pi_2)$. This gives the following semantics for dynamic implication:

$$[\![\pi_1 \Rightarrow \pi_2]\!]_{\mathcal{M}}(S) =$$
$$\{s \in S \mid \textit{for all } s' \in [\![\pi_1]\!]_{\mathcal{M}}(\{s\}) \textit{ it holds that } [\![\pi_2]\!]_{\mathcal{M}}(\{s'\}) \neq \emptyset\}.$$

We abbreviate $\{s \in M^V \mid [\![\pi]\!]_{\mathcal{M}}(\{s\}) \neq \emptyset$ as $\downarrow[\![\pi]\!]_{\mathcal{M}}$. Using this, we can define satisfaction in a model for a state set. Program π is satisfied in $\mathcal{M}$ by state set S if $S \subseteq \downarrow[\![\pi]\!]_{\mathcal{M}}$. We say that π is satisfied in $\mathcal{M}$ if π is satisfied in $\mathcal{M}$ by some nonempty $S \subseteq M^V$. Note the following fact.

(3) *Program π is satisfied in $\mathcal{M}$ iff* $[\![\pi]\!]_{\mathcal{M}}(M^V) \neq \emptyset$.

Program π_1 dynamically entails conclusion π_2 (notation $\pi_1 \models \pi_2$) if for every $\mathcal{M}$ and every $S \subseteq M^V$ the following holds.

(4) $[\![\pi_1]\!]_{\mathcal{M}}(S) \subseteq \downarrow[\![\pi_2]\!]_{\mathcal{M}}$.

Here are some examples of valid dynamic consequences.

(5) $\eta x : man\ x \models man\ x$.

(6) $\eta x : man\ x;\ walk\ x \models \eta y : man\ y;\ walk\ y$.

(7) $(\eta x : man\ x;\ \eta y : woman\ y;\ love(x, y) \Rightarrow kiss(x, y));$
$man\ z;\ woman\ w;\ love(z, w) \models kiss(z, w)$.

For proper contrast, here is an example of a non-valid dynamic consequence.

(8) $\eta x : man\ x \not\models man\ y.$

In Groenendijk and Stokhof [6], dynamic meaning as defined by $[\![\cdot]\!]$ is called *distributive* because it is defined pointwise on individual variable states.

Lemma 1 (Distribution Lemma) *For all programs π of DPL, for all models $\mathcal{M}$ and all $S \subseteq M^V$ the following holds:*

$$[\![\pi]\!]_{\mathcal{M}}(S) = \bigcup_{s \in S} [\![\pi]\!]_{\mathcal{M}}(\{s\}).$$

Dynamic meaning as given by $[\![\cdot]\!]$ is also *non eliminative*, to use another term from [6], in the sense that $[\![\pi]\!]_{\mathcal{M}}(S)$ need not in general be a subset of S. The reason for this is, of course, that the indefinite assignment statements can *change* variable states.

3 Epistemic Update Logic

Following Veltman [13], we can define the language of epistemic update logic over a set of proposition letters P as the smallest set L_P such that the following hold:

1. If $p \in P$ then $p \in L_P$.
2. If $\pi \in L_P$, then $\neg\pi \in L_P, \Diamond\pi \in L_P$.
3. If π and $\pi' \in L_P$, then $(\pi;\pi') \in L_P, (\pi \cup \pi') \in L_P$.

The semantics of L_P is again given in terms of input-output behaviour. We take the set W of worlds over P to be the set $\mathcal{P}P$. Any subset of $\mathcal{P}W$ is an information set. Progams are interpreted as functions from information sets to information sets, i.e., as functions in $\mathcal{P}W \to \mathcal{P}W$. The clauses are as follows:

1. $[\![p]\!](I) = I \cap \{w \mid p \in w\}$.
2. $[\![\pi;\pi']\!](I) = [\![\pi']\!]([\![\pi]\!](I))$.
3. $[\![\pi \cup \pi']\!](I) = [\![\pi]\!](I) \cup [\![\pi']\!](I)$.
4. $[\![\neg\pi]\!](I) = I - [\![\pi]\!](I)$.
5. $[\![\Diamond\pi]\!](I) = \begin{cases} I & \text{if } [\![\pi]\!](I) \neq \emptyset, \\ \emptyset & \text{otherwise.} \end{cases}$

Note that negation is Boolean complement. Intuitively, a program of the form $\Diamond\pi$ does not provide information about the world but about available information. A program $\Diamond\pi$ is true, given an information set I, if there is at least one world $w \in I$ for which π is true in the sense that $w \in [\![\pi]\!](I)$. If such a w can be found, the output information set of $\Diamond\pi$ is equal to its input information set; this agrees with the intuition that $\Diamond\pi$ does not say anything at all about what the world is like. In the other case, i.e., the case were $[\![\pi]\!](I) = \emptyset$, the output information set of $\Diamond\pi$ equals $\emptyset$.

A program π is acceptable in I if $[\![\pi]\!](I) \neq \emptyset$.

A program π is accepted in I if $I = [\![\pi]\!](I)$.

A program π is valid (or always accepted) if for all I it holds that $[\![\pi]\!](I) = I$.

One notion of entailment for Update Logic is the following. Premiss π_1 entails conclusion π_2 (notation $\pi_1 \models \pi_2$) if for all I it holds that $[\![\pi_1]\!](I) = [\![\pi_2]\!]([\![\pi_1]\!](I))$.

It is clear that the semantic clause for $\Diamond\pi$ introduces an element of *non distributivity* into the semantics, in the sense that unions of input states do not distribute over output states. There are π, I such that (9).

(9) $$[\![\pi]\!](I) \neq \bigcup_{i \in I} [\![\pi]\!](\{i\}).$$

More specifically, it does not in general hold that $[\![\pi]\!](I) \subseteq \bigcup_{i \in I} [\![\pi]\!](\{i\})$. Counterexample: take π equal to $\Diamond p$ and let $I = \{w, w'\}$ with $p \in w$ and $p \notin w'$. Then $[\![\Diamond p]\!](\{w\}) \cup [\![\Diamond p]\!](\{w'\}) = \{w\}$, but $[\![\Diamond p]\!](I) = I = \{w, w'\}$.

On the other hand, a simple induction on the complexity of π shows that Lemma 2 holds for all $\pi \in L_P$ and all information sets I. In the terminology of Groenendijk & Stokhof [6]: epistemic update logic is *eliminative*.

Lemma 2 (Elimination Lemma) *For all programs π of Update Logic and all information sets I:* $[\![\pi]\!](I) \subseteq I$.

4 Epistemic Modalities in Dynamic Predicate Logic

A clear prerequisite of the formulation of a dynamic version of modal predicate logic, where the modal operator $\Diamond$ is interpreted epistemically, as in Veltman's update logic, is that the apparent conflict between non eliminativity of DPL and eliminativity of update logic be resolved (see the following table).

	DPL	UL
eliminative	−	+
distributive	+	−

The two attempts that we know of to resolve this apparent conflict both proceed by first making DPL eliminative and then combining the result with UL to create an eliminative, non-distributive logic. There are two ways of making DPL eliminative: curtailing the syntax and changing the semantics. The approach pursued by Dekker in [1], is to first curtail DPL to get an eliminative system, and then add a modal operator to the result. The disadvantage of this is that it imposes a rather unnatural restriction on the language: new assignments to already 'active' variables are forbidden. The formula, $\eta x : Px;\ \eta x : \Diamond Qx$, for instance, is not interpretable in Dekker's eliminative dynamic modal predicate logic. Vermeulen [14] takes the other approach. Here DPL is made eliminative by developing a sequence semantics that remembers all old values of variables. This is the method of changing the semantics. Our objection to this is that the original DPL semantics was rather nice and simple. It is not an accident, by the way, that both approaches to the problem of combining DPL and UL have concentrated on changing the features of DPL. Tampering with the non distributivity of UL does not make much sense, as non distribitivity is the essence of the epistemic notion *maybe* that UL tries to analyse.

Still, there is a third way. Our contention is that it is possible to combine DPL and UL without tampering with the the original syntax and semantics of DPL. We must immediately add that this possibility was indepently noticed by Paul Dekker, who briefly discusses it in an appendix to a chapter of his thesis [2]. See below for the precise relationship between his proposal and ours.

In the rest of this section we will show that a distinction of levels engenders a dynamic predicate logic with an epistemic modality for *maybe* which is the result of direct fusion of the distributive and non eliminative character of DPL with the non distributive and eliminative character of UL. The key to the resolution of the conflict is the introduction of a distinction of levels, which gives our logic the features specified in the following picture.

	local level	global level
eliminative	−	+
distributive	+	−

As the table indicates, the clash between the eliminativity of one subsystem and the non eliminativity of the other is avoided by making a distinction between a global and a local level. This results in a system without artificial syntactic limitation which semantically is as close as possible to both DPL and UL.

Our distinction of levels can be seen as an example of playing with Tarskian variations in the sense of Van Benthem [9]: at the local level we vary assignments, while at the global level we vary possible worlds, or more precisely, interpretation functions over a universe.

If one wants to add an epistemic modal operator *maybe* to dynamic predicate logic, how does one proceed? In any case, the meanings of programs of the form $\Diamond\pi$ will have to be phrased in terms of the behaviour of π in different worlds. Intuitively, $\Diamond\pi$ is acceptable in a world $w \in W$ and a state s for that

world if there is some world $w' \in W$ where s is also available and where evaluation of the program π with input s gives truth. Thus, $\Diamond\pi$ is a test. It keeps in, at every world w, those states s at that world that will make the program π succeed at some index w' where s exists.

This intuitive account leads to a notion of an *information map*, a function from the set of worlds (think of those as regular first order models over a given universe U) to the power set of states for U. Thus, information maps can be viewed as sets of subsets of the state set indexed by a set of worlds. Information maps are the correlate of Veltman's information sets, but with the extra internal structure needed to take the DPL states into account. Information maps not only indicate which worlds are still valid alternatives (namely those worlds where the value of the information map does not equal $\emptyset$), but they also keep track of which states are still 'in the running' for each given world.

In a sense, the semantic clause for $\Diamond\pi$ introduces an element of non distributivity into the semantics: if we evaluate $\Diamond\pi$ with respect to an information map i, then the result, for a given world w, will not in general be equal to the result of evaluating π for the information map i' which is like i at w but which maps every $w' \neq w$ to the empty set. This we may call *non distributivity at the global, information map level.* On the other hand, at the local level of the state sets per index it holds that the output of π for input map i at world w can be computed pointwise, by taking the union of the outputs for each individual state in iw. The reason for this is that the distributivity of DPL carries over to DMPL at the appropriate level. We call this *distributivity at the state set level.*

Similarly, the system is globally eliminative, in the sense that worlds that are mapped to $\emptyset$ by a given input information map i will never be mapped to anything but $\emptyset$ in an output information map of a program π working on input i. Locally, at the level of the state sets, the system is not eliminative, as it is not in general the case that the state sets at an index w are only weeded out by the actions of programs.

5 A Semantics for DMPL

As for the formal definitions, the syntax of DMPL is like the syntax of DPL, but with the following clause added.

- If $\pi \in$ DMPL, then $\Diamond\pi \in$ DMPL.

We will follow the usual predicate logical convention of omitting outermost parentheses for readability. It will become evident from the semantic clause for sequential composition that the ; operator is associative. Therefore, we will often take the liberty to write $\pi_1; \pi_2; \pi_3$ instead of $(\pi_1; \pi_2); \pi_3$ or $\pi_1; (\pi_2; \pi_3)$. Also, we use $t_1 \neq t_2$ as an abbreviation for $\neg t_1 = t_2$, $\bot$ as an abbreviation for $\eta x : x \neq x$, and $\top$ as an abbreviation for $\neg\bot$.

For the interpretation of DMPL we have to look at sets W of DPL models (i.e., first order models). It is convenient to give them all the same universe. Thus, a DMPL model $\mathcal{M} = \langle M, \mathbf{W}\rangle$ consists of a universe M and a set $\mathbf{W}$ of DPL models over that universe. Given a DMPL model $\mathcal{M}$ and a set of variables V, the set of states for $\mathcal{M}$ is the set of functions M^V. The set of information maps for $\mathcal{M}$, $\mathbf{I}_\mathcal{M}$, is the set of functions $\mathbf{W} \to \mathcal{P}(M^V)$.

The following notational conventions are useful.

If $s \in M^V$, $x \in V$, then $s[x] \stackrel{\text{def}}{=} \{s(x|d) \mid d \in M\}$.

If $S \subseteq M^V$, $x \in V$, then $S[x] \stackrel{\text{def}}{=} \{s[x] \mid s \in S\}$.

If $i \in \mathbf{I}_\mathcal{M}$, $x \in V$, then $i[x] \in \mathbf{I}_\mathcal{M}$ is given by $(i[x])w = (iw)[x]$. in other words, $i[x] = \lambda w \cdot \{s[x] \mid s \in iw\}$.

If $i \in \mathbf{I}_\mathcal{M}$, $u \in \mathbf{W}$, $S \subseteq M^V$, then $i(u|S)$ is given by:

$$(i(u|S))w = \begin{cases} S & \text{if } w = u, \\ iw & \text{otherwise.} \end{cases}$$

In discussing information maps, we need to be able to refer to functions mapping every world from a given set $W \subseteq \mathbf{W}$ to the same value, and every other world in $\mathbf{W}$ to $\emptyset$. We use c_S^W for the function in

$\mathbf{I}_\mathcal{M}$ which at every $w \in W$ gives the value S and at every $w \in \mathbf{W} - W$ the value $\emptyset$. For convenience we write c_s^W for $c_{\{s\}}^W$.

We abbreviate $c_\emptyset^\mathbf{W}$ as $\mathbf{0}$. $\mathbf{0}$ is the constant $\emptyset$ function, the absurd information map. We use $\mathbf{1}$ for $c_{M^V}^\mathbf{W}$, the constant M^V function. $\mathbf{1}$ is the information map of blissful ignorance (the map of minimal information). Note that $\mathbf{0}(w|S)$ is the function which maps w to S and every other world to $\emptyset$. Assuming $S \neq \emptyset$ this is a map of complete information.

Next, we define a function $g : \mathbf{I}_\mathcal{M} \to \mathcal{P}\mathbf{W}$ that keeps track of the worlds constituting the valid epistemic alternatives in the context of a given information map. This function is given by (10).

(10) $\quad gi = \{w \in \mathbf{W} \mid iw \neq \emptyset\}.$

Note the following: $g(\mathbf{0}) = \emptyset$, $g(\mathbf{1}) = \mathbf{W}$, and if $S \neq \emptyset$ then $g(\mathbf{0}(w|S)) = \{w\}$.

We are now ready to give the interpretation function $[\![\cdot]\!]_\mathcal{M} : \mathbf{I}_\mathcal{M} \to \mathbf{I}_\mathcal{M}$. Again, for atomic programs we resort to the familiar notion $w \models_s A$, for 'A is true in w under state or assignment s' (recall that every w is a full fledged DPL model).

Here is the definition of the interpretation function for DMPL.

Definition 1 (Semantics of DMPL)

1. $[\![Rt_1 \cdots t_n]\!]_\mathcal{M} i = \lambda w \cdot \{s \in iw \mid w \models_s Rt_1 \cdots t_n\}$.
2. $[\![t_1 = t_2]\!]_\mathcal{M} i = \lambda w \cdot \{s \in iw \mid w \models_s t_1 = t_2\}$.
3. $[\![\pi_1;\ \pi_2]\!]_\mathcal{M} i = [\![\pi_2]\!]_\mathcal{M}([\![\pi_1]\!]_\mathcal{M} i)$.
4. $[\![\neg\pi]\!]_\mathcal{M} i = \lambda w \cdot \{s \in iw \mid ([\![\pi]\!]_\mathcal{M} i(w|\{s\}))(w) = \emptyset\}$.
5. $[\![\eta x : \pi]\!]_\mathcal{M} i = [\![\pi]\!]_\mathcal{M}(i[x])$.
6. $[\![\Diamond\pi]\!]_\mathcal{M} i = \lambda w \cdot \{s \in iw \mid \exists w' \in W \text{ with } s \in iw' \text{ and } ([\![\pi]\!]_\mathcal{M} i(w'|\{s\}))(w) \neq \emptyset\}$.

The clauses for basic relations $Rt_1 \cdots t_n$ and identities $t_1 = t_2$ say that atomic predicates serve to weed out the set of variable states according to what the interpretation function of the model tells us at the current index. Note that identity, because it is a logical relation, behaves the same in every world. For every index, the output state set simply consists of those variable states from the input set that satisfy the predicate.

The clause for program concatenation is as one would expect. Note that negation is boolean complement at the world level, but computed pointwise at the state level. The clause for negation ensures that at every index those states are pruned from the state set that do satisfy the program in the scope of the negation sign. As in DPL, dynamic implication $\pi_1 \Rightarrow \pi_2$, is defined as $\neg(\pi_1;\ \neg\pi_2)$. The clause for dynamic implication boils down to the following.

$$[\![\pi_1 \Rightarrow \pi_2]\!]_\mathcal{M} i =$$
$$\lambda w \cdot \{s \in iw \mid \textit{for all } s' \in ([\![\pi_1]\!]_\mathcal{M} i(w|\{s\}))w \textit{ it holds that } [\![\pi_2]\!]_\mathcal{M}(i(w|\{s'\})) \neq \emptyset\}.$$

Thus, dynamic implication weeds out, at every index, precisely those states that satisfy the antecedent program but have at least one state in their output that does not satisfy the consequent program.

The clause for η assignment to x followed by π first extends the input state sets at every index with all their x variants, and then performs π on the result.

Finally, the clause for $\Diamond$ tells us to evaluate iw pointwise at those other worlds where its states are still available. Note that this is the only clause which is not a straightforward adaptation from the case of DPL.

Definition 2 *π_1 and π_2 are equivalent if for all $\mathcal{M}$ and all $i \in \mathbf{I}_\mathcal{M}$ it holds that $[\![\pi_1]\!]_\mathcal{M} i = [\![\pi_2]\!]_\mathcal{M} i$.*

It is useful to be able to distinguish the test programs (i.e., the programs which are locally eliminative) from the rest:

Definition 3 *π is a test program if*

- *π has the form $Rt_1 \cdots t_n$, $t_1 = t_2$, $\neg\pi_1$, $\Diamond\pi_1$, $\pi_1 \Rightarrow \pi_2$, or*
- *π has the form $\pi_1; \pi_2$, and both π_1 and π_2 are test programs.*

Proposition 3 *π is a test program iff for all $\mathcal{M}, i \in \mathbf{I}_\mathcal{M}$ it holds that $[\![\pi]\!]_\mathcal{M} i = [\![\neg\neg\pi]\!]_\mathcal{M} i$.*

Proof: As is easily checked, the test programs are precisely the locally eliminative programs. By the semantic clause for $\neg$ it holds for precisely the locally eliminative π that π and $\neg\neg\pi$ are equivalent. ■

Acceptability in a model is defined in terms of behaviour with respect to the map of minimal information.

Definition 4 *A program π is acceptable in model $\mathcal{M}$ if $[\![\pi]\!]_\mathcal{M}(\mathbf{1}) \neq \mathbf{0}$.*

Acceptance for an information map:

Definition 5 *A program π is accepted by information map i for model $\mathcal{M}$ if $g([\![\pi]\!]_\mathcal{M} i) = gi$.*

Intuitively, a piece of information that is accepted by i does not weed out the set of current alternatives in i, as given by gi.

Next, we define a function that gives us a handle on local truth.

Definition 6 $\downarrow_i [\pi] \stackrel{\text{def}}{=} \lambda w \cdot \{s \in i(w) \mid [\![\pi]\!] i(w|\{s\})(w) \neq \emptyset\}$.

With this we can define the notion of satisfaction by an information map.

Definition 7 *A program π is* satisfied *by information map i for model $\mathcal{M} = \langle M, \mathbf{W}\rangle$ if for every $w \in \mathbf{W}$ it holds that $i(w) \subseteq \downarrow_i [\pi](w)$*

Dynamic consequence is defined as follows.

Definition 8 *π_1 dynamically entails π_2, notation $\pi_1 \models \pi_2$, if for all models $\mathcal{M}$ and all $i \in \mathbf{I}_\mathcal{M}$ it holds that π_2 is satisfied by $[\![\pi_1]\!]_\mathcal{M} i$.*

Proposition 4 *$\pi_1 \models \pi_2$ iff for every model M and every $i \in I_M$, π_2 is accepted by $[\![\pi_1]\!]_\mathcal{M} i$.*

Proof: We want to prove that $[\![\pi_1]\!]_M i(w) \subseteq \downarrow_{[\![\pi_1]\!]_M i} [\pi_2](w)$ for every model M, every information map i and every world w, iff for every model M and every information map i $g([\![\pi_1]\!]_M i) = g([\![\pi_2]\!]_M([\![\pi_1]\!]_M i))$.

First, let us suppose for some M and some i:

$$g([\![\pi_1]\!]_M i) \neq g([\![\pi_2]\!]_M([\![\pi_1]\!]_M i)).$$

This means that there is a world w such that $[\![\pi_1]\!]_M i(w) \neq \emptyset$ and $[\![\pi_2]\!]_M([\![\pi_1]\!]_M i)(w) = \emptyset$. Hence, there must be a state $s \in [\![\pi_1]\!]_M i(w)$ such that $[\![\pi_2]\!]_M([\![\pi_1]\!]_M i(w|\{s\})) = \emptyset$ i.e. such that $s \notin \downarrow_{[\![\pi_1]\!]_M i} [\pi_2]$.

Next, suppose there is some $s \in [\![\pi_1]\!]_M i(w)$ such that

$$s \notin \downarrow_{[\![\pi_1]\!]_M i} [\pi_2].$$

Then, take the i-map $i' = [\![\pi_1]\!]_M i(w|\{s\})$. It is easy to see that for this map we have that: $g([\![\pi_1]\!]_M i') \neq g([\![\pi_2]\!]_M([\![\pi_1]\!]_M i'))$ (because $[\![\pi_1]\!]_M i'(w) = \{s\}$ and $[\![\pi_2]\!]_M([\![\pi_1]\!]_M i')(w) = \emptyset$). ■

From the semantic clause for $\Rightarrow$ and Definition 8 we get:

Proposition 5 *$\pi_1 \models \pi_2$ iff $\models \pi_1 \Rightarrow \pi_2$.*

A comparison of Definition 8 and Proposition 4 shows how dynamic entailment combines the key features of the entailment notions for DPL and for update logic. For purposes of comparison with DPL and update logic the following facts are also noteworthy.

Proposition 6 (Global Eliminativity) *For all $\mathcal{M}, \pi, I$: $g([\![\pi]\!]_\mathcal{M} i) \subseteq gi$.*

Proposition 7 (Local Non Eliminativity) *There are* $\mathcal{M}, \pi, i, w$ *such that:* $[\![\pi]\!]_{\mathcal{M}} i(w) \not\subseteq iw$.

An example of the failure of inclusion in the Proposition 7 is of course the behaviour of η assignment programs, which may create new states.

Proposition 8 (Local Distributivity) *For all* $\mathcal{M}, \pi, i, w$: $([\![\pi]\!]_{\mathcal{M}} i)w = \bigcup_{s \in iw} \{[\![\pi]\!]_{\mathcal{M}}(i(w|\{s\}))\}$.

Proposition 9 (Global Non Distributivity) *There are* $\mathcal{M}, \pi, i, w$ *such that:*

$$([\![\pi]\!]_{\mathcal{M}} i)w \neq ([\![\pi]\!]_{\mathcal{M}} \mathbf{0}(w|iw))w.$$

Here is an example to illustrate Proposition 9. Just think of the evaluation of $\Diamond Px$ at a P-less world w in a model containing some other world where there are Ps. All information states at w will assign individuals to x which are not P. If one is not allowed to look elsewhere then the states which assign an individual that at some other index is P will not be found. This is all completely trivial, of course; the only reason we mention it is because it shows what has happened to the non distributivity of update semantics in the current combined system.

6 DPL + Update Logic = DMPL

To see that DMPL is a conservative extension of Update Logic, let us consider the restriction of $DMPL$ to the update fragment, i.e., the fragment which only has 0-ary predicates and no quantification (call this fragment DMPL_0). First of all, it is easily seen that for DMPL_0 programs eliminativity holds both locally and globally. Because of local eliminativity, the definition of *acceptation* in $DMPL_0$ can be simplified in the following way:

(11) *A program* π *is accepted by information map* i *for model* $\mathcal{M}$ *if* $[\![\pi]\!]_{\mathcal{M}} i = i$.

It follows from (11) that we have the following:

Proposition 10 $\pi_1 \models \pi_2$ *iff for all models* M *and all information maps* i,

$$[\![\pi_1]\!]_{\mathcal{M}} i = ([\![\pi_2]\!]_{\mathcal{M}}([\![\pi_1]\!]_{\mathcal{M}} i)).$$

Interestingly, for $\pi \in \mathrm{DMPL}_0$ we have that $\Diamond\pi$ becomes again an eliminative consistency check with a semantics which boils down to the following.

$$[\![\Diamond\pi]\!]i = \begin{cases} i & \text{if } g([\![\pi]\!]i) \neq \emptyset, \\ \mathbf{0} & \text{otherwise.} \end{cases}$$

On the other hand, we may also consider the DPL fragment of DMPL (all formulas without occurrences of the $\Diamond$ operator). For these we get both local and global distributivity. The global distributivity of the DPL fragment of DMPL enables us to replace the notion of local satisfaction by the following global notion of satisfaction.

(12) $\quad \downarrow [\pi]_{\mathcal{M}} = \lambda w \cdot \{s \in M^V \mid [\![\pi]\!]_{\mathcal{M}} \mathbf{0}(w|\{s\})(w) \neq \emptyset\}$.

From this we get:

Proposition 11 *If* $\pi_1, \pi_2 \in DPL$, *then* $\pi_1 \models \pi_2$ *iff for all* $\mathcal{M}, i$, $[\![\pi_1]\!]_{\mathcal{M}} i(w) \subseteq \downarrow [\pi_2](w)$.

7 DPL × Update Logic = DMPL

We will now establish the fact that the semantics for the 'least common product' of DPL and UL that Paul Dekker proposes in [2] is equivalent to our semantics. Dekker gives the following relational formulation of the semantic clauses.

Definition 9 (Relational Semantics of DPL × UL)

1. $||Rt_1 \cdots t_n||_{\mathcal{M}} = \{\langle s, W, s', w\rangle \mid s = s' \;\&\; w \in W \;\&\; w \models_s Rt_1 \cdots t_n\}$.

2. $||t_1 = t_2||_{\mathcal{M}} = \{\langle s, W, s', w\rangle \mid s = s' \ \&\ w \in W \ \&\ w \models_s t_1 = t_2\}$.

3. $||\pi_1;\ \pi_2||_{\mathcal{M}} = \{\langle s, W, s', w\rangle \mid \exists s'' : \langle s, \{w' \mid \langle s, W, s'', w'\rangle \in ||\pi_1||_{\mathcal{M}}\}, s', w\rangle \in ||\pi_2||_{\mathcal{M}}\}$.

4. $||\neg\pi||_{\mathcal{M}} = \{\langle s, W, s', w\rangle \mid s = s' \ \&\ w \in W \ \&\ \neg\exists s'' : \langle s, W, s'', w\rangle \in ||\pi||_{\mathcal{M}}\}$.

5. $||\eta x : \pi||_{\mathcal{M}} = \{\langle s, W, s', w\rangle \mid \exists s'' : s[x]s'' \ \&\ \langle s'', W, s', w\rangle \in ||\pi||_{\mathcal{M}}\}$.

6. $||\Diamond\pi||_{\mathcal{M}} = \{\langle s, W, s', w\rangle \mid s = s' \ \&\ \exists s'', w' : \langle s, W, s'', w'\rangle \in ||\pi||_{\mathcal{M}}\}$.

Proposition 12 $[\![\pi]\!]_{\mathcal{M}} i = \lambda w \cdot \bigcup_{s \in iw}\{s' \mid \langle s, gi, s', w\rangle \in ||\pi||_{\mathcal{M}}\}$.

Proof: Induction on the structure of π. ∎

Besides the fully functional definition of the DMPL semantics (definition 1) and the fully relational one (definition 9) there is a multi-valued function formulation, where the function arguments are the inputs to the DMPL programs and the function values are its outputs.

Definition 10 (Multi Valued Function Semantics for DMPL)

1. $[\![Rt_1 \cdots t_n]\!]_{\mathcal{M}}(s, W, w) = \{\langle s, \{w' \in W \mid w' \models_s Rt_1 \cdots t_n\}, w\rangle \mid w \models_s Rt_1 \cdots t_n\}$.

2. $[\![t_1 = t_2]\!]_{\mathcal{M}}(s, W, w) = \{\langle s, \{w' \in W \mid w' \models_s t_1 = t_2\}, w\rangle \mid w \models_s t_1 = t_2\}$.

3. $[\![\pi_1;\ \pi_2]\!]_{\mathcal{M}}(s, W, w) =$
$\{\langle s, W', w\rangle \mid \exists s'', W''$ *with* $\langle s'', W'', w\rangle \in [\![\pi_1]\!]_{\mathcal{M}}$ *and* $\langle s', W', w\rangle \in [\![\pi_2]\!]_{\mathcal{M}}(s'', W'', w)\}$.

4. $[\![\neg\pi]\!]_{\mathcal{M}}(s, W, w) = \{\langle s, W', w\rangle \mid W' = \{w' \in W \mid [\![\pi]\!]_{\mathcal{M}}(s, W, w') \neq \emptyset\}$ *and* $w \in W'\}$.

5. $[\![\eta x : \pi]\!]_{\mathcal{M}}(s, W, w) = \{\langle s', W', w\rangle \mid \exists s'' : s[x]s''$ & $\langle s', W, w\rangle \in [\![\pi]\!]_{\mathcal{M}}(s'', W', w)\}$.

6. $[\![\Diamond\pi]\!]_{\mathcal{M}}(s, W, w) = \{\langle s, W, w\rangle \mid [\![\pi]\!]_{\mathcal{M}}(s, W, w) \neq \emptyset\}$.

Proposition 13 $\langle s, W, s', w\rangle \in ||\pi||_{\mathcal{M}}$ *iff there is some* $W' \subseteq W$ *with* $\langle s', W', w\rangle \in [\![\pi]\!]_{\mathcal{M}}(s, W, w)$.

Proof: Induction on the structure of π. ∎

Proposition 14 $[\![\pi]\!]_{\mathcal{M}} i = \lambda w \cdot \bigcup_{s \in iw}\{s' \mid \exists W \subseteq gi$ *with* $\langle s, W, w\rangle \in [\![\pi]\!]_{\mathcal{M}}(s, gi, w)\}$.

Proof: Immediate from propositions 12 and 13. ∎

It is also quite easy to show that Dekker's definition of dynamic entailment gives the same relation as our Definition 8.

The functional and multi-valued function formulations of the semantics of DMPL have the advantage that they show the two levels of parametric variation (the local level of assignments and the global level of worlds) much more clearly than the relational formulation.

8 Some Simple Examples

In order to gain some preliminary insight in the semantics of DMPL it is useful to apply the semantic clauses for some very simple examples. Let us assume model $\mathcal{M}$ with a set of worlds W consisting of three worlds over a universe $\{d, d'\}$, where a one place predicate P is defined as follows.

$F_{w_1}(P) = \{d, d'\}$.
$F_{w_2}(P) = \{d\}$.
$F_{w_3}(P) = \emptyset$.

Now suppose also we only have one variable x in the language. Then M^V consists of just two states, which we can indicate as $x \to d$ and $x \to d'$. Here are the results of evaluating some very simple programs.

$[\![Px]\!]_{\mathcal{M}}(\mathbf{1}) = i$, where the function i is given by

$$
\begin{aligned}
iw_1 &= \{x \rightarrow d, x \rightarrow d'\},\\
iw_2 &= \{x \rightarrow d\},\\
iw_3 &= \emptyset.
\end{aligned}
$$

$[\![\neg Px]\!]_{\mathcal{M}}(\mathbf{1}) = j$, where j is given by

$$
\begin{aligned}
jw_1 &= \emptyset,\\
jw_2 &= \{x \rightarrow d'\},\\
jw_3 &= \{x \rightarrow d, x \rightarrow d'\}.
\end{aligned}
$$

Note that j can be got from i, i.e., from the function $[\![Px]\!]_{\mathcal{M}}(\mathbf{1})$, by taking complements with respect to M^V at every index.

$[\![\eta x : Px]\!]_{\mathcal{M}}(\mathbf{1}) = [\![Px]\!]_{\mathcal{M}}(\mathbf{1})$.

$[\![\Diamond Px]\!]_{\mathcal{M}}(\mathbf{1}) = \mathbf{1}$.

$[\![\Diamond\neg Px]\!]_{\mathcal{M}}(\mathbf{1}) = \mathbf{1}$.

The first litmus test of the system is whether it can deal with the predicate logical equivalent of the Veltman contrast between *'It may be raining... It isn't raining.'* versus *'It isn't raining... *It may be raining.'*. We show that the semantics of DMPL make the translation of *'He may be present... He isn't present.'* acceptable, but *'He isn't present...*He may be present.'* unacceptable:

$$
\begin{aligned}
[\![\Diamond Px;\ \neg Px]\!]_{\mathcal{M}}\mathbf{1} &= [\![\neg Px]\!]_{\mathcal{M}}([\![\Diamond Px]\!]_{\mathcal{M}}\mathbf{1})\\
&= [\![\neg Px]\!]_{\mathcal{M}}\mathbf{1}\\
&= j.
\end{aligned}
$$

$$
\begin{aligned}
[\![\neg Px;\ \Diamond Px]\!]_{\mathcal{M}}\mathbf{1} &= [\![\Diamond Px]\!]_{\mathcal{M}}([\![\neg Px]\!]_{\mathcal{M}}\mathbf{1})\\
&= [\![\Diamond Px]\!]_{\mathcal{M}}j\\
&= \mathbf{0}.
\end{aligned}
$$

Secondly, the *maybe* operator should not eliminate any of the current epistemically possible worlds. A litmus test for this is that a sentence like *He may be present, but he may just as well not be present* should come out true in the situation of complete ignorance. And this is precisely what we find:

$$
\begin{aligned}
[\![\Diamond Px;\ \Diamond\neg Px]\!]_{\mathcal{M}}\mathbf{1} &= [\![\Diamond\neg Px]\!]_{\mathcal{M}}([\![\Diamond Px]\!]_{\mathcal{M}}\mathbf{1})\\
&= [\![\Diamond\neg Px]\!]_{\mathcal{M}}\mathbf{1}\\
&= \mathbf{1}.
\end{aligned}
$$

We want to argue that if *may* in the following examples is taken in its epistemic sense, then (13) should turn out to be acceptable, but (14) should not.

(13) *Everyone may have escaped from the fire... Oh dear! Someone hasn't made it.*

(14) *Someone hasn't escaped from the fire. *Everyone may have escaped.*

One can easily imagine a fire fighter uttering discourse (13) as a comment to a colleague during an inspection round. But (14) would sound very weird in those same circumstances.

It is clear the *may* or *might* has more senses than the purely epistemic sense that we are focussing on here. Note, for instance, that the following discourse is intuitively acceptable.

(15) *Someone hasn't escaped from the fire. Everyone might have escaped.*

The acceptability of (15) means that *might* in this example does not express a purely epistemic modality. The second sentence of the discourse does not express that a state of affairs where everyone has escaped is in accordance with one's epistemic state (it is not, witness the information contained in the first sentence). The second sentence expresses something quite different: things might have turned out different from how they in fact turned out. To account for this kind of example (with an *irrealis* flavour), we would have to introduce a different kind of modal operator which looks at all indices irrespective of whether they are still 'in the game' according to the information map. We will not pursue this issue here.

9 Quantified Dynamic Modal Logic

To gain further insight in DMPL, our next goal is to provide an axiomatization. For this we take our cue from two existing axiomatisations for DPL and Update Logic, respectively. Van Eijck and De Vries [12] use Hoare logic to axiomatize DPL, and in [11] they apply the same methods to Update Logic.

The extension of DPL with the epistemic operator suggests the use a modal predicate logic as assertion language in a Hoare style calculus for DMPL programs, but rather than confining ourselves to Hoare style *implications* we want to be able to use the full range of logical connections between static assertions from modal predicate logic and programs from DMPL. We will therefore define a version of quantified dynamic modal logic (inspired by Pratt's dynamic logic [4, 7]) that gives us the expressive power we need. Compare also Van Eijck [10], where the Hoare style calculus of [12] is reformulated in quantified dynamic logic.

Definition 11 (Syntax of qd_L)

1. *If R is an n-place relation in L, and $v_1, \ldots, v_n \in V$, then $Rv_1 \cdots v_n \in form(qd_L)$.*
2. *If $\varphi, \psi \in form(qd_L)$, then $(\varphi \wedge \psi), \neg\varphi, \Diamond\varphi \in form(qd_L)$.*
3. *If $v \in V$ and $\varphi \in form(qd_L)$, then $\exists v\varphi \in form(qd_L)$.*
4. *If $\boldsymbol{\pi} \in prog(qd_L)$ and $\varphi \in form(qd_L)$, then $\langle\boldsymbol{\pi}\rangle\varphi \in form(qd_L)$.*
5. *If R is an n-place relation in L, and $v_1, \ldots, v_n \in V$, then $\boldsymbol{Rv_1 \cdots v_n} \in prog(qd_L)$.*
6. *If $\boldsymbol{\pi}, \boldsymbol{\pi}' \in prog(qd_L)$ then $\boldsymbol{\pi}; \boldsymbol{\pi}', \neg\boldsymbol{\pi}, \Diamond\boldsymbol{\pi} \in prog(qd_L)$.*
7. *If $v \in V$ and $\boldsymbol{\pi} \in prog(qd_L)$, then $\boldsymbol{\eta v : \pi} \in prog(qd_L)$.*
8. *Nothing else is in $form(qd_L)$ or $prog(qd_L)$.*

Let $\top$ be an abbreviation of $\forall x(x = x)$ and $\bot$ an abbreviation of $\neg\top$. As is customary, we abbreviate $\neg(\neg\varphi \wedge \neg\psi)$ as $(\varphi \vee \psi)$, $\neg(\varphi \wedge \neg\psi)$ as $(\varphi \rightarrow \psi)$, $\neg\Diamond\neg\varphi$ as $\Box\varphi$, $\langle\neg(\boldsymbol{\pi}_1; \neg\boldsymbol{\pi}_2)\rangle\varphi$ as $\langle\boldsymbol{\pi}_1 \Rightarrow \boldsymbol{\pi}_2\rangle\varphi$, $\neg\langle\boldsymbol{\pi}\rangle\neg\varphi$ as $[\boldsymbol{\pi}]\varphi$ and $\neg\exists x\neg\varphi$ as $\forall x\varphi$. Also, we omit outermost parentheses for readability.

We consider information maps i as pointers to universal Kripke models gi; i.e., the worlds gi which constitute the epistemic alternatives admitted by i are valuations for a given signature L in some universe M, with accessibility relation $gi \times gi$. Recall from the literature (see e.g. [4]) that the modal logic determined by the class of finite universal frames is S5. Moreover, for any universal model $\mathcal{M}$ (a universal frame with first order valuations in some domain M assigned to all of its worlds) there is a submodel $\mathcal{M}'$ where all worlds have different valuations validating the same formulae. $\mathcal{M}'$ can be got by throwing away the extra copies of the worlds with identical valuations: because of the universal accessibility this makes no difference to validity.

We can now successively define the notions of *satisfaction* of a qd_L formula by a state s for a model $\mathcal{M}$, a world w and an information map i, of *forcing* of a qd_L formula by a world w for a model $\mathcal{M}$ and an information map i, of *acceptance* of a qd_L formula by an information map i in a model $\mathcal{M}$, of *truth* of a qd_L formula in a model $\mathcal{M}$, and finally of *(universal) validity* of a qd_L formula.

Definition 12 (State Forcing)

1. $\mathcal{M}, i, w, s \Vdash Rx_1 \cdots x_n$ *if* $w \models Rx_1 \cdots x_n[s]$.
2. $\mathcal{M}, i, w, s \Vdash x_1 = x_2$ *if* $w \models x_1 = x_2[s]$.
3. $\mathcal{M}, i, w, s \Vdash \neg\varphi$ *if it is not the case that* $\mathcal{M}, i, w, s \Vdash \varphi$.
4. $\mathcal{M}, i, w, s \Vdash \varphi \wedge \psi$ *if* $\mathcal{M}, i, w, s \Vdash \varphi$ *and* $\mathcal{M}, i, w, s \Vdash \psi$.
5. $\mathcal{M}, i, w, s \Vdash \exists v\varphi$ *if for some* $d \in M$, $\mathcal{M}, i, w, s(v|d) \Vdash \varphi$.
6. $\mathcal{M}, i, w, s \Vdash \Diamond\varphi$ *if there is some* $w' \in W$ *with* $s \in iw'$ *and* $\mathcal{M}, i, w', s \Vdash \varphi$.
7. $\mathcal{M}, i, w, s \Vdash \langle\boldsymbol{\pi}\rangle\varphi$ *if there is some* $s' \in (([\![\pi]\!]_{\mathcal{M}}(i(w|\{s\})))w$ *with* $\mathcal{M}, [\![\pi]\!]_{\mathcal{M}} i, w, s' \Vdash \varphi$.

Definition 13 (World Forcing) $\mathcal{M}, i, w \Vdash \varphi$ *if for all* $s \in i(w)$: $\mathcal{M}, i, w, s \Vdash \varphi$.

Definition 14 (Satisfaction) $\mathcal{M}, i \models \varphi$ *if for all* $w \in g(i)$: $\mathcal{M}, i, w \Vdash \varphi$.

Proposition 15 $\mathcal{M}, i \models \langle \pi \rangle \top$ *iff* π *is satisfied by* i *in* $\mathcal{M}$.

Definition 15 (Truth) $\mathcal{M} \models \varphi$ *if for all* $i \in \mathbf{I}_{\mathcal{M}}$: $\mathcal{M}, i \models \varphi$.

Definition 16 (Validity) $\models \varphi$ *if for all* $\mathcal{M}$: $\mathcal{M} \models \varphi$.

Finally, we define a consequence relation for qd_L.

Definition 17 (Consequence for qd_L) $\Gamma \models \varphi$ *if for all triples* $\mathcal{M}, i, w$ *with* $i \in \mathbf{I}_{\mathcal{M}}$, $w \in \mathbf{W}$ *and* $\mathcal{M}, i, w \Vdash \gamma$ *for every* $\gamma \in \Gamma$, *it holds that* $\mathcal{M}, i, w \Vdash \varphi$.

Note that the consequence notion is phrased in terms of world forcing, which in turn is phrased in terms of state forcing. This ensures the 'local' character of the consequence relation. The relation sets up contexts consisting of a world and a set of states for that world. In other words, it uses world–state pairs as contexts for the interpretation of the epistemic modalities and the indefinite assignments.

It is convenient to define the following operation on information maps.

Definition 18 (Pruning of information map i by φ) $i_\varphi = \lambda w \cdot \{s \in iw \mid \mathcal{M}, i, w, s \Vdash \varphi\}$.

Note that $\downarrow_i [\pi] = i_{\langle \pi \rangle \top}$.

We now define localisations of qd formulae. If φ, ψ are in $form(qd_L)$, then $\varphi\downarrow\psi$, the localisation of φ to ψ, is given by the following definition.

Definition 19 (Localised qd formulae $\varphi\downarrow\psi$)

$$
\begin{array}{lcl}
Rv_1 \cdots v_n\downarrow\psi & = & Rv_1 \cdots v_n \wedge \psi \\
v_1 = v_2\downarrow\psi & = & v_1 = v_2 \wedge \psi \\
(\varphi_1 \wedge \varphi_2)\downarrow\psi & = & (\varphi_1\downarrow\psi) \wedge (\varphi_2\downarrow\psi) \\
(\neg\varphi)\downarrow\psi & = & \psi \wedge \neg(\varphi\downarrow\psi) \\
(\exists v\varphi\downarrow\psi & = & \psi \wedge \exists v(\varphi\downarrow\psi) \\
(\Diamond\varphi)\downarrow\psi & = & \psi \wedge \Diamond(\varphi\downarrow\psi)) \\
\langle R\boldsymbol{v}_1 \cdots \boldsymbol{v}_n\rangle\varphi\downarrow\psi & = & Rv_1 \cdots v_n \wedge (\varphi\downarrow\psi) \\
\langle \boldsymbol{v}_1 = \boldsymbol{v}_2\rangle\varphi\downarrow\psi & = & v_1 = v_2 \wedge (\varphi\downarrow\psi) \\
\langle \boldsymbol{\pi}_1; \boldsymbol{\pi}_2\rangle\varphi\downarrow\psi & = & \langle \boldsymbol{\pi}_1\rangle\langle \boldsymbol{\pi}_2\rangle\varphi\downarrow\psi \\
\langle \neg\boldsymbol{\pi}\rangle\varphi\downarrow\psi & = & (\varphi\downarrow[\boldsymbol{\pi}]\bot)\downarrow\psi \\
\langle \boldsymbol{\eta v} : \boldsymbol{\pi}\rangle\varphi\downarrow\psi & = & \psi \wedge \exists v(\langle \boldsymbol{\pi}\rangle\varphi\downarrow\psi) \\
\langle \Diamond\boldsymbol{\pi}\rangle\varphi\downarrow\psi & = & \varphi\downarrow\psi \wedge \Diamond(\langle \boldsymbol{\pi}\rangle\top\downarrow\varphi).
\end{array}
$$

The localisation operator will play an important role in the axioms to be presented in Section 10. The following lemma makes clear what localisation accomplishes.

Lemma 16 (Localisation) *For all* $\mathcal{M}$ *and all* $i \in \mathbf{I}_{\mathcal{M}}$, $i_{\varphi\downarrow\psi} = (i_\psi)_\varphi$.

Proof: Let $\mathcal{M} = \langle M, \mathbf{W}\rangle$ and $i \in \mathbf{I}_{\mathcal{M}}$ be given. By Definition 18, what we have to prove is that for all $w \in W$ and all $s \in iw$: $\lambda w \cdot \{s \in iw \mid \mathcal{M}, i, w, s \Vdash \varphi\downarrow\psi\}$ is identical to $\lambda w \cdot \{s \in i_\psi w \mid \mathcal{M}, i_\psi, w, s \Vdash \varphi\}$. By another application of Definition 18 we see that what we have to prove is that $\mathcal{M}, i, w, s \Vdash \varphi\downarrow\psi$ iff $\mathcal{M}, i, w, s \Vdash \psi$ and $\mathcal{M}, i_\psi, w, s \Vdash \varphi$. We proceed by induction on the structure of φ.

For atomic formulae $R\bar{x}$ we have: $\mathcal{M}, i, w, s \Vdash R\bar{x}\downarrow\psi$ iff $\mathcal{M}, i, w, s \Vdash R\bar{x} \wedge \psi$ iff ($\mathcal{M}, i, w, s \Vdash \psi$ and $\mathcal{M}, i_\psi, w, s \Vdash R\bar{x}$).

The clauses for conjunction, negation and existential quantification are left to the reader.

For the modal operator $\Diamond$ we have:

$\mathcal{M}, i, w, s \Vdash (\Diamond\varphi)\downarrow\psi$
iff $\mathcal{M}, i, w, s \Vdash \psi \wedge \Diamond(\varphi\downarrow\psi)$
iff $\mathcal{M}, i, w, s \Vdash \psi$ and $\mathcal{M}, i, w', s \Vdash \varphi\downarrow\psi$ for some $w' \in W$ with $s \in w'$
iff (ind hyp) $\mathcal{M}, i, w, s \Vdash \psi$ and $\mathcal{M}, i, w', s \Vdash \psi$ and $\mathcal{M}, i_\psi, w', s \Vdash \varphi$ for some $w' \in W$ with $s \in w'$
iff $\mathcal{M}, i, w, s \Vdash \psi$ and $\mathcal{M}, i_\psi, w, s \Vdash \varphi$ and $s \in i_\psi w')$
iff $\mathcal{M}, i, w, s \Vdash \psi$ and $\mathcal{M}, i_\psi, w, s \Vdash \Diamond\varphi$.

We will give two key cases for the reasoning about the program modalities.

For the case of program negation, it suffices to note the following (from the semantic clause for program negation and the induction hypothesis):

$\mathcal{M}, i, w, s \Vdash \langle\neg\boldsymbol{\pi}\rangle\varphi\downarrow\psi$
iff $\mathcal{M}, i, w, s \Vdash [\boldsymbol{\pi}]\bot$ and $\mathcal{M}, i_{[\boldsymbol{\pi}]\bot}, w, s \Vdash \varphi$
iff $\mathcal{M}, i, w, s \Vdash \psi\downarrow[\boldsymbol{\pi}]\bot$.

For the case of the epistemic program modality, just note that it follows from the semantic clause for epistemic program modality and the induction hypothesis that (16).

(16) $\mathcal{M}, i, w, s \Vdash \langle\Diamond\boldsymbol{\pi}\rangle\varphi$ iff $\mathcal{M}, i, w, s \Vdash \varphi \wedge \Diamond\langle\boldsymbol{\pi}\rangle\top$.

The rest follows by using what was established in the induction clauses about $\wedge$ and $\Diamond$.

This completes the induction argument and the proof. ■

Proposition 17 *For all programs π, all models $\mathcal{M}$ and all $i \in \mathbf{I}_\mathcal{M}$, $i_{\langle\boldsymbol{\pi}\rangle\top} = [\![\neg\neg\pi]\!]i$.*

Proof: $i_{\langle\boldsymbol{\pi}\rangle\top}$
= (definition of i-pruning) $\lambda w \cdot \{s \in iw \mid \mathcal{M}, i, w, s \Vdash \langle\boldsymbol{\pi}\rangle\top\}$
= (semantics of $\langle\boldsymbol{\pi}\rangle$) $\lambda w \cdot \{s \in iw \mid [\![\pi]\!]_\mathcal{M}(i(w|\{s\}) \neq \emptyset\}$
= (semantic clause for program negation) $\lambda w \cdot \{s \in iw \mid [\![\neg\pi]\!]_\mathcal{M}(i(w|\{s\}) = \emptyset\}$
= (semantic clause for program negation) $[\![\neg\neg\pi]\!]i$. ■

Proposition 18 *For all test programs π , all models $\mathcal{M}$ and all $i \in \mathbf{I}_\mathcal{M}$, $\downarrow_i [\pi] = i_{\langle\boldsymbol{\pi}\rangle\top} = [\![\pi]\!]i$.*

Proof: Immediate from Propositions 3 and 17. ■

10 Axioms for QDML

The calculus for the semantics of qd_L based on the semantics of DMPL has five sets of axiom schemata: (i) propositional and quantificational schemata, (ii) S5 schemata for the epistemic modality $\Box$, (iii) K-schemata for the program modalities, (iv) atomic test schemata for the program modalities, and (v) program composition schemata.

Propositional and Quantificational Schemata We start by taking the axiom schemata of propositional logic and first order quantification:

A 1 $\varphi \rightarrow (\psi \rightarrow \varphi)$.

A 2 $(\varphi \rightarrow (\psi \rightarrow \chi)) \rightarrow ((\varphi \rightarrow \psi) \rightarrow (\varphi \rightarrow \chi))$.

A 3 $(\neg\varphi \rightarrow \neg\psi) \rightarrow (\psi \rightarrow \varphi)$.

A 4 $\forall v\varphi \rightarrow [t/v]\varphi$, *provided t is free for v in φ.*

A 5 $\varphi \rightarrow \forall v\varphi$, *provided v has no free occurrences in φ.*

A 6 $\forall v(\varphi \rightarrow \psi) \rightarrow (\forall v\varphi \rightarrow \forall v\psi)$.

A 7 $v = v$.

A 8 $v = w \rightarrow (\varphi \rightarrow \varphi')$, *where φ' results from replacing some v-occurrence(s) in φ by w.*

See e.g. Enderton [3] for discussion and motivation.

The S5 Schemata for $\Box$

A 9 $\Box(\varphi \to \psi) \to (\Box\varphi \to \Box\psi)$.

A 10 $\Box\varphi \to \varphi$.

A 11 $\Box\varphi \to \Box\Box\varphi$.

A 12 $\Diamond\Box\varphi \to \varphi$.

These are the propositional S5 modalities. The next axiom gives the Barcan Schema, which takes care of the interaction of quantifiers and the S5 modal operator.

A 13 $\forall v\Diamond\varphi \to \Diamond\forall v\varphi$.

The K-schemata for the program modalities

A 14 $[\pi](\varphi \to \psi) \to ([\pi]\varphi \to [\pi]\psi)$.

Atomic Test Schemata

A 15 $\langle Rt_1 \cdots t_n\rangle\varphi \leftrightarrow \varphi\downarrow Rt_1 \cdots t_n$.

A 16 $\langle t_1 = t_2\rangle\varphi \leftrightarrow \varphi\downarrow t_1 = t_2$.

Program Composition Schemata The schemata for complex programs.

A 17 $\langle\pi_1;\pi_2\rangle\varphi \leftrightarrow \langle\pi_1\rangle\langle\pi_2\rangle\varphi$.

A 18 $\langle\neg\pi\rangle\varphi \leftrightarrow \varphi\downarrow[\pi]\bot$.

A 19 $\langle\eta x : \pi\rangle\varphi \leftrightarrow \exists x\langle\pi\rangle\varphi$.

A 20 $\langle\Diamond\pi\rangle\varphi \leftrightarrow \varphi \wedge \Diamond\langle\pi\rangle\top$.

Rules of inference The rules of inference are as follows.

R 1 (Universal Generalization) *Conclude from* $\vdash \varphi$ *to* $\vdash \forall v\varphi$.

R 2 (Necessitation for $\Box$) *Conclude from* $\vdash \varphi$ *to* $\vdash \Box\varphi$.

R 3 (Necessitation for program modalities) *For every program modality* $\langle\pi\rangle$*:*
conclude from $\vdash \varphi$ *to* $\vdash [\pi]\varphi$.

R 4 (Modus Ponens) *Conclude from* $\vdash \varphi \to \psi$ *and* $\vdash \varphi$ *to* $\vdash \psi$.

The notion of theoremhood in the calculus is standard.

Definition 20 *Formula φ is a theorem of the calculus, notation $\vdash \varphi$, if φ fits one of the axiom schemata or φ follows from theorems in the calculus by an application of one of the inference rules.*

In formulating a notion of deducibility we have to bear in mind that the notion is intended to match the local notion $\models$ for consequence in *qd*. We must make sure that $\Gamma \vdash \varphi$ means that the hypotheses from Γ are interpreted in such a way that they set up a context for the free variables occurring in them. This contrasts with a setup where the formulae from Γ are being handled as extra axioms (admitting universal generalization).

Definition 21 *Formula φ is deducible in the calculus from formula collection Γ, notation $\Gamma \vdash \varphi$, if there are $\gamma_1, \ldots, \gamma_n \in \Gamma$, with $n \geq 0$, such that $\vdash (\gamma_1 \wedge \cdots \wedge \gamma_n) \to \varphi$.*

For the soundness proof, let us see first of all that all the axioms are valid. We skip the propositional, predicate logical and S5 axioms: these are certainly valid.

Atomic predicates axiom. We have to prove that for every $\mathcal{M}$, for every $i \in I_{\mathcal{M}}$, every $w \in g(i)$ and every $s \in i(w)$:

- $\mathcal{M}, i, w, s \Vdash \langle R\overline{x}\rangle\varphi$ iff $\mathcal{M}, i, w, s \Vdash \varphi{\downarrow}R\overline{x}$

We know (semantic clause for $\langle\ \rangle$) that $\mathcal{M}, i, w, s \Vdash \langle R\overline{x}\rangle\varphi$ iff
there is a $s' \in ([\![R\overline{x}]\!](i(w|\{s\})))w$ such that $\mathcal{M}, [\![R\overline{x}]\!]i, w, s' \Vdash \varphi$ iff (def of $[\![R\overline{x}]\!]$)
$s \in ([\![R\overline{x}]\!](i(w|\{s\})))w$ and $\mathcal{M}, [\![R\overline{x}]\!]i, w, s \Vdash \varphi$ iff (def of $[\![R\overline{x}]\!]$, Proposition 18)
$\mathcal{M}, i, w, s \Vdash R\overline{x}$ and $\mathcal{M}, i_{\langle R\overline{x}\rangle\top}, w, s \Vdash \varphi$ iff
$\mathcal{M}, i, w, s \Vdash R\overline{x}$ and $\mathcal{M}, i_{R\overline{x}}, w, s \Vdash \varphi$ iff (localization lemma)
$\mathcal{M}, i, w, s \Vdash \varphi{\downarrow}R\overline{x}$

Composition axiom. We have to prove that for every $\mathcal{M}$, for every $i \in I_{\mathcal{M}}$, every $w \in g(i)$ and every $s \in i(w)$:

- $\mathcal{M}, i, w, s \Vdash \langle\pi_1;\pi_2\rangle\varphi$ iff $\mathcal{M}, i, w, s \Vdash \langle\pi_1\rangle\langle\pi_2\rangle\varphi$

We know (semantic clause for $\langle\ \rangle$) that $\mathcal{M}, i, w, s \Vdash \langle\pi_1;\pi_2\rangle\varphi$ iff
there is some $s' \in ([\![\pi_1;\pi_2]\!](i(w|\{s\})))w$ such that $\mathcal{M}, [\![\pi_1;\pi_2]\!]i, w, s' \Vdash \varphi$ iff (def of $[\![\pi_1;\pi_2]\!]$)
there is some $s' \in ([\![\pi_2]\!]([\![\pi_1]\!](i(w|\{s\})))w$ such that $\mathcal{M}, [\![\pi_2]\!]([\![\pi_1]\!]i), w, s' \Vdash \varphi$ iff
there is a $s'' \in ([\![\pi_1]\!]i(w|\{s\}))w$ such that
$s' \in ([\![\pi_2]\!]([\![\pi_1]\!]i(w|\{s''\})))w$ and $\mathcal{M}, [\![\pi_2]\!]([\![\pi_1]\!]i), w, s' \Vdash \varphi$ iff (semantic clause for $\langle\ \rangle$)
$\mathcal{M}, [\![\pi_1]\!]i, w, s'' \Vdash \langle\pi_2\rangle\varphi$ iff $\mathcal{M}, i, w, s \Vdash \langle\pi_1\rangle\langle\pi_2\rangle\varphi$

Negation axiom. We have to prove that for every $\mathcal{M}$, for every $i \in I_{\mathcal{M}}$, every $w \in g(i)$ and every $s \in i(w)$:

- $\mathcal{M}, i, w, s \Vdash \langle\neg\pi\rangle\varphi$ iff $\mathcal{M}, i, w, s \Vdash \varphi{\downarrow}[\pi]\bot$

We know (semantic clause for $\langle\ \rangle$) that $\mathcal{M}, i, w, s \Vdash \langle\neg\pi\rangle\varphi$ iff
there is a $s' \in ([\![\neg\pi]\!](i(w|\{s\})))w$ and $\mathcal{M}, [\![\neg\pi]\!]i, w, s' \Vdash \varphi$ iff (def of $[\![\neg\pi]\!]$, Proposition 18)
$s \in ([\![\neg\pi]\!](i(w|\{s\})))w$ and $\mathcal{M}, i_{[\pi]\bot}, w, s \Vdash \varphi$ iff (def of $[\![\neg\pi]\!]$)
$[\![\pi]\!](i(w|\{s\})) = \emptyset$ and $\mathcal{M}, i_{[\pi]\bot}, w, s \Vdash \varphi$ iff (semantic clause for $[\ \ ]$)
$\mathcal{M}, i, w, s \Vdash [\pi]\bot$ and $\mathcal{M}, i_{[\pi]\bot}, w, s \Vdash \varphi$ iff (localization lemma)
$\mathcal{M}, i, w, s \Vdash \varphi{\downarrow}[\pi]\bot$

Assignment axiom. We have to prove that for every $\mathcal{M}$, for every $i \in I_{\mathcal{M}}$, every $w \in g(i)$ and every $s \in i(w)$:

- $\mathcal{M}, i, w, s \Vdash \langle\eta x : \pi\rangle\varphi$ iff $\mathcal{M}, i, w, s \Vdash \exists x\langle\pi\rangle\varphi$

We know (semantic clause for $\langle\ \rangle$) that $\mathcal{M}, i, w, s \Vdash \langle\eta x : \pi\rangle\varphi$ iff
there is a $s' \in ([\![\eta x : \pi]\!](i(w|\{s\})))w$ and $\mathcal{M}, [\![\eta x : \pi]\!]i, w, s' \Vdash \varphi$ iff (def of $[\![\eta x : \pi]\!]$)
there is a $s' \in ([\![\pi]\!]((i(w|\{s\}))[x]))w$ and $\mathcal{M}, [\![\pi]\!]((i(w|\{s\}))[x]), w, s' \Vdash \varphi$ iff
(definition of satisfaction and of $i[x]$)
for some $d \in \mathcal{M}$ there is a $s' \in [\![\pi]\!](i[x](w|\{s(x|d)\}))$ and $\mathcal{M}, [\![\pi]\!](i[x]), w, s' \Vdash \varphi$ iff
(semantic clause for $\langle\ \rangle$)
for some $d \in \mathcal{M}$, $\mathcal{M}, i[x], w, s(x|d) \Vdash \langle\pi\rangle\varphi$ iff (semantic clause for $\exists$)
$\mathcal{M}, i[x], w, s \Vdash \exists x\langle\pi\rangle\varphi$ iff $\mathcal{M}, i, w, s \Vdash \exists x\langle\pi\rangle\varphi$.

Modality axiom. We have to prove that for every $\mathcal{M}$, for every $i \in I_{\mathcal{M}}$, every $w \in g(i)$ and every $s \in i(w)$:

- $\mathcal{M}, i, w, s \Vdash \langle\Diamond\pi\rangle\varphi$ iff $\mathcal{M}, i, w, s \Vdash \varphi \wedge \Diamond\langle\pi\rangle\top$

We know (semantic clause for $\langle\ \rangle$) that $\mathcal{M}, i, w, s \Vdash \langle\Diamond\pi\rangle\varphi$ iff
there is some $s' \in ([\![\Diamond\pi]\!](i(w|\{s\})))w$ and $\mathcal{M}, [\![\Diamond\pi]\!]i, w, s' \Vdash \varphi$ iff (definition of $[\![\Diamond\pi]\!]$)
$s \in ([\![\Diamond\pi]\!](i(w|\{s\})))w$ and $\mathcal{M}, [\![\Diamond\pi]\!]i, w, s \Vdash \varphi$ iff (definition of $[\![\Diamond\pi]\!]$)
$\mathcal{M}, i, w, s \Vdash \varphi$ and there is a w' with $s \in i(w')$ and such that $[\![\pi]\!](i(w'|\{s\})))w \neq \emptyset$) iff
$\mathcal{M}, i, w, s \Vdash \varphi$ and there is a w' with $s \in i(w')$ such that there is a $s'' \in ([\![\pi]\!](i(w'|\{s\})))w$

and $\mathcal{M}, [\![\pi]\!]i, w', s'' \Vdash \top$ iff (semantic clause for $\langle\ \rangle$)
$\mathcal{M}, i, w, s \Vdash \varphi$ and there is a w' with $s \in i(w')$ such that $\mathcal{M}, i, w', s \Vdash \langle\pi\rangle\top$ iff (semantic clause for $\Diamond$)
$\mathcal{M}, i, w, s \Vdash \varphi$ and $\mathcal{M}, i, w, s \Vdash \Diamond\langle\pi\rangle\top$.

Finally, it is easy to check that the rules of inference preserve world forcing.

Theorem 19 (Soundness) *If* $\Gamma \vdash \varphi$ *then* $\Gamma \models \varphi$.

Theorem 20 (Completeness) *If* $\Gamma \models \varphi$ *then* $\Gamma \vdash \varphi$.

Proof: First observe that the following translation function * from qd_L to S5 modal predicate logic over L preserves world forcing. This check of the translation function boils down to the reasoning we employed to establish the soundness of the program modality axioms.

$$
\begin{array}{lcl}
(Rt_1 \cdots t_n)^* & = & Rt_1 \cdots t_n \\
(t_1 = t_2)^* & = & t_1 = t_2 \\
(\varphi \wedge \psi)^* & = & \varphi^* \wedge \psi^* \\
(\neg\varphi)^* & = & \neg\varphi^* \\
(\exists v\varphi)^* & = & \exists v\varphi^* \\
(\langle \boldsymbol{Rt_1 \cdots t_n}\rangle\varphi)^* & = & \varphi^*\downarrow Rt_1 \cdots t_n \\
(\langle \boldsymbol{t_1 = t_2}\rangle\varphi)^* & = & \varphi^*\downarrow t_1 = t_2 \\
(\langle \boldsymbol{\pi_1};\ \pi_2\rangle\varphi)^* & = & (\langle \boldsymbol{\pi_1}\rangle\langle \boldsymbol{\pi_2}\rangle\varphi)^* \\
(\langle \boldsymbol{\neg\pi}\rangle\varphi)^* & = & \varphi^*\downarrow([\boldsymbol{\pi}]\bot)^* \\
(\langle \boldsymbol{\eta v}:\boldsymbol{\pi}\rangle\varphi)^* & = & \exists v(\langle \boldsymbol{\pi}\rangle\varphi)^* \\
(\langle \boldsymbol{\Diamond\pi}\rangle\varphi)^* & = & \varphi^* \wedge \Diamond(\langle \boldsymbol{\pi}\rangle\top)^*
\end{array}
$$

Thus, it follows from $\Gamma \models \varphi$ that $\Gamma \models \varphi^*$. Next, use the completeness of S5 modal predicate logic to conclude from $\Gamma \models \varphi^*$ that $\Gamma \vdash \varphi^*$. Finally, note that the translation steps and their inverses in the definition of * are licenced by the atomic test schemata and the program composition schemata of the calculus. This allows us to conclude from $\Gamma \vdash \varphi^*$ that $\Gamma \vdash \varphi$. ■

11 Calculating the Meanings of DMPL Programs

The translation function * from Theorem 20 derives more or less directly from our calculus. We will now demonstrate that it can be used for calculating the meanings of DMPL programs as formulae of S5 modal predicate logic.

Please note that in this paper we are not concerned with giving a translation algorithm from natural language to our representation language, although we believe that this can be done using standard techniques from Montague grammar. All that we want to establish here is that DMPL is a reasonable representation language for natural language meaning, by showing that the translation function * that is implied by our calculus can be used to derive truth conditions for natural language texts in S5 modal predicate logic.

The demonstration will proceed by analyzing the example natural language texts from Section 1, repeated here for convenience.

(17) *A man walked out. Maybe he was angry.*

A reasonable DMPL translation is the following (note that we ignore tense).

(18) ηx : *man* x; *walk-out* x; $\Diamond$*angry* x.

We want to find a specification of the conditions under which this DMPL program succeeds, for these are the truth conditions of the natural language example (in the intended reading, as specified by the DMPL translation). In other words, we want the truth conditions of the following *qd* formula.

(19) $\langle \boldsymbol{\eta x}$: *man* $\boldsymbol{x}$; *walk-out* $\boldsymbol{x}$; $\Diamond$*angry* $\boldsymbol{x}\rangle\top$.

We apply the translation function.

(20) $(\langle \boldsymbol{\eta x}$: *man* $\boldsymbol{x}$; *walk-out* $\boldsymbol{x}$; $\Diamond$*angry* $\boldsymbol{x}\rangle\top)^*$.

What we get is:

$$
\begin{array}{ll}
(\langle \boldsymbol{\eta x} : man\ \boldsymbol{x};\ walk\text{-}out\ \boldsymbol{x};\ \Diamond angry\ \boldsymbol{x}\rangle\top)^* & = \\
(\langle \boldsymbol{\eta x} : man\ \boldsymbol{x}\rangle\langle walk\text{-}out\ \boldsymbol{x}\rangle\langle \Diamond angry\ \boldsymbol{x}\rangle\top)^* & = \\
\exists x(\langle man\ \boldsymbol{x}\rangle\langle walk\text{-}out\ \boldsymbol{x}\rangle\langle \Diamond angry\ \boldsymbol{x}\rangle\top)^*) & = \\
\exists x((\langle walk\text{-}out\ \boldsymbol{x}\rangle\langle \Diamond angry\ \boldsymbol{x}\rangle\top)^*\downarrow man\ x) & = \\
\exists x(man\ x \wedge (\langle walk\text{-}out\ \boldsymbol{x}\rangle\langle \Diamond angry\ \boldsymbol{x}\rangle\top)^*) & = \\
\exists x(man\ x \wedge (\langle \Diamond angry\ \boldsymbol{x}\rangle\top)^*\downarrow angry\ x) & = \\
\exists x(man\ x \wedge walk\text{-}out\ x \wedge (\langle \Diamond angry\ \boldsymbol{x}\rangle\top)^*) & = \\
\exists x(man\ x \wedge walk\text{-}out\ x \wedge \Diamond(\langle angry\ \boldsymbol{x}\rangle\top)^*) & = \\
\exists x(man\ x \wedge walk\text{-}out\ x \wedge \Diamond(\top\downarrow angry\ x) & = \\
\exists x(man\ x \wedge walk\text{-}out\ x \wedge \Diamond angry\ x). &
\end{array}
$$

Before we proceed to the next example, it is useful to list some derived translation instructions.

$$
\begin{array}{lcl}
([\boldsymbol{Rt_1 \cdots t_n}]\varphi)^* & = & Rt_1 \cdots t_n \rightarrow (\varphi^*\downarrow Rt_1 \cdots t_n) \\
([\boldsymbol{t_1 = t_2}]\varphi)^* & = & t_1 = t_2 \rightarrow (\varphi^*\downarrow t_1 = t_2) \\
([\boldsymbol{\pi_1};\ \boldsymbol{\pi_2}]\varphi)^* & = & ([\boldsymbol{\pi_1}][\boldsymbol{\pi_2}]\varphi)^* \\
([\neg\boldsymbol{\pi}]\varphi)^* & = & [\boldsymbol{\pi}]\bot \rightarrow \varphi^*\downarrow([\boldsymbol{\pi}]\bot)^* \\
(\langle \boldsymbol{\pi_1} \Rightarrow \boldsymbol{\pi_2}\rangle\varphi)^* & = & \varphi^*\downarrow([\boldsymbol{\pi_1}]\langle \boldsymbol{\pi_2}\rangle\top)^* \\
([\boldsymbol{\pi_1} \Rightarrow \boldsymbol{\pi_2}]\varphi)^* & = & ([\boldsymbol{\pi_1}]\langle \boldsymbol{\pi_2}\rangle\top)^* \rightarrow \varphi^*\downarrow([\boldsymbol{\pi_1}]\langle \boldsymbol{\pi_2}\rangle\top)^* \\
([\boldsymbol{\eta v} : \boldsymbol{\pi}]\varphi)^* & = & \forall v([\boldsymbol{\pi}]\varphi)^* \\
([\Diamond\boldsymbol{\pi}]\varphi)^* & = & \Diamond(\langle \boldsymbol{\pi}\rangle\top)^* \rightarrow \varphi^*
\end{array}
$$

We can now tackle the second example of Section 1.

(21) *If a man walks out, then maybe he is angry.*

A reasonable DMPL translation:

(22) $(\eta x : man\ x;\ walk\text{-}out\ x) \Rightarrow \Diamond angry\ x.$

We have to calculate the truth conditions of the following *qd* sentence.

(23) $\langle(\boldsymbol{\eta x} : man\ \boldsymbol{x};\ walk\text{-}out\ \boldsymbol{x}) \Rightarrow \Diamond angry\ \boldsymbol{x}\rangle\top.$

Again the * function allows us to translate this into S5 modal predicate logic.

$$
\begin{array}{ll}
(\langle(\boldsymbol{\eta x} : man\ \boldsymbol{x};\ walk\text{-}out\ \boldsymbol{x}) \Rightarrow \Diamond angry\ \boldsymbol{x}\rangle\top)^* & = \\
\top\downarrow([\boldsymbol{\eta x} : man\ \boldsymbol{x};\ walk\text{-}out\ \boldsymbol{x}]\langle \Diamond angry\ \boldsymbol{x}\rangle\top)^* & = \\
([\boldsymbol{\eta x} : man\ \boldsymbol{x};\ walk\text{-}out\ \boldsymbol{x}]\langle \Diamond angry\ \boldsymbol{x}\rangle\top)^* & = \\
([\boldsymbol{\eta x} : man\ \boldsymbol{x}][walk\text{-}out\ \boldsymbol{x}]\langle \Diamond angry\ \boldsymbol{x}\rangle\top)^* & = \\
\forall x([man\ \boldsymbol{x}][walk\text{-}out\ \boldsymbol{x}]\langle \Diamond angry\ \boldsymbol{x}\rangle\top)^* & = \\
\forall x(man\ x \rightarrow ([walk\text{-}out\ \boldsymbol{x}]\langle \Diamond angry\ \boldsymbol{x}\rangle\top)^*) & = \\
\forall x(man\ x \rightarrow (walk\text{-}out\ x \rightarrow \langle \Diamond angry\ \boldsymbol{x}\rangle\top)^*)) & = \\
\forall x(man\ x \rightarrow (walk\text{-}out\ x \rightarrow \Diamond angry\ x)). &
\end{array}
$$

12 Conclusion

We have investigated the logic of a representation language for natural lanuage meaning that can handle dynamic binding of variables and dynamically interpreted epistemic modalities. It would be a straightforward extension of the representation language to add generalized quantifiers with a mechanism for internal dynamic bindings, along the lines of [12]. But the real challenge is of course to apply the representation language or a suitable modification of it to puzzles of modal subordination [8].

Acknowledgements

This paper has benefited from helpful comments by Paul Dekker, Tim Fernando, Wilfried Meyer Viol, Kees Vermeulen and Albert Visser.

References

[1] P. Dekker. Update semantics and dynamic interpretation. Manuscript, University of Amsterdam (handout Santa Cruz conference on Formal Linguistics), 1991.

[2] P. Dekker. *Dynamic Interpretation of Natural Language.* PhD thesis (in preparation), Department of Philosophy, University of Amsterdam, 1992.

[3] H.B. Enderton. *A Mathematical Introduction to Logic.* Academic Press, 1972.

[4] R. Goldblatt. *Logics of Time and Computation*, volume 7 of *CSLI Lecture Notes.* CSLI, Stanford, 1987. Distributed by University of Chicago Press.

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

[6] J. Groenendijk and M. Stokhof. Two theories of dynamic semantics. In J. van Eijck, editor, *Logics in AI—European Workshop JELIA '90*, pages 55–64, Berlin, 1991. Springer Lecture Notes in Artificial Intelligence.

[7] V. Pratt. Semantical considerations on Floyd–Hoare logic. *Proceedings 17th IEEE Symposium on Foundations of Computer Science*, pages 109–121, 1976.

[8] C. Roberts. Modal subordination and pronominal anaphora in discourse. *Linguistics and Philosophy*, 12:683–721, 1989.

[9] J. van Benthem. General dynamics. *Theoretical Linguistics*, pages 159–201, 1991.

[10] J. van Eijck. Axiomatizing dynamic predicate logic with quantified dynamic logic (abstract). In *ERCIM Workshop on Theoretical and Experimental Aspects of Knowledge Representation*, pages 125–137, Pisa, 1992. CNR.

[11] J. van Eijck and F.J. de Vries. A sound and complete calculus for update logic. Technical Report CS–R9155, CWI, Amsterdam, 1991. To appear in the Proceedings of the Eighth Amsterdam Colloquium.

[12] J. van Eijck and F.J. de Vries. Dynamic interpretation and Hoare deduction. *Journal of Logic, Language, and Information*, 1:1–44, 1992.

[13] F. Veltman. Defaults in update semantics. Manuscript, Department of Philosophy, University of Amsterdam, 1991. To appear in the Journal of Philosophical Logic.

[14] C.F.M. Vermeulen. Sequence semantics for dynamic predicate logic. Logic Group Preprint Series 60, Department of Philosophy, University of Utrecht, January 1991.