Dynamic Semantics

Jan van Eijck and Albert Visser

April 23, 2009

1 Dynamic Semantics

Dynamic semantics is a perspective on natural language semantics that emphasises the growth of information in time. It is an approach to meaning representation where pieces of text or discourse are viewed as instructions to update an existing context with new information, with an updated context as result. In a slogan: meaning is context change potential. Prime source of inspiration for this dynamic turn is the way in which the semantics of imperative programming languages like C is defined.

It is important to be aware of the abstractness of the perspective, to guard against various non-sequiturs. For one thing, one could easily think that dynamic semantics or update semantics is committed, at least in part, to an internalist idea of semantics, since the information states are 'internal', in the sense that they are wholly contained in the individual mind or, if you wish, in the individual brain. In other words, one might think that the information states of dynamic semantics are what Putnam [Put75] calls 'states in the sense of methodological solipsism'. See the entries: Scientific Realism, Computational Theory of Mind, Mental Content. However, the general framework says nothing about what the states are. The state could very well include the environment in which the receiver is embedded and thus contain an 'external' component.

A second possible misunderstanding is that dynamic semantics or update semantics is in complete opposition to classical truth conditional semantics (compare the entries: Classical Logic, First-order Model Theory). In fact, as further specification of the format will show, what dynamic semantics provides is a generalization of truth conditional semantics rather than a radically different alternative. The classical meanings become the *preconditions* for success of the discourse actions. Dynamic semanticists do hold that *compositional meanings* have the nature of functions or relations, while the classical meanings have the status of *projections* of the compositional meanings.

The point of the use of an abstract framework is not to give empirical predictions. This is the task of specific realizations inside the framework. The framework of Dynamic Semantics (i) provides a direction of thinking and (ii) allows us to import methods from the mathematical study of the framework. It follows that the question whether natural language meaning is intrinsically dynamic does not have an empirical answer. Still, what can be said is that the study of interpretation as a temporal process has proven quite fruitful and rewarding.

Since Dynamic Semantics focuses on the discourse actions of sender and receiver, it is, in a sense, close to use-oriented approaches to meaning in Philosophy, such as the work of Wittgenstein and Dummett. However, easy identifications between Dynamic Semantics and these approaches are to be avoided. Dynamic Semantics as an abstract framework is compatible with many philosophical ways of viewing meaning and interpretation. Dynamic semantics aims to *model* meaning and interpretation. You can do that without answering broader philosophical questions, such as the question what it is that makes it possible for the subject to be related to these meanings at all. E.g., in Dynamic Predicate Logic we take the meaning of *horse* as given without saying what constitutes the subject's having the concept of *horse*; we just stipulate the subject has it. This is not to say such questions — which are at the center of the work of Wittgenstein and Dummett — should not ultimately be answered, it is just to say that a model can be of interest even if it does not answer them. Dynamic Semantics tries to give a systematic and compositional account of meaning. This makes it markedly different in spirit from Wittgenstein's later Philosophy.

2 Meanings in Dynamic Semantics

One approach to dynamic semantics is Discourse Representation Theory or DRT. This was initiated by Hans Kamp in his paper [Kam81]. Closely related to Kamp's approach is Irene Heim's File Change Semantics (see [Hei83a]) and the Discourse Semantics of Pieter Seuren (see [Seu85]). Meanings in DRT are a kind of dynamical databases called *discourse representation structures* or *drs's*. Since drs's can be viewed as rudiments of mental representation, they also have a cognitive appeal [Wer00].

In the approach associated to Discourse Representation Theory, meanings are types of things that are used in the process of fitting pieces of information together and contain items that assist in the fitting. One could compare this to the forms of pieces of a jigsaw puzzle: the form of an individual piece is connected to the possible integration of the piece in the puzzle as a whole. A database taken by itself is a static object: it is the result of an information gathering activity. However as soon as one tries to give a compositional semantics for DRT it becomes clear that the true semantic objects have a dynamic side: they contain instructions for merging the databases. Discourse Representation Theory has a separate entry in the Stanford Encyclopedia; here we will concentrate on what it has in common with a second approach that takes meanings to be resetting actions or update functions.

In this second approach to dynamic semantics, associated with Dynamic Predicate Logic (see [GS91a]), the dynamic meanings are types of actions, things that are individuated by the changes they effect. The basic idea of the relational/update approach in dynamic semantics is that a meaning should be considered as the action type of an action that modifies the receiver's information state. Some basic work in the Dynamic Predicate Logic or DPL-tradition is to be found in [GS91a], [GS91b], [Mus91], [Dek93], [Ver93a], [Eij94], [Ver94], [Gro95], [Kra95], [Ber96], [GSV96], [HV96], [Alo97], [Bea97], [MBV97]. A closely related approach is Update Semantics (see [Vel91]). A unification of DPL and Update Semantics is given in [GSV96].

The varieties of dynamic semantics have led to a modification and extension of the model theoretic approach to natural language semantics in the style of Richard Montague [Mon74a, Mon74b, Mon73] (compare the entry: Logical Form). This new version of Montague Grammar is called Dynamic Montague Grammar. See [GS90, Mus96] and below.

3 Context

Semanticists may mean various things when they talk about context (compare the entries: Epistemic Contextualism, Indexicals), and these different views engender varieties of dynamic semantics, and sometimes interesting combinations. There has been a variety of concerns: constructing an appropriate mechanism for pronominal reference (compare the entries: Anaphora, Reference), explaining the semantics of conditionals (compare the entries: Conditionals, The Logic of Conditionals), giving a semantic treatment of the distinction between assertion and presupposition (compare the entries: Assertion, Speech Acts, Implicature, Pragmatics) and developing a theory of 'presupposition projection', explaining how the interpretation of discourse is influenced and guided by the common ground that exists between speaker and hearer, and developing a theory of how this common ground develops as the discourse proceeds (compare the entries: Pragmatics, Implicature).

Context plays a role in two distinct oppositions. The first opposition is the duality between context and that what modifies the context. Here the context is the information state, or, say, a suitable abstraction from the information state (compare the entry: Semantic Conceptions of Information). The context modifier is the information received. The information cannot be received without the correct kind of presupposed information state. The proper analogues in predicate logic (compare the entries: Classical Logic, First-order Model Theory) are as follows. The information state is an assignment (environment) or a set of assignments. The information received is a set of assignments. The second opposition is the duality of context and content. Here the context is something like the storage capacity of the receiver. The content is the information stored. Thus, e.g., the context in this sense could be a set of discourse referents or files. The content would then be some set of assignments or, perhaps, world/assignment pairs on these referents.

Here is an example to illustrate the distinction. Suppose we view an information state as a pair of a finite set of discourse referents and a set of world/assignment pairs, where the assignments have as domain the given set of referents. Such a state would be a context-in-the-first-sense and the set of referents would be a context-in-the-second-sense. One basic kind of update would be update of content: here we constrain the set of world/assignment pairs, and leave the set of referents constant. A second basic kind of update would be extension of the set of referents: we extend our storage capacity. We modify the given world/assignments pairs to pairs of worlds and extended assignments, where our extended assignments are constrained by the old ones, but take all possible values on the new referents. Thus, the update process in our example is two-dimensional: we have both update of content and update of context-in-the-second-sense.

4 Interpretation as a Process

Interpretation of declarative sentences can be viewed as a product or as a process. In the product perspective, one focusses on the notion of truth in a given situation. In the process perspective, interpretation of a proposition is viewed as an information updating step that allows us to replace a given state of knowledge by a new,more accurate knowledge state. Dynamic semantics focuses on interpretation as a process.

4.1 Propositional Logic as an Update Logic

Propositional logic (the logic of negation, disjunction and conjunction) can be viewed as an update logic, as follows. Consider the case where we have three basic propositions p, q and r, and we know nothing about their truth. Then there are eight possibilities $\{\overline{pqr}, p\overline{qr}, \overline{p}q\overline{r}, \overline{pq}r, pq\overline{r}, p\overline{q}r, \overline{p}qr, pqr\}$. Here $\overline{pqr}$ should be read as: neither of p, q, r is true, $p\overline{qr}$ as: p is true but q and r are false, and so on. If now $\neg p$ ('not p') is announced, four of these disappear, and we are left with $\{\overline{pqr}, \overline{p}q\overline{r}, \overline{pq}r, \overline{p}qr\}$. If next $q \vee \neg r$ ('q or not r') is announced, the possibility $\overline{pq}r$ gets ruled out, and we are left with $\{\overline{pqr}, \overline{p}q\overline{r}, \overline{p}qr\}$. And so on. We can view the meaning of propositions like $\neg p$ and $q \vee \neg r$ as maps from sets of possibilities to subsets of these.

Sets of possibilities represent states of knowledge. In the example,

$$\{\overline{pqr}, p\overline{qr}, \overline{p}q\overline{r}, \overline{pq}r, pq\overline{r}, p\overline{q}r, \overline{p}qr, pqr\}$$

represents the state of complete ignorance about propositions p,q,r. Singleton sets like $\{pq\overline{r}\}$ represent states of full knowledge about the propositions, and the empty set $\emptyset$ represents the inconsistent state that results from processing incompatible statements about p, q and r. Here are the dynamic meanings spelled out of the statements of our propositional language:

- Atomic statements. These are p, q, r. The corresponding update action is to select those possibilities from the current context where the letter is not struck out (overlined).
- Negated statements. These are of the form $\neg\phi$. The corresponding update action os to select those possibilities from the current context that form the complement of the set of possibilities selected by the ϕ statement.

- Conjunctions of statements. These are of the form $\phi \wedge \psi$. The corresponding update action is to select those possibilities from the current context that form the intersection of the selections from the curent context made by the ϕ and the ψ statements.

- Disjunctions of statements. These are of the form $\phi \vee \psi$. The corresponding update action is to select those possibilities from the current context that form the union of the selections made by the ϕ and the ψ statements.

This gives the meanings of the propositional connectives, as operations from an old context representing a state of knowledge to a new context representing the state of knowledge that results from processing the propositional information.

As a concrete example, consider the game of Mastermind, played with three positions and four colors red, green, blue, yellow (see also [Ben96]). You are trying to guess a secret code, which is expressed as a sequence of three colours, so the code is one of:

RGB	RBG	GBR	GRB	BRG	BGR
YGB	YBG	GBY	GYB	BYG	BGY
RYB	RBY	YBR	YRB	BRY	BYR
RGY	RYG	GYR	GRY	YRG	YGR

Suppose the secret code is ‘red, blue, yellow’. Since you do not know this, none of the above 24 possibilities is ruled out yet for you. Assume your initial guess is ‘red, green, blue’.

Now the feedback you will get, according to the rules of the game, is in the form of propositional information. Black marks indicate correct colours in correct positions, grey marks indicate correct colours in wrong positions. So the feedback you will get is one black mark (red is in the correct position) and one grey mark (blue is in the wrong position). The propositional formula $b_1 \wedge g_1$ (‘one black and one grey’) expresses this. This rules out all but six possibilies, for you reason as follows: if red is in correct position, then the code is either red blue yellow (RBY) or red yellow green (RYG); if green is in correct position, then the code is either BGR or YGR; if blue is in correct position then the code is either YRB or GYB. What this means is that you interpret the feedback $b_1 \wedge g_1$ as a map from the set of all positions to the set

$$\{RBY, RYG, BGR, YGR, YRB, GYB\}$$

Suppose your next guess is ‘yellow, green, red’. The feedback now is g_2, for red and yellow occur, but in incorrect positions. This rules out any of the six possibilities with a colour in the same position as in the guess, in other words, it rules out

$$\{RYG, BGR, YGR, YRB\}$$

and you are left with the set of possibilities $\{RBY, GYB\}$. Your final guesss ‘green yellow blue’ sollicits the feedback g_2, for blue and yellow occur, but in wrong positions, and your final update yields $\{RBY\}$. The fact that you are left with a singleton set indicates that you now know the secret code.

4.2 Programming Statements and their Execution

Programming statements of imperative languages like C are interpreted (or ‘executed’) in the context of a machine state, where machine states can be viewed as allocations of values to registers. Assume the registers are named by variables x, y, z, and that the contents of the registers are natural numbers. Then the following is a machine state:

x	12
y	117
z	3

If the C statement $z = x$ is executed (or ‘interpreted’) in this state, the result is a new machine state:

x	12
y	117
z	12

If the sequence of statements $x = y; y = z$ is executed in this state, the result is:

x	117
y	12
z	12

This illustrates that the result of the sequence $z = x; x = y; y = z$ is that the values of x and y are swapped, with the side effect that the old value of z gets lost. In other words, the meaning of the program $z = x; x = y; y = z$ can be viewed as as map from an input machine state s to an output machine state s' that differs from s in the fact that $s(x)$ (the value of x in state s) and $s(y)$ are swapped, and that $s'(z)$ (the new value of z) equals $s(y)$.

4.3 Quantifiers as Programs

Now consider the existential quantifier 'there exists x such that A'. Suppose we add this quantifier to a programming language like C. What would be its meaning? It would be an instruction to replace the old value of x by some arbitrary new value, where the new value has property A. We can decompose this into a part 'there exists x' and a test 'A', where the parts are glued together by sequential composition: '$\exists x; A$'. Focusing on the part '$\exists x$', what would be its natural meaning? An instruction to replace the old value of x by some arbitrary new value. This is again a relation between input states and output states, but the difference with definite assignments like $x = y$ is that now the relation is not a function. In fact, this relational meaning of quantifiers shows up in the well known Tarski-style truth definition for first order logic (compare the entry: Tarski's Truth Definitions):

$\exists x \phi$ is true in a model M, under a variable assignment α iff (if and only if) there is some β with β different from α at most in the value that gets assigned to x, and ϕ s true in M under assigment β.

Implicit in this is a relation $[x]$ that holds between α and β iff for all variables y it is the case that $y \not\equiv x$ implies $\alpha(y) = \beta(y)$.

5 Dynamic Predicate Logic

The scope conventions of ordinary predicate logic (compare the entry: First-order Model Theory) are such that the scope of a quantifier is always confined to the formula in which it occurs as its main connective. In other words, in a parse tree of any formula, an occurrence of a quantifier Qx at a node ν will only bind occurrences of x occurring below ν in the tree. Now consider the following discourse.

```
A man comes in.  He sees a dog.  He smiles.
```

How would we paraphrase this discourse in predicate logic? Well, we could translate `A man comes in` as

$$(1)\ \ \exists x\ (\texttt{man}(x) \wedge \texttt{comes-in}(x)).$$

Then the obvious move would be to translate `A man comes in. He sees a dog` into:

$$(2)\ \ \exists x\ (\texttt{man}(x) \wedge \texttt{comes-in}(x)) \wedge \exists y\ (\texttt{dog}(y) \wedge \texttt{sees}(x,y)).$$

However, this cannot be correct, since the second occurrence of x is not bound by the occurrence of $\exists x$ in this formula. The right translation is:

$$(3)\ \ \exists x\ (\texttt{man}(x) \wedge \texttt{comes-in}(x) \wedge \exists y\ (\texttt{dog}(y) \wedge \texttt{sees}(x,y))).$$

So far, so good. There is nothing wrong, per se, with (3). It just that, unlike (2), it is not *produced* in a compositional way from (1), since rearranging the brackets is not an operation that can be reflected at the semantical level (compare the entry: Compositionality). Similarly, if we want to translate our full sample discourse we have to 'break open' our previous results again to produce:

$$(4)\ \ \exists x\ (\texttt{man}(x) \wedge \texttt{comes-in}(x) \wedge \exists y\ (\texttt{dog}(y) \wedge \texttt{sees}(x,y)) \wedge \texttt{smiles}(x)).$$

Thus, if we think of translation of discourses *as a process*, we cannot, in general, produce intermediate translations: we are forced to translate the discourse as a whole. For another example of this, consider Geach's donkey sentence (compare the entry: Anaphora):

`If a farmer owns a donkey, he beats it.`

The obvious translation into predicate logic would be:

$$(\exists x\ (\texttt{farmer}(x) \wedge \exists y\ (\texttt{donkey}(y) \wedge \texttt{owns}(x,y))) \to \texttt{beats}(x,y)).$$

However, again the last occurrences of x and y will be free in this paraphrase. So the paraphrase will not capture the intended meaning. The correct translation would be something like:

$$\forall x\ (\texttt{farmer}(x) \to \forall y\ ((\texttt{donkey}(y) \wedge \texttt{owns}(x,y)) \to \texttt{beats}(x,y))).$$

This last translation is clearly not obtained in a compositional way from Geach's donkey sentence (see [Gea80], and the entry on Anaphora).

Dynamic Predicate Logic (DPL) was invented by Jeroen Groenendijk and Martin Stokhof [GS91a] to make compositional translation of examples such as the ones above possible. It is the simplest possible variant of predicate logic in which we have existential quantifiers with extendible scope. The universal quantifiers on the other hand are the familiar ones of predicate logic. DPL is a theory of testing and resetting of variables/registers. These are fundamental operations in computer science. Thus, apart from its use in Logical Semantics, DPL is a simple theory of these basic operations.

To understand the basic idea behind DPL, look at the example `A man comes in. He sees a dog` again. Why is it the case that in ordinary predicate logic we cannot take the meaning associated with `A man comes in` and combine it with the meaning of `He sees a dog` to obtain the meaning of `A man comes in. He sees a dog`? Well, the meaning of `A man comes in` is a set of assignments. Suppose e.g. there is a man in the domain of discourse entering some specified place. Then, `a man comes in` would be true. Its meaning would be the set of *all* assignments on some given set of variables. There is no way to get at the object or objects introduced by the sentence, just by looking at this set. It could also be the meaning of `A dog sees a cat`. What we need is an alternative meaning that 'remembers' and 'produces' the man introduced in the discourse.

We get our clue on how to do this from staring at the definition of existential quantification in ordinary predicate logic. Suppose we work with total assignments on a fixed set of variables VAR over a fixed domain D. Let the meaning of $P(x)$ be the set of assignments F. Thus, F is the set of all assigments α with property that αx is an object satisfying P.

Define

$$\alpha[x]\beta :\Leftrightarrow \forall v \in \mathsf{VAR} \setminus \{x\}\ \alpha v = \beta v.$$

So $[x]$ is the relation 'β is a result of resetting α on x'. Now the meaning, say G, of $\exists x\ P(x)$, will be:

$$G := \{\alpha{\in}\mathsf{ASS} \mid \exists\beta{\in}F\ \alpha[x]\beta\}.$$

Thus, G is the set of assignments that can be successfully reset w.r.t. x to an assignment in F. Viewed differently G is *the domain of* the relation R given by

$$\alpha R\beta :\Leftrightarrow \alpha[x]\beta \text{ and } \beta \in F.$$

We could say that G is the precondition for the resetting action R. Now the idea of DPL is to take the meaning of $\exists x\ P(x)$ not the precondition G

but the resetting action R. In this way we do not lose information since G can always be obtained from R. Moreover, the range elements β of R are constrained to be in F and have x-values in the interpretation of P. These are precisely the x's that do P, that we were looking for.

More generally, we take as DPL-meanings *binary relations* between assignments. Such relations can be seen as (modelings of) *resetting actions*. This is an instance of a well-known, admittedly simplistic way, of modeling actions: an action is viewed as a relation between the states of the world before the action and the corresponding states after the action.

Here is the full definition. Let a non-empty domain D and a set of variables VAR be given. Let a model $\mathcal{M} = \langle D, I\rangle$ of signature Σ be given. Atomic conditions π are of the form $P(x_0, \cdots, x_{n-1})$, where P is in Σ of arity n. Atomic resets ε are of the form $\exists v$, where v is a variable. The language of predicate logic for Σ is given by:

$$\phi ::= \bot \mid \top \mid \pi \mid \varepsilon \mid \phi \cdot \phi \mid {\sim}(\phi).$$

Assignments are elements α, β, ..., of $\mathsf{ASS} := D^{\mathsf{VAR}}$. We define the relational meaning of the language, as follows:

- $\alpha[\bot]\beta :\Leftrightarrow 0 \neq 0$.
- $\alpha[\top]\beta :\Leftrightarrow \alpha = \beta$.
- $\alpha[P(x_0, \cdots, x_{n-1})]\beta :\Leftrightarrow \alpha = \beta$ and $\langle \alpha x_0, \cdots, \alpha x_{n-1}\rangle \in I(P)$, where P is a predicate symbol of Σ with arity n.
- $\alpha[\exists v]\beta :\Leftrightarrow \alpha[v]\beta$, where $\alpha[v]\beta$ iff $\alpha w = \beta w$, for all variables $w \not\equiv v$.
- $\alpha[\phi \cdot \psi]\beta :\Leftrightarrow \exists\gamma\ \alpha[\phi]\gamma[\psi]\beta$.
- $\alpha[{\sim}(\phi)]\beta :\Leftrightarrow \alpha = \beta$ and $\forall\gamma\ \neg\ \alpha[\phi]\gamma$.

Truth is defined in terms of relational meaning:

$$\alpha \models \phi :\Leftrightarrow \exists\beta\ \alpha[\phi]\beta.$$

We can define implication $\phi \rightarrow \psi$ as ${\sim}(\phi \cdot {\sim}\psi)$. Applying the truth definition to this gives:

$$\alpha \models \phi \rightarrow \psi \ :\Leftrightarrow \ \text{for all } \beta: \ (\alpha[\phi]\beta \Rightarrow \beta \models \psi).$$

Relational meaning also yields the following beautiful definition of dynamic implication:

$$\phi \models \psi :\Leftrightarrow \forall \alpha, \beta\ (\alpha[\phi]\beta \Rightarrow \exists \gamma\ \beta[\psi]\gamma).$$

This definition was first introduced by Hans Kamp in his pioneering paper [Kam81]. Note that $\sim\phi$ is equivalent to $(\phi \to \bot)$, and that $(\phi \to \psi)$ is true iff $\phi \models \psi$. We can define $\forall x\ (\phi)$ as $(\exists x \to \phi)$.

A possible alternative notation for $\exists v$ would be $[v :=?]$ (random reset). This emphasizes the connection with random assignment in programming.

The interpretations of predicate symbols are *conditions*. They are subsets of the diagonal $\{\langle \alpha, \alpha \rangle \mid \alpha \in \mathsf{ASS}\}$. A condition is a test: it passes on what is OK and throws away what is not OK, but it modifies nothing. The mapping diag that sends a set F of assignments to a condition $\{\langle \alpha, \alpha \rangle \mid \alpha \in F\}$ is the link between the classical and the dynamic world. E.g. the composition of the diagonals of F and G is the diagonal of their intersection.

Ordinary Predicate Logic can be interpreted in DPL as follows. We suppose that the predicate logical language has as connectives and quantifiers: $\top$, $\bot$, $\wedge$, $\to$, $\exists x$. We translate as follows:

- $(\cdot)^*$ commutes with atomic formulas and with $\to$
- $(\phi \wedge \psi)^* := \phi^* \cdot \psi^*$
- $(\exists x(\phi))^* := \neg\neg(\exists x \cdot \phi^*)$

We get that $[\phi^*]$ is the diagonal of the classical interpretation of ϕ. Our translation is compositional. It shows that we may consider Predicate Logic as a *fragment* of DPL.

It is, conversely, possible to translate any DPL-formula ϕ to a predicate logical formula ϕ°, such that the domain of $[\phi]$ is the classical interpretation of ϕ°. One of the ways to define this translation is by means of a precondition calculus, with Floyd-Hoare rules [EdV92]. The following is a variation on this. Take the language of standard predicate logic, with diamond modalities $\langle \psi \rangle \phi$ added, where ψ ranges over DPL formulas, with meaning $\alpha \models \langle \psi \rangle \phi$ if there is an assignment β with $\alpha[\psi]\beta$, and $\beta \models \phi$. Then the following equivalences show that this extension does not increase expressive power.

- $\langle \bot \rangle \phi \leftrightarrow \bot$.

- $\langle \top \rangle \phi \leftrightarrow \phi$.
- $\langle P(x_1 \cdots x_n) \rangle \phi \leftrightarrow (P(x_1 \cdots x_n) \wedge \phi)$.
- $\langle \exists v \rangle \phi \leftrightarrow \exists v \phi$.
- $\langle \psi_1 \cdot \psi_2 \rangle \phi \leftrightarrow \langle \psi_1 \rangle \langle \psi_2 \rangle \phi$.
- $\langle {\sim}(\psi) \rangle \phi \leftrightarrow (\neg(\langle \psi \rangle \top) \wedge \phi)$.

So in a weak sense 'nothing new happens in DPL'. We cannot define a set that we cannot also define in Predicate Logic.

The equivalences for the modalities fix a translation $(\cdot)^\circ$ that yields the weakest precondition for achieving a given postcondition. As an example, we compute $\langle \psi \rangle \top$, where ψ is the Geach sentence (the weakest precondition for success of the Geach sentence):

$$\begin{array}{ll} & (\langle (\exists x \cdot Fx \cdot \exists y \cdot Dy \cdot Hxy) \rightarrow Bxy \rangle \top)^\circ \\ \Leftrightarrow & (\langle {\sim}((\exists x \cdot Fx \cdot \exists y \cdot Dy \cdot Hxy) \cdot {\sim} Bxy) \rangle \top)^\circ \\ \Leftrightarrow & \neg(\langle (\exists x \cdot Fx \cdot \exists y \cdot Dy \cdot Hxy) \cdot {\sim} Bxy \rangle \top)^\circ \\ \Leftrightarrow & \cdots \\ \Leftrightarrow & \neg \exists x (Fx \wedge \exists y (Dy \wedge Hxy \wedge \neg Bxy)). \end{array}$$

The translation gives the Geach sentence its correct meaning, but it is not compositional: the example illustrates that the way in which the existential quantifier gets handled depends on whether it is in the scope of $\sim$.

6 Update Semantics, the Very Idea

Update semantics is a theory of meaning based on a very simple idea. We start with a simple model of a hearer / receiver who receives items of information sequentially. At every moment the hearer is in a certain state: she possesses certain information. This state is modified by the incoming information in a systematic way. We now analyze the meaning of the incoming items as their contribution to the change of the information state of the receiver. Thus, the meaning is seen as an action.

Note that the meanings are are in fact action types. They are not the concrete changes of some given state into another, but what such concrete changes have in common.

The most abstract mathematical format for update semantics is as a transition system. We have a set of states and a set of labeled transitions between these states. The meaning of a given item is modeled by the relation corresponding to a label that is assigned to the item. Here are two examples of labeled transitions systems. The system $A \xrightarrow{a} B \xrightarrow{b} C$ is a functional transition system. The system $D \xleftarrow{a} A \xrightarrow{a} B \xrightarrow{b} C$ is non-functional. Note the abstractness of the notion of state. Nothing has been said about what states *are*, and this lack of commitment is intentional. Information states are often called *contexts*, since the state is a precondition for the 'interpretation' of the informational item. Also the use of the word 'context' makes it clear that we are not interested in the total state of the receiver but only in aspects of it relevant to the kind of information we are focussing on. Thus, meanings are often called *context change potential* in the dynamic tradition.

7 Updates on a Boolean Algebra

A very simple and appealing model is to consider updates on a Boolean algebra (compare the entry: The Mathematics of Boolean Algebra), or, more concretely, on the power set of a given sets of possibilities. Thus, we have certain operations on our states available like conjunction / intersection and negation / complement.

7.1 Semantics for Maybe

One realization of this idea is Frank Veltman's Update Semantics for *maybe.* See [Vel91], [Vel96]. Note that the discourse *Maybe it is raining. It is not raining* is coherent. However, the discourse *It is not raining. Maybe it is raining* is not. (We are assuming that the environment about which we receive information does not change during the discourse.) The aim of Veltman's Update Semantics is to analyze this phenomenon of non-commutativity.

The language of update semantics is that of propositional logic, with the possibility operator '$\Diamond$' added. This operator stands for '*maybe*'. Here is the specification of the language, where 'p' ranges over of propositional variables. We prefer a dot over the usual conjunction sign to stress that conjunction means sequential composition in the order of reading.

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

The interpretation is a simple extension of the ‘update perspective’ on propositional logic sketched above, where the update interpretations of propositional atoms, of negation, of conjunction and of disjunction were given. The update interpretation of ‘maybe’ is given by:

> Maybe statements are of the form $\Diamond\phi$. The corresponding update action on the current context is to check whether an update with ϕ in the context yields a non-empty set of possibilities. In the affirmative case, the update with $\Diamond\phi$ returns the whole context, otherwise it returns the empty set.

For spelling this out more formally, we fix a Boolean algebra $\mathcal{B}$. Interpretations are functions from $\mathcal{B}$ to $\mathcal{B}$. Let α be an assignment from the propositional variables to the domain of $\mathcal{B}$. We define, for s in $\mathcal{B}$.

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

Definition of truth in an information state:

$$s \models_\alpha \phi :\Leftrightarrow s \leq [\phi]_\alpha s.$$

Instead of $s \models_\alpha \phi$ we also say that ϕ *is accepted in* s.

Definition of consequence, relative to an assigmnent:

$$\psi \models_\alpha \phi :\Leftrightarrow \forall s\ [\psi]_\alpha s \models \phi.$$

Consistency, relative to an assignment:

> ϕ is *consistent* iff, for some state s, we have $[\phi]_\alpha s \neq \bot$.

Note that ϕ is consistent iff $\phi \not\models_\alpha \bot$,

We easily see that $s \models_\alpha \phi \cdot \psi$ iff $s \models_\alpha \phi$ and $s \models_\alpha \psi$. So the difference between *Maybe it is raining. It is not raining* and *It is not raining. Maybe* *it is raining* cannot be understood at the level of acceptance: none of the two discourses is ever accepted. However, *Maybe it is raining. It is not raining* is clearly consistent, where *It is not raining. Maybe it is raining* is inconsistent.

If we drop the semantics for *maybe*, Veltman's semantics collapses modulo isomorphism into classical semantics, the relevant mappings being $F \mapsto F\top$ and $p \mapsto \lambda s \cdot (s \wedge p)$. These mappings spell out the relation between the usual semantics for propositional logic and its update semantics.

7.2 Properties of Update Functions

Here are some important properties of update functions. The first and second ones hold of updates that commute with (possibly finite) disjunctions. The third one holds of updates that narrow down the range of possibilities.

- An update function f is *finitely distributive* iff $f\bot = \bot$ and, for any s, s', $f(s \vee s') = fs \vee fs'$.
- An update function f is *distributive* iff $f(\bigvee X) = \bigvee(fX)$, for any set X of states. (So, distributivity means that f is an morphism of $\mathcal{B}$ to itself, where $\mathcal{B}$ is considered as a complete upper semi lattice.)
- An update function f is *eliminative* or *regressive* iff, for any s, we have $fs \leq s$

Note that if a Boolean algebra is finite, then it is automatically complete. Moreover in this case distributive and finitely distributive coincide. Here is an example of an update function that is finitely distributive, but not distributive. Consider the Boolean Algebra of subsets of the natural numbers. Take $F(X) := \top$ if X is infinite and $F(X) = \bot$ if X is finite. Then, clearly F is finitely distributive, but not distributive.

The update functions of Veltman that can be generated in his system for *maybe* are eliminative, but not distributive. E.g., suppose $\bot < s < \top$ and $\alpha(p) = s$. Then, $[\Diamond p]_\alpha(s \vee \neg s) = \top$ and $([\Diamond p]_\alpha(s) \vee [\Diamond p]_\alpha(\neg s)) = (s \vee \bot) = s$.

We will see that the update functions associated DPL are distributive, but not eliminative, due to the presence of $\exists v$. If we view eliminativity as an explication of information growth, the non-eliminativity means that DPL contains destructive updates. This is intuitively plausible, since $\exists v$ does indeed throw away previous values stored under v. Full distributivity means that the update functions can be considered as relations.

7.3 Dynamic Predicate Logic in Update Form

In Update Semantics for DPL, we represent information states as sets of assignments and we represent transitions *as functions* from sets of assignments to sets of assignments.

Distributive update functions and relations over a given domain can be correlated to each other in a quite general way. Let A be a non-empty set. Let R range over binary relations on A and let F range over functions from $\wp A$ to $\wp A$. We define:

- $F_R(X) := \{y{\in}A \mid \exists x{\in}X \; xRy\}$,
- $xR_Fy :\Leftrightarrow y \in F(\{x\})$.

We can show that, if F is distributive, then $F_{R_F} = F$ and $R_{F_R} = R$. We can transform the relations of DPL to update functions via the mapping $F_{(\cdot)}$.

Here is the direct definition. Let a non-empty domain D and a set of variables VAR be given. Let a model $\mathcal{M} = \langle D, I\rangle$ of signature σ be given. Atomic conditions π are of the form $P(x_0, \cdots, x_{n-1})$, where P is in σ of arity n. Atomic resets ε are of the form $\exists v$, where v is a variable. We repeat the definition of the language of dynamic predicate logic for σ:

- $\phi ::= \bot \mid \top \mid \pi \mid \varepsilon \mid \phi \cdot \phi \mid {\sim}(\phi)$.

A *state* is a set of assignments, i.e. of functions $\mathsf{VAR} \to D$. We consider the states as a complete Boolean algebra $\mathcal{B}$ with the usual operations.

Formulas ϕ of predicate logic are interpreted as *update functions* $[\phi]$, i.e. as functions $\mathsf{States} \to \mathsf{States}$. We define:

- $[\bot]s := \emptyset$.
- $[\top]s := s$.
- $[P(x_0, \cdots, x_{n-1})]s := \{\alpha{\in}s \mid \langle \alpha x_0, \cdots, \alpha x_{n-1}\rangle \in I(P)\}$,
  where P is a predicate symbol of σ with arity n.

- $[\exists v]s := \{\beta \mid \exists \alpha{\in}s\ \alpha[v]\beta\}$, where $\alpha[v]\beta$ iff $\alpha w = \beta w$, for all variables $w \not\equiv v$.
- $[\phi \cdot \psi]s := [\psi][\phi]s$.
- $[{\sim}\phi]s := \{\alpha{\in}s \mid [\phi]\{\alpha\} = \emptyset\}$.

The truth definition now takes the following shape:

$$s \models \phi :\Leftrightarrow \forall \alpha{\in}s\ [\phi]\{\alpha\} \neq \emptyset.$$

And here is the definition of dynamic implication in its new guise:

$$\phi \models \psi :\Leftrightarrow \forall s\ [\phi]s \models \psi.$$

7.4 Van Benthem's Bottle

Johan van Benthem enclosed the dynamic fly in a static bottle by showing that update semantics on a Boolean algebra collapses into classical semantics if we demand both finite distributivity and eliminativity [Ben89].

This argument seems to show that the non-eliminativity of a relational semantics like DPL is a necessary feature. The cost of distributivity is that we accept destructive updates. After giving the argument we will indicate the way out of the bottle in Subsection 7.5. Here is the argument.

Theorem 7.1 (van Benthem) *Suppose we are given a Boolean algebra $\mathcal{B}$ and an update function f over $\mathcal{B}$. Suppose further that f is finitely distributive and eliminative. I.e.,*

- $f\bot = \bot,\ f(s \vee s') = fs \vee fs'$,
- $fs \leq s$.

Then, we have $fs = (s \wedge f\top)$.

Proof

$$\begin{array}{rcl} s \wedge f\top & = & s \wedge f(s \vee \neg s) \\ & = & s \wedge (fs \vee f(\neg s)) \\ & = & (s \wedge fs) \vee (s \wedge f(\neg s)) \\ & = & fs \end{array}$$

❑

The map $\tau : f \mapsto f\top$. is a bijection between finitely distributive and eliminative update functions and the elements of our Boolean algebra. Moreover, for finitely distributive and eliminative f and g,

$$gfs = (fs \wedge g\top) = ((s \wedge f\top) \wedge g\top) = (s \wedge (f\top \wedge g\top)).$$

So $\tau(gf) = \tau(f) \wedge \tau(g)$. One can also show that τ commutes with negation. Thus, τ is an isomorphism between finitely distributive and eliminative updates with their intrinsic operations and $\mathcal{B}$.

7.5 Dimensions of Information

We can escape the bottle by treating information as 'more dimensional'. Consider, e.g., the operation $\exists x$ in DPL. This means intuitively:

Extend the information state with a discourse referent x.

This does not change the content of what the receiver knows in the worldly sense, but it changes the algebra of possible propositions. Thus, it is reasonable to say that this operation takes us to a different algebra.

This suggests the following setting. An update function is given by (i) a canonical embedding E between Boolean algebras $\mathcal{B}_0$ and $\mathcal{B}_1$ and a mapping f from $\mathcal{B}_0$ to $\mathcal{B}_1$. Here E tells us which proposition in the new algebra contains the same worldly information as a proposition in the old one. The salient (possible) properties of f now translate to:

- *finite distributivity*: $f\bot_0 = \bot_1$, $f(s \vee s') = fs \vee fs'$,
- *distributivity*: $f \bigvee X = \bigvee\{fx \mid x \in X\}$,
- *eliminativity*: $fs \leq_1 \mathsf{E}s$.

In the new context, van Benthem's theorem tells us that, if f is eliminative and finitely distributive, then $fs = (\mathsf{E}s \wedge f\top_0)$. Thus, the modified result shows that an eliminative and finitely distributive update function can be completely characterized by the pair $\langle \mathsf{E}, f\top_0\rangle$. This pair can be more or less considered as the semantic discourse representation structure, or semantic drs, associated with f. So, in a sense, van Benthem's result explains the possibility of Discourse Representation Theory.

Frank Veltman, Jeroen Groenendijk en Martin Stokhof succeeded in integrating update semantics for *maybe* and Dynamic Predicate Logic by realizing a two-dimensional semantics, where the elements of the relevant Boolean algebras are sets of assignment/world pairs. The change in the algebras is caused by the extension of the domains of the assignments: introducing a discourse referent enlarges the storage capacity (see [GSV96]).

7.6 Introducing a Referent

What happens if we try to introduce a discourse referent, when it is already present? This phenomenon is in fact the source of destructiveness of classical DPL. The imagined situation is deeply unnatural. How can one intend to introduce a new referent and fail at the job? From the technical point of view there are many ways to handle the problem. A first way is to simply forbid the occurrence of such a repeated introduction. This amounts to introducing a *constraint on syntax*. This way is embodied in versions of Discourse Representation Theory: the drs-construction algorithm always invites us to choose a *fresh* variable letter when a new discourse referent is needed.

A second way is to store a stack of objects under every variable letter. In this way one obtains versions of Vermeulen's Sequence Semantics. See [Ver93a]. One could view Vermeulen's idea either as 'different objects under one referent' or as 'different referents under one variable name'. See below.

The most satisfactory way, is to say that the imagined occurrence of double introduction is an instance of the fallacy of misplaced concreteness. It rest on the confusion of the discourse referent and its label, or, to change the simile, it confuses the file and a pointer to the file. Only the label could already be present. The referent is new *ex stipulatione.*

Again there are various ways of handling a situation of overloading of labels/pointers. For example, we could say that in case of a clash, the new referent will win and the old referent will loose it label. This gives us Vermeulen's Referent Systems. See [Ver95]. Alternatively we could let the old referent win. This possibility is embodied in Zeevat's compositional version of Discourse Representation Theory (see [Zee91]). Finally, we could allow referents to share a label. Vermeulen's Sequence Semantics can be viewed as one way of implementing this idea.

Frank Veltman, Jeroen Groenendijk and Martin Stokhof in their [GSV96] use one version of referent systems in their integration of Update Semantics for *maybe* and Dynamic Predicate Logic.

8 Dynamic Semantics and Presupposition

We have looked at anaphora and at epistemic modalities. Other natural language phenomena with a dynamic flavour are presuppositions, and assumption-introducing expressions like 'suppose'. This section sketches a dynamic perspective on presuppositions.

8.1 Different Styles

Dynamic logic comes in various flavours, the main distinction being that between single-storey and dual-storey architectures. In the single-storey approach everything is dynamic, and therefore formulas denote relations. In the dual-storey approach there is a level of state changes and a level of states, and it is possible to move back and forth between the two levels. An example of a switch from the state change level to the state level is the postcondition operator, which gives the postcondition of a state change for some initial condition. Another operator is the precondition operator, which gives the (weakest) precondition of a state change for a given postcondition. Below we give an example of a dual-storey approach, with precondition operators. It is not hard to work out a single-storey version or a version with postcondition operators.

Pragmatic accounts of presupposition and presupposition projection were given by Karttunen [Kar73, Kar74] and Stalnaker [Sta72, Sta74]. These authors proposed an explanation for the fact that the presupposition of a conjunction ϕ and ψ consists of the presupposition of ϕ conjoined with the implication $\text{ass}_\phi \to \text{pres}_\psi$. When a speaker utters this conjunction, she may take it for granted that the audience knows ϕ after she has uttered this first conjunct. So even if ϕ is not presupposed initially, it will be presupposed by the time she gets to assert ψ, for now the context has shifted to encompass ϕ.

Various authors have tried to make the idea of shifting context precise, most notably Heim [Hei83b]. Presupposition projection has been a major topic in dynamic semantics of natural language ever since [Bea01]. Formal accounts of presupposition within the DRT framework (e.g., [San92], or the textbook treatment based on this in [BBar]) combine the dynamics for setting up appropriate contexts for anaphoric linking with the dynamics for presupposition handling and presupposition accommodation. Although anaphora resolution and presupposition handling have things in common, we will treat them as orthogonal issues. For a dissenting voice on the marriage of dynamic semantics and presupposition handling, see [Sch07].

8.2 Presuppositions Failure and Error Transitions

Presupposition failures are errors that occur during the left to right processing of a natural language text. On the assumption that sequential processing changes context dynamically, a dynamic account of presupposition failure models presupposition failure as transition to an error state.

So we postulate the existence of an error state, and we say that a process 'aborts' (or: 'suffers from presupposition failure') in a given context if it leads to the error state from that context.

Propositional Dynamic Error Logic is a logic of formulas and programs that is interpreted in labeled transition systems over a state set that includes the error state *error*.

Let P be a set of proposition letters.

Formulas

$$\phi ::= \top \mid p \mid \neg\phi \mid \phi_1 \wedge \phi_2 \mid E(\pi) \mid \langle\pi\rangle\phi \mid [\pi]\phi$$

Programs

$$\pi ::= \textbf{abort} \mid \phi? \mid \pi_1;\pi_2 \mid \pi_1 \cup \pi_2$$

Abbreviations:

$$\begin{array}{rcl} \bot & :\equiv & \neg\top \\ \phi_1 \vee \phi_2 & :\equiv & \neg(\neg\phi_1 \wedge \neg\phi_2) \\ \phi_1 \rightarrow \phi_2 & :\equiv & \neg(\phi_1 \wedge \neg\phi_2) \end{array}$$

Let M be a pair (S, V) with $\mathit{error} \in S$ (the set of states includes the error state) and $V : P \rightarrow \mathcal{P}(S - \{\mathit{error}\})$ (the valuation assigns a set of proper states to every proposition letter in P). Interpretation of formulas and programs by mutual recursion.

All relational meanings will be subsets of $S \times S$, with two additional properties:

1. $(\mathit{error}, \mathit{error})$ is an element of every relational meaning,
2. $(\mathit{error}, s) \in R$ implies $s = \mathit{error}$.

The composition $R;T$ of two relations R and T is defined in the usual way:

$$R;T := \{(s, s') \mid \exists s'' \in S : (s, s'') \in R \text{ and } (s'', s') \in T\}.$$

Note that it follows from the definition and the properties of R and T that $R;T$ will also have these properties. In particular, $(\mathit{error}, s) \in R;T$ implies $s = \mathit{error}$. In other words, there is no recovery from error.

In the truth definition we assume that s is a proper state.

$$\begin{array}{rcl}
M, s \models \top & & \text{always} \\
M, s \models p & :\equiv & s \in V(p) \\
M, s \models \neg\phi & :\equiv & \text{not } M, s \models \phi \\
M, s \models \phi_1 \wedge \phi_2 & :\equiv & M, s \models \phi_1 \text{ and } M, s \models \phi_2 \\
M, s \models E(\pi) & :\equiv & (s, \mathit{error}) \in [\![\pi]\!]^M \\
M, s \models \langle\pi\rangle\phi & :\equiv & \text{there is an } s' \in S - \{\mathit{error}\} \\
 & & \text{with } (s, s') \in [\![\pi]\!]^M \text{ and } M, s' \models \phi
\end{array}$$

$$\begin{array}{rcl}
[\![\mathbf{abort}]\!]^M & := & \{(s, \mathit{error}) \mid s \in S\} \\
[\![\phi?]\!]^M & := & \{(s, s) \mid s \in S - \{\mathit{error}\} \text{ and } M, s \models \phi\} \cup \{(\mathit{error}, \mathit{error})\} \\
[\![\pi_1; \pi_2]\!]^M & := & [\![\pi_1]\!]^M; [\![\pi_2]\!]^M \\
[\![\pi_1 \cup \pi_2]\!]^M & := & [\![\pi_1]\!]^M \cup [\![\pi_2]\!]^M.
\end{array}$$

This language has an obvious axiomatisation: the axioms of propositional logic, an axiom stating the relation between program diamonds and boxes,

$$\langle\pi\rangle\neg\phi \leftrightarrow \neg[\pi]\phi$$

the distribution axiom for programs

$$[\pi](\phi_1 \to \phi_2) \to [\pi]\phi_1 \to [\pi]\phi_2$$

the reduction axioms for sequential composition, choice and test:

$$\begin{array}{rcl} \langle \pi_1; \pi_2 \rangle \phi & \leftrightarrow & \langle \pi_1 \rangle \langle \pi_2 \rangle \phi \\ \langle \pi_1 \cup \pi_2 \rangle \phi & \leftrightarrow & \langle \pi_1 \rangle \phi \vee \langle \pi_2 \rangle \phi \\ \langle \phi_1 ? \rangle \phi_2 & \leftrightarrow & \phi_1 \wedge \phi_2 \end{array}$$

reduction axioms for error behaviour of composition, choice and test:

$$\begin{array}{rcl} E(\pi_1; \pi_2) & \leftrightarrow & E(\pi_1) \vee \langle \pi_1 \rangle E(\pi_2) \\ E(\pi_1 \cup \pi_2) & \leftrightarrow & E(\pi_1) \vee E(\pi_2) \\ E(\phi ?) & \leftrightarrow & \bot \end{array}$$

the axiom stating that **abort** leads to error and to no other state,

$$E(\textbf{abort}) \wedge [\textbf{abort}]\bot$$

and the rules of inference modus ponens (from ϕ_1 and $\phi_1 \rightarrow \phi_2$ infer ϕ_2) and program necessitation (from ϕ infer $[\pi]\phi$).

Let us see now how this applies to presupposition projection, Given a pair of formulas consisting of a presupposition *pres* and an assertion *ass*, the general recipe of forging a program from this is by means of

$$(\neg \mathit{pres}?; \textbf{abort}) \cup \mathit{ass}?$$

This uses the toolset of dynamic logic (compare the entry: Propositional Dynamic Logic) to enforce the desired behaviour: if the presupposition is not fulfilled the program aborts and otherwise the program behaves as a test for the assertion.

Apply this to the case of being a bachelor. The presupposition is being male and being adult (say $m \wedge a$), and the assertion is being unmarried, for which we use n. According to the recipe above the program **bachelor** is defined as

$$(\neg(m \wedge a)?; \textbf{abort}) \cup n?.$$

Similarly, being male has presupposition $\top$ and assertion m, so the program **male** is defined as $\bot?; \textbf{abort} \cup m?$, which reduces to $m?$. What this says is that **male** is a program without presupposition (the program never aborts), whereas **bachelor** does have a presupposition (the program aborts if the test $m \wedge a$ fails).

To get the assertion back from a program π, we can use

$$\langle \pi \rangle \top$$

which gives the conditions under which the program has at least one transition that does not lead to *error*. Here is a proof, for a program π of the form $(\neg \mathit{pres}?; \mathbf{abort}) \cup \mathit{ass}?$:

$$\begin{aligned}
\langle (\neg \mathit{pres}?; \mathbf{abort}) \cup \mathit{ass}? \rangle \top &\equiv \langle \neg \mathit{pres}?; \mathbf{abort} \rangle \top \vee \langle \mathit{ass}? \rangle \top \\
&\equiv \langle \neg \mathit{pres}? \rangle \langle \mathbf{abort} \rangle \top \vee \mathit{ass} \\
&\equiv \langle \neg \mathit{pres}? \rangle \bot \vee \mathit{ass} \\
&\equiv (\neg \mathit{pres} \wedge \bot) \vee \mathit{ass} \\
&\equiv \mathit{ass}
\end{aligned}$$

To get the presupposition, we can use

$$\neg E(\pi)$$

which gives the conditions under which the program will not have a transition to *error*. Here is a proof:

$$\begin{aligned}
E((\neg \mathit{pres}?; \mathbf{abort}) \cup \mathit{ass}?) &\equiv E(\neg \mathit{pres}?; \mathbf{abort}) \vee E(\mathit{ass}?) \\
&\equiv E(\neg \mathit{pres}?) \vee \langle \neg \mathit{pres}? \rangle \mathbf{abort}) \vee \bot \\
&\equiv \bot \vee \langle \neg \mathit{pres}? \rangle E(\mathbf{abort}) \vee \bot \\
&\equiv \langle \neg \mathit{pres}? \rangle E(\mathbf{abort}) \\
&\equiv \neg \mathit{pres} \wedge \top \\
&\equiv \neg \mathit{pres}
\end{aligned}$$

It follows that $\neg E((\neg \mathit{pres}?; \mathbf{abort}) \cup \mathit{ass}?) \equiv \mathit{pres}$.

Now consider the composition **male**; **bachelor**, the result of first uttering **male**, and next **bachelor**. The assertion of this composed utterance is:

$$\begin{aligned}
\langle \mathbf{male}; \mathbf{bachelor} \rangle \top &\equiv \langle \mathbf{male} \rangle \langle \mathbf{bachelor} \rangle \top \\
&\equiv \langle \mathbf{male} \rangle n \\
&\equiv m \wedge n.
\end{aligned}$$

Its presupposition is

$$\begin{aligned}
\neg E(\mathbf{male}; \mathbf{bachelor}) &\equiv \neg E(\mathbf{male}) \wedge \neg \langle \mathbf{male} \rangle E(\mathbf{bachelor}) \\
&\equiv \top \wedge [\mathbf{male}] \neg E(\mathbf{bachelor}) \\
&\equiv m \rightarrow (m \wedge a) \\
&\equiv m \rightarrow a.
\end{aligned}$$

program	$\pi_1; \pi_2$
assertion	$ass_{\pi_1} \wedge ass_{\pi_2}$
presupposition	$pres_{\pi_1} \wedge (ass_{\pi_1} \rightarrow pres_{\pi_2})$

Figure 1: Projection table for sequential composition.

program	**not** π
assertion	$\neg ass_{\pi}$
presupposition	$pres_{\pi}$

Figure 2: Projection table for negation.

The problem of presupposition projection, for our language of programs with error abortion, is to express the presupposition of π in terms of presuppositions and assertions of its components. For that, it is useful to define ass_π as $\langle\pi\rangle\top$ (or, equivalently, as $\neg[\pi]\bot$), and $pres_\pi$ as $\neg E(\pi)$.

It is a simple logical exercise to express the assertion and presupposition of $\pi_1; \pi_2$ in terms of the assertions and presuppositions of its components. The result is in Table 1.

What does it mean to negate a program π? The most straightforward requirement is let **not** π be a test that succeeds if π fails, and that aborts with error if π aborts. The following definition of **not** π implements this:

$$\mathbf{not}\ \pi \ :\equiv\ (E(\pi)?;\mathbf{abort}) \cup ([\pi]\bot)?$$

Using this to work out the meaning of **not bachelor**, we get:

$$\begin{aligned}\mathbf{not\ bachelor} &\equiv (E(\mathbf{bachelor})?;\mathbf{abort}) \cup ([\mathbf{bachelor}]\bot)? \\ &\equiv (\neg(m \wedge a)?;\mathbf{abort}) \cup ([\mathbf{bachelor}]\bot)? \\ &\equiv (\neg(m \wedge a)?;\mathbf{abort}) \cup \neg n?\end{aligned}$$

Again, it is a simple logical exercise to express the assertion and presupposition of **not** π in terms of assertion and presupposition of its component π. See Table 2.

The implication of $\pi_1 \Rightarrow \pi_2$ has as natural definition **not** $(\pi_1; \mathbf{not}\ \pi_2)$, and

program	$\pi_1 \Rightarrow \pi_2$
assertion	$ass_{\pi_1} \rightarrow ass_{\pi_2}$
presupposition	$pres_{\pi_1} \wedge (ass_{\pi_1} \rightarrow pres_{\pi_2})$

Figure 3: Projection table for implication.

its projection behaviour can be worked out from this definition.

$$\begin{array}{rcl}
\langle \pi_1 \Rightarrow \pi_2 \rangle \top & \equiv & \langle \mathbf{not}\ (\pi_1; \mathbf{not}\ \pi_2)\rangle \top \\
& \equiv & \neg\langle \pi_1; \mathbf{not}\ \pi_2 \rangle \top \\
& \equiv & \neg\langle \pi_1 \rangle \langle \mathbf{not}\ \pi_2 \rangle \top \\
& \equiv & \neg\langle \pi_1 \rangle \neg\langle \pi_2 \rangle \top \\
& \equiv & [\pi_1]\langle \pi_2 \rangle \top \\
& \equiv & [\pi_1] ass_{\pi_2} \\
& \equiv & [\neg pres_{\pi_1}?; \mathbf{abort} \cup ass_{\pi_1}?] ass_{\pi_2} \\
& \equiv & [\neg pres_{\pi_1}?; \mathbf{abort}] ass_{\pi_2} \wedge [ass_{\pi_1}?] ass_{\pi_2} \\
& \equiv & [\neg pres_{\pi_1}?][\mathbf{abort}] ass_{\pi_2} \wedge [ass_{\pi_1}?] ass_{\pi_2} \\
& \equiv & \neg pres_{\pi_1} \rightarrow \top \wedge [ass_{\pi_1}?] ass_{\pi_2} \\
& \equiv & [ass_{\pi_1}?] ass_{\pi_2} \\
& \equiv & ass_{\pi_1} \rightarrow ass_{\pi_2}
\end{array}$$

The calculation of presupposition failure conditions:

$$\begin{array}{rcl}
E(\pi_1 \Rightarrow \pi_2) & \equiv & E(\mathbf{not}\ (\pi_1; \mathbf{not}\ \pi_2)) \\
& \equiv & E(\pi_1; \mathbf{not}\ \pi_2) \\
& \equiv & E(\pi_1) \vee \langle \pi_1 \rangle E(\mathbf{not}\ \pi_2) \\
& \equiv & E(\pi_1) \vee \langle \pi_1 \rangle E(\pi_2) \\
& \equiv & \neg pres_{\pi_1} \vee (ass_{\pi_1} \wedge \neg pres_{\pi_2})
\end{array}$$

It follows that $pres(\pi_1 \Rightarrow \pi_2) \equiv \neg E(\pi_1 \Rightarrow \pi_2) \equiv pres_{\pi_1} \wedge (ass_{\pi_1} \rightarrow pres_{\pi_2})$. Table 3 gives the projection table for implication.

Applying this to the example of bachelorhood we get, using the facts that

program	π_1 **or** π_2
assertion	$ass_{\pi_1} \vee (pres_{\pi_1} \wedge ass_{\pi_2})$
presupposition	$pres_{\pi_1} \wedge (\neg ass_{\pi_1} \rightarrow pres_{\pi_2})$

Figure 4: Projection table for (sequential) disjunction.

$ass_{\mathbf{male}} \equiv m$, $pres_{\mathbf{male}} \equiv \top$, $ass_{\mathbf{bachelor}} \equiv n$ and $pres_{\mathbf{bachelor}} \equiv m \wedge a$:

$$\begin{aligned} & \mathbf{male} \Rightarrow \mathbf{bachelor} \\ \equiv\ & \mathbf{not}\ (\mathbf{male}; \mathbf{not\ bachelor}) \\ \equiv\ & (m \rightarrow \neg(m \wedge a)?; \mathbf{abort} \cup (m \rightarrow n)? \\ \equiv\ & (m \rightarrow \neg a)?; \mathbf{abort} \cup (m \rightarrow n)? \end{aligned}$$

Finally, what does it mean to process two programs π_1 and π_2 disjunctively? Taking linear order into account, one proceeds one by one: first execute π_1; if this succeeds then done, otherwise execute π_2. This leads to the following definition of π_1 **or** π_2:

$$\pi_1 \ \mathbf{or}\ \pi_2 :\equiv \pi_1 \cup (\mathbf{not}\ \pi_1; \pi_2).$$

Again we apply this to our running example:

$$\begin{aligned} & \mathbf{male\ or\ bachelor} \\ \equiv\ & \mathbf{male} \cup (\mathbf{not\ male}; \mathbf{bachelor}) \\ \equiv\ & \mathbf{male} \cup (\neg m?; \mathbf{bachelor}) \\ \equiv\ & \mathbf{male} \cup (\neg m?; \mathbf{abort}) \end{aligned}$$

The projection table for this is given in Table 4.

8.3 Presupposition Accommodation

In many cases where a presupposition of an utterance is violated, the utterance is nevertheless understood. This is called presupposition accommodation: the audience implicitly understands the presupposition as an additional assertion.

In our technical framework, we can define an operation **ACC** mapping utterances π to $\mathbf{ACC}(\pi)$ by accommodating their presuppositions. The definition of **ACC** is given by

$$\mathbf{ACC}(\pi) := \mathit{pres}_\pi?; \mathit{ass}_\pi?$$

For the running example case of **bachelor**, we get:

$$\mathbf{ACC}(\mathbf{bachelor}) = (m \wedge a)?; n?$$

The presupposition of $\mathbf{ACC}(\pi)$ is always $\top$.

8.4 Presuppositions and Dynamic Epistemic Logic

Epistemic logic, the logic of knowledge, is a branch of modal logic where the modality 'i knows that' is studied (compare the entries: Epistemic Logic, The Logic of Belief Revision). The dynamic turn in epistemic logic, which took place around 2000, introduced a focus on change of state, but now with states taken to be representations of the knowledge of a set of agents.

If we fix a set of basic propositions P and a set of agents I, then a knowledge state for P and I consists of a set W of possible worlds, together with a valuation function V that assigns a subset of P to each w in W (if $w \in W$, then $V(w)$ lists the basic propositions that are true in w) and for each agent $i \in I$, a relation R_i stating the epistemic similaries for i (if wR_iw', this means that agent i cannot distinguish world w from world w'). Epistemic models $M = (W, V, \{R_i \mid i \in I\})$ are known as multimodal Kripke models. Pointed epistemic models are epistemic models with a designated world w_0 representing the actual world.

What happens to a given epistemic state $(M, w_0) = ((W, V, \{R_i \mid i \in I\}), w_0)$ if a public announcement ϕ is made? Intuitively, the world set W of M is restricted to those worlds $w \in W$ where ϕ holds, and the valuation function V and epistemic relations R_i are restricted accordingly. Call the new model $M \mid \phi$. In case ϕ is true in w_0, the meaning of the public announcement ϕ can be viewed as a map from (M, w_0) to $(M \mid \phi, w_0)$. In case ϕ is false in w_0, no update is possible.

Veltman's update logic can be accommodated in public announcement logic (compare the entry: Common Knowledge) by allowing public announcements of the form $\Diamond\phi$, where the modality is read as reachability under common knowledge. If an S5 knowledge state for a set of agents (compare the entry: Epistemic Logic) is updated with the public announcement $\Diamond\phi$, then in case ϕ is true somewhere in the model, the update changes nothing (for in this case $M \mid \Diamond\phi$ equals M), and otherwise the update yields inconsistency (since public announcements are assumed to be true). This is in accordance with the update logic definition.

The logical toolbox for epistemic logic with communicative updates is called dynamic epistemic logic or DEL. DEL started out from the analysis of the epistemic and doxastic effects of public announcements [Pla89, Ger99]. Public announcement is interesting because it creates common knowledge. There is a variety of other kinds of announcement — private announcements, group announcements, secret sharing, lies, and so on — that also have well-defined epistemic effects. A general framework for a wide class of update actions was proposed in [BMS99] and [BM04]. A further generalization to a complete logic of communication and change, with enriched actions that allow changing the facts of the world, is provided in [BvEK06]. A textbook treatment of dynamic epistemic logic is given in [DvdHK06].

To flesh out what "transition to an error state" means one may represent the communicative situation of a language utterance with presupposition in more detail, as follows. Represent what a speaker assumes about her audience knows or believes, in a multi-agent belief (or knowledge) state, and model the effect of the communicative action on the belief state.

A simple way to handle utterances with presupposition in dynamic epistemic logic is by modelling a presupposition P as a public announcement "it is common knowledge that P". In cases where it is indeed common knowledge that P, an update with this information changes nothing. In cases where P is not common knowledge, however, the utterance is false, and public announcements of falsehoods yield an inconsistent knowledge state: the analogue of the error state in Subsection 8.2 above.

9 Encoding Dynamics in Typed Logic

Compositionality has always been an important concern in the use of logical systems in natural language semantics (see the entry: Compositionality). Through the use of higher order logics (see the entries: Second-order and Higher-order Logics, Church's Type Theory) a thoroughly compositional account of, e.g., the quantificational system of natural language can be achieved, as is demonstrated in classical Montague Grammar [Mon74a, Mon74b, Mon73] (compare the entry: Logical Form). We will review how the dynamic approach can be extended to higher order systems. The link between dynamic semantics and type theory is more like a liaison than a stable marriage: there is no intrinsic need for the connection. The connection is treated here to explain the historical influence of dynamic semantics on Montague grammar.

Most proposals for dynamic versions of Montague grammar develop what are in fact higher order versions of Dynamic Predicate Logic (DPL). This holds for [GS90, Chi92, Mus95, Mus96, Mus94, Eij97, EK97, KKP96, Kus00]. These systems all inherit a feature (or bug) from the DPL approach: they make re-assignment destructive. DRT does not suffer from this problem: the discourse representation construction algorithms of [Kam81] and [KR93] are stated in terms of functions with finite domains, and carefully talk about 'taking a fresh discourse referent' to extend the domain of a verifying function, for each new noun phrase to be processed.

In extensional Montague grammar 'a man' translates as:

$$\lambda P \exists x(\text{man } x \wedge Px).$$

Here P, of type $e \to t$, is the variable for the VP slot: it is assumed that VPs denote sets of entities.

In Dynamic Montague Grammar (DMG) in the style of [GS90], the translation of an indefinite NP does introduce an anaphoric index. The translation of 'a man' would look like

$$\lambda P \lambda a \lambda a'.\exists x(\text{man } x \wedge Pu_i(u_i|x)aa').$$

Instead of the basic types e and t of classical extensional Montague grammar, DMG hass basic types e, t and m (m for marker). States pick out entities for markers, so they can be viewed as objects of type $m \to e$. Abbreviating $m \to e$ as s (for 'state'), we call objects of type $s \to s \to t$ state transitions. The variable P in the DMG translation of 'a man' has type $m \to s \to s \to t$, so VP meanings have been lifted from type $e \to t$ to this type. Note that $\to$ associates to the right, so $m \to s \to s \to t$ is shorthand for $m \to (s \to (s \to t))$.

Indeed, DMG can be viewed as the result of systematic replacement of entities by markers and of truth values by state transitions. A VP meaning for 'is happy' is a function that maps a marker to a state transition. The state transition for marker u_i will check whether the input state maps u_i to a happy entity, and whether the output context equals the input context.

The variables a, a' range over states, and the expression $(u_i|x)a$ denotes the result of resetting the value of u_i in a to x, so the old value of u_i gets destroyed (destructive assignment).

The anaphoric index i on reference marker u_i is introduced by the translation. In fact, the translation starts from the indexed indefinite noun phrase 'a man$_i$'.

An alternative treatment is given in Incremental Typed Logic (ITL), an extension to typed logic of a 'stack semantics' that is based on variable free indexing and that avoids the destructive assignment problem. The basic idea of the stack semantics for DPL, developed in [Ver93b], is to replace the destructive assignment of ordinary DPL, which throws away old values when resetting, by a stack valued one, that allows old values to be re-used. Stack valued assignments assign to each variable a stack of values, the top of the stack being the current value. Existential quantification pushes a new value on the stack, but there is also the possibility of popping the stack, to re-use a previously assigned value. ITL [Eij00] is in fact a typed version of stack semantics, using a single stack.

Assuming a domain of entities, contexts are finite lists of entities. If c is a context of length n, then we refer to its elements as $c[0]$, …, $c[n-1]$, and to its length as $|c|$. We will refer to the type of contexts of length i as $[e]^i$. If c is a context in $[e]^i$, then objects of type $\{0, .., i-1\}$ can serve as indices into c. If $c \in [e]^i$ and $j \in \{0, .., i-1\}$, then $c[j]$ is the object of type e that occurs at position j in the context.

A key operation on contexts is extension with an element. If $c :: [e]^i$ and $x :: e$ (c is a context of length i and x is an entity) then $c\hat{}x$ is the context of length $i+1$ that has elements $c[0], \ldots, c[i-1], x$. Thus $\hat{}$ is an operator of type $[e]^i \to e \to [e]^{i+1}$.

Also note that types like $[e]^i$ are in fact polymorphic types, with i acting as a type variable. See [Mil78].

In ITL there is no destructive assignment, and indefinite noun phrases do not carry indexes in the syntax. The ITL translation of 'a man' picks up an index from context, as follows:

$$\lambda P \lambda c \lambda c'. \exists x (\text{man } x \wedge P|c|(c\hat{}x)c').$$

Here P is a variable of type $\{0, .., i\} \to [e]^{i+1} \to [e]^j \to t$, while c is a variable of type $[e]^i$ representing the input context of length i, and c' is a variable of type $[e]^j$ representing the output context. Note that the type $\{0,..,i\} \to [e]^{i+1} \to [e]^j \to t$ for P indicates that P first takes an index in the range $\{0,..,i\}$, next a context fitting this range (a context of length $i+1$), next a context of a yet unknown length, and then gives a truth value. P is the type of unary predicates, lifted to the level of context changers, as follows. Instead of using a variable to range over objects to form an expression of type e, a lifted predicate uses a variable ranging over the size of an input context to form an expression that denotes a changer for that context.

The ITL translation of 'a man' has type $(\{0,..,i\} \to [e]^{i+1} \to [e]^j \to t) \to [e]^i \to [e]^j \to t$. In $P|c|(c\hat{}x)c'$, the P variable marks the slot for the VP interpretation; $|c|$ gives the length of the input context to P; it picks up the value i, which is the position of the next available slot when the context is extended. This slot is filled by an object x denoting a man. Note that $c\hat{}x[|c|] = c\hat{}x[i] = x$, so the index i serves to pick out that man from the context.

To see that a dynamic higher order system is expressible in ITL, it is enough to show how to define the appropriate dynamic operations. Assume ϕ and ψ have the type of context transitions, i.e. type $[e] \to [e] \to t$ (using $[e]$ for arbitrary contexts), and that c, c', c'' have type $[e]$. Then we can define the dynamic existential quantifier, dynamic negation and dynamic composition as follows:

$$\begin{array}{rcl} \mathcal{E} & := & \lambda cc'.\exists x(c\hat{}x = c') \\ {\sim}\phi & := & \lambda cc'.(c = c' \wedge \neg\exists c''\phi cc'') \\ \phi \mathbin{;} \psi & := & \lambda cc'.\exists c''(\phi cc'' \wedge \psi c''c') \end{array}$$

Dynamic implication, $\Rightarrow$, is defined in the usual way, by means of ${\sim}(\phi \mathbin{;} {\sim}\psi)$.

10 Conclusion

Hopefully, the above has given the reader a sense of Dynamic Semantics as a fruitful and flexible approach to meaning and information processing. Dynamic semantics comes with a set of flexible tools, and with a collection of 'killer applications', such as the compositional treatment of Donkey sentences, the account of anaphoric linking, the account of presupposition projection, and the account of epistemic updating. It is to be expected that advances in dynamic epistemic logic will lead to further integration. Taking a broader perspective, dynamic semantics can be viewed as the application of dynamic epistemic logic in natural language semantics, although this is controversial.

References

[Alo97] M. Aloni. Quantification in dynamic semantics. In P. Dekker, editor, *Proceedings of the Eleventh Amsterdam Colloquium*, pages 73–78, 1997.

[BBar] P. Blackburn and J. Bos. *Working with Discourse Representation Theory: An Advanced Course in Computational Semantics.* CSLI Publications, To appear. Available from `http://homepages.inf.ed.ac.uk/jbos/comsem/book2.html`.

[Bea97] D. Beaver. Presupposition. In J. van Benthem and A. ter Meulen, editors, *Handbook of logic and Language*, pages 939–1008. Elsevier, 1997.

[Bea01] D. Beaver. *Presupposition and Assertion in Dynamic Semantics.* CSLI Publications, Stanford, 2001.

[Ben89] J. van Benthem. Semantic parallels in natural language and computation. In H.-D. Ebbinghaus et al., editors, *Logic Colloquium, Granada, 1987*, pages 331–375, Amsterdam, 1989. Elsevier, Amsterdam.

[Ben96] J. van Benthem. *Exploring Logical Dynamics.* CSLI & Folli, 1996.

[Ber96] M.H. van den Berg. *The Internal Structure of Discourse.* PhD thesis, ILLC Dissertation Series 1996-3, March 1996.

[BM04] A. Baltag and L.S. Moss. Logics for epistemic programs. *Synthese*, 139(2):165–224, 2004.

[BMS99] A. Baltag, L.S. Moss, and S. Solecki. The logic of public announcements, common knowledge, and private suspicions. Technical Report SEN-R9922, CWI, Amsterdam, 1999. With many updates.

[BvEK06] J. van Benthem, J. van Eijck, and B. Kooi. Logics of communication and change. *Information and Computation*, 204(11):1620–1662, 2006.

[Chi92] G. Chierchia. Anaphora and dynamic binding. *Linguistics and Philosophy*, 15(2):111–183, 1992.

[Dek93] P. Dekker. *Transsentential Meditations, ups and downs in dynamic semantics.* PhD thesis, University of Amsterdam, ILLC, May 1993.

[DvdHK06] H.P. van Ditmarsch, W. van der Hoek, and B. Kooi. *Dynamic Epistemic Logic*, volume 337 of *Synthese Library*. Springer, 2006.

[EdV92] J. van Eijck and F.J. de Vries. Dynamic interpretation and Hoare deduction. *Journal of Logic, Language, and Information*, 1:1–44, 1992.

[Eij94] J. van Eijck. Presupposition failure – a comedy of errors. *Aspects of Computing*, 6A:766–787, 1994.

[Eij97] J. van Eijck. Typed logics with states. *Logic Journal of the IGPL*, 5(5):623–645, 1997.

[Eij00] J. van Eijck. The proper treatment of context in NL. In Paola Monachesi, editor, *Computational Linguistics in the Netherlands 1999; Selected Papers from the Tenth CLIN Meeting*, pages 41–51. Utrecht Institute of Linguistics OTS, 2000.

[EK97] J. van Eijck and H. Kamp. Representing discourse in context. In J. van Benthem and A. ter Meulen, editors, *Handbook of Logic and Language*, pages 179–237. Elsevier, Amsterdam, 1997.

[Gea80] P.T. Geach. *Reference and Generality: An Examination of Some Medieval and Modern Theories.* Cornell University Press, Ithaca, 1962 (Third revised edition: 1980).

[Ger99] J. Gerbrandy. Dynamic epistemic logic. In L.S. Moss et al., editors, *Logic, Language and Information, Vol. 2.* CSLI Publications, Stanford, 1999.

[Gro95] W. Groeneveld. *Logical investigations into dynamic semantics.* PhD thesis, University of Amsterdam, 1995.

[GS90] J. Groenendijk and M. Stokhof. Dynamic Montague Grammar. In L. Kalman and L. Polos, editors, *Papers from the Second Symposium on Logic and Language*, pages 3–48. Akademiai Kiadoo, Budapest, 1990.

[GS91a] J. Groenendijk and M. Stokhof. Dynamic predicate logic. *Linguistics and Philosophy*, 14:39–100, 1991.

[GS91b] 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.

[GSV96] J. Groenendijk, M. Stokhof, and F. Veltman. Coreference and modality. In S. Lappin, editor, *Handbook of Contemporary Semantic Theory*, pages 179–213. Blackwell, Oxford, 1996.

[Hei83a] 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.

[Hei83b] I. Heim. On the projection problem for presuppositions. *Proceedings of the West Coast Conference on Formal Linguistics*, 2:114–126, 1983.

[HV96] M. Hollenberg and C. Vermeulen. Counting variables in a dynamic setting. *Journal of Logic and Computation*, 6(5):725–744, 1996.

[Kam81] H. Kamp. A theory of truth and semantic representation. In J. Groenendijk, T. Janssen, and M. Stokhof, editors, *Formal Methods in the Study of Language*, pages 277–322. Mathematisch Centrum, Amsterdam, 1981.

[Kar73] L. Karttunen. Presuppositions of compound sentences. *Linguistic Inquiry*, 4:169–193, 1973.

[Kar74] L. Karttunen. Presupposition and linguistic context. *Theoretical Linguistics*, pages 181–194, 1974.

[KKP96] M. Kohlhase, S. Kuschert, and M. Pinkal. A type-theoretic semantics for λ-DRT. In P. Dekker and M. Stokhof, editors, *Proceedings of the Tenth Amsterdam Colloquium*, Amsterdam, 1996. ILLC.

[KR93] H. Kamp and U. Reyle. *From Discourse to Logic.* Kluwer, Dordrecht, 1993.

[Kra95] E. Krahmer. *Discourse and Presupposition.* PhD thesis, Tilburg University, 1995.

[Kus00] S. Kuschert. *Dynamic Meaning and Accommodation.* PhD thesis, Universität des Saarlandes, 2000. Thesis defended in 1999.

[MBV97] R. Muskens, J. van Benthem, and A. Visser. Dynamics. In J. van Benthem and A. ter Meulen, editors, *Handbook of Logic and Language*, pages 587–648. Elsevier, Amsterdam & MIT Press, Cambridge, 1997.

[Mil78] R. Milner. A theory of type polymorphism in programming. *Journal of Computer and System Sciences*, 17, 1978.

[Mon73] R. Montague. The proper treatment of quantification in ordinary English. In J. Hintikka, editor, *Approaches to Natural Language*, pages 221–242. Reidel, 1973.

[Mon74a] R. Montague. English as a formal language. In R.H. Thomason, editor, *Formal Philosophy; Selected Papers of Richard Montague*, pages 188–221. Yale University Press, New Haven and London, 1974.

[Mon74b] R. Montague. Universal grammar. In R.H. Thomason, editor, *Formal Philosophy; Selected Papers of Richard Montague*, pages 222–246. Yale University Press, New Haven and London, 1974.

[Mus91] R. Muskens. Anaphora and the logic of change. In J. van Eijck, editor, *JELIA '90, European Workshop on Logics in AI*, Springer Lecture Notes, pages 414–430. Springer, 1991.

[Mus94] R. Muskens. A compositional discourse representation theory. In P. Dekker and M. Stokhof, editors, *Proceedings 9th Amsterdam Colloquium*, pages 467–486. ILLC, Amsterdam, 1994.

[Mus95] R. Muskens. Tense and the logic of change. In U. Egli et al., editor, *Lexical Knowledge in the Organization of Language*, pages 147–183. W. Benjamins, 1995.

[Mus96] R. Muskens. Combining Montague Semantics and Discourse Representation. *Linguistics and Philosophy*, 19:143–186, 1996.

[Pla89] J. A. Plaza. Logics of public communications. In M. L. Emrich, M. S. Pfeifer, M. Hadzikadic, and Z. W. Ras, editors, *Proceedings of the 4th International Symposium on Methodologies for Intelligent Systems*, pages 201–216, 1989.

[Put75] Hilary Putnam. The meaning of 'meaning'. In *Philosophical Papers, Vol 2*. Cambridge University Press, 1975.

[San92] R.A. van der Sandt. Presupposition projection as anaphora resolution. *Journal of Semantics*, 9:333–377, 1992. Special Issue: Presupposition, Part 2.

[Sch07] Philippe Schlenker. Anti-dynamics: Presupposition projection without dynamic semantics. *Journal of Logic, Language and Information*, 16(3):325–356, 2007.

[Seu85] P. Seuren. *Discourse Semantics*. Blackwell, Oxford, 1985.

[Sta72] R. Stalnaker. Pragmatics. In D. Davidson and G. Harman, editors, *Semantics of Natural Language*, pages 380–397. Reidel, 1972.

[Sta74] R. Stalnaker. Pragmatic presuppositions. In M.K. Munitz and P.K. Unger, editors, *Semantics and Philosophy*, pages 197–213. New York University Press, 1974.

[Vel91] F. Veltman. Defaults in update semantics. In H. Kamp, editor, *Conditionals, Defaults and Belief Revision*. Dyana Deliverable R2.5A, Edinburgh, 1991.

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

[Ver93a] C.F.M. Vermeulen. Sequence semantics for dynamic predicate logic. *Journal of Logic, Language and Information*, 2:217–254, 1993.

[Ver93b] C.F.M. Vermeulen. Sequence semantics for dynamic predicate logic. *Journal of Logic, Language, and Information*, 2:217–254, 1993.

[Ver94] C.F.M. Vermeulen. *Explorations of the Dynamic Environment.* PhD thesis, Utrecht University, september 1994.

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

[Wer00] Paul Werth. *Text Words: Representing Conceptual Space in Discourse.* Pearson Education/Longman, 2000.

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

Related Entries

Anaphora, Assertion, Classical Logic, Common Knowledge, Computational Theory of Mind, Conditionals, Compositionality, Discourse Representation Theory, Epistemic Conceptualism, Epistemic Logic, First-order Model Theory, Implicature, Indexicals, Logical Form, Mental content: Externalism about mental content (content-externalism), Mental content: narrow mental content (content-narrow), The Logic of Belief Revision, The Logic of Conditionals, The Mathematics of Boolean Algebra, Pragmatics, Propositional Dynamic Logic, Reference, Scientific Realism, Semantic Conceptions of Information, Speech Acts, Tarski's Truth Definitions.