Interaction of Modalities in Discourse: a Monoidal Account

Kees Vermeulen

Preprint nr. 009 October 1999

Artificial Intelligence Preprint Series

Interaction of modalities in discourse: a monoidal account

C.F.M. Vermeulen

September 1999

Abstract

The topic of the present paper is the interaction of modalities in discourse. We make some initial observations about this phenomenon and discuss some connections with other discourse phenomena. Then we present two styles of dynamic semantics for modal interaction: an update semantics in the style of Veltman [Vel96] and a monoidal semantics in the style of Visser [VV96]. The monoidal style is an improvement on the update style, as it forces us to break down the complex update procedures into elementary operations on locations in memory, thus providing a rational reconstruction of the dynamics underlying the update functions. By means of a translation it is shown that the monoidal semantics of the interaction of modalities preserves all the pleasant properties of the update semantics, among which is the decidability result of [Ver99].

1 Introduction

This paper is about the semantic interaction of modalities in discourse. The crucial examples of this phenomenon have been known for a long time.[1] They usually go under the heading *modal subordination.* Let's look at some examples:

(1a) A lion might come in.	(2a) A lion might come in.
(1b) It would eat you first.	(2b) It might eat you first.
(1c) It might eat me later.	(2c) It might choose me instead.

A superficial glance at these discourses suggests a representation as in:

(1) $\Diamond p \wedge \Box q \wedge \Diamond r$ (2) $\Diamond p \wedge \Diamond q \wedge \Diamond r$

But some further consideration already suffices to see that this cannot be right. We see that the appropriate interpretations involve an *interaction* of the modalities: what the lion will do in the (b)-parts is to be considered only in situations where the lion actually has come in, as indicated in the (a)-parts. As an attempt to get this interaction right, we could try to get away with a simple nesting of modalities:

[1]There are many interesting points of reference. We suggest [Rob89] and [Ash96] as starting points. A more recent contribution in the dynamic style is in [Kib94].

(1) $\Diamond(p \wedge \Box(q \wedge \Diamond r))$ (2) $\Diamond(p \wedge \Diamond(q \wedge \Diamond r))$

But again it is easy to see that this won't do. For one, the claim in (1b) clearly is *not* that it might be the case that a lion comes in and would eat you first: so, we do not want anything like $\Diamond(p \wedge \Box q)$. (1b) claims that *whenever* a lion comes in, it would eat you first. And similarly for (2b). Therefore, nesting the modalities is not a good idea.
Furthermore, there is a notable distinction in the (c)-parts of the examples which is not accounted for in either of the representation-attempts: the information in (1c) is about situations where a lion has come in *and* eats you first. In example (2c), however, we are just considering situations in which a lion has come in.[2] This means that, in the general case, there is not just one extra situation that we have to keep in mind. Apart from the real world, there are *several* other options available as a context for the interpretation of modal expressions. This fact is easily overlooked.
The best approximations of the meanings of the examples in ordinary propositional modal logic are as follows.[3]

(1) $\Diamond p$
$\wedge\ (p \rightarrow \Box q)$
$\wedge\ (p \wedge q \rightarrow \Diamond r)$

(2) $\Diamond p$
$\wedge\ (p \rightarrow \Diamond q)$
$\wedge\ (p \rightarrow \Diamond r)$

Crucial facts about these approximations are: (i) some form of conditionalisation is required; (ii) the antecedents of the implications contain no modalities. (i) is a good reason for describing the phenomenon as *subordination.* (ii) is a confirmation of the observation above that there is no nesting of modalities involved. Both (i) and (ii) provide a serious obstacle for any attempt at a compositional translation of modal subordination into an ordinary modal language. Although we do not provide such a translation here, this is one good motivation for considering other languages for the representation of the phenomena.
We consider yet another example of modal interaction. This example is of a more complex kind than the ones usually considered in the literature.[4] It shows that we are not just dealing with subordination (=conditionalisation).

(3)		(3')	
	Morgen kan ze zwanger (blijken te) zijn.		Tomorrow we might find out she is pregnant.
	Het kan ook nog vandaag.		Or we might find out already today.
	Het kan van de behanger zijn,		It could be from the upholsterer,
	Of van een Franse zanger zijn,		Or from a French chansonnier,
	Of iemand uit Den Haag.		Or someone from The Hague.

[2]Arguably it is even assumed that it has *not* eaten you. But this seems to be a subtlety of the meaning of *instead*, that we do not claim to represent here. See for example [Ash96] on the negative implications of such indicators of contrast.

[3]It could be argued that the examples also have a 'generic' flavour. In a simple modal language this generic flavour can be approximated too: we can prefix the implications with an extra $\Box$: $\Box(p \rightarrow \Box q)$. However, the semantics of generics seems to be an issue that is orthogonal to our main concerns here. So, we will ignore such complications in the remainder of this paper.

[4]The Dutch text is from the famous Dutch song *Op een mooie Pinksterdag*, lyrics: A. Schmidt, music: H. Bannink.

The simple modal approximation of this example is something like:

$$
\begin{aligned}
&(\Diamond \text{ tomorrow} \\
&\wedge \Diamond \text{ today}) \\
&\wedge (\text{ tomorrow} \vee \text{ today} \rightarrow \Diamond \text{ upholsterer}) \\
&\wedge (\text{ tomorrow} \vee \text{ today} \rightarrow \Diamond \text{ chansonnier}) \\
&\wedge (\text{ tomorrow} \vee \text{ today} \rightarrow \Diamond \text{ the-hague})
\end{aligned}
$$

The example shows that apart from traditional subordination effects also more involved patterns of interaction are available. In (3) the 'antecedent' of the modal interaction is formed as the *union* of two options mentioned in the discourse sofar: tomorrow or today. The options mentioned form a list of alternatives that could be the case in this 'antecedent'-situation. The important point about this example is that is not a simple adding up of modal subordinations as in (1) or (2). Hence in a general account of modal interaction different operations on modal 'antecedents' will have to be present.
Unfortunately an account for complex examples such as these is not within reach yet. Simply adding lots of operations in the logic to form more patterns of interaction is an unattractive option: this would all too easily lead to silly forms of overgeneration. Especially the relation with discourse structure is systematic source of constraints which should be investigated. Some work has been done here—for example in [Ash96]—but additional investigations seem to be called for, before a satisfactory formalisation becomes a realistic goal. Therefore we postpone the formalisation of the complex cases and start with analysing the simple cases, such as (1) and (2).
For the analysis of these examples we first consider a propositional language with indexed modalities. We provide this language with an update semantics. Then we transform the setting to obtain an algebraic semantics of the phenomena along the lines of [VV96]. The relation between the two styles of semantics is discussed and made precise in several steps.

1.1 Quantificational subordination and E-type anaphora

Before we continue with the formalisation of modal interaction in discourse, we point briefly at the relation with two other important phenomena in the semantics of discourse: quantificational subordinations and E-type anaphora. It seems that quantifiers in discourse interact in very much the same way as modal expressions. This is not a new observation. And it also is not surprising, given the well-established semantic correspondence between quantifiers and modalities (cf. [Ben85]). But it confirms that the phenomenon we are considering is not an isolated atrocity: it is an instance of general patterns of interaction in discourse.
We give some examples parallel to (1) and (2) above.

(4) (It started to rain.)
Some people came in.
Most (of them) had
no umbrella.

(5) (It started to rain.)
Some people came in.
Some of them had
no umbrella.

Some did not even have a rain coat.	Some (others) were afraid to catch a cold.

We see that (4) is just like (1) and that (5) is very similar to (2). Good approximations in a standard approach to (generalised) quantification would be as follows.[5]

(4) $\mathsf{Some}(p, c)$
$\wedge\ \mathsf{Most}(p \wedge c, u)$
$\wedge\ \mathsf{Some}(p \wedge c \wedge u, r)$

(5) $\mathsf{Some}(p, c)$
$\wedge\ \mathsf{Some}(p \wedge c, u)$
$\wedge\ \mathsf{Some}(p \wedge c, a)$

This shows how the accumulation of material in the restrictor of the quantifiers works along the same lines as the accumulation of material in the conditions of the implications above. It also shows that there is no nesting of quantifiers involved. We have discussed such examples in [Ver97].[6] Below we concentrate on the examples with modals, but the similarities with the patterns of interaction of quantifiers are so striking that the account of modal subordination we present, can be converted immediately into a semantics for the quantificational dependencies.
E-type anaphora[7] seems to be yet another related phenomenon. In fact, E-types are very much like the examples of quantificational interaction that we saw above. The difference is that in the E-type examples the quantifier tends to remain implicit. Consider the following case.

(6) (It started to rain.)
Some people came in.
They had no umbrella.
They did not even have a rain coat.
$\mathsf{Some}(p, c)\ \wedge\ \mathsf{All}(p \wedge c, u)\ \wedge\ \mathsf{All}(p \wedge c \wedge u, r)$

Finally we note that there are also mixed examples, where quantifiers modals and E-types interact. It is not difficult to make up lots of examples, but let's stick to the rainy-day situation.

(7) It could be a rainy day.
Then most people would come inside.
They may not have an umbrella.
Some will not even have a raincoat.
Or they will simply be afraid to catch a cold.

We conclude that the data suggest that in discourse patterns of interaction exist between all kinds of expressions. Examples are: modals, quantifiers, E-type anaphors. These patterns are an irresistable invitation to consider logics that describe such interactions. Below we consider a logic that accounts for

[5]As we are interested in the general patterns the names involved do not really matter. Still, it might be helpful to read: p as *people*; c as *comers-in*; u as *umbrellaless*; r as *rain-coat-less*; a as *afraid*.

[6]The effects are also mentioned already, for example, in [GNP92] and [Rob89].

[7]See [Eva80] for discussion.

some of the interaction between modalities. Analogous logics can then easily be developed for the other phenomena.
As we pointed out above, our logic will not allow for all the complexity of interaction that we found in the examples. It seems too early for a serious attempt to formalise the complex patterns as there is no clear general view on the possibilities and restrictions.

2 Indexed modalities in update semantics

In the analysis of modal interaction we start with a language with indexed modalities $\mathcal{L}_m$:

Definition 2.1 Let a set PROP of basic propositions and a set IND of indices be given (PROP $\cap$ IND $= \emptyset$). We use $p, q, r, \ldots$ for elements of PROP and $i, j, k, \ldots$ for elements of IND. From these primitives formulas are built up as follows:
$\phi \ ::= \ \bot \mid p \mid \phi{\wedge}\phi \mid \neg\phi \mid \Diamond^i_j\phi$ $\Box$

The indices can then be used to describe the patterns of interaction we presented above. By means of the indices we let a modal expression select one of the available contexts of interpretation *and* produce a new context for further discourse. This way modalities are similar at the same time to anaphors—they select an antecedent context—and to antecedents—they set up a new context that can be selected by subsequent modalities. In the notation subscripts describe the selection of old antecedents, superscripts describe the production of new antecedents. We only have $\Diamond$-modalities in the representation language. They will correspond, roughly, to *might* in the examples. Clearly, indexed boxes $\Box^i_j$ could easily be added to analyse the behavior of *would*.[8] In our representation of (1) we sneek in such an operator. Then we get for examples (1) and (2):

(1) $\Diamond^i_j p \ \wedge \ \Box^k_i q \ \wedge \ \Diamond^l_k r$ (2) $\Diamond^i_j p \ \wedge \ \Diamond^k_i q \ \wedge \ \Diamond^l_i r$

The subscripts give the required variation in subordination behavior: different subscripts select different contexts for interpretation. The superscripts indicate that new points of selection have become available.
We present an update semantics for the modal language. Each formula $\phi \in \mathcal{L}_m$ gives rise to an operation $[\![\phi]\!]$ on information states. Information states are pairs $\langle I, V\rangle$, where $I \subseteq W$ and $V : \{\text{PROP} \cup \text{IND}\} \to \text{POW}(W)$. I tells us which of the possible worlds in W we still regard as possibilities. $V(p)$ ($p \in$ PROP) is the usual interpretation of an atomic formula. In addition V tells us, for each index $i \in$ IND, which set of possibilties i stands for. Hence, i refers to alternative state of mind, where we think that only the possibilities in $V(i)$ are still left open.

[8]But note that $\Box^i_j$ and $\Diamond^i_j$ are not simply dual in the semantics!

Definition 2.2 [Update Semantics]

$$
\begin{array}{lcl}
\langle I, V\rangle[\![p]\!] & = & \langle I \cap V(p), V\rangle \\
\langle I, V\rangle[\![\bot]\!] & = & \langle \emptyset, V\rangle \\
\langle I, V\rangle[\![\phi \wedge \psi]\!] & = & (\langle I, V\rangle[\![\phi]\!])[\![\psi]\!] \\
\langle I, V\rangle[\![\neg\phi]\!] & = & \langle I \backslash ((\langle I, V\rangle[\![\phi]\!])_0, V\rangle \\
\langle I, V\rangle[\![\diamond^i_j \phi]\!] & = & \left\{ \begin{array}{l}
\langle I, V'\rangle \text{ in case } (\langle V(j), V\rangle[\![\phi]\!])_0 \;\neq\; \emptyset \\
\quad \text{where} \left\{ \begin{array}{ll} V'(p) = V(p) & (p \in \textsc{prop}) \\ V'(i) = (\langle V(j), V\rangle[\![\phi]\!])_0 & \\ V'(k) = V(k) & (k \neq i) \end{array} \right. \\
\langle \emptyset, V'\rangle \text{ else} \\
\quad \text{where} \left\{ \begin{array}{ll} V'(p) = V(p) & (p \in \textsc{prop}) \\ V'(i) = \emptyset & \\ V'(k) = V(k) & (k \neq i) \end{array} \right.
\end{array} \right.
\end{array}
$$

❑

Of course one can vary the details of the formalism to taste. For example, in the current set up we have no way to make a modality pick up the current state of mind, I. But it is easy to add a special index for this purpose. This would enable us, for example, to use the current state as a default context for the interpretation of modalities. Also note that we do *not* demand that $V(i) \subseteq I$. This option was followed in [Ver99], but it seems to block any chance of representing counterfactual information.

We can now check that the representation of example (2) does indeed have the required effect. Let's start in *the global maximum*, i.e. we start with $I = W$ and $V(i) = W$ for all indices i. The result is $\langle I', V'\rangle$, where $I' = W$ provided:

(a) $V(p) \neq \emptyset$: test for *a lion might come in*

(b) $V(p) \cap V(q) \neq \emptyset$: test for *if a lion comes in, it might eat you first*, and

(c) $V(p) \cap V(r) \neq \emptyset$: test for *if a lion comes in, it might eat me first*

In all other cases $I' = \emptyset$. This describes the effect of (2) as far as the truth conditions are considered. At the same time (2) has the effect of introducing several classes of situations for further reference: new values for the indices i, k, l:

$V'(i) = W \cap V(p) = V(p)$, the coming-in situations

$V'(k) = V(p) \cap V(q)$, the coming-in-and-eating-you-first situations

$V'(l) = V(p) \cap V(r)$ the coming-in-and-eating-me-first situations

In [Ver99] we have shown how the update semantics allows for a *standard translation* to capture its success conditions: we can map each formula ϕ of the indexed modal language to a formula $\tau(\phi)$ of second order modal logic in such a way that $\tau(\phi)$ is true exactly if updating with ϕ does not lead to contradictions. In [Ver99] we obtained a decidability result from this translation: the translations for the update language are all part of a decidable fragment of second order modal logic.

3 Monoidal semantics

Above we have used an update semantics to account for the interaction of modalities in discourse. Next we will be working out an algebraic semantics, as in [VV96], and compare the two styles of semantics. Before we turn to the technical details we repeat some of the motivation and explanation for monoidal semantics.

Why? Montague's convincing arguments for the need of a uniform and systematic method in natural language semantics established a tradition in which a bottom-up formulation of the semantics has become the standard. It is clear, however, that working bottom-up is not always good enough. In order to really do justice to the way natural language interpretation proceeds *in time*, the meaning specifications have to satisfy some additional requirements. Here *dynamic semantics* comes in: it investigates interpretation as a *process* and shows how often in natural language interpretation details about the procedure matter. For example, looking at long expressions—such as discourses or texts—makes it clear that also the left-to-right order should be respected: what comes first in the text we read should be interpreted first in the formal semantics. Also the interpretation of anaphora is a point in case. We have argued that in fact *associativity* is a sensible extra constraint on a semantics. This results in the program of *monoidal semantics* in [VV96].[9]

What? Recall that a monoid is an associative structure with a unit element, i.e. it is of the form

$$\mathcal{M} = \langle M, \bullet, \mathsf{e} \rangle,$$

$$\text{where: } m_1 \bullet (m_2 \bullet m_3) \;=\; (m_1 \bullet m_2) \bullet m_3$$

$$\text{and: } \mathsf{e} \bullet m \;=\; m \;=\; m \bullet \mathsf{e}.$$

In the monoidal approach to dynamics the semantic universe is such a monoid. This monoid is built up from simple monoids by constructions that automatically preserve the monoidal character of the universe. Each simple monoid corresponds to one kind of information. The combination of these kinds of information by the constructions allows us to account for the interaction between different kinds of information relevant for natural language interpretation.
In the end we obtain a (rather) complex monoid of *information states*, with a (rather) complex associative operation. However, the fact that we have constructed this complex monoid by the systematic combination of simple monoids, ensures that we have a rational reconstruction of all the complexity that we end up with. Hence none of the complexities in the semantics are *ad hoc*.
The monoids that we start from are about the performance of simple memory management tasks: opening a new file, storing basic information units in files,

[9]There you can also find more detailed arguments. Earlier arguments for associativity are already in [Ver94], for example. Note that the monoidal program is not at all in conflict with the *compositionality* requirement: it is just a non-standard way of meeting this requirement.

closing a file, etc. This way we prevent magical update operations from creeping in.

Information states The elements of the semantic universe will be called information states or m-STATES . The information states in the monoid play two roles. First, they serve as meaning objects, i.e. they are the things that expressions denote. Secondly, they stand for states of mind: the state of mind of the hearer who is interpreting the text.
Information states are built up with the GROT-construction. In the appendices A and B we describe in some detail how this works.

The general appearance of an m-state is $\langle X, F\rangle$. Here X is a set of finite stacking cells: one for each variable name in VAR. VAR contains all the indices in IND, an identifier *val* that stands for the current state of information, and also some auxiliary identifiers in which we can temporarily store information. In all we get: $\text{VAR} = \text{IND} \cup \{val\} \cup \{l, n_1, n_2, m\}$. From X we obtain a set of *referents* or *files* R_X^+. R_X^+ contains for each $x \in \text{VAR}$ the set of memory locations that are in some way connected to the identifier x. One location will contain the current value of x, but there also may be some locations around, for previous values of x for example.[10]
F is a set of mappings from all the referents of R_X to sets of possible worlds: $F \subseteq [R_X^+ \to \text{POW}(W)]$. It represents the truth conditional information that we have about all the referents.
A few finite stacking cells have a special name:[11] id, id^+, pop^+, push^+, new. When such a finite stacking cell σ is assigned to a syntactic variable x, this is indicated by writing x in the subscript: σ_x. Hence push_x^+ indicates that the stacking cell push^+ has been assigned to x. This way we get:

- ▷ id_x: when an identifier x is not really being used at the moment. Hence id_x does not contribute to R_X^+.
- ▷ id_x^+: assigned to a 'neutral' occurrence of x; contributes just one referent to R_X^+, called x_{st1}
- ▷ pop_x^+: used for an 'old' occurrence of x; contributes just one referent, called x_{po1}.
- ▷ push_x^+: used when a 'new' occurrence of x is introduced; contributes one referent, called x_{pu1}.
- ▷ $\mathsf{new}_x = \mathsf{pop}_x^+ \bullet \mathsf{push}_x^+$: used when an old way of using x is exchanged for a new use of x. Hence two referents are involved: x_{po1} and x_{pu1}.

We extend the notation convention by writing $\sigma_{x_1} \bullet \ldots \bullet \sigma_{x_n}$ for the situation where all variables other then the x_i have the trivial stacking cell id as a value.

[10]Here appendix A gives the details.
[11]Again see appendix A for more extensive discussion.

This method of mentioning only the non-trivial information is extended to the level of m-STATES . So, for example, push^+_x will also be used for the m-state that: (i) assigns push^+ to x; (ii) assigns id to all other variables; (iii) does not constrain the value of the only referent in this situation.
In addition we will use the following shorthand for m-STATES : $x = \mathsf{t}$, where t is a term. The basic idea of this notation is that:

> $x = \mathsf{t}$ stands for $\langle Y, F\rangle$
> where Y assigns id^+_y to all variables occuring in the equation
> and F constrains the value of x_{st1} as indicated by t.

We will encounter the following instances of t:

- ▷ y for a variable y
- ▷ c for a set of possible worlds $c \in \textsc{pow}(W)$
- ▷ $\mathsf{t}_1 \backslash \mathsf{t}_2$ for terms t_1, t_2
- ▷ $\left\{ \begin{array}{l} \mathsf{t}_1 \Leftarrow (y = \emptyset) \\ \mathsf{t}_2 \Leftarrow (y \neq \emptyset) \end{array} \right.$ for a variable y and terms t_1, t_2

The disjunctive construction (similar to a 'discriminator term' in algebraic logic) in the last clause only occurs as main-connective. So, in particular, it cannot occur inside any of the t_i above.
For this specific (inductively defined) set of terms we can be more specific about the notation $x = \mathsf{t}$. Given f we can define $\overline{\mathsf{t}}$ for terms t as introduced in the first three clauses above:

- ▷ $\overline{y} := f(y_{st1})$
- ▷ $\overline{c} := c$
- ▷ $\overline{\mathsf{t}_1 \backslash \mathsf{t}_2} := \overline{\mathsf{t}_1} \backslash \overline{\mathsf{t}_2}$

Let's write $\mathsf{id}^+_{\mathsf{t}}$ for $\mathsf{id}^+_{x_1} \bullet \ldots \bullet \mathsf{id}^+_{x_n}$, in case $\{x_1, \ldots, x_n\}$ is the set of variables occurring in t. (Note that id_{t} is insensitive to the order in which we merge the $\mathsf{id}^+_{x_i}$.) Then:

- ▷ $x = y$ stands for $\langle \mathsf{id}^+_x \bullet \mathsf{id}^+_y, \{f : \overline{x} = \overline{y}\}\rangle$
- ▷ $x = c$ stands for $\langle \mathsf{id}^+_x, \{f : \overline{x} = \overline{c}\}\rangle$
- ▷ $x = \mathsf{t}_1 \backslash \mathsf{t}_2$ stands for $\langle \mathsf{id}^+_x \bullet \mathsf{id}^+_{t_1} \bullet \mathsf{id}^+_{t_2}, \{f : \overline{\mathsf{t}} = \overline{\mathsf{t}_1 \backslash \mathsf{t}_2}\}\rangle$
- ▷ $x = \left\{ \begin{array}{l} \mathsf{t}_1 \Leftarrow (y = \emptyset) \\ \mathsf{t}_2 \Leftarrow (y \neq \emptyset) \end{array} \right.$ stands for
  $\langle \mathsf{id}^+_x \bullet \mathsf{id}^+_y \bullet \mathsf{id}^+_{t_1} \bullet \mathsf{id}^+_{t_2}, \{f : (\overline{x} = \overline{\mathsf{t}_1} \ \& \ \overline{y} = \emptyset) \text{ or } (\overline{x} = \overline{\mathsf{t}_2} \ \& \ \overline{y} \neq \emptyset)\}\rangle$

It helps to do some basic calculations with basic m-STATES . For example, one can check that:[12]

$$\langle X, F\rangle \bullet \mathsf{new}_x \;=\; \langle X \;\bullet\; \mathsf{new}_x, \{g :\; \exists f \in F :\;\; f[x_{pu1}]g\}\rangle$$

allowing us to reset the value of x to an arbitrary new value. Also:

$$\langle X, F\rangle \;\bullet\; \mathsf{new}_x \;\bullet\; x = \mathsf{t} \;\text{ equals}$$
$$\langle X \;\bullet\; \mathsf{new}_x, \{g :\; \exists f \in F \;\; (f[x_{pu1}]g \;\&\; g(x_{pu1}) = \bar{\mathsf{t}})\}\rangle.$$

Similar result apply with push i.s.o. new. For the sets of assignments in these examples we write $F[x :=?]$ and $F[x := \mathsf{t}]$ respectively. So we get:

$$\begin{aligned}
\langle X, F\rangle \bullet \mathsf{new}_x &= \langle X \;\bullet\; \mathsf{new}_x, F[x :=?]\rangle\\
\langle X, F\rangle \bullet \mathsf{new}_x &= \langle X \;\bullet\; \mathsf{new}_x, F[x := \mathsf{t}]\rangle\\
\langle X, F\rangle \bullet \mathsf{push}_x &= \langle X \;\bullet\; \mathsf{push}_x, F[x :=?]\rangle\\
\langle X, F\rangle \bullet \mathsf{push}_x &= \langle X \;\bullet\; \mathsf{push}_x, F[x := \mathsf{t}]\rangle
\end{aligned}$$

4 Giving the monoidal semantics

After this lengthy introduction of the m-STATES and the notation that we will use for them, it is now time to get to business. For the monoidal interpretation we use a representation language that consists of strings. The string language we use is:

$$\mathcal{L}_a \;::=\; (\{\bot, \triangleleft_\neg, \triangleright_\neg\} \cup \{\triangleleft^i_j :\; i, j \in \textsc{ind}\} \cup \{\triangleright^i_j :\; i, j \in \textsc{ind}\} \cup \{\oplus_i, \ominus_i :\; i \in \textsc{ind}\} \cup \{p :\; p \in \textsc{prop}\})^*$$

Each of the symbols in the alphabet of $\mathcal{L}_a$ gets an m-state as its interpretation. The interpretation of longer strings is then obtained using the (associative!) operation in m-STATES . The basic interpretations are:

$$\begin{aligned}
[\![p]\!] &:= \mathsf{push}^+_l \;\bullet\; (l = (V(p) \cap \mathit{val})) \;\bullet\; \mathsf{new}_{\mathit{val}} \;\bullet\; (\mathit{val} = l) \;\bullet\; \mathsf{pop}_l\\
[\![\bot]\!] &:= \mathsf{new}_{\mathit{val}} \;\bullet\; (\mathit{val} = \emptyset)\\
[\![\triangleleft_\neg]\!] &:= \mathsf{push}^+_{n_1} \;\bullet\; (n_1 = \mathit{val})\\
[\![\triangleright_\neg]\!] &:= \mathsf{push}^+_{n_2} \;\bullet\; (\mathit{val} = n_2) \;\bullet\; \mathsf{new}_{\mathit{val}} \;\bullet\; (\mathit{val} = (n_1 \backslash n_2)) \;\bullet\; \mathsf{pop}_{n_2} \;\bullet\; \mathsf{pop}_{n_1}\\
[\![\oplus_i]\!] &:= \mathsf{push}^+_l \;\bullet\; (l = i) \;\bullet\; \mathsf{push}^+_i \;\bullet\; (i = l) \;\bullet\; \mathsf{pop}_l\\
[\![\ominus_i]\!] &:= \mathsf{pop}_i\\
[\![\triangleleft^i_j]\!] &:= \mathsf{push}^+_m \;\bullet\; (m = \mathit{val}) \;\bullet\; \mathsf{new}_{\mathit{val}} \;\bullet\; (\mathit{val} = j)\\
[\![\triangleright^i_j]\!] &:= \mathsf{new}_i \;\bullet\; (i = \mathit{val}) \;\bullet\; \mathsf{new}_{\mathit{val}} \;\bullet\; (\mathit{val} = \left\{ \begin{array}{l} \emptyset \Leftarrow (i = \emptyset) \\ m \Leftarrow (i \neq \emptyset) \end{array} \right. \;) \;\bullet\; \mathsf{pop}_m
\end{aligned}$$

The operations $[\![p]\!]$ and $[\![\bot]\!]$ are self-explanatory: they set the appropriate new value for val. The pair of triangles $[\![\triangleleft_\neg]\!]$ and $[\![\triangleright_\neg]\!]$ is to be compared with the negation operation in the update semantics. For example, in $\triangleleft_\neg \cdot p \cdot \triangleright_\neg$, we see

[12]The notation $f[r]g$ is used to indicate that f and g differ at most on r. To make this notation really precise one can go into the details of the appendix A, making all the embeddings of the R-sets explicit.

how using a few auxialiary variables n_i allows us to substract the p-worlds from the current value for val: first we set the auxiliary variable n_1 equal to val; then we compute $val \cap V(p)$; this is stored in n_2; then we set assign a new value to $val := n_1 \backslash n_2$.
The triangles $\triangleleft$ and $\triangleright$ do a similar job in replacing $\diamond$ in the update setting. It's easy to check that:

$$\triangleleft^i_j \cdot p \cdot \triangleright^i_j \cdot \triangleleft^k_i \cdot q \cdot \triangleright^k_i \cdot \triangleleft^l_i \cdot r \cdot \triangleright^l_i$$

does what we want for example (2). This leaves the mysterious elements $\oplus_i$ and $\ominus_i$. We will explain what they are for below, when we discuss the systematic translation of $\mathcal{L}_m$ into $\mathcal{L}_a$.

5 From update actions to action algebras

We compare the update semantics for $\mathcal{L}_m$ with the monoidal semantics for $\mathcal{L}_a$ by giving a translation of $\mathcal{L}_m$ into $\mathcal{L}_a$. For this translation it is convenient to first write the formulas of $\mathcal{L}_m$ in normal form. This can be done as follows:

ϕ	$\mathsf{nf}(\phi)$
p	p
$\bot$	$\bot$
$(\phi \wedge \psi)$	$\phi_1 \wedge \mathsf{nf}(\phi_2 \wedge \psi)$
	in case $\mathsf{nf}(\phi)$ is of the form $(\phi_1 \wedge \phi_2)$
$(\phi \wedge \psi)$	$\mathsf{nf}(\phi) \wedge \mathsf{nf}(\psi)$
	else
$\neg\phi$	$\neg\mathsf{nf}(\phi)$
$\diamond^i_j\phi$	$\diamond^i_j\mathsf{nf}(\phi)$

In the normal forms we regroup brackets: we systematically rewrite $((\phi\wedge\psi)\wedge\chi)$ to $(\phi \wedge (\psi \wedge \chi))$. We then translate these normal forms into the string language $\mathcal{L}_a$ as follows:

ϕ	ϕ^a	ϕ^b
p	p	p
$\bot$	$\bot$	$\bot$
$(\phi \wedge \psi)$	$\phi^a \cdot \psi^a$	$\oplus_i \cdot \phi^a \cdot \psi^b \cdot \ominus_i$
		in case ϕ is of the form $\diamond^i_j\chi$
$(\phi \wedge \psi)$	$\phi^a \cdot \psi^a$	$\phi^b \cdot \psi^b$
		in case ϕ is not of the form $\diamond^i_j\chi$
$\neg\phi$	$\triangleleft_\neg \cdot \phi^b \cdot \triangleright_\neg$	$\triangleleft_\neg \cdot \phi^b \cdot \triangleright_\neg$
$\diamond^i_j\phi$	$\triangleleft^i_j \cdot \phi^b \cdot \triangleright^i_j$	$\oplus_i \cdot (\diamond^i_j\phi)^a \cdot \ominus_i$

$\phi^a \in \mathcal{L}_a$ is the 'algebraic' translation of $\phi \in \mathcal{L}_m$. $\phi^b \in \mathcal{L}_a$ is an auxiliary construct. In the translation from $\mathcal{L}_m$ to $\mathcal{L}_a$ we lose the brackets. This will not cause any semantic ambiguities, as the semantics of the string-language $\mathcal{L}_a$ is monoidal, and hence the required associativity conditions are provided for.

A more confusing feature of the translation is the use of the normal forms and the auxiliary mapping ϕ^b. Let's try to see what this is all about.
First recall that in the update semantics $\neg$ and $\diamond$ have very limited externally dynamic effects. For example, $\diamond^i_j p$ will assign a new value to the index i, but $\neg\diamond^i_j p$ will *not* give a new value to i. Therefore we introduce an auxiliary translation ϕ^b which has no dynamic side-effects. In the definition of ϕ^a we can then use this auxiliary translation whenever we want to limit the dynamic effects. But we will want to limit only the 'external' dynamics!
When we introduce these extra $\oplus$- and $\ominus$-actions we have to be careful not to disturb the internal dynamics as well. For example, in $\neg(\diamond^i_j p \wedge \diamond^k_i q)$ there is some internal dynamics: *within* the negation the new value for i is passed on from the first conjunct to the second conjunct. Therefore we should postpone the $\ominus$-action until after the second conjunct. This is where the normal forms come to rescue: they allow us to systematically postpone the $\ominus$-action. As an example, consider $\neg((\diamond^i_j p \wedge \diamond^k_i q) \wedge \diamond^m_i r)$. We want to ensure that the new value of i is passed on from the first conjunct to the second conjunct and also to the third. But we do not want the new value of i to travel beyond the negation. Therefore we push_i the value of i, then perform the computation and finally pop_i the extra copy. We know that we have to push_i as soon as we see i appear as a superscript in $\diamond^i_j$ in the subformula $(\diamond^i_j p \wedge \diamond^k_i q)$. But we should postpone the pop_i-action until after the *next* subformula, i.e. $\diamond^m_i r$. How do we achieve such systematic postponements?
There are several options[13] for achieving the postponement. Above we do this by first rebracketing: this is what the normal forms do for us. By rebracketing all conjuncts $(\phi \wedge \psi) \wedge \chi$ systematically to $\phi \wedge (\psi \wedge \chi)$, we achieve a situation where the proper location for the pop-action is easy to indicate: always pop after the next conjunct.[14]
This completes the intuitive explanation of the translation. Now we can check that it really works.

6 The result

Above we gave a translation from $\mathcal{L}_m$ into $\mathcal{L}_a$. To check that this translation really works we first have to embed the information states of the update semantics into the universe of m-STATES . This works as follows:

$$\langle I, V\rangle^a = \langle X, F\rangle$$

where X assigns a value to all indices and auxiliary variables: we get push^+_x for all $x \in \text{IND}$ and also push^+_{val}. To the other, auxiliary variables X assigns id. F contains just one mapping, f. It assigns I to the referent induced by val and

[13]For example, we could make the definition work on pairs consisting of a formula and a multiset of variables: the multiset of variables still to be popped. This would enable us to give a compositional translation, but the translation would be defined on the enriched objects.

[14]It may be helpful to the well informed reader to bring the 'translation' from DPL into first order logic to mind, as discussed in [GS91]. There a similar move was made to handle the dynamics of DPL.

$V(i)$ to the referent induced by i ($i \in \textsc{ind}$). (Note that the auxiliary variables do not generate referents.)
Now we claim that for all $\langle I, V\rangle$ and all $\phi \in \mathcal{L}_m$:

Proposition 6.1

(a) $\langle I, V\rangle[\![\phi]\!]^a \;=\; \langle I, V\rangle^a \;\bullet\; [\![\phi^a]\!]$ (modulo garbage)

(b) $\langle(\langle I, V\rangle[\![\phi]\!])_0, V\rangle^a \;=\; \langle I, V\rangle^a \;\bullet\; [\![\phi^b]\!]$ (modulo garbage)

$\square$

The proposition says that the update function $[\![\phi]\!]$ has the same effect as merging with the information state $[\![\phi^a]\!]$. Hence the two styles of semantics agree. The only difference is in the *garbage*. This means that the only difference is in the way the monoidal semantics tends to remember details about old values for the identifiers that we do not bother to keep track of in the update semantics. In appendix A there are more details about what garbage is exactly. The proof of proposition 6.1 is given in appendix C. It is a simultaneous induction on the complexity of $\phi \in \mathcal{L}_m$.
Recall that we set out to develop the monoidal semantics for $\mathcal{L}_a$ with coverage of the data equal to the update semantics of $\mathcal{L}_m$. The proposition shows that we have succeeded. The additional advantage of the monoidal semantics is that it gives a precise reconstruction of how all the updates are constructed from basic actions on memory locations.

7 Conclusion

We have discussed the interaction between modal expressions in discourse. Several examples were presented to illustrate the phenomena involved. We saw that also quantifiers in a discourse display similar patterns of interaction.
For the formal analysis of this type of interaction we have considered two logical languages: first we considered $\mathcal{L}_m$ in which the interaction is represented by indexed modalities. The update semantics for $\mathcal{L}_m$ was presented and we have seen how the basic interaction patterns could be represented.
As an alternative we have considered $\mathcal{L}_a$, a string language for which a monoidal semantics can be given along the lines of [VV96]. We have shown that this language can also deal with the phenomena. This was done in two ways: first by the presentation of examples. But more convincingly we gave a translation of $\mathcal{L}_m$ into $\mathcal{L}_a$ to show that $\mathcal{L}_a$ is suddicently expressive.
The advantage of $\mathcal{L}_a$ over $\mathcal{L}_m$ is that it forces/allows us to take the metaphors of dynamic semantics seriously: we break down the interpretation into elementary operations on locations in memory. From these basic operations we generate the complex procedures in a systematic and uniform way. This provides us with a very precise account of what goes on, dynamically speaking. It also guarantees that there is no 'funny business' in the semantics: all can be reduced to elementary operations on files. Hence we provide a rational reconstruction of all the complex update procedures required in the semantics of $\mathcal{L}_m$ which prevents wild and unrealistic procedures from creeping in.

An additional advantage of the systematics of the monoidal approach is that it will facilitate the combination of our account of modal interaction with other results, for example the similar accounts for the interaction of quantifiers and the semantics of E-type ananphors that are now available.
Of course, a lot of further work awaits. For the interaction phenomena in discourse that we have seen in the examples in section 1, the next tasks seem to be: (i) a taxonomy of the interaction patterns that actually occur in discourse; (ii) a good account of the way these patterns of interaction depend on discourse structure. We hope to attend to these matters in the near future.

A Finite stacking cells

We consider some algebras for memory managment, leading up to the category of finite stacking cells. Throughout we will be looking at the memory locations (files, referents, variables, indices) only. We will consider at a later stage how the locations get linked up with *names* and with *values*.
Notation: we use postfix notation for operations in the algebras, prefix notation for the R-operations and the embeddings between the algebras.

A.1 Stacks

A basic tool for the description of the way we think about memory mangement it the well known data type of *stacks*. This data types consists of a stack of files on which we can perform two operations: pushing a new value on top of the stack and popping a file from the top of the stack. An alternative name for stacks is LIFO, short for *last-in-first-out*.
As, for now, we are only concerned with the locations/files proper and not with their names or their content, stacks simply are finite sequences over an arbitrary one element set. Or: stacks are really just natural numbers with operations for successor and predecessor.
All this is perfectly true, but we have to keep in the back of our minds that we are interested in files that will obtain names and in which we will want to store things. Then it will be important to keep track of which file ends up where and how. The functor R reminds us of this deeper motivation and keeps track of what goes on below the surface. In particular it gives us, for all the operations involved, the proper embeddings of the files underlying the objects.

Definition A.1 [STACKS]

$$\mathcal{S} \;=\; \langle \{1\}^*, \mathsf{push}, \mathsf{pop}, \langle\rangle \rangle$$
where: $(1^n)\mathsf{push} \;=\; 1^{n+1}$ and $(1^{n+1})\mathsf{pop} \;=\; 1^n$.

We specify $R_{\mathcal{S}}$, immediately applying the convention of omitting superfluous subscripts:

$R:$	$\mathcal{S}$	$\longrightarrow$	FINSET
	$\langle\rangle$	$\longmapsto$	$\emptyset$
	1^n	$\longmapsto$	$\{l_1, \ldots, l_n\}$
$R(\mathsf{push})$:	$R(1^n)$	$\longrightarrow$	$R(1^{n+1})$
	l_i	$\longmapsto$	l_{i+1}
$R(\mathsf{pop})$:	$R(1^{n+1})$	$\longrightarrow$	$R(1^n)$
	l_{i+1}	$\longmapsto$	l_i

$\Box$

Note that pop is a partial operation and, as a consequence, so is $R(\mathsf{pop})$. Also note that $\mathcal{S}$ is (isomorphic to) natural numbers with successor and predecessor.

A.2 Stacking cells

Basic intuition about stacking cells: they are to stacks what integers are to natural numbers. Integers: (i) help us get rid of the partiality of substraction operation; (ii) allow us to represent *actions* as *elements* of the algebra. (i) should be clear. For (ii): consider how the integer $-n$ corresponds to the operation $x \mapsto x - n$. Similarly $+n$ corresponds to the operation $x \mapsto x + n$.
Similarly stacking cells unrich the universe of stacks with auxiliary elements so that: (i) pop becomes a total operation; (ii) the push- and pop-actions correspond to special stacking cells.

Definition A.2 [STACKING CELLS]

$\mathcal{SC} = \langle IN \times IN \times IN,\ \bullet, \mathsf{id}\rangle$

where: $\mathsf{id} = \langle 0,0,0\rangle$;

$\langle n_1, m_1, X_1\rangle \bullet \langle n_2, m_2, X_2\rangle = \langle n_1 + (n_2 \dot{-} m_1), m_2 + (m_1 \dot{-} n_2),\ X_1 + X_2 + \mathsf{min}(m_1, n_2)\rangle$

We specify $R_{\mathcal{SC}}$:

R:	$\langle n, m, X\rangle$	$\longmapsto$	$\{\mathrm{po}_1, \ldots, \mathrm{po}_n\} \cup \{\mathrm{pu}_1, \ldots, \mathrm{pu}_m\}$ $\cup\{g_1, \ldots, g_X\} \cup \{\mathrm{st}_i\ :\ i > n\}$
emb_1:	$R(\langle n_1, m_1, X_1\rangle)$	$\longrightarrow$	$R(\langle n_1, m_1, X_1\rangle \bullet \langle n_2, m_2, X_2\rangle)$
	po_i	$\longmapsto$	po_i
$(i > n_2)$	pu_i	$\longmapsto$	pu_{i+m_2}
$(i \leq n_2)$	pu_i	$\longmapsto$	$g_{(X_1+X_2+i)}$
$(m_1 < n_1 + i \leq n_2)$	st_{n_1+i}	$\longmapsto$	po_{n_1+i}
else	st_{n_1+i}	$\longmapsto$	st_{n_1+i}
	g_i	$\longmapsto$	g_i
emb_2:	$R(\langle n_2, m_2, X_2\rangle)$	$\longrightarrow$	$R(\langle n_1, m_1, X_1\rangle \bullet \langle n_2, m_2, X_2\rangle)$
	pu_i	$\longmapsto$	pu_i
$(i > m_1)$	po_i	$\longmapsto$	po_{i+n_1}
$(i \leq m_1)$	po_i	$\longmapsto$	$g_{(X_1+X_2+i)}$
$(n_2 < n_2 + i \leq m_1)$	st_{n_2+i}	$\longmapsto$	pu_{m_2+i}
else	st_{n_2+i}	$\longmapsto$	st_{n_2+i}
	g_i	$\longmapsto$	$g_{(X_1+i)}$

$\Box$

It can be checked now that emb_1 and emb_2 are jointly surjective, i.e.,

$$\mathsf{emb}_1(R(\langle n_1, m_1, X_1\rangle)) \cup \mathsf{emb}_2(R(\langle n_2, m_2, X_2\rangle)) = \\ R(\langle n_1, m_1, X_1\rangle \bullet \langle n_2, m_2, X_2\rangle).$$

In reading the definitions one can roughly think of a stacking cell as two stacks: a push-stack and a pop-stack. As before, we count the files on the stack from top-to-bottom: 1 is the top element of the stack.
Notice that in the embeddings a pu-file of the first stacking cell and a po-file of the second stacking cell one would expect the file to disappear. Instead we let them produce a garbage level. Garbage levels are included in the general picture of [VV96] for several reasons. Here we can think of them as a way of making really sure that we are not confronted with an irretrievable loss of information: when something appears to have gone missing, we can always check the garbage.
If there are more pu-levels in the first stacking cell than po-levels in the second stacking cell, then we get extra pu-levels in the merger of the two cells. Conversely, if we have an excessive number of po-levels in the second stacking cell, this gives rise to extra po-levels in the merger.
The st-files can be thought of as auxiliary files: they are a bit like a notebook where we make temporary notes. They can also be thought of as files with an as-yet-indeterminate status: we don't know yet whether they are pu-files or po-files. In [VV96] we allow for an unlimited amount of st-files. Here we will need at most one (at a time).

A.3 From actions to algebras

We can now define the embedding of $\mathcal{S}$ into $\mathcal{SC}$. This will show how we convert the operations of $\mathcal{S}$ into elements of $\mathcal{SC}$. Again the fishy part is one the level of the R-sets.

Definition A.3 We specify the embedding that sends stacks to stacking cells.

emb:	$\mathcal{S}$	$\longrightarrow$	$\mathcal{SC}$
	0	$\longmapsto$	$\langle 0,0,0\rangle$
	n	$\longmapsto$	$\langle 0,n,0\rangle$
	push	$\longmapsto$	$\langle 0,1,0\rangle$
	pop	$\longmapsto$	$\langle 1,0,0\rangle$
emb_R:	$R_\mathcal{S}(n)$	$\longrightarrow$	$R_\mathcal{SC}(\mathsf{emb}(n))$
	l_i	$\longmapsto$	pu_i

❏

By now there is a whole cube of commutative diagrams to be checked. We list the crucial claims for the push-operation. Similar claims can be checked for pop.

Proposition A.4

- ▷ Within $\mathcal{S}$:

  $\mathsf{push} \circ R_{\mathcal{S}} \;=\; R_{\mathcal{S}} \circ R_{\mathcal{S}}(\mathsf{push})$

- ▷ Within $\mathcal{SC}$:

  $(\_ \bullet \mathsf{emb}(\mathsf{push})) \circ R_{\mathcal{SC}} \;=\; R_{\mathcal{SC}} \circ \mathsf{emb}_1$

- ▷ From $\mathcal{S}$ to $\mathcal{SC}$:

  $\mathsf{emb} \circ R_{\mathcal{SC}} \;=\; R_{\mathcal{S}} \circ \mathsf{emb}_R$ ▯

Combining the claims above we get lots of commutativity facts of the definitions above. An unmistakable indication that there is a bit of category theory going on in the background. Check [VV96] for details.

A.4 Finite stacking cells

In the previous section we saw stacking cells in their full generality. They contain a lot of storage facilities to make sure that we can play lots of tricks with them in our general framework. Most of the time, however, we do not want to play all those tricks. In particular, we do not generally require an infinite amount of (auxiliary) files. We typically work in a restricted version of the category of stacking cells where only finitely many files are really used. Here we model this by splitting the set $R(\langle n, m, X\rangle)$ into a (finite) set $R^+(\langle n, m, X\rangle)$ and a set $R^-(\langle n, m, X\rangle)$. The R^+-set contains the files that are actively used. We will also say: the files that are *present*. In [VV96] it is shown that the distinction of these sets is a simple instance of the Grothendieck construction. Here we specify the necessary details directly.

Definition A.5 Let $R_1^+ = R^+(\langle n_1, m_1, X_1\rangle)$ and $R_2^+ = R^+(\langle n_2, m_2, X_2\rangle)$ be given. Then:

$$R^+(\langle n_1, m_1, X_1\rangle \bullet \langle n_2, m_2, X_2\rangle) \;=\;$$
$$(\mathsf{emb}_1(R_1^+) \cup \mathsf{emb}_2(R_2^+)) \backslash \{g_i \;:\; i > (X_1 + X_2)\}$$ ▯

In all our applications here we will regard garbage files as *not* really present. The definition above makes sure that this is still so after the merger.
Finally we introduce some handy notation for special elements of the monoid of finite stacking cells. All the stacking cells we meet in the papaer are obtain by merger these simple stacking cells. Hence they can be described by sequences of the names we introduce.

push	$\langle 0,1,0\rangle$	$R^+ = \emptyset$
pop	$\langle 1,0,0\rangle$	$R^+ = \emptyset$
id	$\langle 0,0,0\rangle$	$R^+ = \emptyset$
push$^+$	$\langle 0,1,0\rangle$	$R^+ = \{\mathrm{pu}_1\}$
pop$^+$	$\langle 1,0,0\rangle$	$R^+ = \{\mathrm{po}_1\}$
id$^+$	$\langle 0,0,0\rangle$	$R^+ = \{\mathrm{st}_1\}$
garb	$\langle 0,0,1\rangle$	$R^+ = \emptyset$

B Storing information in files

In this appendix we briefly discuss the constructions from [VV96] for the storage of information in files. The general situation is that we have files $a \in \mathcal{A}$ and for each file a a monoid of potential 'values' $\mathcal{B}(a)$. It is our job to store a value from $\mathcal{B}(a)$ into file a.
The trick is to do this is such a way that we automatically obtain a new monoid in which we can easily recognise the files in $\mathcal{A}$ and the contents in the $\mathcal{B}(a)$s. The general solution that we present in [VV96] uses techniques from category theory. Below we will avoid category and provide just a rough sketch of what is really going on. We mainly dicuss just the examples crucial for this paper.

General Let a monoid $\mathcal{A}$ and a family of monoids $\mathcal{B}(a)$ $(a \in \mathcal{A})$ be given. Then we write $(\Sigma a \in \mathcal{A})\mathcal{B}(a)$ for the monoid that consists of pairs: $\langle a, b\rangle$ $(a \in \mathcal{A},\ b \in \mathcal{B}(a))$, where the new monoidal operation is defined as:

$$\langle a, b\rangle \bullet \langle a', b'\rangle \ = \ \langle a \bullet a', \mathsf{emb}_1(b) \bullet \mathsf{emb}_2(b')\rangle.$$

We see that the new $\bullet$-operation works almost component-wise. The subtlety is in the use of the emb_i functions. This subtlety arises, because $b \in \mathcal{B}(a)$ and $b' \in \mathcal{B}(a')$ do not live in the same monoid. Therefore we have to send them to embed them into a new monoid before we can combine them. Of course this monoid has to be $\mathcal{B}(a \bullet a')$. So we can only perform the construction if with the monoids $\mathcal{A}$ and $\mathcal{B}(a)$ appropriate embedding functions are given.
Category theory provides the uniquely suitable setting for including all the details about such embedding functions. We will see in the examples below what they can be in practise.

Example Consider as an example: $\textsc{files} := (\Sigma a \in \{\textsc{var}\})[\textsc{var} \to \mathcal{SC}]$
Here $\{\textsc{var}\}$ is a one element monoid. Hence the construction gives, at first sight, just fancy notation for the set of all mappings f from $\textsc{var}$ to $\mathcal{SC}$. But looking more carefully we see that we also get an associative operation on the mappings: we get $f \bullet f'(x) \ = \ f(x) \bullet f'(x)$. Hence the merger of the mappings $f \bullet f'$ assigns to each $x \in \textsc{var}$ the merger of the stacking cells $f(x)$ and $f(x')$.
Note: in this case $\mathcal{B}(a) = \mathcal{B}(a')$ for all a, a'. So, we do not see the embeddings emb_i.

Second example We sketch how m-$\textsc{states}$ come about. Consider $\textsc{files}$ from the previous example. Each $f \in \textsc{files}$ gives rise to a set of locations in memory:

$$R_{\text{FILES}}(f) \ = \ \bigoplus_{x \in \text{VAR}} R^{+}_{\mathcal{SC}}(f(x)).$$

Here $R^{+}_{\mathcal{SC}}$ is as explained in appendix A. Hence R_{FILES} gives us the locations in memory that are currently active. We want to assign a set of possible worlds to each of these locations in memory. Then each m-state will contain a set of such assignments. We get this by setting

$\text{m-STATES} = (\Sigma X \in \text{FILES})\text{ASS}(X)$,

where $\text{ASS}(X)$ is the following monoid:

the elements are sets of mappings $F \subseteq [R_{\text{FILES}}(X) \rightarrow \text{POW}(W)]$

the merger operation is intersection: $F \bullet G = F \cap G$

Now, if we combine two m-STATES $\langle X, F\rangle$ and $\langle Y, G\rangle$, we obtain $\langle X, F\rangle \bullet \langle Y, G\rangle = \langle X \bullet Y, \ \mathsf{emb}_1(F) \cap \mathsf{emb}_2(G)\rangle$. I.e., we merge the FILES-components first. Then we want to take the intersection of F and G. But this will not work: the mappings $f \in F$ have $dom(f) = R_{\text{FILES}}(X)$ and the mappings $g \in G$ have domain $dom(g) = R_{\text{FILES}}(Y)$. Therefore, we have to use embedding functions. The new, common domain of $\mathsf{emb}_1(f)$ and $\mathsf{emb}_2(g)$ will be $R_{\text{FILES}}(X \bullet Y)$. We have seen in appendix A how the embedding of the referents of $R_{\text{FILES}}(X)$ and $R_{\text{FILES}}(Y)$ into $R_{\text{FILES}}(X \bullet Y)$ can be achieved. We can then embed F as follows:

$h \in \mathsf{emb}_1(F)$ iff $\exists f \in F$ such that $f(r) = h(\mathsf{emb}_1(r))$ for all $r \in R_{\text{FILES}}(X)$.

and simlarly for G. This way we obtain sets of functions $\mathsf{emb}_1(F)$ and $\mathsf{emb}_2(G)$ with a common domain. So, we can now take $\mathsf{emb}_1(F) \cap \mathsf{emb}_2(G)$.

We have now sketched the general construction of $(\Sigma a \in \mathcal{A})\mathcal{B}(a)$ as well as the examples that are crucial for this paper. We admit that we have not been completely specific about some of the details, especially regarding the embeddings of files that are always going on in the background. But these additional details can be looked up in [VV96].

C The proof of the main result

In this appendix we prove proposition 6.1, the main result of the paper, which we repeat here for convenience.

(a) $\langle I, V\rangle[\![\phi]\!]^a = \langle I, V\rangle^a \bullet [\![\phi^a]\!]$ (modulo garbage)

(b) $\langle((\langle I, V\rangle[\![\phi]\!])_0, V\rangle^a = \langle I, V\rangle^a \bullet [\![\phi^b]\!]$ (modulo garbage)

The proof of (a) and (b) is a simultaneous induction on the complexity of $\phi \in \mathcal{L}_m$. We give some initial remarks before turning to the details of this proof. We write $=$ for equality modulo garbage. (This is relevant at all points where the new stacking cell occurs.) There is some annotation of proof: we show where the induction hypothesis is applied by a subscript *i.h.*. Throughout the associativity of $\bullet$ is crucial. (We mention this only once (in the case for $\wedge$).) We abundantly use the simple facts about information states, as discussed in section 1 above. A lot of the computation steps below also involve replacing terms by their value to make them less order dependent. We have seen an example of this above. Below we are often in a situation where the input state $\langle I, V\rangle$ provides a value for a variable: in $\langle I, V\rangle^a \bullet l = i$ the value of l is determined by the

input $\langle I, V\rangle$, which causes order sensitivity. But as soon as we replace i by its value $V(i)$, we can safely permute the two conjuncts: $\langle I, V\rangle^a \bullet l = i$ is equal to $\langle I, V\rangle^a \bullet l = V(i)$, which is equal to $l = V(i) \bullet \langle I, V\rangle^a$.
By making such switches we then sometimes end up in a situation where a push action is simply followed by a pop action. Then we can use $\mathsf{push} \bullet \mathsf{pop} = \mathsf{id}$ (modulo garbage!).

Proof: We do not present all cases: mostly the b-cases are identical to the a-cases, so we can leave them out. Exceptions are $((\Diamond^i_j\phi) \wedge \psi)^b$ and $(\Diamond\phi)^b$. We present the first case. The second case is then again very similar to the first case (with ψ the empty formula, as it were), so we can be brief about it.

- ▷ ∘ $p^a = p$

  ∘ $[\![p^a]\!]$
  $(= \mathsf{push}_l \bullet l = val \bullet \mathsf{new}_{val} \bullet val = l \cap V(p) \bullet \mathsf{pop}_l) \quad =$
  $\langle \mathsf{new}_{val},\ \{f : f(val_{pu1}) = f(val_{po1}) \cap V(p)\}\rangle$

  ∘ $\langle I, V\rangle^a \bullet [\![p^a]\!] \quad =$
  $\langle X \bullet \mathsf{new}_{val}, F[val := I \cap V(p)]\rangle \quad =$
  $\langle I \cap V(p), V\rangle \quad =$
  $(\langle I, V\rangle[\![\phi]\!])^a$

- ▷ ∘ $\bot^a = \bot$: similar.

- ▷ ∘ $(\phi \wedge \psi)^a = \phi^a \cdot \psi^a$

  ∘ $(\langle I, V\rangle[\![(\phi \wedge \psi)]\!])^a \quad =$
  $((\langle I, V\rangle[\![\phi]\!])[\![\psi]\!])^a \quad {}_{i.h.} =$
  $(\langle I, V\rangle[\![\phi]\!])^a \bullet [\![\psi^a]\!] \quad {}_{i.h.} =$
  $(\langle I, V\rangle^a \bullet [\![\phi^a]\!]) \bullet [\![\psi^a]\!] \quad {}_{ass.} =$
  $\langle I, V\rangle^a \bullet ([\![\phi^a]\!] \bullet [\![\psi^a]\!]) \quad =$
  $\langle I, V\rangle^a \bullet ([\![\phi^a \cdot \psi^a]\!]) \quad =$
  $\langle I, V\rangle^a \bullet ([\![(\phi \cdot \psi)^a]\!])$

- ▷ ∘ $(\Diamond^i_j\chi \wedge \psi)^b \ = \ \oplus_i \cdot (\Diamond^i_j\chi)^a \cdot \psi^b \cdot \ominus_i$

  ∘ $\langle I, V\rangle^a \bullet [\![(\Diamond^i_j\chi \wedge \psi)^b]\!] \quad =$
  $\langle I, V\rangle^a \bullet [\![\oplus_i]\!] \bullet [\![(\Diamond^i_j\chi)^b]\!] \bullet [\![\psi^b]\!] \bullet \mathsf{pop}_i \quad =$
  $\langle I, V\rangle^a \bullet \mathsf{push}^+_i \bullet i = V(i) \bullet [\![(\Diamond^i_j\chi)^b]\!] \bullet [\![\psi^b]\!] \bullet \mathsf{pop}_i \quad =$
  $\mathsf{push}^+_i \bullet i = V(i) \bullet \langle I, V\rangle^a \bullet [\![(\Diamond^i_j\chi)^b]\!] \bullet [\![\psi^b]\!] \bullet \mathsf{pop}_i \quad {}_{i.h.} =$
  $\mathsf{push}^+_i \bullet i = V(i) \bullet (\langle\langle I, V\rangle[\![\Diamond^i_j\chi]\!]_0, V\rangle)^a \bullet [\![\psi^b]\!] \bullet \mathsf{pop}_i \quad {}_{i.h.} =$
  $\mathsf{push}^+_i \bullet i = V(i) \bullet (\langle\langle I, V\rangle[\![\Diamond^i_j\chi \wedge \psi]\!]_0, V\rangle)^a \bullet \mathsf{pop}_i \quad =$
  $(\langle\langle I, V\rangle[\![\Diamond^i_j\chi \wedge \psi]\!]_0, V\rangle)^a$

- ▷ ∘ $(\neg\phi)^a \ = \ \lhd_\neg \cdot \phi^b \cdot \rhd_\neg$

  ∘ $(\langle I, V\rangle[\![\neg\phi]\!])^a \ = \ \langle I \backslash (\langle I, V\rangle[\![\phi]\!])_0, V\rangle^a \ =$
  $\langle X, F[val := \ (\langle I, V\rangle[\![\phi]\!])_0]\rangle$

- $\circ$ $[\![\triangleleft_\neg \cdot \phi^b \cdot \triangleright_\neg]\!] =$
  $[\![\triangleleft_\neg]\!] \bullet [\![\phi^b]\!] \bullet [\![\triangleright_\neg]\!] =$
  $\mathsf{push}^+_{n_1} \bullet n_1 = val \bullet [\![\phi^b]\!] \bullet [\![\triangleright_\neg]\!]$
- $\circ$ $\langle I, V\rangle^a \bullet \triangleleft_\neg \bullet [\![\phi^b]\!] \bullet [\![\triangleright_\neg]\!] =$
  $\mathsf{push}^+_{n_1} \bullet n_1 = I \bullet \langle I, V\rangle^a \bullet [\![\phi^b]\!] \bullet [\![\triangleright_\neg]\!] \quad i.h. =$
  $\mathsf{push}^+_{n_1} \bullet n_1 = I \bullet \langle(\langle I, V\rangle[\![\phi]\!])_0, V\rangle^a \bullet [\![\triangleright_\neg]\!] =$
  $\mathsf{push}^+_{n_1} \bullet n_1 = I \bullet \langle(\langle I, V\rangle[\![\phi]\!])_0, V\rangle^a \bullet \mathsf{push}^+_{n_2} \bullet$
  $n_2 = val \bullet \mathsf{new}_{val} \bullet val = (n_1 \backslash n_2) \bullet \mathsf{pop}_{n_1} \bullet \mathsf{pop}_{n_2} =$
  $\mathsf{push}^+_{n_1} \bullet n_1 = I \bullet \langle(\langle I, V\rangle[\![\phi]\!])_0, V\rangle^a \bullet \mathsf{push}^+_{n_2} \bullet$
  $n_2 = (\langle I, V\rangle[\![\phi]\!])_0 \bullet \mathsf{new}_{val} \bullet val = (I \backslash (\langle I, V\rangle[\![\phi]\!])_0) \bullet$
  $\mathsf{pop}_{n_1} \bullet \mathsf{pop}_{n_2} =$
  $\langle(\langle I, V\rangle[\![\phi]\!])_0, V\rangle^a \bullet \mathsf{new}_{val} \bullet val = (I \backslash (\langle I, V\rangle[\![\phi]\!])_0) =$
  $\langle(\langle I, V\rangle[\![\neg\phi]\!])_0, V\rangle^a$

$\triangleright$
- $\circ$ $(\Diamond^i_j \phi)^a = \triangleleft^i_j \cdot \phi^b \cdot \triangleright^i_j$
- $\circ$ $\langle I, V\rangle^a \bullet [\![\Diamond^i_j \phi]\!]^a =$
  $\langle I, V\rangle^a \bullet \mathsf{push}^+_m \bullet m = val \bullet \mathsf{new}_{val} \bullet val = j \bullet [\![\phi^b]\!] \bullet [\![\triangleright^i_j]\!] =$
  $\langle I, V\rangle^a \bullet \mathsf{push}^+_m \bullet m = I \bullet \mathsf{new}_{val} \bullet val = V(j) \bullet$
  $[\![\phi^b]\!] \bullet [\![\triangleright^i_j]\!] =$
  $\mathsf{push}^+_m \bullet m = I \bullet \bullet \langle V(j), V\rangle^a \bullet [\![\phi^b]\!] \bullet [\![\triangleright^i_j]\!] \quad i.h. =$
  $\mathsf{push}^+_m \bullet m = I \bullet \langle(\langle V(j), V\rangle[\![\phi]\!])_0, V\rangle^a \bullet [\![\triangleright^i_j]\!] =$
  $\mathsf{push}^+_m \bullet m = I \bullet \langle(\langle V(j), V\rangle[\![\phi]\!])_0, V\rangle^a \bullet \mathsf{new}_i \bullet i = val \bullet$
  $\mathsf{new}_{val} \bullet val = \begin{cases} \emptyset \Leftarrow (i = \emptyset) \\ m \Leftarrow (i \neq \emptyset) \end{cases} \bullet \mathsf{pop}_m =$
  $\mathsf{push}^+_m \bullet m = I \bullet \langle(\langle V(j), V\rangle[\![\phi]\!])_0, V\rangle^a \bullet \mathsf{new}_i \bullet$
  $i = (\langle V(j), V\rangle[\![\phi]\!])_0 \bullet \mathsf{new}_{val} \bullet val = \begin{cases} \emptyset \Leftarrow (i = \emptyset) \\ I \Leftarrow (i \neq \emptyset) \end{cases} \bullet \mathsf{pop}_m =$
  $\langle(\langle V(j), V\rangle[\![\phi]\!])_0, V\rangle^a \bullet \mathsf{new}_i \bullet$
  $i = (\langle V(j), V\rangle[\![\phi]\!])_0 \bullet \mathsf{new}_{val} \bullet val = \begin{cases} \emptyset \Leftarrow (i = \emptyset) \\ I \Leftarrow (i \neq \emptyset) \end{cases} =$
  $(\langle I, V\rangle[\![\Diamond^i_j \phi]\!])^a$

$\triangleright$
- $\circ$ $(\Diamond^i_j \phi)^b = \oplus_i \cdot (\Diamond^i_j \phi)^a \cdot \ominus_i$
- $\circ$ $\langle I, V\rangle^a \bullet [\![\phi^b]\!] =$
  $\mathsf{push}^+_i \bullet i = V(i) \bullet \langle I, V\rangle^a \bullet [\![(\Diamond^i_j \phi)^a]\!] \bullet \mathsf{pop}_i \quad i.h. =$
  $\mathsf{push}^+_i \bullet i = V(i) \bullet (\langle I, V\rangle[\![(\Diamond^i_j \phi)]\!])^a \bullet \mathsf{pop}_i =$
  $(\langle I, V\rangle[\![(\Diamond^i_j \phi)]\!])^a \bullet \mathsf{new}_i \bullet i = V(i) =$
  $\langle(\langle I, V\rangle[\![\Diamond^i_j \phi]\!])_0, V\rangle$

❑

This completes the proof of the proposition.

References

[Ash96] N. Asher. *Reference to abstract objects in discourse.* Kluwer, Dordrecht, 1996.

[Ben85] J. van Benthem. *Modal Logic and Classical Logic.* Bibliopolis, Naples, 1985.

[Eva80] G. Evans. Pronouns. *Linguistic inquiry*, 11:337–362, 1980.

[GNP92] J. Gawron, J. Nerbonne, and S. Peters. The absorption principle and e-type anaphora. In *Situation Theory and its applications, volume II.* CSLI Lecture Notes 26, Stanford, 1992.

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

[Kib94] R. Kibble. Dynamics of epistemic modality and anaphora. In H. *et al* Bunt, editor, *Proceedings of IWCS94.* Tilburg University, the Netherlands, 1994.

[Rob89] C. Roberts. Modal subordination and pronominal anaphora in discourse. *Linguistics and Philosophy*, 12:683–721, 1989.

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

[Ver94] C. Vermeulen. Incremental semantics for propositional texts. *Notre Dame Journal of Formal Logic*, 35:243–271, 1994.

[Ver97] C. Vermeulen. (some) dynamics of quantification, 1997. Workshop on quantification, ESSLLI97, manuscript available from author.

[Ver99] C. Vermeulen. Type declaration for modal subordination. In H. *et al* Bunt, editor, *Proceedings of IWCS99.* Tilburg University, the Netherlands, 1999.

[VV96] C. Vermeulen and A Visser. Dynamic bracketing and discourse representation. *Notre Dame Journal of Formal Logic*, 37(2):321–365, 1996.