Updates that Grow on Trees

Albert Visser

Philosophy, Faculty of Humanities, Utrecht University,
Janskerkhof 13, 3512BL Utrecht, The Netherlands
a.visser@uu.nl

Abstract

In this paper we solve the following problem: what is to Update Semantics as Discourse Representation Theory is to Dynamic Predicate Logic?

This paper is dedicated to Jeroen, Martin en Frank. Their work always was, is and will be an inspiration!

1 Introduction

The system of *Coreference and Modality* ([GSV96]) is, without any doubt, the zenith of the work of the Amsterdam school of dynamic semantics. It integrates the great earlier contributions of Jeroen Groenendijk & Martin Stokhof ([GS91]) and of Frank Veltman ([Vel96]). The authors succeed in unifying the *prima facie* quite different ideas of validity in dynamic predicate logic on the one hand and update semantics on the other. This unification involved understanding how Dynamic Predicate Logic can be viewed as *eliminative*, in other words: how to develop a non-destructive notion of reset.[1] Jeroen, Martin and Frank managed to keep the complexity of the definitions within reasonable bounds.

However great the achievement, many riddles concerning how the system really works still remain. One such riddle is the problem of the *space of meanings*. In the present note, I want to move a step back to Frank's work to study *the space of meanings* in propositional update semantics. The real target of this should be in the end to understand meanings in *Coreference and Modality* better

The dynamic 'logics' created in [GS91], [Vel96], [GSV96] lack true logicality. This means that the meanings that *you are allowed to* assign to the atomic formulas are only a small subclass of the possible meanings. Thus, the class of meanings that can be generated using the operations of the logic from the atomic meanings is only a tiny subclass of the total space of meanings —even if you are allowed to start with all possible static conditions. One can look upon this fact as a bug, but one can see it also as an opportunity as I will explain below.

In the case of Dynamic Predicate Logic, the meanings are relations between assignments or, if you wish, distributive update functions of sets of assignments. In the set-up of Coreference and Modality, the meanings are eliminative update functions over sets of world assignment pairs: a truly awe-inspiring totality. The problem is not so much the greater cardinality, but that we have no good global insight on how the meanings behave even for the generated fragment. Particularly, the intuition of *distributivity* is very strong. One feels somewhat lost without it.

[1] The work of Kees Vermeulen on Referent Systems ([Ver95]) provided an important piece of the puzzle.

In Dynamic Predicate we can reduce the class of meanings to DRS-like objects: pairs of a dynamic context and a set of assignments.[2] The idea of this note is to do something analogous for update semantics: find more 'representational' objects to represent update functions. Doing this we reduce the class of meanings in a substantial way, even if we still preserve the generated meanings. The basic insight is that Frank Veltman's update functions are indeed non-distributive, but that the amount of non-distributivity is *under control*: it is just due to a finite number of distributivity breaking actions.

We only give a realization of the idea in the propositional case. The proposed meanings are binary trees of choices of elements of a Boolean algebra. How to give a full realization for the system of Coreference and Modality remains an open question.

2 Update Semantics

The language of update semantics is defined as follows, where 'p' ranges over a set of propositional variables.

- $\phi ::= \bot \mid \top \mid p \mid \phi \cdot \psi \mid \Diamond(\phi) \mid \sim(\phi)$.

The notation '$\Diamond$' stands for *maybe*. For the interpretation, we fix a Boole algebra $\mathcal{B}$. We let a, b, ..., range over the domain of $\mathcal{B}$. We first specify Veltman's original semantics for the system. Interpretations are functions from $\mathcal{B}$ to $\mathcal{B}$. Let α be an assignment from the propositional variables to the domain of $\mathcal{B}$. We define, for s in $\mathcal{B}$:

- $s[p]_\alpha := (s \wedge \alpha(p))$.
- $s[\phi \cdot \psi]_\alpha := s([\phi]_\alpha \cdot [\psi]_\alpha) := s[\phi]_\alpha[\psi]_\alpha$.
- $s[\Diamond\phi]_\alpha := \begin{cases} s & \text{if } s[\phi]_\alpha \neq \bot \\ \bot & \text{if } s[\phi]_\alpha = \bot \end{cases}$.
- $s[\sim \phi]_\alpha := s \wedge \neg(s[\phi]_\alpha)$.
- $s \models_\alpha \phi :\Leftrightarrow s \leq s[\phi]_\alpha$.

Our alternative semantics for updates consists of binary boolean labeled trees. The idea is that the binary branchings represent the choices provided by the *maybe* operator. The depth of the tree stands for the amount of 'distributivity breaking'.

The variable 'a' ranges over elements of our Boole algebra. Here is the definition of the set of labeled trees.

- $\sigma ::= a \mid \mathsf{b}_a(\sigma, \sigma)$.

The composition $\tau \cdot \nu$ of trees τ and ν, is the result of applying the following procedure to each leaf of τ. Suppose the leaf is labeled a. We first relabel ν by replacing each label b by $a \wedge b$. Then, we append the relabeled copy of ν below the leaf, where we identify the leaf with the root of the relabeld copy of ν. Here is the definition by recursion.

[2] If you use Vermeulen-style referent systems you get a very neat variant of the DRT-semantics of Henk Zeevat ([Zee91]). In this variant, the mapping from syntactical object to DRS, the mapping from DRS to semantic DRS and the mapping from semantic DRS to DPL-style reset-relation behave compositionally. This is the main argument to dissolve the muddle where people viewed non-compositionality as the hallmark of a representational semantics.

$$
\begin{array}{rcl}
(a_0 \cdot a_1) \cdot a_2 & = & (a_0 \wedge a_1) \wedge a_2 \\
& = & a_0 \wedge (a_1 \wedge a_2) \\
& = & a_0 \cdot (a_1 \cdot a_2) \\
(a_0 \cdot a_1) \cdot \mathsf{b}_{a_2}(\tau_{20}, \tau_{21}) & = & (a_0 \wedge a_1) \cdot \mathsf{b}_{a_2}(\tau_{20}, \tau_{21}) \\
& = & \mathsf{b}_{(a_0 \wedge a_1) \wedge a_2}((a_0 \wedge a_1) \cdot \tau_{20}, (a_0 \wedge a_1) \cdot \tau_{21}) \\
& = & \mathsf{b}_{a_0 \wedge (a_1 \wedge a_2)}((a_0 \cdot a_1) \cdot \tau_{20}, (a_0 \cdot a_1) \cdot \tau_{21}) \\
& \overset{\text{IH}}{=} & \mathsf{b}_{a_0 \wedge (a_1 \wedge a_2)}(a_0 \cdot (a_1 \cdot \tau_{20}), a_0 \cdot (a_1 \cdot \tau_{21})) \\
& = & a_0 \cdot \mathsf{b}_{a_1 \wedge a_2}(a_1 \cdot \tau_{20}, a_1 \cdot \tau_{21}) \\
& = & a_0 \cdot (a_1 \cdot \mathsf{b}_{a_2}(\tau_{20}, \tau_{21})) \\
(a_0 \cdot \mathsf{b}_{a_1}(\tau_{10}, \tau_{11})) \cdot \tau_2 & = & \mathsf{b}_{a_0 \wedge a_1}(a_0 \cdot \tau_{10}, a_0 \cdot \tau_{11}) \cdot \tau_2 \\
& = & \mathsf{b}_{a_0 \wedge a_1}((a_0 \cdot \tau_{10}) \cdot \tau_2, (a_0 \cdot \tau_{11}) \cdot \tau_2) \\
& \overset{\text{IH}}{=} & \mathsf{b}_{a_0 \wedge a_1}(a_0 \cdot (\tau_{10} \cdot \tau_2), a_0 \cdot (\tau_{11} \cdot \tau_2)) \\
& = & a_0 \cdot \mathsf{b}_{a_1}(\tau_{10} \cdot \tau_2, \tau_{11} \cdot \tau_2) \\
& = & a_0 \cdot (\mathsf{b}_{a_1}(\tau_{10}, \tau_{11}) \cdot \tau_2) \\
(\mathsf{b}_{a_0}(\tau_{00}, \tau_{11}) \cdot \tau_1) \cdot \tau_2 & = & \mathsf{b}_{a_0}(\tau_{00} \cdot \tau_1, \tau_{11} \cdot \tau_1) \cdot \tau_2 \\
& = & \mathsf{b}_{a_0}((\tau_{00} \cdot \tau_1) \cdot \tau_2, (\tau_{11} \cdot \tau_1) \cdot \tau_2) \\
& \overset{\text{IH}}{=} & \mathsf{b}_{a_0}(\tau_{00} \cdot (\tau_1 \cdot \tau_2), \tau_{11} \cdot (\tau_1 \cdot \tau_2)) \\
& = & \mathsf{b}_{a_0}(\tau_{00}, \tau_{11}) \cdot (\tau_1 \cdot \tau_2)
\end{array}
$$

Figure 1: Associativity

- $a \cdot b := a \wedge b$,
- $a \cdot \mathsf{b}_b(\tau_0, \tau_1) := \mathsf{b}_{a \wedge b}(a \cdot \tau_0, a \cdot \tau_1)$,
- $\mathsf{b}_a(\sigma_0, \sigma_1) \cdot \tau := \mathsf{b}_a(\sigma_0 \cdot \tau, \sigma_1 \cdot \tau)$.

We verify a desired property of the operaration $() \cdot ()$.

Theorem 2.1. *The operation $\cdot$ on trees gives us a monoid with $\top$ as unit.*

Proof. The fact that $\top \cdot \tau = \tau \cdot \top = \tau$ can be verified by two simple inductions. The verification by induction of associativity is given in Figure 1 ❑

The *maybe* operation on trees is as follows: append below each leaf one branch to the left with new leaf labeled $\top$ and one leaf to the right with new leaf labeled $\bot$. Here is the definition with recursion.

- $\Diamond a := \mathsf{b}_a(\top, \bot)$,
- $\Diamond \mathsf{b}_a(\sigma_0, \sigma_1) := \mathsf{b}_a(\Diamond \sigma_0, \Diamond \sigma_1)$.

The increase in depth of the tree corresponds to the increase in distributivity breaks. The branching stands for the binary choice introduced by *maybe*.

Negation is very simple: relabel each leaf with the negation of its label.

- $\sim a := \neg a$,
- $\sim \mathsf{b}_a(\sigma_0, \sigma_1) := \mathsf{b}_a(\sim \sigma_0, \sim \sigma_1)$.

Thus we have defined our algebra on trees.

We evaluate trees on a state s in $\mathcal{B}$ as follows. We start at the root of the tree. Suppose we are at a node with label a. If the node is not a leaf, we check whether $(s \wedge a) \neq \bot$ or $(s \wedge a) = \bot$. If the first, we move down the left branch, if the last down the right. If the node is a leaf, we output value $s \wedge a$. Here is the recursion.

- $s \star a := s \wedge a$,
- $s \star \mathsf{b}_a(\tau_0, \tau_1) := \begin{cases} s \star \tau_0 & \text{if } (s \wedge a) \neq \bot \\ s \star \tau_1 & \text{if } (s \wedge a) = \bot \end{cases}$.

The operation $\star$ is a bit like the collapse of the wave function in quantum mechanics. In the tree the choices are left open. By applying the operation they are actualized.

The following theorem articulates the basic insight concerning $\star$.

Theorem 2.2. *We have:*

1. $s \star (\tau \cdot \nu) = (s \star \tau) \star \nu$,
2. $s \star \Diamond\tau = \begin{cases} s & \textit{if } s \star \tau \neq \bot \\ \bot & \textit{if } s \star \tau = \bot \end{cases}$.
3. $s \star \sim \tau = (s \wedge \neg(s \star \tau))$.

Proof. The proof is by three inductions. The induction for (i) is given in Figure 2. The induction for (ii) is given in Figure 3. The induction in (iii) is given in Figure 4. ❑

Let α be again an assignment of elements of the Boole algebra to the propositional variables. We define tree-semantics as follows:

- $\tau_{p,\alpha} := \alpha(p)$,
- $\lambda\phi \cdot \tau_{\phi,\alpha}$ commutes with composition, $\Diamond$, and $\sim$.

If, for a given formula ϕ, we first find its associated tree and, subsequently, evaluate it for s, the result is the same as evaluating the associated update function for s.

Theorem 2.3. $s[\phi]_\alpha = s \star \tau_{\phi,\alpha}$.

The proof is a simple induction using Theorem 2.2.

One can show that many trees give the same update function. In case $\mathcal{B}$ is finite, one can easily show that each eliminative update function is tree definable. In the infinity case this is unclear.
Open Question 2.4.

1. Can we find normal forms for trees such that each definable update function corresponds to precisely one normal form?

$$
\begin{array}{rcl}
s \star (a \cdot b) & = & s \star (a \wedge b) \\
 & = & s \wedge (a \wedge b) \\
 & = & (s \wedge a) \wedge b \\
 & = & (s \star a) \star b \\
s \star (a \cdot \mathsf{b}_b(\nu_0, \nu_1)) & = & s \star \mathsf{b}_{a \wedge b}(a \cdot \nu_0, a \cdot \nu_1) \\
 & = & \begin{cases} s \star (a \cdot \nu_0) & \text{if } s \wedge (a \wedge b) \neq \bot \\ s \star (a \cdot \nu_1) & \text{if } s \wedge (a \wedge b) = \bot \end{cases} \\
 & \overset{\text{IH}}{=} & \begin{cases} (s \star a) \star \nu_0 & \text{if } (s \wedge a) \wedge b \neq \bot \\ (s \star a) \star \nu_1 & \text{if } (s \wedge a) \wedge b = \bot \end{cases} \\
 & = & (s \star a) \star \mathsf{b}_b(\nu_0, \nu_1) \\
s \star (\mathsf{b}_a(\tau_0, \tau_1) \cdot \nu) & = & s \star \mathsf{b}_a(\tau_0 \cdot \nu, \tau_1 \cdot \nu) \\
 & = & \begin{cases} s \star (\tau_0 \cdot \nu) & \text{if } s \wedge a \neq \bot \\ s \star (\tau_1 \cdot \nu) & \text{if } s \wedge a = \bot \end{cases} \\
 & \overset{\text{IH}}{=} & \begin{cases} (s \star \tau_0) \star \nu & \text{if } s \wedge a \neq \bot \\ (s \star \tau_1) \star \nu & \text{if } s \wedge a = \bot \end{cases} \\
 & = & (s \star \mathsf{b}_a(\tau_0, \tau_1)) \star \nu
\end{array}
$$

Figure 2: $s \star (\tau \cdot \nu) = (s \star \tau) \star \nu$

$$
\begin{array}{rcl}
s \star \Diamond a & = & s \star \mathsf{b}_a(\top, \bot) \\
 & = & \begin{cases} s \star \top & \text{if } s \wedge a \neq \bot \\ s \star \bot & \text{if } s \wedge a = \bot \end{cases} \\
 & = & \begin{cases} s & \text{if } s \star a \neq \bot \\ \bot & \text{if } s \star a = \bot \end{cases} \\
s \star \Diamond \mathsf{b}_a(\tau_0, \tau_1) & = & s \star \mathsf{b}_a(\Diamond \tau_0, \Diamond \tau_1) \\
 & = & \begin{cases} s \star \Diamond \tau_0 & \text{if } s \wedge a \neq \bot \\ s \star \Diamond \tau_1 & \text{if } s \wedge a = \bot \end{cases} \\
 & = & \begin{cases} s & \text{if } s \wedge a \neq \bot \text{ and } s \star \tau_0 \neq \bot \\ \bot & \text{if } s \wedge a \neq \bot \text{ and } s \star \tau_0 = \bot \\ s & \text{if } s \wedge a = \bot \text{ and } s \star \tau_1 \neq \bot \\ \bot & \text{if } s \wedge a = \bot \text{ and } s \star \tau_1 = \bot \end{cases} \\
 & = & \begin{cases} s & \text{if } s \wedge a \neq \bot \text{ and } s \star \tau_0 \neq \bot \\ s & \text{if } s \wedge a = \bot \text{ and } s \star \tau_1 \neq \bot \\ \bot & \text{if } s \wedge a \neq \bot \text{ and } s \star \tau_0 = \bot \\ \bot & \text{if } s \wedge a = \bot \text{ and } s \star \tau_1 = \bot \end{cases} \\
 & = & \begin{cases} s & \text{if } s \star \mathsf{b}_a(\tau_0, \tau_1) \neq \bot \\ \bot & \text{if } s \star \mathsf{b}_a(\tau_0, \tau_1) = \bot \end{cases}
\end{array}
$$

Figure 3: $s \star \Diamond \tau = s$ if $s \star \tau \neq \bot$ and $s \star \Diamond \tau = \bot$ if $s \star \tau = \bot$

$$
\begin{aligned}
s \star \sim a &= s \star \neg a \\
&= s \wedge \neg a \\
&= s \wedge \neg (s \wedge a) \\
&= s \wedge \neg (s \star a) \\
s \star \sim \mathsf{b}_a(\tau_0, \tau_1) &= s \star \mathsf{b}_a(\sim \tau_0, \sim \tau_1) \\
&= \begin{cases} s \star \sim \tau_0 & \text{if } s \wedge a \neq \bot \\ s \star \sim \tau_1 & \text{if } s \wedge a = \bot \end{cases} \\
&= \begin{cases} s \wedge \neg (s \star \tau_0) & \text{if } s \wedge a \neq \bot \\ s \wedge \neg (s \star \tau_1) & \text{if } s \wedge a = \bot \end{cases} \\
&= s \wedge \neg (s \star \mathsf{b}_a(\tau_0, \tau_1))
\end{aligned}
$$

Figure 4: $s \star \sim \tau = s \wedge \neg (s \star \tau)$

2. Is every tree-definable update function definable in Veltman's system?
3. Can we generalize the tree representation to the *Coreference and Modality* system of [GSV96]?

❑

References

[GS91] J. Groenendijk and M. Stokhof. Dynamic predicate logic. *Linguistics and Philosophy*, 14:39–100, 1991.

[GSV96] 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.

[Vel96] F. Veltman. Defaults in Update Semantics. *Journal of Philosophical Logic*, 25:221–261, 1996.

[Ver95] C.F.M. Vermeulen. Merging without mystery, variables in dynamic semantics. *Journal of Philosophical Logic*, 24:405–450, 1995.

[Zee91] H. Zeevat. A compositional approach to DRT. *Linguistics and Philosophy*, 12:95–131, 1991.