Algebras for Boolean Presuppositions[*]

Kees Vermeulen kees.vermeulen@phil.uu.nl
Els Wolters ewolters@worldaccess.nl
Albert Visser albert.visser@phil.uu.nl

October 29, 1998

Abstract

We present an algebraic approach to the semantics of presuppositions in dynamic semantics: preconditions are introduced explicitly as separate components in the semantic algebra. The approach is worked out for a propositional language that is interpreted in a Boolean setting. We provide several (meta-)mathematical results about the example —some completeness issues are discussed and a decision method is presented— and we compare the approach with the presupposition-as-preconditions approach, the major alternative treatment of presuppositions in dynamic semantics. It turns out that our way of introducing presuppositions into a presuppositionless semantics gives satisfactory results for the examples inside its range.

Key words: Dynamic Semantics, Algebraic Semantics, Presuppositions, Natural Language, Residuation Lattices

MSC codes: 03B50, 03B60, 03B65, 03G05, 03G10, 68S05

[*]The first author wishes to acknowledge the financial support of the Netherlands Organisation for Scientific Research (NWO), project 300-75-306.

Contents

1 Introduction

Consider a sentence such as (John is a bachelor) What does this sentence mean? Surely, it can only be true if John is not married. But, somehow, it is also required that John has some additional properties: it seems that, for the sentence to be, true John should also be male and adult. Still, it is clear that these additional requirements on John are of a different kind than the demand that John should be unmarried. It is felt that, if John is not a male adult, the sentence is not simply false: it should never have been uttered in the first place. So if John turns out to be a little girl, the sentence will not be judged to be *false*, but *inappropriate* in another way. The technical term for this kind of inappropriateness is *presupposition failure*: we say that the sentence *asserts* that John is not married and *presupposes* that John is male and adult.

There are many examples in which a similar distinction between the assertion and the presupposition of an expression can be made. In the above example the presupposition is triggered by the word bachelor. Other examples of lexical presupposition triggers are: dead —presupposes animate— ... We call a presupposition that is triggered by such a word a *lexical* presupposition. But there are also other kinds of *presupposition triggers*. For example, in (the king of France is bold) it seems to be presupposed that France has a (unique) king. Here the definite article the triggers the presupposition. And in the sentence (Sue regrets that it rains) the factive verb to regret induces the presupposition that it actually rains.[1] For a purely truth conditional approach to semantics, presuppositions have always been a bit of a nuisance: after all the idea of a presupposition is precisely that it is *not* simply a truth condition. In dynamic semantics the concept of meaning is more fundamental than the concept of truth: truth is (only?) a derived notion. Therefore we can hope that in a dynamic approach to semantics a natural and elegant treatment of presuppositions becomes available.

There have been several attempts to provide such a dynamic account of the semantics of presuppositions.[2] In this paper we focus on one such attempt. The approach, due to Visser [17], proposes an algebraic model of the creation of presuppositions. In his algebraic model, Visser implements the basic intuition that a presupposition gives *negative* information, while the assertion counts as *positive* information. He works with information states which consist of two components: one component for the positive information and a separate one for the negative information. The construction of such pairs is similar to the construction of the (positive and negative) integers from the naturals. Below we will see in detail how the approach works out when it is applied in a simple propositional setting. In addition we compare our example with a major dynamic alternative account of presuppositions: the presuppositions-as-preconditions account. We will then provide several (meta-)mathematical results about the system.

Our main conclusion will be that, in the cases considered, the algebraic framework provides a good format for the systematic introduction of presup-

[1] See [1] for a thorough introduction to the subject.
[2] See for example [13] or [1] for discussion and references.

positions into a presuppositionless setting. This encourages us to develop the approach further to include more involved examples in more expressive formalisms.

1.1 Two Approaches

In dynamic semantics it is held that the meaning of expressions is, for a large part, determined by their *information change potential.* In each situation or information state the interpretation of a sentence will bring about some change: the information in the sentence is combined with the information that is already available and the result is a new information state. If we can characterise this potential of the sentence to change information states, we will have captured a lot —if not all— of its meaning.[3] Although dynamic semantics is a fairly young field of research, there are already two 'schools' of dynamic semantics.[4] Each school uses a different format for the formal description of information change potentials. One school of dynamic semantics describes meanings as *relations* between information states: by interpreting a sentence ϕ in an input state $\mathcal{I}$ an output state $\mathcal{O}$ can arise iff the pair $\langle \mathcal{I}, \mathcal{O} \rangle$ stands in the relation R_ϕ. In such a relational approach to dynamic semantics the semantic universe will consist of (binary) relations.[5] A second school associates a *date base* with each sentence. The data base will have a slot for *each* aspect of the meaning of the sentence. Two data bases can be *merged* into one larger data base, by an operation $\bullet$ which we shall call *the merger.* Then the information change potential can be described in terms of the merger of data bases: if our initial information is correctly described by database $\mathcal{I}$ and ϕ itself corresponds to the data base $\mathcal{I}_\phi$, then interpreting ϕ in state $\mathcal{I}$ results in $\mathcal{I} \bullet \mathcal{I}_\phi$. In a data base approach to dynamic semantics the semantic universe contains data bases and a merger operation to combine them.

The first school of dynamic semantics naturally finds inspiration in relational algebra and the (operational) semantics of programming languages. The second approach leads to a different style of formalisation: algebras[6] of the form $\langle X, \bullet \rangle$ become the main topic of investigation where X is the set of data bases and $\bullet$ the merger operation. The crucial question now is how the appropriate algebra $\langle X, \bullet \rangle$ —which contains a slot for *each* aspect of the meaning of a sentence— can be constructed systematically from several simpler algebras $\langle X_i, \bullet_i \rangle$, each of which describes only *one* aspect of the meaning of a sentence. We regard Groenendijk and Stokhof [3] and Veltman [16] as crucial examples of the first approach and consider Kamp [9], Heim [7] and Kamp and Reyle [10] as examples of the second approach. Groenendijk and Stokhof [4] compare different formulations of the first approach and also Groeneveld [5] is useful in this respect.

[3]See [14] for an introduction and references.

[4]Fortunately in practice the different schools as well as their results turn out to be extremely compatible.

[5]Alternatively one could use *(update) functions*: interpreting ϕ in state $\mathcal{I}$ results in state $\mathcal{O}$ iff $f_\phi(\mathcal{I}) = \mathcal{O}$. In such a functional set up of dynamic semantics the semantic universe will consist of (update) functions. (Cf. Veltman [16]).

[6]Or categories: cf. [19].

Zeevat [20] provides the algebraic formulation of the data base approach and Visser and Vermeulen [19] develop this idea further.

Thus we obtain two main styles of formalisation in dynamic semantics: one describes meanings as programs, the other describes meanings as data bases. In both schools quite some thought has been given to the treatment of presuppositions. We will see the basics of the ideas of *both* theories below, illustrated in the case of a simple propositional language. However, our main concern will be with the meanings-as-data-bases approach to dynamics.

The remainder of this paper will be organised as follows. In section 2 we present the technical apparatus of the algebraic approach to presuppositions. This section also contains some philosophical motivation for the choices made in designing this apparatus. Still, the reader who is not interested in generalities may want to jump to subsection 2.5 immediately, where we summarise the technicalities that will be used in the rest of the paper. Then, in section 3, we present some applications of the algebraic techniques. We discuss examples and compare the results with other approaches in the literature. In section 4, we prove some metamathematical results about the algebras that we use in section 3.

2 The Algebraic Approach: Residuation Lattices of Pairs

In this section we present the formal apparatus that was developed in Visser [17] for the treatment of actions under presuppositions. In the next section we will use a simplified instance of this approach for the analysis of simple cases of presuppositions. This simplified version is presented in subsection 2.5. Perhaps the reader will want to skip the more general subsections 2.1, 2.3 and 2.4 at first reading.

To give you a clear picture of the apparatus we present it in several steps. First we discuss the distinction between the synchronic and the diachronic perspective on information. To capture both perspectives in our model we will use the notion of a residuation lattice. Then we discuss the issue of partiality involved in the analysis of presuppositions. We regard presupposed information as missing information and develop the formalism accordingly: this leads to the introduction of algebras of pairs with a 'positive' and a 'negative' component. Then interaction of these two ideas is established in residuation lattices of pairs. We present these in full generality: they are our general proposal for the formalisation of the behaviour of presuppositions.

Finally, we present the instances of such lattices that we will see in this paper. As we will mainly be concerned with Boolean presuppositions and assertions, we only need consider pair algebras where both components come from a Boolean algebra. In Boolean algebras some of the computations are easier than in the general case and it will be convenient for reference later on to have a separate description of this simple case.

2.1 Two Perspectives on Information

The apparatus that we will present is part of *dynamic semantics*: the tradition in formal semantics that tries to model the exchange of information as it takes place in discourse interpretation. This means that we have in mind a situation in which an agent tries to process the information that another agent makes available in —written or spoken— discourse. Our agent will have some initial information available and will process the new information in the light of this initial information. This will lead to a new state of information.

In a realistic model of this situation the order in which the information is presented will play an important role. Clearly it can make a big difference which information comes first. Typical examples of this are simple anaphors as in (John comes in. He is smiling). Here the pronoun he in the second sentence depends on information from the first sentence for its interpretation: it requires John as an antecedent. Also presuppositions are a typical example of cases where order matters. Consider (at that time France was a monarchy. The king of France was rich). Here the information presupposed by the second sentence is provided by the first sentence. As a result the discourse as a whole does not carry a presupposition.

These examples indicate why we need to consider information as it is given *in time*. We call this the *diachronic* perspective on information.

Still, in a realistic model of information exchange we also want to compare information that is not ordered in time. This hardly needs an argument, but let's look at an example to fix our thoughts. Consider an agent ready and waiting for new information. Now we have two options: we can either tell him: (John had a drink last night) or (John drank some wine last night). Now these sentences are not to be compared in some temporal order: they are alternative bits information at one and the same point in time. Clearly the second sentence is more informative. We will want to include in our model a way of comparing information is this sense. We call this the *synchronic* view on information. Our model of discourse interpretation will have to embody these two perspectives on information: the synchronic and the diachronic perspective.

The starting point for our formalisation is a set of information states A, the things we called data bases above. We obtain a diachronic perspective on these states by including a *monoidal* operation that we will call the *merger*. For the merger we will use the notation $\bullet$. So $\bullet$ will be a binary operation that is sensitive to order —not, in general, $a \bullet a' = a' \bullet a$— but not to bracketing —$a \bullet (a' \bullet a'') = (a \bullet a') \bullet a''$— and we assume that a unit element 1 is available such that $1 \bullet a = a \bullet 1 = a$. 1 stands for the *tabula rasa* state of information: it has no information content and is completely harmless.

For the synchronic perspective on A we introduce a lattice ordering $\leq$. This means that $\leq$ is a partial order along which finite suprema and infima exist. We will use the usual notation $\vee$ and $\wedge$ for the suprema and the infima.

We also introduce a way of relating the two perspectives: we assume that two *residuals* of $\bullet$ are present, called $\rightarrow$ and $\leftarrow$. The defining property for residuals is:

$$a_1 \bullet a_2 \leq a_3 \;\Leftrightarrow\; a_1 \leq (a_3 \leftarrow a_2) \;\Leftrightarrow\; a_2 \leq (a_1 \rightarrow a_3)$$

One can check that in cases where $\bullet$ is not order sensitive, the operations $\rightarrow$ and $\leftarrow$ will co-incide. In this case the property defines only one operation. Examples are: Boolean algebra, Heyting algebra, linear logic. In all these cases the property simply defines the implication of the logic. Hence, one way to think about residuals is as order sensitive forms of implication. There are also several examples of residuals in the literature where order sensitivity does play a role, e.g. categorial grammar and relation algebra. In general, the way to read the defining property is as follows: it states that $(a_3 \leftarrow a_2)$ is the largest element x such that $x \bullet a_2 \leq a_3$. Similarly $(a_1 \rightarrow a_3)$ is the largest element x such that $a_1 \bullet x \leq a_3$. By introducing the residuals in our model we introduce the assumption that such largest elements are always available in A. Hence we introduce an assumption on the relation between $\bullet$ and $\leq$, over and above the assumptions that we have made for each of them separately.

The following definition summarises the discussion so far.

Definition 2.1 A residuation lattice is an algebra

$$\mathcal{A} = \langle A, \vee, \wedge, \top, \mathbf{0}, \bullet, 1, \rightarrow, \leftarrow \rangle$$

which satisfies the following extra requirements. Define $a \leq b := \; a \vee b = b$. We have:

- $\langle A, \vee, \wedge, \top, 0 \rangle$ is a lattice, with top $\top$ and bottom 0;
- $\langle A, \bullet, 1 \rangle$ is a monoid;
- $a \bullet b \leq c \iff a \leq c \leftarrow b \iff b \leq a \rightarrow c$.

$\leftarrow$ is *left residuation* or *post-implication*. $\rightarrow$ is *right residuation* or *pre-implication*. ◻

In some applications we do not really need *all* the operations present in a residuation lattice. For example, we do not always care about the top and bottom of the lattice ordering. Then we can work with a somewhat reduced signature. We will say a bit more about that in section 2.6. But first we start with the 'full' signature. Our notation $\top$, for top, is standard. The notation for the bottom can be explained by the fact that in a residuation lattice the bottom of the lattice ordering, will always be a *zero element* or *annihilator* of the monoid: $0 \bullet a = a \bullet 0 = 0$.[7]

2.2 Natural Language Implication in Residuation Lattices

The operation $\bullet$ in a residuation lattice is supposed to represent order dependence: $\alpha \bullet \beta$ stands for the the information α *followed by* the information β.

[7]For this fact, and several other facts about residuation lattices, we refer to [17] p.205.

Evidently, we cannot expect in general that $\alpha \bullet \beta = \beta \bullet \alpha$. A similar remark applies to the use of implications in natural language. Let's consider an example: If John owns a donkey, then he feeds it. Here we will need information from the antecedent of the implication to be able to process the consequent. A lot can be said —and has been said— about such examples. For us it means that the following informal characterisation of so-called *dynamic implication* makes sense:

> $\phi \Rightarrow \psi$ means:
> if we add information ϕ, we reach a state of information in which the interpretation of ψ does not lead to an essential increase of information.

Note how the order in which we process the different information items is crucial here: starting in our current state, s say, we first add the information from ϕ to reach a new information state, s' say. Then we add ψ in state s'. After that we will have reached a state, s'' say, which contains no more information than s'. In terms of residuation lattices this notion of implication can be approximated as follows:

$$(\alpha \Rightarrow \beta) \;=\; (\alpha \bullet \beta \leftarrow \alpha)$$

This ensures that $1 \;\leq\; (\alpha \Rightarrow \beta)$ iff $\alpha \;\leq\; \alpha \bullet \beta$ and, hence, we will know that, in each state σ, $\sigma \bullet \alpha \;\leq\; \sigma \bullet \alpha \bullet \beta$.[8] This means that the enrichment of a state σ with information α will already be at least as informative (synchronically speaking) as the enrichment of σ with $\alpha \bullet \beta$, as required.

Note that in the final comparison we use the *synchronic* ordering of information $\leq$: it is clear from the natural language examples above that *diachronically* speaking a lot of things will happen if we add β to $\sigma \bullet \alpha$: β may bring new topics to our attention explicitly which will thereby become available for (anaphoric) reference. But this kind of enrichment does not seem to matter for the evaluation —i.e. 'truth'— of natural language implications.

2.3 Partiality as Unsaturatedness

Apart from the diachronic/synchronic distinction there is another ingredient that is crucial to any realistic model of information exchange: partiality. Here we are interested in partiality in the following sense. An agent who interprets discourse will do this in the light of the information that is already available to him/her. The current information state has to support the interpretation. As examples of required support we can look again at the cases of anaphora and presupposition indicated above. The state of information of the agent after interpretation of the first sentence has to provide an antecedent for the anaphor he in the first example. In the second example the information state has to provide the information that is presupposed by the second sentence. Hence in both examples there is some information 'missing' in the second sentence. The

[8] This follows, since monotonicity of $\bullet$ in both arguments holds in any residuation lattice (cf. [17]).

information state that we associate with the second sentence will have to reflect this unsaturatedness. This requires a distinction between states of information that are saturated, or: complete, and states of information that are unsaturated, or: partial. We will call the set of saturated states of information S and will assume that we know which subset $S \subseteq A$ is.

In line with the remarks about the order sensitivity of discourse interpretation above, also this notion of unsaturatedness is sensitive to order: we are interested in unsaturatedness towards the *past* and assume that it cannot be made up for in the future. In other words, unsaturatedness will persist:

$$a' \bullet a \in S \;\Rightarrow\; a' \in S \text{ (OTAT)}$$

Visser [17] calls this the OTAT principle: *Once a Thief, Always a Thief.* So we model partiality by having a set $S \subseteq A$ that contains the saturated information states. S will satisfy the OTAT principle. *From this point on we will assume that our residuation lattice satisfies OTAT.*

Let's think a bit about the right choice of S in our particular case. As indicated by the example above, we will consider expressions that carry presuppositions as unsaturated. The presupposed information is regarded as information that is *missing.* In contrast the asserted information is regarded as information that is *given* by the expression. Now if we assign an information state $a \in A$ to some expression then a stronger presupposition means that more information is missing. Hence stronger presupposition makes the state less informative. Conversely, a stronger assertion means that more information is given. Hence a stronger assertion will make the state more informative. We intend 1 as a completely harmless bit of information: $a \bullet 1 = a$ for all $a \in A$. Hence 1 cannot carry a presupposition. But for the same reason it cannot make much of an assertion either: 1 is a point of no presupposition and no assertion. So 1 will form a point of division between the unsaturated and the saturated. Thereby it makes sense to set $S = \{a \in A |\; a \leq 1\}$. We will opt for this choice of S in what follows.

Note that, using our new characterisation of S, OTAT becomes equivalent to: $(1 \leftarrow a) \leq 1$.

Incorporating partiality in the way sketched above is quite general and quite elegant. Still it may be a bit puzzling to work with a *total* operation $\bullet$ to model *partiality* of information. Perhaps the reader would expect the partiality of information to correspond to partiality of operations. Fortunately the set up with a set A of (partial) states and a set $S \subseteq A$ of saturated states gives us a natural way of generating partial functions that transform states into states. We will call such functions *update functions.* So an update function is a partial function[9] $F : S \to S$. Before we show how to generate such update functions we make a little excursion.

[9]We will find it convenient to use postfix notation for update functions: we will write sF for the result of applying F to s. We will use the notation $\Downarrow$ and $\Uparrow$ for definedness and undefinedness respectively of partial functions.

We can regard the update functions themselves as items of information: applying the update function is like adding the information that the update function stands for. Then, if we try to order update functions according to our considerations about partiality, the following definition makes sense:

$$F \leq G \;\Leftrightarrow\; \mathsf{dom}(G) \subseteq \mathsf{dom}(F) \;\&\; \forall s \in \mathsf{dom}(G):\; sF \leq sG.$$

Here the first condition displays the point about the relation between unsaturatedness, partiality and missing information: $\mathsf{dom}(G) \subseteq \mathsf{dom}(F)$ says that G misses more information than F. The second clause simply says that on their common domain —i.e. $\mathsf{dom}(G)$— the result sF is more informative than the result sG.

Within the class of all such updates there is a natural subclass that can be generated in an elegant way from the underlying algebra $\mathcal{A}$. This class fits nicely with our ideas about presuppositions and therefore is our candidate for the analysis of presuppositions in natural language. Recall that we regard presupposed information as *missing* information: it is in a negative place. Asserted information on the other hand is in a positive place: it is information that is added by the expression. Now let's consider the situation where both the presupposition and the assertion are elements of $\mathcal{A}$: say $a' \in A$ is the presupposition and $a \in A$ the assertion. Given the presupposition the assertion should not produce unsaturatedness, i.e. $a' \bullet a \in S$. By OTAT this implies $a' \in S$. Now we can associate with such a pair $\alpha = \langle a', a\rangle$ an update function Ψ_α as follows:[10]

$$s\Psi_\alpha \;=\; s \bullet a \text{ for } s \leq a' \text{ and is } s\Psi_\alpha \Uparrow \text{ otherwise}$$

The update function Ψ_α as defined here can be understood as follows. First Ψ_α tests whether $s \leq a'$, i.e. it tests whether the presupposed material a' is provided by s. If so, Ψ adds the asserted information a to s and we obtain $s \bullet a$. If the presupposed material is not provided by s, the update is unsuccessful.

Note that the function Ψ_α is completely fixed by the pair $\alpha = \langle a', a\rangle$. Below we will confuse the set $\{\Psi_\alpha :\; \alpha = \langle a', a\rangle \;\&\; a' \bullet a \in S\}$ with $\{\langle a', a\rangle :\; a' \bullet a \in S\}$.[11]

2.4 The Residuation Lattice of Pairs

Now we have introduced two ingredients that we find essential for a realistic model of the information exchange involved in discourse interpretation: the synchronic-diachronic distinction and the notion of unsaturatedness of information. The next step is to get the two ingredients to live together. For this

[10] Note that this definition requires that: $s \leq a' \;\Rightarrow\; s \bullet a \in S$. This holds by monotonicity of $\bullet$ in its left argument —this property holds in any residuation lattice— and downward closure of S.

[11] There also is another, smaller class of natural update functions, which are called Φ_a ($a \in A$) in [17]. These updates are defined as follows:

$$s\Phi_a \;=\; s \bullet a \text{ if } s \bullet a \in S \text{ and is } s\Phi_a \Uparrow \text{ otherwise}$$

Given the choices made above, we see that Φ_a can be obtained as $\Psi_{\langle(1 \leftarrow a), a\rangle}$.

purpose we need a residuation lattice consisting of the update functions that we discussed above. So let's try to transform the set $\{\langle a', a\rangle : \ a' \bullet a \in S\}$ into a residuation lattice. To make this construction work, we have to add the following additional assumption, Ω, on $\mathcal{A}$:

$$s' \bullet a \leq s' \bullet b \;\&\; s \leq s' \;\Rightarrow\; s \bullet a \leq s \bullet b \qquad (\Omega)$$

This property has no intuitive content: it is simply the weakest condition that makes things work. However, we will see later on that several more natural conditions are around that imply Ω. As we will have intuitive grounds for these stronger assumptions, we cannot be accused of doing 'funny business'.

First we look at the ordering of the pairs. A natural candidate for ordering update functions was given above, so let's try to use this ordering. One can show, using Ω, that in terms of the pairs this ordering reads as:[12]

$$\langle a', a\rangle \leq \langle b', b\rangle \;\Leftrightarrow\; b' \leq a' \;\&\; b' \bullet a \leq b' \bullet b$$

Next we consider the merger of the pairs. Again a natural candidate for the merger is available: composition of (partial) functions. If we write this down in terms of the pairs this leads to the following definition:[13]

$$\langle a', a\rangle \bullet \langle b', b\rangle \;=\; \langle a' \wedge (b' \leftarrow a), a \bullet b\rangle$$

This is simply the way the composition of $\Psi_{\langle a', a\rangle}$ and $\Psi_{\langle b', b\rangle}$ looks in terms of the pairs, so we need no further motivation for this funny expression. Still some remarks about the merger of pairs may be helpful. First note that on the presupposition side we find $a' \wedge (b' \leftarrow a)$. This shows how the presupposition of $\langle a', a\rangle$ is inherited by $\langle a', a\rangle \bullet \langle b', b\rangle$. Furthermore, we see that the presupposition of $\langle b', b\rangle$ also re-occurs, but in a weakened form: $(b' \leftarrow a)$. This stands for what is left of b' after a will have been asserted. Hence the temporal order of the merger —first $\Psi_{\langle a', a\rangle}$, then $\Psi_{\langle b', b\rangle}$— is reflected. On the assertion side we simply find that first a will be asserted and then b, as expected.

Now the question is whether this choice for $\leq$ and $\bullet$ gives rise to a residuation lattice of pairs. It can be shown that it almost works: to make everything work smoothly we only have to add a few artificial elements to the set of pairs: if we look at the ordering $\leq$ on the pairs we see that we have to add a new 0. We will not go into the details here and simply quote Visser's result:

Proposition 2.2 Let $\mathcal{A}$ be a residuation lattice that satisfies Ω. Then there is a unique residuation lattice $\mathcal{U}(\mathcal{A}) = \langle Y, \vee, \wedge, \top, 0, \bullet, 1, \rightarrow, \leftarrow\rangle$ such that:

[12] There is a subtlety here: $\leq$ was a partial order on the update functions Ψ_α. But different pairs $\langle a', a\rangle$ and $\langle b', b\rangle$ may generate the same update function. Hence on the pairs the ordering $\leq$ is only a preorder. We can correct this by dividing out the following equivalence:

$$\langle a', a\rangle \equiv \langle b', b\rangle \;\Leftrightarrow\; b' = a' \;\&\; b' \bullet a = b' \bullet b$$

So strictly speaking we should be talking about *equivalence classes* of pairs instead of pairs simpliciter. We will ignore this in most of what follows.

[13] Compare with Fact 12.2 in [17].

- $Y = \{0\} \cup \{\langle a', a\rangle \mid a' \bullet a \in S\}$
- on the pairs $\langle a', a\rangle$ the operations of the residuation lattice agree with function composition and the ordering of update functions discussed above

$\blacksquare$

The proof simply is the definition of the required connectives. Lengthy computations lead to the following definitions:

$$\begin{array}{rcl}
\top & := & \langle 0,0\rangle \\
\langle a',a\rangle \vee \langle b',b\rangle & := & \langle a' \wedge b', a \vee b\rangle \\
\langle a',a\rangle \wedge \langle b',b\rangle & := & \langle a' \vee b', (a' \to a' \bullet a) \wedge (b' \to b' \bullet b)\rangle \\
\langle a',a\rangle \bullet \langle b',b\rangle & := & \langle a' \wedge (b' \leftarrow a), a \bullet b\rangle \\
\langle b',b\rangle \leftarrow \langle a',a\rangle & := & \langle b', (b' \to a') \wedge (b' \to (b' \bullet b \leftarrow a))\rangle \\
 & = & \langle b', b' \to (a' \wedge (b' \bullet b \leftarrow a))\rangle \\
\langle a',a\rangle \to \langle b',b\rangle & := & \langle b' \bullet a, b' \bullet a \to b' \bullet b\rangle \text{ if } b' \leq a',\ := 0 \text{ otherwise}
\end{array}$$

We see that for a residuation lattice $\mathcal{A}$ that satisfies Ω we can construct the algebra of pairs $\mathcal{U}(\mathcal{A})$. $\mathcal{U}(\mathcal{A})$ is a residuation lattice: hence it incorporates the synchronic-diachronic distinction. Moreover, the pairs correspond in a natural way to update functions that, in turn, embody the view of partiality-as-unsaturatedness that we want for our analysis of presuppositions. So we now have developed a general apparatus for the dynamic analysis of presuppositions. Of course this general apparatus has to be tested. Apart from more philosophical motivation, along the lines that we have followed above, testing the apparatus means two things: the first test is in the application to real examples of presuppositions in natural language. An important second test is in an investigation of the formal properties of the apparatus.

Below we will follow both lines of testing, but only for a simplified case: the case where the base algebra $\mathcal{A}$ is a Boolean algebra. In the next subsection we will see that for Boolean base algebras some of the constructions work out a bit nicer than in the general case.

2.5 Simplification: Going Boolean

Below we will apply the apparatus to cases where both the presupposed and the asserted information come from a Boolean algebra $\mathcal{A}$. As an example one can think of the case where $\mathcal{A} = \wp(W)$ for a set of possible worlds W. In this subsection we summarise how the construction simplifies in such a case.

Fact 2.3 A Boolean algebra $\mathcal{A} = \langle A, \cap, \cup, \neg, \top, \bot\rangle$ 'is' a residuation lattice $\mathcal{A} = \langle A, \vee, \wedge, 0, 1, \bullet, \to, \leftarrow\rangle$ if we read the connectives in the residuation lattice as follows.

Residuation Algebra	Boolean Algebra	Boolean Algebra of Sets
$\vee$	$\vee$	$\cup$
$\wedge$	$\wedge$	$\cap$
0	$\bot$	$\emptyset$
1	$\top$	A
$\top$	$\top$	A
$\bullet$	$\wedge$	$\cap$
$\rightarrow$	$\neg(-)\vee(-)$	$\neg(-)\cup(-)$
$\leftarrow$	$\neg(-)\vee(-)$	$\neg(-)\cup(-)$

Note that in a Boolean algebra $\mathcal{A}$ the canonical choice for S turns out to be A itself! ❑

We assume that the base algebra $\mathcal{A}$ is Boolean. It does not follow that now also $\mathcal{U}(\mathcal{A})$ will be a Boolean algebra. Still the computations in $\mathcal{U}(\mathcal{A})$ simplify considerably. First we note that already a much weaker assumption on $\mathcal{A}$ leads to a simplification:

Fact 2.4 Assume that for all $s \in S \subseteq \mathcal{A}$ we have: $s \bullet s = s$. We call this property S-idempotency. Now the following holds:[14]

- S-idempotency implies Ω
- S-idempotency implies: $\forall a', b\ \exists! a \le a' :\ \Psi_{\langle a',b\rangle} = \Psi_{\langle a',a\rangle}$ ❑

Hence, for S-idempotent residuation lattices, the pair construction will work. Furthermore we only have to look at pairs $\langle a', a\rangle$ where $a' \le a$. *Below we will always assume that our pairs have this 'normalised' form.* Note that any Boolean algebra is S-idempotent!

Finally we look at the definitions of the connectives for Boolean base algebras. These definitions simplify quite a bit, as we can simply calculate the Boolean way in each of the components of the pairs. In our notation we will sometimes interchange the Boolean notation and the notation from the residuation lattices. This should not lead to confusion.

Fact 2.5

$$\begin{array}{rcl}
\top & = & \langle\bot,\bot\rangle \\
1 & = & \langle\top,\top\rangle \\
0 & & \text{is the new 0 that we add in the construction} \\
\langle a',a\rangle \vee \langle b',b\rangle & = & \langle a'\wedge b', (a\vee b)\wedge a'\wedge b'\rangle \\
\langle a',a\rangle \wedge \langle b',b\rangle & = & \langle a'\vee b', (a'\vee b')\wedge(a'\rightarrow a)\wedge(b'\rightarrow b)\rangle \\
\langle a',a\rangle \bullet \langle b',b\rangle & = & \langle a'\wedge(a\rightarrow b'), a\wedge b\rangle \\
\langle a',a\rangle \leftarrow \langle b',b\rangle & = & \langle a', a'\wedge b'\wedge(b\rightarrow a)\rangle \\
\langle a',a\rangle \rightarrow \langle b',b\rangle & = & \langle b'\wedge a, b\wedge a\rangle \text{ if } b'\le a', =\ 0 \text{ o.w.}
\end{array}$$

❑

[14]Cf. Visser [17], subsection 12.5.1, for additional details.

In the Boolean case the relationship between the original algebra $\mathcal{A}$ and the corresponding pair algebra becomes particularly easy. We define the standard embedding function $\mathsf{emb}_{\mathcal{A}}$ from $\mathcal{A}$ to the states of $\mathcal{U}(\mathcal{A})$ by: $\mathsf{emb}_{\mathcal{A}}(a) := \langle 1, a\rangle$.[15] We have:

Fact 2.6 1. $\mathsf{emb}_{\mathcal{A}}$ is a bijection between the elements of $\mathcal{A}$ and $S_{\mathcal{U}(\mathcal{A})} \setminus \{0\}$, i.e. the states of $\mathcal{U}(\mathcal{A})$ except 0.

2. $\mathsf{emb}_{\mathcal{A}}$ commutes with 1, $\vee$, $\wedge$, $\bullet$ and $\leftarrow$.[16]

3. $\mathsf{emb}_{\mathcal{A}}$ does not commute with $\top$, since

$$\mathsf{emb}_{\mathcal{A}}(\top_{\mathcal{A}}) = \langle 1,1\rangle \neq \langle 0,0\rangle = \top_{\mathcal{U}(\mathcal{A})}.$$

It does not commute with $\perp$, since:

$$\mathsf{emb}_{\mathcal{A}}(\perp_{\mathcal{A}}) = \langle 1,0\rangle \neq 0 = \perp_{\mathcal{U}(\mathcal{A})}.$$

Finally it does not commute with $\rightarrow$.

$\Box$

From this point on the reader will only encounter Boolean pair lattices. So she may forget about the whole of section 2 and just remember that we will be working with pairs $\langle a', a\rangle$ where $a' \leq a \in \mathcal{A}$ for some Boolean algebra $\mathcal{A}$, where special operations on these pairs are defined as in fact 2.5. To make things even simpler, the reader may take $\mathcal{A} = \wp(W)$, for some set of worlds W.

2.6 Reduced Signatures

In the construction of a residuation lattice of pairs from a residuation lattice we do not get a bottom element for free: we have to add a new bottom element 0 by hand. Sometimes we will not be interested in this artificial element: we will just look at the real pairs. In the Boolean case this will cause no problems, for most of the operations in the constructed residuation lattice, except for $\rightarrow$, only generate non-zero elements from non-zero elements (cf. definition 2.2). We call the set of real pairs $U^o(\mathcal{A})$ and use $\mathcal{U}^o(\mathcal{A})$ as notation for the algebra

$$\langle U^o(\mathcal{A}), \vee, \wedge, \top, \bullet, 1, \leftarrow\rangle.$$

Thus, we obtain $\mathcal{U}^o(\mathcal{A})$ from $\mathcal{U}(\mathcal{A})$, be removing 0 from the domain and by taking 0 and $\rightarrow$ from the signature.

In other cases we will disregard some further operations that are automatically available on pairs. It will be convenient also to have a notation for such situations. As a rule we will write the operations that remain in the subscript: $\mathcal{U}^{(o)}_{[op_1,\ldots,op_n]}(\mathcal{A})$ will be the algebra $\mathcal{U}^{(o)}(\mathcal{A})$ with the reduced signature $[op_1, \ldots, op_n]$. To summarise:

[15] Remember that in the Boolean case $1 \leftarrow a = 1$.
[16] Remember that the Boolean $\bullet$ is just $\wedge$.

- $\mathcal{U}^o(\mathcal{A})$ is $\mathcal{U}(\mathcal{A})$ without 0 in the domain and without $0, \to$ in the signature.
- $\mathcal{U}^{(o)}_{[op_1,\ldots,op_n]}(\mathcal{A})$ is the reduct of $\mathcal{U}^{(o)}(\mathcal{A})$ to the signature $[op_1, \ldots, op_n]$.

In addition we will employ the following syntactical convention. We use the notation $\mathcal{T}_{[op_1,\ldots,op_n]}(P)$ for the set of terms of signature $[op_1, \ldots, op_n]$ on the set of propositional atoms P.

3 Applications: Boolean Presuppositions

In this section we are concerned with the application of the general theory outlined above to concrete examples of presuppositions in natural language. Then we will evaluate our results by a comparison with other approaches in the literature, in particular the presuppositions-as-preconditions approach of Van Eijck [2] and the three valued logic of Kracht [12].

In the applications we will simplify matters in two ways: first we use a propositional language to represent the natural language examples. This means that we ignore the process by which a presupposition trigger occurring somewhere inside a proposition produces the presupposition of the proposition as a whole. It also means that the presuppositions we predict for expressions will always be propositions, thereby ignoring other presuppositional effects. The second simplification we make is in the choice of the base algebra. Below we will assume that a Boolean base algebra $\mathcal{A}$ is given.[17] These simplifications of the general situation will surely help to see through some of the technical details involved in the general case. Furthermore, it is precisely the fact that already this simplified situation gives such nice results, that shows the potential of the general construction.

3.1 Examples

Let's consider a few simple sentences:

John is male.
John is adult.
John is unmarried.
John is smiling.
John is a bachelor.

If we want to represent such simple sentences in a propositional language, it makes sense to just represent them as atomic propositions, say m, a, u, s and b respectively. Normally we would then go on to interpret such atomic propositions straightforwardly in a truth conditional way. For example, we would assume some set of possible worlds W to be given and we would have each atomic proposition denote those possible worlds where the atomic proposition

[17] Recall that this does not imply in general that also $\mathcal{U}(\mathcal{A})$, the pair algebra over $\mathcal{A}$ is a Boolean algebra!

is true. In this way (John is male) would denote the set of worlds where John is male, etc.

As long as we only consider sentences such as (John is male), this is probably all we want to do. But there is a more interesting example in our list which makes us think again: John is a bachelor. This simple sentence carries a presupposition: it is *presupposed* that John is a male adult. In addition the sentence *asserts* that John is unmarried. So the atomic proposition b seems to have the proposition $m \wedge a$ as a presupposition and in addition asserts only u. This means that already the atomic propositions require a complex denotation: an atomic proposition p does not simply stand for an element $[\![p]\!] \in \wp(W)$, but each atomic proposition comes with *two* such elements: one for its presupposition and one for its assertion. These two elements jointly will give an inhabitant of $\mathcal{U}(\wp(W))$, the pair algebra over base algebra $\wp(W)$. This gives the basic idea for the representation of simple sentences in the algebraic setting. We can now go on to interpret more complex examples such as:

> John is adult. He is smiling.
> John is male. He is a bachelor.
> If John is adult, then he is a bachelor.
> If John is a bachelor, then he is smiling.
> If John is a bachelor, then he is unmarried.

The interpretations of these complex sentences will be produced from the interpretations of the simple sentences using only the operations of the lattice of pairs. This way we will *predict* the presuppositions of the complex sentences based on the information about the presuppositions of the simple sentences: the so-called projection problem. These predictions can then be compared with our intuitions as natural language users, thus providing a first test of the construction.

Let's make this precise. First we define a propositional representation language $\mathcal{L}$. Let a set of propositional variables P be given. p will range over P. Define:

$$\mathcal{L}: \quad \phi \ ::= \ p \mid \bot \mid \phi \cdot \phi \mid (\phi \Rightarrow \phi).$$

We see that $\mathcal{L}$ will contain some atomic propositions to represent simple sentences. Let's assume that P includes representations of the simple sentences above, say $m, a, u, s, b \in P$, where m represents (John is male), etc. $\bot$ will be used as falsum: a proposition that is false in all situations. The connectives correspond in the expected way to ways of building complex sentences in natural language. $\cdot$ stands for natural language concatenation (or: conjunction), $\Rightarrow$ for implication. We have no connectives for disjunction and negation yet, but these will be discussed later on.

Next we consider the interpretation of this propositional language in a Boolean pair lattice $\mathcal{U}(\mathcal{A})$.[18] We already discussed the interpretation of the atoms in P: each atom will denote the pair consisting of a presupposition and an

[18]So $\mathcal{A}$ could be $\wp(W)$ for some set of possible worlds.

assertion. We will assume that the information of the presupposition is always included in the assertion of a proposition. This will be inessential.[19] Let's assume in addition that the base algebra $\mathcal{A}$ contains elements m, a, u, $s \in \mathcal{A}$ that stand for the assertions of the corresponding propositions.[20] Then we can set:

$$\begin{array}{rcl} [\![m]\!] & = & \langle 1, m\rangle \\ [\![a]\!] & = & \langle 1, a\rangle \\ [\![u]\!] & = & \langle 1, u\rangle \\ [\![s]\!] & = & \langle 1, s\rangle \end{array}$$

These simple sentences do not really carry a presupposition, so we get 1 as a first component.[21] The second component simply is the associated element in $\mathcal{A}$. For b the situation is more interesting: b should behave as (John is a bachelor). As was discussed above, b does carry a real presupposition. We get:

$$[\![b]\!] \quad = \quad \langle m \wedge a, m \wedge a \wedge u\rangle$$

For $\perp$ the following interpretation is available: $\perp$ is a proposition that is false *in all situations*. So it should not carry a presupposition and has as an assertion $\perp \in \mathcal{A}$, the falsum of the base algebra.

$$[\![\perp]\!] \quad = \quad \langle 1, \perp\rangle$$

Note that $\perp$ is *not* the bottom of $\mathcal{U}(\wp(W))$. Next we turn to the interpretation of the connectives. We use $\cdot$ for natural language concatenation and conjunction. So $\cdot$ stands for the combination of information in time. This is exactly what the operation $\bullet$ in $\mathcal{U}(\mathcal{A})$ should represent. So we set:

$$[\![\phi \cdot \psi]\!] \quad = \quad [\![\phi]\!] \bullet [\![\psi]\!]$$

Recall that on Boolean pairs $\bullet$ behaves as follows:

$$\langle a', a\rangle \bullet \langle b', b\rangle \quad = \quad \langle a' \wedge (a \to b'), a \wedge b\rangle$$

So if we concatenate ϕ and ψ with interpretation $\langle a', a\rangle$ and $\langle b', b\rangle$ respectively, the result is predicted to have presupposition $a' \wedge (a \to b')$. This means that the presupposition of ϕ carries over to $\phi \cdot \psi$, while the presupposition of ψ is weighed against the assertion of ϕ first: the result, $a \to b'$, is then added to the presupposition of $\phi \cdot \psi$. For the first two examples this means that we predict:

$$\begin{array}{rcl} [\![a \cdot s]\!] & = & \langle 1, a \wedge s\rangle \\ [\![m \cdot b]\!] & = & \langle (m \to a), m \wedge a \wedge u\rangle \end{array}$$

We see in the second example that b occurs in a context where part of the presupposition of b is given. By the operation $\bullet$ on the pairs this part disappears from the presupposition component, as required.

[19] It corresponds directly to only having pairs $\langle a', a\rangle$ where $a \leq a'$. This move is just a matter of choosing a uniquely determined representative from an equivalence class. (Cf. section 2.5.)

[20] Note the overloading of the notation!

[21] Recall that $1 \in \mathcal{A}$ co-incides with $\top \in \mathcal{A}$, as $\mathcal{A}$ is a Boolean algebra!

Next we turn to $\Rightarrow$, the connective corresponding to natural language implication. Above we have argued that in a residuation lattice $(\alpha \bullet \beta \leftarrow \alpha)$ is a promising candidate for the representation of natural language implication in residuation lattices. So here we set:

$$[\![(\phi \Rightarrow \psi)]\!] \quad = \quad [\![\phi]\!] \bullet [\![\psi]\!] \leftarrow [\![\phi]\!]$$

From now on we use $(\alpha \Rightarrow \beta)$ as shorthand for $(\alpha \bullet \beta \leftarrow \alpha)$. Let's look at the consequences for our examples:

$$\begin{array}{lcl} [\![(a \Rightarrow b)]\!] & = & \langle (a \to m), (a \to m \wedge u) \rangle \\ [\![(b \Rightarrow s)]\!] & = & \langle m \wedge a, m \wedge a \wedge (u \to s) \rangle \\ [\![(b \Rightarrow u)]\!] & = & \langle m \wedge a, m \wedge a \rangle \end{array}$$

Again we can compare the projection behaviour that we obtain with our intuitions. For example, in (if John is an adult, then he is a bachelor), we predict the required weakening of the presupposition of b. Looking at the assertion sides of the examples, we have to take into account that we assume $a \leq a'$ for any $\langle a', a \rangle$. We discuss some other connectives, $\cup$ and $\sim$, later, when we compare our results with the presupposition-as-precondition approach of Van Eijck [2].

Presupposition production Above we assigned presuppositions to atomic propositions directly. We simply set $[\![b]\!] = \langle m \wedge a, m \wedge a \wedge u \rangle$. Surely, this is the obvious way of proceeding: b is an atomic proposition, so we may assume direct access to its meaning. There is, however, the lingering feeling that ordinary Boolean propositions are in a sense more fundamental. We want our propositions under presuppositions to be defined from ordinary propositions. Fortunately our framework is rich enough to implement this intuition. This insight derives from the following fact.

Fact 3.1
Let $c, c' \in \mathcal{A}$ be given (for some Boolean algebra $\mathcal{A}$). Then

$$\langle c', c \wedge c' \rangle = (\langle 1, c' \rangle \to \langle 1, c \rangle) \qquad \square$$

Thus, we can 'add' the presupposition $c' \in \mathcal{A}$ to the assertion $c \in \mathcal{A}$ by the residuation operation $\to$. It is reasonable to require $c \leq c'$. But if this requirement is not satisfied, our definition of $\to$ sets things right automatically, since intersection with c' is built-in for the second component.

We may conclude that $\to$ can be used as a presupposition operator. Suggestions for the use of such operators can also be found in the literature: for example in Beaver [1] or Kracht [12]. In our setting the algebraic framework *provides* such an operator 'spontaneously'.

To satisfy the intuition that ordinary Boolean propositions are primary, we may now set up things as follows. We enrich our language $\mathcal{L}$ with the connective $\to$. We interpret all propositional atoms by pure Boolean propositions, i.e. all interpretations will be of the form $\langle 1, c \rangle$, where $c \in \mathcal{A}$. Propositions under presuppositions are to be defined using $\to$. For example: $b := (m \cdot a \to u)$, will be a definition introducing the *abbreviation* b.

Note that $\top_{\mathcal{U}(\mathcal{A})}$ is $\langle \bot_{\mathcal{A}}, \bot_{\mathcal{A}} \rangle$. It can be defined in the extended language by $(\bot \to \bot)$. Similarly $0_{\mathcal{U}(\mathcal{A})}$ can be defined by $((\bot \to \bot) \to \bot)$.

If we do not add $\to$ but e.g. $\leftarrow$ to our language, we cannot make the transition from purely Boolean propositions to propositions under presuppositions. This is immediate from fact 2.6: we simply cannot leave the non-zero states using only $\leftarrow$. We will see in subsection 4.1 that the combination of $\leftarrow$ and a symbol for $\top_{\mathcal{U}(\mathcal{A})}$ *does* suffice to introduce that transition!

3.2 Presuppositions as Preconditions

Above we have applied the pair construction on residuation lattices to analyse Boolean presuppositions. In the literature on dynamic semantics we find another popular formal model of presupposition behaviour: we can treat presuppositions as preconditions of programs. This idea was suggested already in Van Benthem [15]. Here we consider the way it is worked out for the case of Boolean presuppositions by Van Eijck in [2].[22]

The starting point for the model of presuppositions-as-programs is the popular dynamic analogy between the interpretation of natural language and the execution of programs. Starting from a certain initial state of information the execution of a program leads to an output state. In a similar way the interpretation of a natural language expression in a certain state of information will lead the interpreter to a new state of information. This analogy leads to formal models of natural language interpretation in which the representation language is a programming language to which an operational semantics is assigned.

In operational semantics the notion of *(weakest) preconditions* plays an important role. The preconditions of some program are the conditions that an input state has to satisfy to enable the program to be executed. This is closely analogous to the concept of a presupposition of a natural language expression: presuppositions are the conditions that are required for the evaluation of an expression. This is the basic idea of the analysis of presuppositions as preconditions.

So when the preconditions of an expression are met, we can evaluate the expression. But of course this evaluation still can have two results: the expression may be true or it may be false. This way we get a three way distinction on the information states:

- states in which the presuppositions are not met
- states in which the expression is true
- states in which the expression is false

[22] [2] already points out that for *Boolean* presuppositions his predictions agree with Karttunen and Peters [11].

Van Eijck [2] introduces such a three way distinction in the operational semantics of **PUL**, the **P**artial **U**pdate **L**ogic that he uses for the analysis of Boolean presuppositions.

$$\textbf{PUL}: \quad \pi \ ::= \ p \mid \bot \mid \pi;\pi \mid \pi \Rightarrow \pi$$

The intended interpretation of these programs is roughly as follows. p ranges over a set of propositional variables, P say. The program p simply checks whether the current information state supports p. $\bot$ is the program that always fails. The intended interpretation of the connectives ; and $\Rightarrow$ is as usual: $\pi;\pi'$ stands for concatenation of programs π and π'. $\pi \Rightarrow \pi'$ is the dynamic implication of the programs π and π'. This is a *test* that checks whether each successful run of π can be continued with a successful run of π'.

In the operational semantics for this language an element of *partiality* is introduced: we start by assigning, to each $p \in P$, two information states: $[p]^+$ and $[p]^-$. $[p]^+$ is the largest information state that *supports* or *asserts* the information p. Dually $[p]^-$ is the largest information state that *rejects* or *denies* the information p. In a total semantics it is taken for granted that a state rejects p iff it does not support p, but here things are different. We will assume that $[p]^+ \wedge [p]^- = \bot$ (consistency), but it is not taken for granted that $[p]^+ \vee [p]^- = \top$. This element of partiality allows Van Eijck to represent the behaviour of presuppositions. For an atomic test p we know that we have to be in state $[p]^+$ to be able to safely assert p. When we have at least information $[p]^-$, then we can safely deny p. Hence if we have at least the information that $[p]^+ \vee [p]^-$, we can be sure that we can evaluate p. In this way we get for each atomic test p the following three information states:

$$\begin{aligned}\mathsf{ass}(p) &= [p]^+, \text{ the } \textit{assertion} \text{ of } p\\ \mathsf{den}(p) &= [p]^-, \text{ the } \textit{denial} \text{ of } p\\ \mathsf{pre}(p) &= \mathsf{ass}(p) \vee \mathsf{den}(p), \text{ the } \textit{presupposition} \text{ of } p\end{aligned}$$

A suitable generalisation of these concepts to arbitrary programs produces the required analyses of presuppositions of complex programs. This generalisation is obtained by the following partial version of operational semantics for the programs in **PUL**:[23]

Definition 3.2 Let a Boolean algebra $\mathcal{A}$ be given. Assume that for each $p \in P$, $[p]^+$, $[p]^- \in \mathcal{A}$ are given such that $[p]^+ \wedge [p]^- = \bot$. Then we assign to each

[23]In this presentation of Van Eijck [2] we make some harmless changes in the details. For example, in [2] $\neg\pi$ is the basic connective and $\Rightarrow$ and $\bot$ are abbreviations. Also [2] only considers the case where $\mathcal{A}$ is the power set of some set W of possible worlds.

program $\pi \in \mathbf{PUL}$ mappings $[\pi]^+,\ [\pi]^- \in \mathcal{A}^\mathcal{A}$ as follows:

$$\begin{array}{rcl}
a[p]^+ & = & a \wedge [p]^+ \\
a[p]^- & = & a \wedge [p]^- \\
a[\bot]^+ & = & \bot \\
a[\bot]^- & = & \top \\
a[\pi;\pi']^+ & = & (a[\pi]^+)[\pi']^+ \\
a[\pi;\pi']^- & = & a[\pi]^- \ \cup\ (a[\pi]^+)[\pi']^- \\
a[\pi \Rightarrow \pi']^+ & = & a[\pi]^- \ \cup\ (a[\pi]^+)[\pi']^+ \\
a[\pi \Rightarrow \pi']^- & = & (a[\pi]^+)[\pi']^-
\end{array}$$

$\Box$

Here $\mathcal{A}$ is the algebra of information states. In the operational semantics we assign to each input state $a \in \mathcal{A}$ an output state $a[\pi]^+$, as usual. In addition we also define $a[\pi]^-$. This can be understood in two ways. First we can see $a[\pi]^-$ simply as an auxiliary notion that we need as a consequence of the partiality in the semantics. But we can also make intuitive sense of $a[\pi]^-$. $[p]^-$ is the largest —i.e. least informative— state where p is rejected. Then $a[p]^-$ is the largest state below a where p is rejected. This way the idea of denial (and assertion) is relativised to any input state $a \in \mathcal{A}$. Now the next step is the generalisation to arbitrary programs $\pi \in \mathbf{PUL}$: we get $a[\pi]^-$ as the generalised concept of denial for arbitrary programs.[24] So now we have the following generalisation of the three concepts discussed above.

Definition 3.3 Let $a \in \mathcal{A}$ and $\pi \in \mathbf{PUL}$ be given. Then:

- $\mathsf{ass}(a,\pi)\ =\ a[\pi]^+$
- $\mathsf{den}(a,\pi)\ =\ a[\pi]^-$
- $\mathsf{pre}(a,\pi)\ =\ \mathsf{ass}(a,\pi) \vee \mathsf{den}(a,\pi)$

$\Box$

It is easy to see that for an atomic test p we simply get $\mathsf{ass}(\top,p) = [p]^+$, $\mathsf{den}(\top,p) = [p]^-$ and $\mathsf{pre}(\top,p) = [p]^+ \vee [p]^-$. So we get what we want for atomic tests.

Actually it is mainly the case where $a = \top$ that we are interested in. But the extra parameter $a \in \mathcal{A}$ is used in the computation of the presupposition behaviour of complex programs. It can be shown that:

Fact 3.4 Let $\pi \in \mathbf{PUL}$, $a \in \mathcal{A}$ be given. Then:

- $\mathsf{ass}(a,\pi)\ =\ a \wedge \mathsf{ass}(\top,\pi)$
- $\mathsf{den}(a,\pi)\ =\ a \wedge \mathsf{den}(\top,\pi)$

$\Box$

[24]To analyse this partial update logic Van Eijck follows the Hoare-logic approach: an *assertion language* **APUL** is introduced to describe the input-output behaviour of the programs in **PUL**. We will not discuss the details of this formalisation. Instead we make a shortcut to get quickly to the resulting analysis of presupposition.

Now we can compute the presupposition and the assertion of programs of **PUL**. We restrict ourselves to the case where $a = \top$ and suppress $\top$ in the notation. The result is:

program π	presupposition $\mathsf{pre}(\pi)$	assertion $\mathsf{ass}(\pi)$
p	$[p]^+ \vee [p]^-$	$[p]^+$
$\bot$	$\top$	$\bot$
$\pi;\pi'$	$\mathsf{pre}(\pi) \wedge (\mathsf{ass}(\pi) \rightarrow \mathsf{pre}(\pi'))$	$\mathsf{ass}(\pi) \wedge \mathsf{ass}(\pi')$
$\pi \Rightarrow \pi'$	$\mathsf{pre}(\pi) \wedge (\mathsf{ass}(\pi) \rightarrow \mathsf{pre}(\pi'))$	$\mathsf{den}(\pi) \vee (\mathsf{ass}(\pi) \wedge \mathsf{ass}(\pi'))$

This way Van Eijck [2] obtains predictions about the presupposition behaviour of complex programs.

We can compare these results with our own predictions. Recall that our predictions are a direct consequence of the principled choice for residuation lattices in the analysis of presuppositions. For the comparison we translate each program $\pi \in$ **PUL** into a formula ϕ_π of $\mathcal{L}$. Then we can compare $[\![\phi_\pi]\!] \in \mathcal{U}(\mathcal{A})$ and $\mathsf{pre}(\pi)$ and $\mathsf{ass}(\pi)$ as given above: hoping that $[\![\phi_\pi]\!] = \langle \mathsf{pre}(\pi), \mathsf{ass}(\pi)\rangle$.

There are obvious candidates for the translation of ; and $\Rightarrow$. We translate ; by $\cdot$ as both connectives stand for combination *in time*. And also the translation of $\Rightarrow$ by $\Rightarrow$ is an obvious choice. So the translation is given in the following schema:

π	ϕ_π
p	p
$\bot$	$\bot$
$\pi;\pi'$	$\phi_\pi \cdot \phi_{\pi'}$
$(\pi \Rightarrow \pi')$	$(\phi_\pi \Rightarrow \phi_{\pi'})$

It is straightforward to check that this indeed produces the required goods: assuming that $[\![\phi_{\pi_i}]\!] = \langle \mathsf{pre}(\pi_i), \mathsf{ass}(\pi_i)\rangle$ (for $i = 1, 2$), we get:

$$\begin{aligned} [\![\phi_{\pi_1} \cdot \phi_{\pi_2}]\!] &= \langle \mathsf{pre}(\pi_1;\pi_2), \mathsf{ass}(\pi_1;\pi_2)\rangle \\ [\![(\phi_{\pi_1} \Rightarrow \phi_{\pi_2})]\!] &= \langle \mathsf{pre}(\pi_1 \Rightarrow \pi_2), \mathsf{ass}(\pi_1 \Rightarrow \pi_2)\rangle \end{aligned}$$

We conclude that in the Boolean case the choice for residuation lattices of pairs in the analysis of presuppositions produces the same results as Van Eijck's [2] presuppositions-as-preconditions approach. However: the arguments for the use of residuation lattices of pairs were quite general and we can hope for generalisations and extensions along these lines. Van Eijck's result were more ad hoc: they were produced by a careful implementation of intuitions about the meaning of specific program constructs. We could say that we have obtained a *rational reconstruction* of his results.

Excursion: other connectives Van Eijck [2] discusses several extensions of **PUL** with other connectives. He introduces *strong negation* $\widetilde{\pi}$, *choice* $\cup$ and *disjunction* $\sqcup$ of programs to capture several examples of presupposition behaviour in natural language.[25] The relevant presuppositional properties of these additional connectives are as follows:[26]

program	presupposition	assertion
$\widetilde{\pi}$	$\top$	$\neg\mathsf{ass}(\pi)$
$\pi \cup \pi'$	$(\mathsf{ass}(\pi') \vee \mathsf{pre}(\pi)) \wedge$ $(\mathsf{ass}(\pi) \vee \mathsf{pre}(\pi'))$	$\mathsf{ass}(\pi) \vee \mathsf{ass}(\pi')$
$\pi \sqcup \pi'$	$\mathsf{pre}(\pi) \wedge (\neg\mathsf{ass}(\pi) \to \mathsf{pre}(\pi'))$	$\mathsf{ass}(\pi) \vee (\mathsf{den}(\pi) \wedge \mathsf{ass}(\pi'))$

We see that the strong negation of π can always be processed: it has presupposition $\top$. The assertion of $\widetilde{\pi}$ is the negation of the assertion of π. It is sometimes argued that there are cases of negation in natural language that display this behaviour. Here (it is not true that the king of France is bald) could count as an example. Such an explicit negation could easily be followed by, for example: (there simply is no king of France) indicating that the first sentence can be uttered whether France has a king or not. Van Eijck's strong negation approximates this effect: $\widetilde{\pi}$ has no presupposition.

The behaviour of natural language disjunction is a notoriously hard problem both for dynamic semantics in general and for the semantics of presuppositions in general. It is no surprise that the problem is particularly hard for a dynamic treatment of presuppositions. Several proposals have been made in dynamic semantics for the formal representation of natural language disjunction. Van Eijck [2] discusses two such proposals: $\cup$ and $\sqcup$. $\pi \cup \pi'$ stand for the choice of programs: when running $\pi \cup \pi'$ the processor may choose whether it wants to run π or π'. This construction is not sensitive to the order of the programs in $\pi \cup \pi'$. It is often argued that order sensitivity is an essential property of natural language disjunction. As evidence we find for example the sentence:

Either the chicken has escaped or it is in a funny place.

The occurrence of it in the second disjunct depends on the first disjunct for its interpretation: it has the chicken as an antecedent. This is why the construction $\pi \sqcup \pi'$ is proposed as a more promising candidate for the representation of natural language 'or'. We see that $\pi \sqcup \pi'$ is order sensitive: it first checks for the presupposition of π. Then it tries to assert $\mathsf{ass}(\pi)$. Only if this fails the program π' is activated. We see in the table above that only if $\mathsf{ass}(\pi)$ does not hold we check for the presupposition of π' and then process the assertion of π'.

Van Eijck [2] already notices that $\pi \sqcup \pi'$ can be defined as $\pi \cup ((\pi \Rightarrow \bot); \pi')$. So we can also regard $\sqcup$ as an abbreviation for this complex expression containing $\cup$.

[25]We have no room here to summarise the discussion on the behaviour of presuppositions under negation and disjunction. Consult Van Eijck [2] and Beaver [1] for more details. Below we just give one or two examples to indicate the problems involved.

[26]For completeness we also give the semantic clauses for $\cup$ and $\sim$: $a[\pi \cup \pi']^+ = (a[\pi]^+ \cup a[\pi']^+)$, $a[\pi \cup \pi']^- = (a[\pi]^- \cap a[\pi']^-)$, $a[\widetilde{\pi}]^+ = a \cap \neg(a[\pi]^+)$, $a[\widetilde{\pi}]^- = a[\pi]^+$. The abbreviation given below will produce the operational semantics of $\sqcup$.

The language $\mathcal{L}$ is not rich enough to express these additional connectives.[27] So we will have to extend $\mathcal{L}$ if we want to capture the proposals of Van Eijck [2] for the representation of natural language negation and disjunction. Of course we have a strong preference for connectives that fall within the paradigm of residuation lattices: if residuation lattices are indeed the right paradigm for the study of natural language presuppositions, then the connectives of residuation lattices should suffice in the representation language $\mathcal{L}$.

Fortunately, the residuations indeed do the trick. If we extend $\mathcal{L}$ with the other residuation lattice connectives we can write:

$$
\begin{aligned}
&\overset{\sim}{\phi} = \ (\phi \to (\phi \leftarrow \top)) \wedge 1 \\
&\phi \cup \phi' \ = \ (\phi \vee \psi) \wedge (\phi \to \phi) \wedge (\psi \to \psi)
\end{aligned}
$$

Here we use the connectives $\to$, $\leftarrow$, $\wedge$, $\vee$ and the constants $\top$ and 1. This seems to be a bit much. It is only natural to wonder how many of the connectives of a residuation lattice we really need for the treatment of the simple examples that we have been looking at so far. We will see in subsection 4.1 that we can in fact do all the constructions that we have seen so far using only $\bot$, $\top$, $\leftarrow$ and $\bullet$.

Note that Van Eijck first introduces $\cup$ to the language and then goes on to discuss how $\sqcup$ is probably a better representation of natural language disjunction. We have seen now that we can represent $\cup$ in the full language of residuation lattices using a rather complicated expression. But if it is really $\sqcup$ that we are after, we can do much better. We find that:

$$\phi_{\pi \sqcup \pi'} \ = \ \phi_\pi \vee (\overset{\sim}{\phi_\pi} \cdot \phi_{\pi'})$$

also works.

3.3 Three Valued Logics

In this subsection we discuss connections with three valued logic. We will show how one can see our system as a three valued logic. Then we will see that our dynamic implication $\Rightarrow$ corresponds to a proposal by Kracht [12] for a treatment of presuppositions in three valued logics.

In order to read our proposal as a three valued logic, we first have to provide the truth values. Starting from the Boolean truth values, we can construct three suitable pairs: $\langle \bot, \bot \rangle$, $\langle 1, \bot \rangle$ and $\langle 1, 1 \rangle$. We can see these as truth values: $\langle 1, \bot \rangle$ gives an update function which is always defined and has the assertion $\bot$. It is the natural candidate for falsum in our setting. We denote it by $\overline{0}$. Similarly $\langle 1, 1 \rangle$ is the natural candidate for verum. We denote it by $\overline{1}$. $\langle \bot, \bot \rangle$ then is the third 'truth value'. It has the property that it both presupposes and asserts everything. So the corresponding update function is never defined, but if it *were*

[27] A quick argument involving some of the abbreviations of the next subsection: set $\overline{2} \sqsubset \overline{0}$ and $\overline{2} \sqsubset \overline{1}$, the definedness ordering on the truth values. The connectives of $\mathcal{L}$ are monotonous along the definedness ordering. Hence all expressions of $\mathcal{L}$ are monotonous along the definedness ordering. But we find $\overset{\sim}{\overline{2}} = \overline{1} \not\sqsubseteq \overset{\sim}{\overline{1}} = \overline{0}$.

defined it would be extremely informative. We denote it by $\overline{2}$. Note that in the ordering on the pairs $\overline{0} \leq \overline{1} \leq \overline{2}$.

In section 4 we will discuss in more detail how this idea of a correspondence with three valued logic works out formally: we will give a three valued representation theorem and prove a functional completeness theorem. Here we give the truth tables for the most interesting connectives.

$\Rightarrow$	$\overline{0}$	$\overline{1}$	$\overline{2}$
$\overline{0}$	$\overline{1}$	$\overline{1}$	$\overline{1}$
$\overline{1}$	$\overline{0}$	$\overline{1}$	$\overline{2}$
$\overline{2}$	$\overline{2}$	$\overline{2}$	$\overline{2}$

$\cdot$	$\overline{0}$	$\overline{1}$	$\overline{2}$
$\overline{0}$	$\overline{0}$	$\overline{0}$	$\overline{0}$
$\overline{1}$	$\overline{0}$	$\overline{1}$	$\overline{2}$
$\overline{2}$	$\overline{2}$	$\overline{2}$	$\overline{2}$

These connectives are not new in the literature on three valued logic:[28] they correspond to the connectives $\overset{\triangleright}{\to}$ and $\overset{\triangleright}{\wedge}$ of Kracht. He arrives at these connectives by informal considerations regarding the *direction* and the *economy* of computation: the $\triangleright$ indicates that the intended direction of computation is from left to right. By the principle of economy of computation, Kracht assigns a truth value as soon as the left-to-right computation allows us to predict the outcome of the procedure as a whole.

The fact that we have produced a familiar three valued logic is encouraging. The systematic and general considerations of Visser [17] agree with the specific considerations of Kracht about directed connectives. So we can hope that also when we apply the techniques of [17] in more general situations the results will keep making sense.

4 Results about the System

An important question for the algebraic approach is: which signature should our algebras have? Here we have chosen for residuation lattices. Residuation lattices have a *synchronic* component —a lattice— and a *diachronic* component —a monoid— that are related by the residuations. We have argued informally for the presence of both a synchronic and a diachronic component in the analysis of information flow. But are the residuation lattices really the proper way to substantiate these intuitions? Similarly we have to test the implementation of the ideas concerning *partiality* that we have discussed above.

It is our opinion that the answer to such questions should consist of further philosophical considerations (as in section 2), test applications (as in section 3) and —last but not least— a careful investigation of the formal properties of the system. In this section we discuss four results about the formal properties of the system that we have seen in section 3. There we used a simple Boolean algebra

[28] We already pointed out that in the literature on presupposition our Boolean system corresponds to Karttunen and Peters [11] approach. For details on the connection with Kracht consider p.95 of [12]. Kracht traces the origin of $\overset{\triangleright}{\to}$ back to Hayes [6]. Kracht's truth values correspond with our truth values as follows: T= $\overline{1}$, F= $\overline{0}$, U= $\overline{2}$. Note that in Kracht's ordering relation, the definedness ordering, T and F are incomparable, where we obtain $\overline{0} < \overline{1} < \overline{2}$.

to construct an algebra of pairs in the way discussed in section 2. The resulting algebra was used for the interpretation of a simple propositional language with presuppositions. The results that we will prove are:

1. expressive completeness
2. a representation theorem for Boolean pair algebras
3. functional completeness
4. a decision method for the valid equations of Boolean pair algebras

These results are presented in the subsections below. Each subsection starts with a statement of the result of that subsection. This makes it possible for the impatient reader to skip the details of the proofs at first reading.

We end the section with some remarks and questions about the axiomatisation of valid equations and sequents for Boolean pair algebras.

4.1 Expressive Completeness

In this subsection we prove an expressive completeness result. The question that we consider is as follows: in section 3 we have assumed that the notion of information content is represented by a Boolean algebra $\mathcal{A}$: the elements of $\mathcal{A}$ are the things we wish to talk 'about'. Then we went on to construct the pair algebra over this Boolean algebra to capture the idea of *negative* information. In principle there is no restriction on the way positive (or negative) information can be combined: if a and b both occur as positive (or negative) information, then also the Boolean combinations —$a \wedge b$, $a \vee b$, $(a \to b)$,...— can be expected. So we would like to be able to express all these Boolean combinations of the components. This is an expressivity requirement. Here we will discuss how the requirement can be met. To be able to present our results in a nice way, we first introduce some standard operations on products and disjoint unions.[29]

Definition 4.1 Let A_1, A_2, B, C, D_1, D_2 be sets. Consider the Cartesian product $A_1 \times A_2$. π_1, π_2 are the usual projection functions on $A_1 \times A_2$:

- $\pi_i(a_1, a_2) = a_i$

Suppose $f_i : B \to A_i$. Then $(f_1, f_2) : B \to (A_1 \times A_2)$ is the function with

- $(f_1, f_2)(b) := \langle f_1(b), f_2(b) \rangle$

We write $\circ$ for composition of functions, reading $g \circ h$ as: first h, then g. Then:

- $f_i = \pi_i \circ (f_1, f_2)$

The disjoint union $A_1 \oplus A_2$ of A_1 and A_2 is the set $(\{1\} \times A_1) \cup (\{2\} \times A_2)$. ι_1 and ι_2 are the usual embedding functions to $A_1 \oplus A_2$:

[29]Also the notation conventions from subsection 2.6 will be used abundantly.

- $\iota_i(a) := \langle i, a\rangle$

Suppose $g_i : A_i \to C$. Then $[g_1, g_2] : A_1 \oplus A_2 \to C$ is the function with:

- $[g_1, g_2](\langle i, a\rangle) := g_i(a)$

We have: $g_i = [g_1, g_2] \circ \iota_i$. Finally, if $h_i : A_i \to D_i$, then $h_1 \times h_2 : A_1 \times A_2 \to D_1 \times D_2$ is the function with:

- $h_1 \times h_2(\langle a_1, a_2\rangle) = \langle h_1(a_1), h_2(a_2)\rangle$ $\Box$

Let's fix a Boolean algebra $\mathcal{A}$. We will not work in the full structure $\mathcal{U}(\mathcal{A})$, but in the reduced structure $\mathcal{U}^{o}_{[1,\top,\bullet,\leftarrow]}(\mathcal{A})$, where we leave out 0 and reduce the signature to 1, $\top$, $\bullet$ and $\leftarrow$. We will be working with $[1, \top, \bullet, \leftarrow]$ for most of this subsection, so it will be convenient to write $\mathcal{W}(\mathcal{A})$ for $\mathcal{U}^{o}_{[1,\top,\bullet,\leftarrow]}(\mathcal{A})$ here. Let a set of propositional variables P be given. p will range over P. Define $\mathcal{T}$ as follows:

$$\mathcal{T} : \qquad \phi \ ::= \ p \mid 1 \mid \top \mid \phi \cdot \phi \mid (\phi \leftarrow \phi).$$

An assignment σ is a function from P to the elements of $\mathcal{W}(\mathcal{A})$. We interpret the formulas of $\mathcal{T}$ in $\mathcal{W}(\mathcal{A})$ as follows:

- $[\![p]\!]\,\sigma := \sigma(p)$, $[\![1]\!]\,\sigma := \langle 1, 1\rangle$, $[\![\top]\!]\,\sigma := \langle 0, 0\rangle$,
- $[\![\phi \cdot \psi]\!]\,\sigma := [\![\phi]\!]\,\sigma \bullet [\![\psi]\!]\,\sigma$,
- $[\![(\phi \leftarrow \psi)]\!]\,\sigma := ([\![\phi]\!]\,\sigma \leftarrow [\![\psi]\!]\,\sigma)$.

We consider $[\![\phi]\!]$ as a function from assignments to the elements of $\mathcal{W}(\mathcal{A})$. We can interpret the language $\mathcal{L}$ into $\mathcal{T}$ via, say, $(.)^*$ as follows:

- $p^* := p$, $\bot^* := (1 \leftarrow \top)$, $\top^* := 1$,
- $(\phi \cdot \psi)^* := \phi^* \cdot \psi^*$,
- $(\phi \Rightarrow \psi)^* := (\phi^* \cdot \psi^* \leftarrow \phi^*)$

In the notation system provided by $\mathcal{T}$ we stay close to the notation in residuation algebras: we use 1 for the unit of the $\bullet$-operation and reserve $\top$ for the top element of the pair algebra. Our choice of connectives and notation in $\mathcal{L}$ was based on natural language considerations. The translation shows how the two languages are connected: this way $\mathcal{T}$ can be considered as an extension of $\mathcal{L}$.

We will also use a language $\mathcal{V}$ to speak about the original structure $\mathcal{A}$. $\mathcal{V}$ will be the standard language for ordinary propositional logic with, as set of propositional atoms, $P^+ := P \oplus P$ and with connectives $\top$, $\bot$, $\wedge$ and $\to$. We will write p^i for $\langle i, p\rangle$. $\mathcal{V}$ is interpreted in the standard way into $\mathcal{A}$. Say this interpretation function is $[\![.]\!]_{\mathsf{b}} : \mathcal{V} \to \mathcal{A}$. Now we have three languages:

- $\mathcal{L}$, as introduced before. $\mathcal{L}$ is the language we use for representing ordinary reasoning with presuppositions. It contains the 'dynamic implication' $\Rightarrow$. An atomic formula $p \in \mathcal{L}$ has a presupposition and an assertion, but these remain implicit in the notation, just as in natural language.

- $\mathcal{V}$, the language of ordinary propositional logic, but we have doubled up the number of atomic propositions: for each $p \in \mathcal{P}$ there will be two atomic propositions p^1 and p^2 in $\mathcal{V}$, corresponding intuitively to the presupposition and assertion of $p \in \mathcal{L}$. $\mathcal{V}$ is interpreted in the Boolean algebra $\mathcal{A}$.
- $\mathcal{T}$, the language that is closest to the pair algebras. Both $\mathcal{T}$ and $\mathcal{L}$ are interpreted in the pair algebra $\mathcal{W}(\mathcal{A})$. The main difference is in the implications: we have $(\phi \Rightarrow \psi)$ in $\mathcal{L}$ corresponding to $(\phi \cdot \psi \leftarrow \phi)$ in $\mathcal{T}$.

We map assignments σ for $\mathcal{T}$ to assigments $\breve{\sigma}$ for $\mathcal{V}$ as follows: $\breve{\sigma}(p^i) := \pi_i(\sigma(p))$. Let's say that $\breve{\sigma} =: \Theta(\sigma)$.[30] Let us return for a moment to the specification of the meaning function for $\mathcal{T}$. E.g. consider the clause

- $[\![\phi \cdot \psi]\!]\,\sigma := [\![\phi]\!]\,\sigma \bullet [\![\psi]\!]\,\sigma$

When we spell this out, we get:

- Suppose $[\![\phi]\!]\,\sigma = \langle a', a\rangle$ and $[\![\psi]\!]\,\sigma = \langle b', b\rangle$. Then:
  $[\![\phi \cdot \psi]\!]\,\sigma := \langle a' \wedge (a \to b'), a \wedge b\rangle$.

The specification of $[\![\phi \cdot \psi]\!]\,\sigma$ employs Boolean terms in the *metalanguage*. We could have specified $[\![.]\!]$ in a different way, employing rather the Boolean terms of our *object language* $\mathcal{V}$: first translate $\mathcal{T}$-terms to *pairs of* $\mathcal{V}$*-terms* and then interpret the components of these pairs via $[\![.]\!]_{\mathsf{b}}$. Here is the way this works. We define the translation function $(.)^\pi : \mathcal{T} \to \mathcal{V} \times \mathcal{V}$ as follows.

- $p^\pi := \langle p^1, p^2\rangle$, $1^\pi := \langle \top, \top\rangle$, $\top^\pi := \langle \bot, \bot\rangle$,
- $(\phi \cdot \psi)^\pi := \langle \pi_1(\phi^\pi) \wedge (\pi_2(\phi^\pi) \to \pi_1(\psi^\pi)), (\pi_2(\phi^\pi) \wedge \pi_2(\psi^\pi))\rangle$,
- $(\phi \leftarrow \psi)^\pi := \langle \pi_1(\phi^\pi), \pi_1(\phi^\pi) \wedge \pi_1(\psi^\pi) \wedge (\pi_2(\phi^\pi) \to \pi_2(\psi^\pi))\rangle$.

Suppose $\phi^\pi = \langle \phi_1, \phi_2\rangle$. It is easy to see that we have:

$$[\![\phi]\!]\,\sigma = \langle\, [\![\phi_1]\!]_{\mathsf{b}}\breve{\sigma},\, [\![\phi_2]\!]_{\mathsf{b}}\breve{\sigma}\rangle.$$

In other words: $[\![\phi]\!] = (\,[\![\phi_1]\!]_{\mathsf{b}}, [\![\phi_2]\!]_{\mathsf{b}}) \circ \Theta$.[31]

[30] Θ is in fact the standard isomorphism between $(A \times A)^P$ and $A^{P \oplus P}$. We have: $\Theta(\sigma) = [\pi_1 \circ \sigma, \pi_2 \circ \sigma]$. It's inverse Θ^{-1} is given by $\Theta^{-1}(\tau) = (\tau \circ \iota_1, \tau \circ \iota_2)$. We have e.g.

$$\begin{aligned}([\pi_1 \circ \sigma, \pi_2 \circ \sigma] \circ \iota_1, [\pi_1 \circ \sigma, \pi_2 \circ \sigma] \circ \iota_2) &= (\pi_1 \circ \sigma, \pi_2 \circ \sigma)\\ &= \sigma\end{aligned}$$

[31] We can write this in an even more fancy way by considering $[\![.]\!]$ as a binary function taking formulas and assignments as arguments:

$$[\![\cdot]\!] = (\,[\![\cdot]\!]_{\mathsf{b}} \times [\![\cdot]\!]_{\mathsf{b}}) \circ (\pi_1 \times \mathsf{id}, \pi_2 \times \mathsf{id}) \circ ((.)^\pi \times \Theta).$$

Note that we will have: $[\![\phi_1]\!]_{\mathsf{b}}\breve{\sigma} \leq [\![\phi_2]\!]_{\mathsf{b}}\breve{\sigma}$. We are going to prove the converse of the above observation: consider *any* pair of $\mathcal{V}$-formulas ϕ_1, ϕ_2. We will produce a $\mathcal{T}$-term $\phi := T(\phi_1, \phi_2)$ such that

$$[\![\phi]\!]\,\sigma = (\,[\![\phi_1]\!]_{\mathsf{b}}\breve{\sigma},\, [\![\phi_1]\!]_{\mathsf{b}}\breve{\sigma} \wedge [\![\phi_2]\!]_{\mathsf{b}}\breve{\sigma})$$

Note that in case $[\![\phi_1]\!]_{\mathsf{b}}\breve{\sigma} \leq [\![\phi_2]\!]_{\mathsf{b}}\breve{\sigma}$, this reduces to

$$[\![\phi]\!] = (\,[\![\phi_1]\!]_{\mathsf{b}},\, [\![\phi_2]\!]_{\mathsf{b}}) \circ \Theta.$$

Thus, for suitable pairs, T will be the inverse of $(.)^{\pi}$.

Summarising, if we make up a new assertion and a new presupposition as a Boolean combination of a old assertions and presuppositions, then there will be a term in $\mathcal{T}$ that produces exactly this new assertion and this new presupposition. We will call this property: *expressive completeness*, or, in our specific case where the underlying algebra is Boolean, *Boolean completeness.*

We have already met the projection functions π_i. These functions bring us from elements of $\mathcal{W}(\mathcal{A})$ to elements of $\mathcal{A}$. It is quite convenient to 'lift' these functions to functions from $\mathcal{W}(\mathcal{A})$ to $\mathcal{W}(\mathcal{A})$. Thus we are led to the following definition.

Definition 4.2 We consider the following *projections.*

- $\mathfrak{p}_i(\langle a_1, a_2\rangle) = \langle 1, a_i\rangle$.

If we put $\mathsf{emb}(a) := \mathsf{emb}_{\mathcal{A}}(a) = \langle 1, a\rangle$, we can rewrite our definition as:
$\mathfrak{p}_i := \mathsf{emb} \circ \pi_i$. ▯

At this point it is a good idea to acquire some experience in computing in $\mathcal{W}(\mathcal{A})$.

1. As pointed out in fact 2.6, we have:

   (a) $\langle 1, a\rangle \bullet \langle 1, b\rangle = \langle 1, a \wedge b\rangle$,

   (b) $(\langle 1, b\rangle \leftarrow \langle 1, a\rangle) = \langle 1, (a \to b)\rangle$.

2. $(1 \leftarrow \top) = \langle 1, \bot\rangle$.

3. $(1 \leftarrow \langle a', a\rangle) = \langle 1, a'\rangle$. This means that $\mathfrak{p}_1$ is definable.

4. $\langle 1, a'\rangle \bullet \langle a', a\rangle = \langle 1, a\rangle$. I.o.w. $\mathfrak{p}_1(\xi) \bullet \xi = \mathfrak{p}_2(\xi)$. Since we already saw that $\mathfrak{p}_1$ is definable, this tells us that $\mathfrak{p}_2$ is definable.

5. $\langle 1, a\rangle \bullet \top = \langle \neg a, \bot\rangle$. Note that $(\langle 1, \bot\rangle \leftarrow \langle 1, a\rangle) = \langle 1, \neg a\rangle$. Ergo:
   $(\langle 1, \bot\rangle \leftarrow \langle 1, a\rangle) \bullet \top = \langle a, \bot\rangle$. So the mapping $\Downarrow : \langle 1, a\rangle \mapsto \langle a, \bot\rangle$ is definable.

6. $\langle a, \bot\rangle \leftarrow \langle b, \bot\rangle = \langle a, a \wedge b\rangle$.
   We see that:
   $\Downarrow\langle 1, a\rangle \leftarrow \Downarrow\langle 1, b\rangle = \langle a, a \wedge b\rangle$.

In this exercise in computing with pairs we show a few important tricks. So it really is worth your while to look at the 'exercise' in some detail. Next we are ready for some definitions. Let ϕ, ϕ' be $\mathcal{T}$-terms.

- $\tilde{\mathfrak{p}}_1(\phi) := (1 \leftarrow \phi)$. Note that $\tilde{\mathfrak{p}}_1$ maps syntax to syntax. We have: $[\![\tilde{\mathfrak{p}}_1(\phi)]\!]\,\sigma = \mathfrak{p}_1([\![\phi]\!]\,\sigma)$. In a more elegant notation, this tells us: $[\![\tilde{\mathfrak{p}}_1(\phi)]\!] = \mathfrak{p}_1 \circ [\![\phi]\!]$.
- $\tilde{\mathfrak{p}}_2(\phi) := \tilde{\mathfrak{p}}_1(\phi) \cdot \phi$. We find: $[\![\tilde{\mathfrak{p}}_2(\phi)]\!] = \mathfrak{p}_2 \circ [\![\phi]\!]$.
- Define $\tilde{\bot} := (1 \leftarrow \top)$. So $[\![\tilde{\bot}]\!]\,\sigma = \langle 1, \bot\rangle$.
- Define $\downarrow(\phi) := (\tilde{\bot} \leftarrow \phi) \cdot \top$.
  We find: $[\![\downarrow(\phi)]\!] = \Downarrow \circ\, [\![\phi]\!]$.

We translate $\mathcal{V}$ via, say, $(.)^\circ$ into $\mathcal{T}$ in the following way.

- $(p^1)^\circ := \tilde{\mathfrak{p}}_1(p)$
- $(p^2)^\circ := \tilde{\mathfrak{p}}_2(p)$
- $\bot^\circ := \tilde{\bot}$
- $(\phi \wedge \psi)^\circ := \phi^\circ \cdot \psi^\circ$
- $(\phi \to \psi)^\circ := (\psi^\circ \leftarrow \phi^\circ)$

Lemma 4.3 Let ψ be a term of $\mathcal{V}$. We have: $[\![\psi^\circ]\!]\,\sigma = \langle 1, [\![\psi]\!]_{\mathsf{b}}\breve{\sigma}\rangle$. In other words: $[\![\psi^\circ]\!] = \mathsf{emb} \circ [\![\psi]\!]_{\mathsf{b}} \circ \Theta$. ❑

The proof is a simple induction on ψ, using fact 2.6.

The lemma shows that for a Boolean combination of p^i's —let's call it ψ—, we can give a term $\psi^\circ \in \mathcal{T}$ such that the *assertion* of ψ° is exactly ψ. This is an important step towards proving expressive completeness: if we can do something similar for *presuppositions*, then we are done!

Fortunately we do not have to do all the work again for the presupposition side. We can simply use the same translation $(.)^\circ$ in combination with the operation $\downarrow$ that was introduced earlier. Consider two $\mathcal{V}$-terms ϕ_1 and ϕ_2. We have:

$$[\![(\downarrow(\phi_1^\circ) \leftarrow \downarrow(\phi_2^\circ))]\!]\,\sigma \quad = \quad \langle [\![\phi_1]\!]\,\breve{\sigma},\, [\![\phi_1]\!]\,\breve{\sigma} \wedge [\![\phi_2]\!]\,\breve{\sigma}\rangle$$

We may conclude that the formula $T(\phi_1, \phi_2)$ that we were looking for is:

$$T(\phi_1, \phi_2) := (\downarrow(\phi_1^\circ) \leftarrow \downarrow(\phi_2^\circ)).$$

This gives us our expressive completeness result.

Another way of looking at our result is as follows: we have shown that in the pair algebra over a Boolean algebra the Boolean connectives $\wedge$, $\vee$ can be introduced as abbreviations! For, the connectives on pairs are defined in terms of Boolean combinations of their presuppositions and assertions and here we have shown that such Boolean combinations can already be defined using the connective in $\mathcal{T}$. For $\rightarrow$ we have a slightly weaker result: $\rightarrow$ makes essential use of the artificial bottom element in the pair algebra. Therefore there is no hope to define $\rightarrow$ in $\mathcal{T}$. But we *can* define a function that coincides with $\rightarrow$ on the inputs where $\rightarrow$ does *not* give 0.

4.2 Three Valued Representation Theorem

For Boolean algebras we have the famous Stone representation theorem. This theorem says that any Boolean algebra is (isomorphic to) a subalgebra of a power set algebra. Or, equivalently, any Boolean algebra is a subalgebra of an algebra $\mathbf{2}^M$ for a suitable M. Here $\mathbf{2}$ is the Boolean algebra of truth values $\{\bot, \top\}$.[32]

For the pair algebras we obtain a similar result, but this time we have to use *three* truth values. Let's use the notation $\overline{2} := \langle \bot, \bot \rangle$, $\overline{1} := \langle 1, 1 \rangle$ and $\overline{0} := \langle 1, \bot \rangle$. Note that $\overline{0} < \overline{1} < \overline{2}$ and

$$\overline{i} \bullet \overline{j} = \begin{cases} \overline{0} & \text{if } i = 0 \\ \overline{j} & \text{if } i = 1 \\ \overline{2} & \text{if } i = 2 \end{cases}$$

I.o.w. $\overline{i} \bullet \overline{j} = \overline{\min(i \cdot (i + j - 1), 2)}$. We define $\mathbf{3} = \mathcal{U}^o(\mathbf{2})$. So $\mathbf{3}$ is the unique reduced residuation algebra on $\{\overline{0}, \overline{1}, \overline{2}\}$, with signature $[\vee, \wedge, \top, \bullet, 1, \leftarrow]$, where $\leq$ and $\bullet$ are as specified above.

Proposition 4.4 Let $\mathcal{A}$ be a Boolean algebra. Then $\mathcal{U}^o(\mathcal{A})$ is a subalgebra of $\mathbf{3}^M$ for a suitable M. ▯

Proof

We start by observing that $\mathcal{A}$ is a subalgebra of $\mathbf{2}^M$. This means that each $a \in \mathcal{A}$ can also be considered as a mapping $[a] \in \mathbf{2}^M$. Given any $\langle a', a \rangle \in U^o(\mathcal{A})$, we define $f_{\langle a', a \rangle} \in \mathbf{3}^M$ by

$$f_{\langle a', a \rangle}(m) = \langle [a'](m), [a](m) \rangle$$

Remember that we assume that $a \leq a'$, hence $\langle [a'](m), [a](m) \rangle$ will be indeed in $\mathbf{3}$. Now it is easy to check that everything commutes with the operators. For example:

$$\begin{aligned} & f_{\langle a', a \rangle \bullet \langle b', b \rangle}(m) \;=\; f_{\langle a' \wedge (a \rightarrow b'), a \wedge b \rangle}(m) = \\ & \langle [a' \wedge (a \rightarrow b')](m), [a \wedge b](m) \rangle = \\ & \langle \min([a'](m), \max(1 - [a](m), [b'](m))), \min([a](m), [b](m)) \rangle = \\ & \langle [a'](m), [a](m) \rangle \bullet \langle [b'](m), [b](m) \rangle \;=\; f_{\langle a', a \rangle}(m) \bullet f_{\langle b', b \rangle}(m) \end{aligned}$$

[32] In fact for any Boolean algebra $\mathcal{A}$ we can use $M = \mathsf{Maxfil}(\mathcal{A})$, the set of maximal filters of $\mathcal{A}$.

and

$$\begin{array}{l} f_{\langle a',a\rangle \leftarrow \langle b',b\rangle}(m) \ = \ f_{\langle a', a' \wedge b' \wedge (b \to a)\rangle}(m) = \\ \langle [a'](m), [a' \wedge b' \wedge (b \to a)](m)\rangle = \\ \langle [a'](m), \min([a'](m), [b'](m), \max(1-[b](m),[a](m)))\rangle = \\ f_{\langle a',a\rangle}(m) \leftarrow f_{\langle b',b\rangle}(m) \end{array}$$

and

$$f_{\overline{0}}(m) \ = \ \langle [1](m), [\bot](m)\rangle \ = \ \langle 1, \bot\rangle \ = \ \overline{0}$$

So we really do obtain a sub*algebra*, as required. ❑

The representation theorem shows us that on Boolean algebras the pair construction simply gives a three valued logic. This shows again that in the simple case of Boolean algebras the pair construction makes sense. We already came to that conclusion in section 3, where we noticed that for simple examples we obtained 'the expected results'.

4.3 Functional Completeness

In this subsection we show functional completeness of our semantics in three valued logic. Let $\mathcal{T} := \mathcal{T}_{[1,\top,\bullet,\leftarrow]}(P)$, where $P = \{p_1,\ldots,p_m\}$. Each term $\tau \in \mathcal{L}$ defines a truth function $f_\tau \in \mathbf{3}^{(\mathbf{3}^P)}$, viz. $f_\tau = \lambda\sigma{\in}\mathbf{3}^P.\, [\![\tau]\!]\,\sigma$ Now it is only natural to wonder which $f \in \mathbf{3}^{(\mathbf{3}^P)}$ can be obtained as the truth function f_τ of some term τ. We will answer this question by proving *functional completeness*.

Proposition 4.5 [Functional completeness] Let $f : \mathbf{3}^P \to \mathbf{3}$ be given. We prove that there is a $\tau \in \mathcal{T}$ such that $[\![\tau]\!]\,\sigma \ = \ f(\sigma)$. ❑

We will give two different proofs of this proposition, providing rather different terms. The first proof uses our result of subsection 4.1.

Proof

("via Boolean Completeness") Let $\nu : P \oplus P \to \mathbf{2}$. We can send ν to $\Xi(\nu) := \sigma : P \to \mathbf{3}$, by setting: $\sigma(p) := \langle \nu(p^1), \nu(p^1) \wedge \nu(p^2)\rangle$. It is easy to see that if $\Xi(\Theta(\sigma)) = \sigma$, where Θ is as given in subsection 4.1. Moreover, if for all p, $\nu(p^2) \le \nu(p^1)$, then $\Theta(\Xi(\nu)) = \nu$.

Consider $f : \mathbf{3}^P \to \mathbf{3}$. Define $\Phi_i(f) : \mathbf{2}^{P\oplus P} \to \mathbf{2}$, as follows.

$$\Phi_i(f)(\nu) := \pi_i(f(\Xi(\nu))).$$

I.o.w. $\Phi_i(f) = \pi_i \circ f \circ \Xi$. Ordinary two valued functional completeness provides t_i that generate these mappings $\Phi_i(f)$, i.e., for all ν, $\Phi_i(f)(\nu) = [\![t_i]\!]_{\mathsf{b}}\nu$. We find that $f(\sigma) = \langle\, [\![t_1]\!]\,\Theta(\sigma),\, [\![t_2]\!]\,\Theta(\sigma)\rangle$, where Θ is defined as in subsection 4.1. We may apply the result of subsection 4.1 to obtain the desired term τ. ❑

So we see that functional completeness is a consequence of Boolean completeness. Still, it would be nice to have a proof that is more analogous to the usual one for ordinary propositional logic, generalising the use of disjunctive normal forms. Such a construction is given by the following proof.

Proof

("construction of Disjunctive Normal Forms") In this proof we will use the connectives $\wedge$ and $\vee$. We have already seen that these are definable in section 4.1.

Our technique for proving functional completeness is a generalisation of the idea of *disjunctive normal forms.* The τ_p that we introduce below generalise the idea of a *literal* (Step 1). Then conjunctions of literals describe *valuations* (Step 2). Disjunctions of such terms describe sets of valuations, i.e. *propositions* (Step 2).

Step 0: $n = 0$: obvious: $\overline{2}$ and $\overline{1}$ are constants of the language and $\overline{0}$ can be defined as $(\overline{1} \leftarrow \overline{2})$.

Step 1: $n = 1$. First we consider the basic functions.

p	$\tau_{\overline{0}} = (\overline{0} \leftarrow p)$	$\tau_{\overline{1}} = \tilde{\mathfrak{p}}_2(p)$	$\tau_{\overline{2}} = (\overline{0} \leftarrow \tilde{\mathfrak{p}}_1(p))$	$\tau_{\overline{0}} \cdot \overline{2}$	$\tau_{\overline{1}} \cdot \overline{2}$	$\tau_{\overline{2}} \cdot \overline{2}$
$\overline{0}$	$\overline{1}$	$\overline{0}$	$\overline{0}$	$\overline{2}$	$\overline{0}$	$\overline{0}$
$\overline{1}$	$\overline{0}$	$\overline{1}$	$\overline{0}$	$\overline{0}$	$\overline{2}$	$\overline{0}$
$\overline{2}$	$\overline{0}$	$\overline{0}$	$\overline{1}$	$\overline{0}$	$\overline{0}$	$\overline{2}$

This table gives us the basic truth functions in one variable. We can now use $\vee$ to get an arbitrary truth function in one variable.

Step 2: other n. Again we create the basic functions first. So let f and $\sigma \in \mathbf{3}^P$ be given such that $f(\sigma) = \overline{1}$ and $f(\nu) = \overline{0}$ for all other inputs ν. In step 1 we have provided $\tau_v(p)$ for all $v \in \mathbf{3}$. Then:

$$\tau_f \;=\; \tau_{\sigma(p_1)}(p_1) \wedge \ldots \wedge \tau_{\sigma(p_n)}(p_n)$$

works. Next we can produce g such that $g(\sigma) = \overline{2}$ and $g(\nu) = \overline{0}$ for all other inputs ν by stipulating: $\tau_g = \tau_f \cdot \overline{2}$. The other functions in $\mathbf{3}^{(\mathbf{3}^P)}$ can be obtained from the basic ones using $\vee$. ❑

4.4 A Decision Method

We present a decision method for the equational theory of Boolean pair algebras. We indicate how to decide whether an equation $\tau = \tau'$ in the language $\mathcal{T} := \mathcal{T}_{[1,\top,\bullet,\leftarrow]}(P)$ of Boolean pair algebras, $\mathcal{W}(\mathcal{A}) := \mathcal{U}^o_{[1,\top,\bullet,\leftarrow]}(\mathcal{A})$, is valid, i.e. whether it has the property that, for all Boolean algebras $\mathcal{A}$ and all $\sigma \in W(\mathcal{A})^P$,

$$[\![\tau]\!]_{\mathcal{W}(\mathcal{A})}\sigma \;=\; [\![\tau']\!]_{\mathcal{W}(\mathcal{A})}\sigma.$$

We consider only terms of signature $[1, \top, \bullet, \leftarrow]$. There are two ways to justify this choice: one is the fact that this set is Boolean complete. So all other connectives can be introduced as abbreviations. This is a fairly convincing, but purely technical argument: for example, it will no longer hold if our applications require more than Boolean algebras to start with. There also is a less technical reason for the choice: we regard these connectives as the dynamic kernel of a residuation algebra. Clearly $\bullet$ and 1 are truly dynamic connectives: they are about addition of information *in time*. The other two, $\leftarrow$ and $\top$, are not purely diachronic (for example we use $\leq$ to define $\leftarrow$) but it seems that they are a natural and indispensable choice when we want to give the formalism the sort of expressivity required for the dynamic analysis of natural language (cf. section 3).

We employ the translation $(\cdot)^\pi$ from subsection 4.1. Say $\tau^\pi = \langle \tau_1, \tau_2 \rangle$ and $\tau'^\pi = \langle \tau'_1, \tau'_2 \rangle$. Remember that the τ_i, τ'_i are in the language $\mathcal{V}$. The following theorem follows directly from our definitions.

Theorem 4.6 *$\tau = \tau'$ is valid iff* (†) $\{p^2 \to p^1 \mid p \in P\} \vdash (\tau_1 \leftrightarrow \tau'_1) \wedge (\tau_2 \leftrightarrow \tau'_2)$. *Here $\vdash$ is derivability in ordinary propositional logic.*

So it is sufficient to apply the familiar decision method for propositional logic to (†).

4.5 Some Remarks on Axiomatisation

In this subsection, we make a few remarks and we pose a few questions concerning axiomatisation. By our previous results the axioms of the following four groups are valid in all Boolean pair algebras.

pairing	τ	$=$	$(\downarrow(\tilde{\mathfrak{p}}_1(\tau)) \leftarrow \downarrow(\tilde{\mathfrak{p}}_2(\tau)))$
projection	$\tilde{\mathfrak{p}}_1(1)$	$=$	$1 = \tilde{\mathfrak{p}}_2(1)$
	$\tilde{\mathfrak{p}}_1(\top)$	$=$	$\tilde{\bot} = \tilde{\mathfrak{p}}_2(\top)$
	$\tilde{\mathfrak{p}}_1(\alpha \cdot \beta)$	$=$	$\tilde{\mathfrak{p}}_1(\alpha) \cdot (\tilde{\mathfrak{p}}_1(\beta) \leftarrow \tilde{\mathfrak{p}}_2(\alpha))$
	$\tilde{\mathfrak{p}}_2(\alpha \cdot \beta)$	$=$	$\tilde{\mathfrak{p}}_2(\alpha) \cdot \tilde{\mathfrak{p}}_2(\beta)$
	$\tilde{\mathfrak{p}}_1(\alpha \leftarrow \beta)$	$=$	$\tilde{\mathfrak{p}}_1(\alpha)$
	$\tilde{\mathfrak{p}}_2(\alpha \leftarrow \beta)$	$=$	$\tilde{\mathfrak{p}}_1(\alpha) \cdot \tilde{\mathfrak{p}}_1(\beta) \cdot (\tilde{\mathfrak{p}}_2(\alpha) \leftarrow \tilde{\mathfrak{p}}_2(\beta))$
boolean	Boolean algebra applies to expressions built up from $\tilde{\mathfrak{p}}_i(p)$		
normal form	$\tilde{\mathfrak{p}}_1(p) \cdot \tilde{\mathfrak{p}}_2(p)$	$=$	$\tilde{\mathfrak{p}}_2(p)$

When, in specifying the axiom 'boolean', we say *Boolean algebra applies ...* we mean with 1 in the role of $\top$, $\cdot$ in the role of $\wedge$, $\leftarrow$ in the role of $\to$.

Suppose $\tau = \tau'$ is valid. We employ the translation $(\cdot)^\circ$ of subsection 4.1. We use an auxiliary translation $(\cdot)^\rho$ defined just like $(\cdot)^\pi$ of subsection 4.1 with the exception that we translate p to $\langle p^1, p^1 \wedge p^2 \rangle$. Let $\tau^\rho = \langle \tau_1, \tau_2 \rangle$ and $\tau'^\rho = \langle \tau'_1, \tau'_2 \rangle$. Clearly $\tau_1 = \tau'_1$ and $\tau_2 = \tau'_2$ hold in all Boolean algebras. It follows, by the completeness theorem for ordinary propositional logic and by the boolean axioms above, that: $\vdash \tau_i^\circ = \tau'^\circ_i$. Hence, by the normal form axioms and the projection axioms, $\vdash (\downarrow(\tilde{\mathfrak{p}}_1(\tau)) \leftarrow \downarrow(\tilde{\mathfrak{p}}_2(\tau))) = (\downarrow(\tilde{\mathfrak{p}}_1(\tau')) \leftarrow \downarrow(\tilde{\mathfrak{p}}_2(\tau')))$. We may conclude, by the pairing axiom, that $\vdash \tau = \tau'$.

We can also see that the Boolean pair algebras form a variety, i.e. that every structure satisfying the equations of Boolean pair algebras is isomorphic to a Boolean pair algebra. Given any structure that satisfies the above equations, we can extract a Boolean algebra, say $\mathcal{B}$, by restricting ourselves to the elements of the form $\tilde{\mathfrak{p}}_i(a)$. (Every such element can both be written in the form $\tilde{\mathfrak{p}}_1(c)$ and $\tilde{\mathfrak{p}}_2(d)$.) The mapping $\langle b_1, b_2\rangle \mapsto (\downarrow b_1 \leftarrow \downarrow b_2)$ gives us an isomorphism between $\mathcal{W}(\mathcal{B})$ and $\mathcal{A}$. E.g. the injectivity of the mapping follows from:

$$\tilde{\mathfrak{p}}_i(\downarrow(\tilde{\mathfrak{p}}_{j_1}(a_1)) \leftarrow \downarrow(\,\tilde{\mathfrak{p}}_{j_2}(a_2))\,) = \tilde{\mathfrak{p}}_{j_i}(a_i).$$

We admit that the above axiomatisation is not very satisfactory. So we pose the following question.

Open Question 4.7 Is there a *satisfactory* axiomatisation of the valid identities for Boolean pair algebras for the signature $[1, \top, \bullet, \leftarrow]$? ▯

Axiomatising the equational theory of Boolean pair algebras for the given signature is but one of the tasks at hand. Here is another.

Open Question 4.8 Is there a *satisfactory* axiomatisation of the valid identities for Boolean pair algebras for the signature $[1, \bot, \bullet, \Rightarrow]$? ▯

Remember that $\bot$ stands for the pair $\langle 1, \bot\rangle$ and not for 0. Our question 4.8 seems even more salient than question 4.7, since it asks for an axiomatisation of the part of our language that really is concerned with presuppositional reasoning. A brief inspection shows that the equational principles valid for $[1, \bot, \bullet, \Rightarrow]$ are markedly different from the principles valid in 'the propositional logic of DPL'. See [8] and [18]. Thus, presuppositional reasoning is truly a distinct branch of dynamics from relational resetting. (In fact in the classical system DPL of Groenendijk en Stokhof the presuppositional aspect is fully eliminated by their use of *total* assignments on a *fixed* set of variables.).

In dynamic approaches to logic there are several notions of *valid inference*, each of which deserves to be axiomatised. We briefly discuss one alternative that is crucial to the semantics of natural language. It is closely related to the dynamic notion of implication that we saw in section 3. For $\alpha, \beta \in W(\mathcal{A})$:

Definition 4.9 $\alpha \models \beta \Leftrightarrow \alpha \leq \alpha \bullet \beta \leq 1$ ▯

The idea behind this notion is that, in the context α, β provides no new information (synchronically speaking): $\alpha \leq \alpha \bullet \beta$. So it gives a natural combination of the two ways of looking at information: we consider β in context, which is a typically diachronic move. But the ordering of information that we use in this context simply is the synchronic ordering $\leq$.[33]

[33] Alternatively $\models$ can be obtained by setting $U = \{\overline{2}\}$ and $D = \{\overline{1}\}$ in the setting of Kracht [12] p.94.

Now the definition of $\models$ results from one additional requirement: we demand that the context α is a state. Note that in Boolean algebras this is an empty requirement: all $a \in \mathcal{A}$ are states. But still, in the Boolean *pair* algebra $\mathcal{U}(\mathcal{A})$ it is not an empty requirement at all. For Boolean pairs we have chosen as the set of states $S = \{\alpha \mid \alpha \leq 1\}$ and $1 = \langle 1, 1 \rangle$ is not the top of the pair algebra![34]

It will be clear to the reader that a decision procedure for this notion of inference follows immediately from the decision procedure for the equational theory. We end with a last pair of questions.

Open Question 4.10 Give a sequent system axiomatising $\models$ for each of the signatures $[1, \top, \bullet, \leftarrow]$ and $[1, \bot, \bullet, \Rightarrow]$. $\square$

5 Conclusion

Above we have presented an algebraic approach to the semantics of presuppositions. Presuppositions are regarded as negative information and a general construction is given that captures this intuition. To test the construction we have worked it out for the special case of Boolean presuppositions. We have discussed the representation of examples, compared our representation with other approaches to Boolean presuppositions and discussed formal properties of the system.

We conclude that the results are encouraging. The applications are in agreement with our basic intuitions about the examples we discuss; the system that the general construction produces agrees with systems known from the linguistic literature; and the system we arrive at has rather nice formal characteristics. This encourages us to try to apply the construction also to more complicated examples in future research.

It should be pointed out that what we have proposed is a *general* construction. This is why the points of agreement with what-was-already-known are encouraging and *not* disappointing. The generality of our approach suggests that we will be able to extend satisfactory results for simple cases in a *systematic* way to other cases. And we may hope that these extensions will be equally satisfactorily.

Of course, the generalisations still have to be carried out. And we have to admit that the most naive extensions have already proved problematic. But there are many less naive options left open that we intend to explore in further research.

[34] We recommend for example [5] for further discussion of dynamic notions of inference.

References

[1] D. Beaver. Presupposition. In J. van Benthem and A. ter Meulen, editors, *Handbook of logic and Language*, pages 939–1008. Elsevier, 1997.

[2] J. van Benthem. General dynamics. *Theoretical Linguistics*, 17:159–201, 1991.

[3] J. van Eijck. Presupposition failure – a comedy of errors. *Aspects of Computing*, 6A:766–787, 1994.

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

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

[6] W. Groeneveld. *Logical investigations into dynamic semantics.* PhD thesis, University of Amsterdam, 1995.

[7] P Hayes. Three-valued logic and computer science. Technical Report CSM-6, University of Essex, 1975.

[8] I. Heim. File change semantics and the familiarity theory of definiteness. In R. Bäuerle, C. Schwarze, and A. von Stechow, editors, *Meaning, Use and Interpretation of Language*, pages 164–189. De Gruyter, Berlin, 1983.

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

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

[11] H. Kamp and U. Reyle. *From Discourse to Logic*, volume I, II. Kluwer, Dordrecht, 1993.

[12] L. Karttunen and S. Peters. Conventional implicature. In Oh C.-K. and D. Dinneen, editors, *Syntax and semantics (11): Presupposition*, pages 1–56. Academic Press, 1979.

[13] M.A. Kracht. Logic and control: How they determine the behaviour of presuppositions. In J. van Eijck and A. Visser, editors, *Logic and Information Flow.* MIT Press, Cambridge, Mass., 1994.

[14] E. Krahmer. *Discourse and Presupposition.* PhD thesis, Tilburg University, 1995.

[15] R. Muskens, J. van Benthem, and A. Visser. Dynamics. In J. van Benthem and A. ter Meulen, editors, *Handbook of Logic and Language*. Elsevier, Amsterdam & MIT Press, Cambridge, 1997.

[16] F. Veltman. Defaults in update semantics. In H. Kamp, editor, *Conditionals, Defaults and Belief Revision*. Dyana Deliverable R2.5A, Edinburgh, 1991.

[17] A. Visser. Actions under presuppositions. In J. van Eijck and A. Visser, editors, *Logic and Information Flow*, pages 196–233. MIT Press, Cambridge, Mass., 1994.

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

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

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

Kees Vermeulen
Els Wolters
Albert Visser

Department of Philosophy
Utrecht University
Heidelberglaan 8
3584 CS Utrecht
The Netherlands