Chapter 8
On Groenendijk and Stokhof's "Dynamic Predicate Logic"

Anthony S. Gillies

Abstract This paper can serve as a gentle introduction to (but not exhaustive overview of) Groenendijk and Stokhof's classic paper "Dynamic predicate logic". If you are not new to DPL and so not in need of an introduction to DPL, this paper can alternatively be read as an invitation to re-read it.

Keywords Dynamic semantics · Dynamic logic · Update semantics · Anaphora

Introduction

Sometimes it happens that commentaries on or textbook treatments of important theories or ideas are better to read than the originals. On these occasions the commentary or textbook version is able to turn the benefits of time, hindsight, and intellectual division of labor into a cleaner or more elegant presentation of the essential insights of the original. This is not such an occasion.

Instead this paper is an introduction to, or alternatively an invitation to re-read, Groenendijk and Stokhof's (1991) "Dynamic predicate logic" (DPL): a useful warm-up exercise containing some background and sign-posts, gestures at some nearby connections, and (hopefully) something that will enrich the DPL experience.[1]

Here is what you can expect. We will begin (section "Farmers, Donkeys, Files, Boxes") with stage-setting: an argument that no quantificational analysis of indefinites can be right and how this set the stage for DPL. Section "Dynamic Worldview" is an introduction to the dynamic worldview, got at by a gentle introduction to (propositional) dynamic logic. Sections "DPL Basics: $\wedge$ and $\exists$"

[1] I will use 'DPL' to pick out both the system of dynamic predicate logic and the article itself, hoping that context will disambiguate.

A. S. Gillies (✉)
Department of Philosophy, University of Arizona, Tucson, AZ, USA

L. McNally, Z. G. Szabó (eds.), *A Reader's Guide to Classic Papers in Formal Semantics*, Studies in Linguistics and Philosophy 100,
https://doi.org/10.1007/978-3-030-85308-2_8

and "DPL Basics: ¬, →, and ∀ (and ∨, Too)" illustrate the mechanics of the core of canonical DPL (what you will find in the paper itself) and section "Truth, Entailment, Equivalence" is an overview of the theoretical properties of the system (truth, entailment, equivalence). Sections "Updating Information States, DPL Updates, and Dynamic Differences" shift gears a bit, recasting DPL as an update system and looking a little into the ways in which such systems can depart from classical frameworks. Section "Extensions, Amendments, Revisions" concludes by offering a pointer to one main area of subsequent work in the spirit of DPL. The appendix contains some official definitions.

Farmers, Donkeys, Files, Boxes

Suppose we focus on a small fragment of natural language, one where we can talk about a collection of farmers, livestock, and some properties they have. This kind of fragment is one where the tools of first-order logic can be put to good use. So assume that we have a language L_{Fo} of first order logic with individual variables $x, y, \ldots$, a stock of predicates, negation ($\neg$), conjunction ($\wedge$), and the existential quantifier ($\exists$).[2]

Some things we might want to report:

(1) a. A farmer walked in the field and he whistled.

b. A farmer walked in the field. He whistled.

There is not much mystery what (1a) and (1b) express: both are true if and only if there is at least one farmer who walked in the field and whistled. But thinking about how they manage to express this (surprisingly) raises some hard issues.[3]

The anaphoric pronoun *he* in both examples seems to pick out a farmer that walked in the field, adding that that farmer whistled. This seems to suggest that the indefinite noun phrase *a farmer* that is the antecedent for this *he* also picks out that field-walking farmer and thus that the two devices are coreferential. But indefinite noun phrases don't in general seem to be devices for referring to individuals, much less co-referring. You can see this by embedding them in various ways, for instance under a wide-scope negation:

(2) Pedro doesn't own a donkey.

[2] See Definition 16 for the official definition. Throughout, I will assume a working knowledge of what it takes for sentences of L_{Fo} to be true in a model $M = \langle D, I \rangle$ with respect to an assignment function g.

[3] Numbering conventions: numbers-(plus-letters) (as in (1a) and (2) are for example sentences; number. Number labels (as in (8.1) and (8.10)) are for equations and formulas, with the first number picking out the section of the paper in which the equation/formula makes its first appearance.

Here *a donkey* doesn't refer to any donkey. So understanding the relationship between indefinites and pronouns as coreference doesn't in general seem right.

If not referring, then what? The other salient option is to treat *a farmer* as a quantificational expression. This makes a lot of sense. And given the humble fragment we are investigating, we can think of them as regular first-order existential quantificational expressions. After all, the examples seem to say that some farmer walked and whistled. This works well for (1a) but runs into problems for (1b).

$$\exists x(\texttt{farmer}\, x \wedge \texttt{walked-in-field}\, x \wedge \texttt{whistled}\, x) \tag{8.1}$$

$$\exists x(\texttt{farmer}\, x \wedge \texttt{walked-in-field}\, x) \wedge \texttt{whistled}\, x \tag{8.2}$$

With a little imagination and practice, you can see a natural path from (1a) to the representation in (8.1). On this way of thinking, the relationship between the indefinite *a farmer* and *he* is a familiar one: the occurrences of the variable x in $\texttt{farmer}\, x \wedge \texttt{walked-in-field}\, x$ and the occurrence of x in $\texttt{whistled}\, x$ all lie in the scope of the quantifier $\exists x$. This quantifier thus binds these variables. Interpret these first-order representations in the usual way and you therefore get the prediction that (8.1)—and so by extension (1a)—is true if and only if something is a farmer that walked in the field and is also something that whistled.

Things are trickier with (1b). Assume that our representations should go sentence-by-sentence and that a sequence of sentences can be usefully modeled as conjunction: however you represent *A farmer walked in the field* should be conjoined with how you represent *he whistled*. Then we arrive at (8.2). But now the x in $\texttt{whistled}\, x$ is not in the scope of $\exists x$ and so we don't predict the right relationship between it and something that is a farmer that walked in the field. As a result we get the wrong truth conditions: (8.2) is true in a model with respect to an assignment g if one farmer walked in the field, no farmers whistled, and $g(x)$ is a non-farmer who whistled.

The strict sentence-by-sentence approach prevented us from getting the right relationship between the quantificational apparatus and the downstream pronouns. Can we insist that the indefinite *a farmer* must have scope over *he whistled*? This gets us back to the representation in (1a) and does deliver the right truth conditions. But it is dodgy. How can the syntactic scope of indefinites cross sentence boundaries? Probably, it can't. This strategy also moves from taking the individual sentences in (1b) as the principal thing semantics assigns meanings to and shifts focus to the whole discourse.[4] With that comes a whiff of non-compositionality: what looked like a meaningful sentence in natural language (the first sentence of (1b)) isn't being interpreted on its own and the whole discourse therefore has a meaning not directly figure-out-able from the meanings of its parts.

[4] This is the strategy suggested in Geach (1962/1980). This discourse-first motto is taken up, but for a non-quantificational approach, in DRT (Kamp, 1981).

Whether or not this or some other approach can be defended in this case, things get worse when we again shift to other embedding environments. Notably, in so-called *donkey conditionals*:

(3) a. If a farmer owns a donkey, he feeds it hay.

b. Every farmer feeds hay to every donkey he owns.

The conditional in (3a) has a reading where it means what (3b) means. The hypothesis that *a farmer* and *a donkey* are existential quantifiers again runs into trouble. Two attempts:

$$\exists x(\texttt{farmer}\, x \wedge \exists y(\texttt{donkey}\, y \wedge \texttt{owns}\, xy)) \rightarrow \texttt{feeds-hay-to}\, xy \tag{8.3}$$

$$\exists x(\texttt{farmer}\, x \wedge \exists y((\texttt{donkey}\, y \wedge \texttt{owns}\, xy) \rightarrow \texttt{feeds-hay-to}\, xy)) \tag{8.4}$$

The representation in (8.3) is pretty good: it is a conditional and *a farmer owns a donkey* shows up as the conditional's antecedent, which seems right. But $\texttt{feeds-hay-to}\, xy$ is not in the scope of $\exists x$ or $\exists y$ and so the truth conditions are wrong: the bound reading of the pronouns in *he feeds it hay* are not predicted. Meanwhile the representation in (8.4) ensures that the existential quantifiers both reach all the way to the consequent. Insisting that indefinites always take wide scope is surprising—deep down (3a) isn't a conditional!—and not worth it since the truth conditions are wrong: (8.4) is true if there is a farmer and a donkey he doesn't own but (3a) seems to be true if and only if every farmer feeds every donkey he owns hay.

So: the relationship between indefinites and anaphoric pronouns doesn't seem to be the relationship of coreference and it doesn't seem to be the relationship of quantifier and a variable it scopes over. This is a rough version of arguments in both Kamp (1981) and Heim (1982) that aim to motivate non-quantificational (and non-referential) theories of indefinites and both Discourse Representation Theory (DRT) and File Change Semantics (FCS) deliver just that.[5]

Two features of DRT to keep in mind as a backdrop for reading DRT. First: like the Geach-strategy, it is discourses not sentences that are the primary bearers of semantic values. It is not in general true in DPL that the truth conditions of whole discourses can be decomposed into the truth conditions of the individual sentences that make them up. Second: the analysis of indefinites is thoroughly non-quantificational and representational (making heavy use of discourse referents and an intermediate and distinctive level of representation).[6] DRT can be seen as a reaction to both of these features.

[5] DRT was first developed in Kamp (1981); see Geurts (2019) for a reader's guide to it. The canonical version of DRT is developed in Kamp and Reyle (1993); see Geurts et al. (2016) for an introduction. FCS is developed in Heim (1982) and refined somewhat in Heim (1983); see Yalcin (2012) for a recent introduction to FCS.

[6] Discourse referents originate in Karttunen (1976).

But it is not a baby/bathwater situation: for the target fragment — roughly, anaphoric relationships that can be sensibly represented using the resources of L_{Fo} — DPL makes exactly the same empirical predictions as DRT.[7]

Dynamic Worldview

Famously, DPL embraces the dynamic perspective on meaning. Groenendijk and Stokhof put it this way:

> The general starting point of the kind of semantics that DPL is an instance of, is that the meaning of a sentence does not lie in its truth conditions, but rather in the way it changes (the representation of) the information of the interpreter. The utterance of a sentence brings us from a certain state of information to another one. The meaning of a sentence lies in the way it brings about such a transition. (DPL, p. 43)

This is powerful imagery and makes dynamic semantics seem in equal parts exotic and revolutionary.[8]

Before getting to DPL proper, and seeing how it makes this imagery concrete, let's look at a nearby idea: dynamic logics.[9] These are (modal) logics for reasoning about programming languages. They aren't the same thing as dynamic semantics, and that is useful. Like regular modal logics, sentences express propositions (sets of satisfying points) but they also interpret program-denoting parts of the language. Thus you can view the semantics for these logics as an intermediate stepping stone on the road to dynamic semantics.

Here we will just sketch the simplest such logic, propositional dynamic logic (PDL).[10]

In classical propositional modal logic, sentences are assigned meanings with respect to a model that includes a non-empty set of worlds and a binary accessibility relation over that set. All sentences have sets of worlds as meanings: $[\![\phi]\!]$ is the set of worlds at which ϕ is true, a proposition. For modal claims like $\Diamond\phi$ the propositions depend on what's accessible to what: $[\![\Diamond\phi]\!]$ is the set of w's such that there is an accessible world from w at which ϕ is true. So far, so familiar.[11]

[7] We won't go into the details, but Section 4.2 of DPL (pp. 76–83) is devoted to showing this.

[8] Dynamics really got a foothold in the 1980s and 1990s: dynamic semantics, belief dynamics, dynamic epistemic logic were all in boom cycles (see, for instance, Muskens et al. 1997). Thanks to the trio of David Beaver, Paul Dekker, and Willem Groeneveld (who, unbelievably in hindsight, taught an online course in dynamic semantics way back then) I got swept up by all of it. Heady times.

[9] A good textbook: Harel et al. (2000).

[10] This is a way to ease into the dynamic worldview and serves as preparation for tackling Section 4.3 of DPL, where you can see how DPL can be embedded into a fragment of Quantified Dynamic Logic.

[11] If not, see Gamut (1991).

PDL is similar in both language and interpretation but is expressively richer. It also has room to express things like `set the value of register b to 3` and `do this and then do that`. The language of PDL has both atomic *sentences* and atomic *programs*. Atomic sentences are sentences that are basic and have no interesting parts. So too for atomic programs: they represent the simple, non-decomposable actions. Sentences of PDL can be negated and conjoined to create more complex sentences, but there are also ways of making more complex programs out of simpler ones. For instance, you can *compose* them: $\pi_1; \pi_2$ is a program whenever π_1 and π_2 are programs. Declarative sentences and programs have two points of contact. First, given a sentence ϕ, you can form a *test*: $\phi?$ is a program whenever ϕ is a sentence. Second, the modalities are generated by programs: $\langle\pi\rangle\phi$ and $[\pi]\phi$ are sentences if π is a program and ϕ is a sentence. These take programs and sentences and create possibility and necessity claims in terms of them.

The semantics for PDL is built on two ideas, one familiar and the other new. The familiar: declarative sentences express propositions, the sets of points or worlds or (as they're called in this context) *states* at which they are true. The new: programs express what they do. What a program does depends on the state in which it is executed. Executing the program `take 3 steps to the right` when I am standing here results in a different state of affairs than it does when it is executed when I am standing over there. Same goes for all programs in whatever state. They link (input) states to (output) states. Which is to say: programs express relations between states.[12]

So, following the set-up for modal logic, a PDL model $M = \langle W, V\rangle$ is a non-empty set of states W and a function V that assigns atomic *sentences* propositions and atomic *programs* relations on W. Where p is an atomic sentence and a is an atomic program:

$$V(p) \subseteq W \text{ and } V(a) \subseteq W \times W \tag{8.5}$$

Just as $V(p)$ is the set of states s such that (atomic) p is true at s, $V(a)$ is the set of pairs of states $\langle s, t\rangle$ such that executing (atomic) a in s possibly lands you in state t.[13] From here we have to say how more complex meanings are assigned. Focus on three aspects of that assignment, leaving the rest to the official version in Definition 19.

First: possibility modal claims $\langle\pi\rangle\phi$. In PDL possibility is restricted twice over: $\langle\pi\rangle\phi$ at a state s says that ϕ is true at some state reachable *by executing* π in s.

$$s \in [\![\langle\pi\rangle\phi]\!] \text{ iff there is a } t \text{ s.t. } \langle s, t\rangle \in [\![\pi]\!] \text{ and } t \in [\![\phi]\!] \tag{8.6}$$

[12] Sentences and programs differ in semantic type, but still fit together. That is important because even though program constructs are not themselves formulas in the language, they do appear as parts of declarative formulas. The modalities are a case in point.

[13] Since program meanings are relations, not necessarily functions, we are allowing that programs are not deterministic. Hence the "possibly" in the gloss.

We have lots of modalities in PDL: two for each program we can construct. But where are the accessibility relations? They are the meanings of the programs! The compositional way that complex programs express relations on W as a function of the relations expressed by the simpler programs that make them up is key here.

Second: program composition. Whenever π_1 is a program and π_2 is a program we know $\pi_1; \pi_2$ is, too. And we know that $[\![\pi_1; \pi_2]\!]$ will serve as the accessibility relation for the modals $\langle \pi_1; \pi_2 \rangle$ and $[\pi_1; \pi_2]$. As the name suggests such composed programs express the composition of the relations expressed by the programs they are composed of.

$$[\![\pi_1; \pi_2]\!] = [\![\pi_1]\!] \circ [\![\pi_2]\!] \tag{8.7}$$

where $\circ$ is, literally, relational composition. Thus

$$\langle s, t \rangle \in [\![\pi_1; \pi_2]\!] \text{ iff, for some } r, \langle s, r \rangle \in [\![\pi_1]\!] \text{ and } \langle r, t \rangle \in [\![\pi_2]\!] \tag{8.8}$$

The linking here through the intermediate state r is key.

And third: test programs. These show the other way that things fit together by turning a sentence ϕ into a program $\phi?$. The point of such programs is to *check whether* ϕ is true. They don't themselves change the state you're in.

$$\langle s, t \rangle \in [\![\phi?]\!] \text{ iff } s = t \text{ and } s \in [\![\phi]\!] \tag{8.9}$$

Test programs are sometimes described as those programs only defined on the diagonal. Draw an array of ordered pairs of states: test programs denote subsets of the diagonal relation $\langle s, s, \rangle, \langle t, t \rangle, \langle u, u, \rangle, \ldots$.

PDL models can get represented graphically, as in Fig. 8.1: states are nodes, and each atomic program is represented by a labeled arrow (labeled by the program). Double-circled states represent *fixed-point* states: these are states where there is a successful test program pictured (i.e., a labeled self-loop).[14]

DPL Basics: $\wedge$ and $\exists$

Indefinites are shape-shifters, sometimes seeming to contribute existential truth conditions (as is in (1a) where *a farmer* scopes under *and*) and sometimes seeming to contribute universal force (as in (3a) where *a farmer* scopes under *if*). A quantificational analysis of indefinites thus seems hopeless. There is a loophole, though, and DPL exploits it.

[14] In the model in Fig. 8.1 suppose $[\![p]\!] = \{s, t\}$ and $[\![q]\!] = \{s, u\}$. It might be useful to put pencil to paper and figure out $[\![a; b]\!]$, $[\![(a; b); a]\!]$, $[\![q?]\!]$, and $[\![\langle c \rangle p]\!]$.

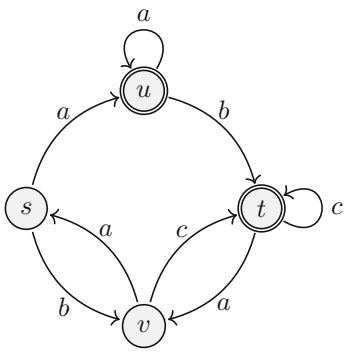

Fig. 8.1 PDL model

DPL offers a new way of interpreting a language of first order logic. It looks at programs in PDL for inspiration and says *every* sentence is a program. Thus meanings will all be relations: sets of ordered pairs of "states". This will provide a way of treating indefinites as existential quantifiers (albeit, dynamic ones) that can account for their shape-shifting behavior.

What counts as a state (of information) depends on what we are modeling. Given the limited focus here (extensional first order stuff) this means that we are tracking, with respect to a given first order model M, what we are talking about. Thus, information is *discourse information*: which variables are hooked up with which objects in D. That is information usefully encoded by assignment functions.

Definition 1 (Information States in (Relational) DPL) Let V be the set of variables of L_{Fo} and $M = \langle D, I \rangle$ be a FO model. The set S of DPL information states is the set:

$$g \in S \text{ iff } g \in D^V$$

where D^V is the set of assignment functions from V to D.

Putting the pieces together: sentences are programs or instructions for changing information. Information is information about variables. So sentences denote sets of input-output pairs of assignment functions.[15]

Definition 2 (Meanings in (Relational) DPL) For any ϕ:

$$[\![\phi]\!]^M \subseteq S \times S$$

[15] The general idea that interpreting things might both depend on and affect assignment functions can also be found in Barwise (1987).

This doesn't tell us just which relation is expressed by ϕ. We'll get to that. Intuitively: $\langle g, h\rangle \in [\![\phi]\!]$ iff executing (the program expressed by) ϕ in state g (possibly) terminates in state h. Since $[\![\phi]\!]$ is a relation I'll opt for infix notation, $g[\![\phi]\!]h$ being a somewhat snappier thing to read and write than $\langle g, h\rangle \in [\![\phi]\!]$.

What remains is to look at the specific ways different sorts of sentences say to change an information state g.[16] Begin with (1b) and its sentence-by-sentence representation (8.2). We will work up to the DPL treatment of this by talking about simple cases showing how the clauses in the official version of relational DPL work (Definition 20). By the end, we will see why this is true:

Fact 1 *$g[\![\exists x Fx \wedge Gx]\!]h$ only if $h(x) \in I(F)$ and $h(x) \in I(G)$.*

Start with the DPL meaning assigned to atomic formulas like Fx. This simply says that whatever we have been using x to talk about is one of the F's.[17] So Fx expresses a test program to see if that's the case.

$$g\,[\![Fx]\!]\,h \;\text{ iff : } h = g \text{ and } g(x) \in I(F) \tag{8.10}$$

Thus $[\![Fx]\!]$ contains those pairs $\langle g, g\rangle$, $\langle h, h\rangle$, $\langle f, f\rangle$, ... of assignments that map x to a thing in D that is in $I(F)$.

The instruction that $\exists x Fx$ expresses is different. It is a two step procedure. It tells us, first, to move from g to a state that differs at most from g in what it associates with x: the state $g[x]k$.[18] From this state, does interpreting Fx move you to h? The original g is linked to h by way of $[\![\exists x Fx]\!]$ if and only if the answer is yes.

$$g\,[\![\exists x Fx]\!]\,h \;\text{ iff : there is a } k \text{ s.t. } g[x]k \text{ and } k\,[\![Fx]\!]\,h \tag{8.11}$$

Given (8.10), the right hand side simplifies:

$$g\,[\![\exists x Fx]\!]\,h \text{ iff : } \begin{cases} \text{there is a } k \text{ s.t. } g[x]k \text{ and} \\ k = h \text{ and} \\ h(x) \in I(F) \end{cases} \tag{8.12}$$

That is:

[16] It is important that all of this happens within a given model M. We'll typically suppress mention of this, though.

[17] This isn't too far away from the idea in FCS that definites are used to update existing files.

[18] Officially, define:

$$g[x]k \text{ iff } k = g[x/d] \text{ for some } d \in D$$

This is an unspecific x-alternative to g. It is also an information state: one just like g except (possibly) in what it says about x. Note that we are writing this '$g[x]k$' while Groenendijk and Stokhof write '$k[x]g$'. The left-to-right way seems more natural to me, but your mileage may vary.

$$g \llbracket \exists x F x \rrbracket h \text{ iff } : g[x]h \text{ and } h(x) \in I(F) \tag{8.13}$$

So when you interpret $\exists x F x$ you wind up in a state in which x is associated with a thing that is an F.

In DPL $\wedge$ expresses program composition. So to interpret $\exists x F x \wedge G x$ we find a linking information state between $\llbracket \exists x F x \rrbracket$ and $\llbracket G x \rrbracket$.

$$g \llbracket \exists\ x\ F x\ \wedge\ G x \rrbracket h \text{ iff } : \begin{cases} \text{there is a } k \text{ s.t. } g \llbracket \exists x\ F x\ \rrbracket k \text{ and} \\ k \llbracket G x \rrbracket h \end{cases} \tag{8.14}$$

Using (8.13) we can substitute in the right-hand side:

$$g \llbracket \exists x F x \rrbracket k \text{ iff } g[x]k \text{ and } k(x) \in I(F) \tag{8.15}$$

So the denotation of our conjunction is:

$$g \llbracket \exists x F x \wedge G x \rrbracket h \text{ iff } : \begin{cases} \text{there is a } k \text{ s.t. } g[x]k \text{ and} \\ k(x) \in I(F) \text{ and} \\ k \llbracket G x \rrbracket h \end{cases} \tag{8.16}$$

Similarly, for $\llbracket G x \rrbracket$:

$$k \llbracket G x \rrbracket h \text{ iff } h = k \text{ and } h(x) \in I(G) \tag{8.17}$$

Putting those together and skipping a step:

$$g \llbracket \exists x F x \wedge G x \rrbracket h \text{ iff } : \begin{cases} g[x]h \text{ and} \\ h(x) \in I(F) \text{ and} \\ h(x) \in I(G) \end{cases} \tag{8.18}$$

The x in Gx is outside the syntactic scope of $\exists x$ but still gets bound by it. The reason is that, as we saw, $\llbracket \exists x F x \rrbracket$ relates g to h only if $h(x)$ is in $I(F)$. Since conjunction is program sequencing, such h's are the input states for executing $\llbracket G x \rrbracket$. Existentially quantified formulas change downstream assignment functions. Conjunction, a.k.a. relational composition, is primed to be sensitive to those changes. You can see this in Fig. 8.2. That completes the demonstration of Fact 1.

This works because there is an interplay between *external* and *internal* dynamics in DPL. External: $\exists x$ can have effects on things outside its syntactic scope. Internal: $\wedge$ has dynamic effects that carry from first conjunct to second.[19] By the time we get to the x in Gx we are therefore in an information state that maps x to a thing

[19] Conjunction is externally dynamic, too.

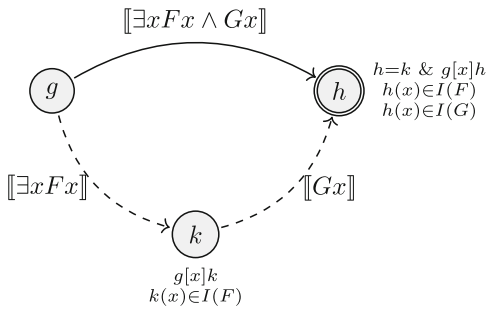

Fig. 8.2 $[\![\exists x Fx]\!] \circ [\![Gx]\!]$

in $I(F)$ and the Gx conjunct thus tacks on the information that that thing is also in $I(G)$. We will return to this idea a few more times.

This particular way that external and internal dynamics play together has the upshot that scope and binding come apart in DPL. Binding outstrips syntactic scope.[20] So taking on board the information carried by $\exists x Fx$ has the downstream consequence that occurrences of x outside the scope of this $\exists x$ can end up being obligatorily associated with things in $I(F)$.

DPL Basics: ¬, →, and ∀ (and ∨, Too)

Our FO language takes ¬, ∧, and ∃ as basic and offers ∨, →, and ∀ as abbreviations. This is another distinctive feature of DPL: classically, it doesn't matter what you take as basic and what you take as derived, but it matters in DPL. We'll see why.

In this section we will work up to the DPL treatment of example (3a) by talking about simple instances of the DPL clauses for ¬ and →. Once again the simple, sentence-by-sentence representation in (8.3) will do in DPL. The main thing is that in DPL an indefinite in a conditional antecedent can bind a pronoun in a conditional consequent and bind it with universal force. Working out the meaning of $\exists x Fx \rightarrow Gx$ is enough to make the point.[21]

Fact 2 *$g\ [\![\exists x Fx \rightarrow Gx]\!]\ h$ only if $h[x]k$ and $k(x) \in I(F)$ imply $k(x) \in I(G)$.*

All of our derived connectives appeal to ¬ so let's start with that. Negation in DPL tests for *failure*. An information state g is $[\![\neg\phi]\!]$-related to h iff $g = h$ and it's not possible to execute $[\![\phi]\!]$ in g at all.

[20] Section 3.3 of DPL probes how scope and binding come apart (DPL, pp. 58–61).

[21] This is code for: you should work out the DPL meaning for (8.3) as a homework exercise.

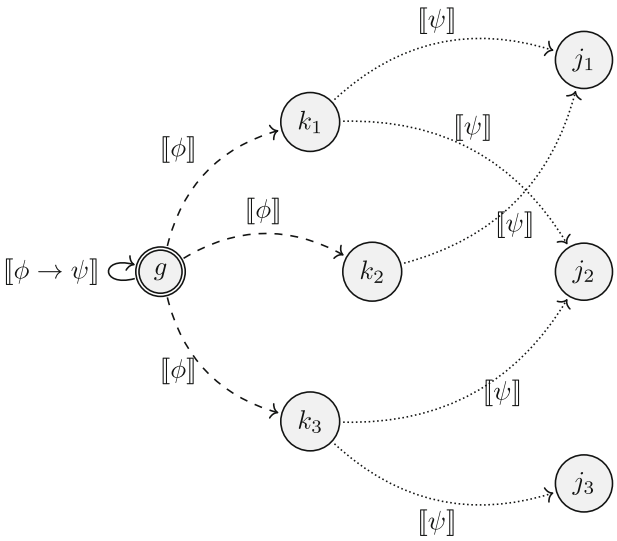

Fig. 8.3 $[\![\phi \rightarrow \psi]\!]$

$$g\ [\![\neg\phi]\!]\ h \text{ iff : } \begin{cases} h = g \text{ and} \\ \text{there is no } k \text{ s.t. } g\ [\![\phi]\!]\ k \end{cases} \tag{8.19}$$

This makes negation a test-maker: $\neg\phi$ is a test no matter what and so is not externally dynamic (a.k.a., static).

Conditionals in DPL are negated conjunctions: $\phi \rightarrow \psi = \neg(\phi \wedge \neg\psi)$. This has two upshots. The first is that the derived meaning for $\phi \rightarrow \psi$ is a test with a wrinkle:

$$g\ [\![\phi \rightarrow \psi]\!]\ h \text{ iff : } \begin{cases} h = g \text{ and} \\ h\ [\![\phi]\!]\ k \text{ implies } k\ [\![\psi]\!]\ j \text{ for some j} \end{cases} \tag{8.20}$$

The first clause ensures that conditionals are tests. The second clause contains the wrinkle. It implicates an intermediate state k that passes values of variables from antecedent to consequent. So, even though they are tests, conditionals are internally dynamic.

The second upshot: conditionals in DPL are *strict*. You can see this in the second clause: in saying "$h\ [\![\phi]\!]\ k$ implies ..." we are universally quantifying over all k's that are reachable from h (i.e., g) by executing $[\![\phi]\!]$. Figure 8.3 graphs this relationship: $\phi \rightarrow \psi$ relates g to itself iff every way of executing $[\![\phi]\!]$ in g lands you in an information state (those are the intermediate k's) where it is possible to execute $[\![\psi]\!]$.

The strictness of conditionals conspires with the wrinkle of internal dynamics in donkey conditionals. Again, the point is already present in the simpler version we're

dealing with. Applying the clauses above what you'll see is this:

$$g \llbracket \exists x Fx \rightarrow Gx \rrbracket h \text{ iff : } \begin{cases} h = g \text{ and} \\ h[x]k \text{ and } k(x) \in I(F) \text{ implies} \\ k(x) \in I(G) \end{cases} \tag{8.21}$$

That is: the x in Gx is bound by the $\exists x$ in the antecedent, and bound with universal force. The result is that $\exists x Fx \rightarrow Gx$ says all Fs are Gs. That's Fact 2.[22]

The universal quantifier in DPL is similar and for the similar reason that it is derived from the interaction of $\exists$ and $\neg$. The derived meaning for $\forall x\phi = \neg\exists x\neg\phi$:

$$g \llbracket \forall x\phi \rrbracket h \text{ iff : } \begin{cases} h = g \text{ and} \\ h[x]k \text{ implies } k \llbracket \phi \rrbracket j \text{ for some } j \end{cases} \tag{8.22}$$

Working out a simple example, you'll see that $g \llbracket \forall x Fx \rrbracket h$, iff $g = h$ and $h[x]k$ implies $k(x) \in I(F)$. Like $\rightarrow$, the universal quantifier is internally dynamic (a linking state k shows up) but externally static (because: negation). We'll return to what this predicts for bound readings of pronouns (and thus why $\forall$ is introduced as an abbreviation).

But first, let's turn to $\vee$: we are taking $\phi \vee \psi$ as shorthand for the official $\neg(\neg\phi \wedge \neg\psi)$. A useful exercise is to verify this:

$$g \llbracket \phi \vee \psi \rrbracket h \text{ iff : } \begin{cases} g = h \text{ and} \\ \text{there is a } j \text{ s.t. } g \llbracket \phi \rrbracket j \text{ or} \\ \text{there is a } j \text{ s.t. } g \llbracket \psi \rrbracket j \end{cases} \tag{8.23}$$

Like all of these negation-derived operators, DPL disjunction is externally static. But it is also internally static: this is because the first disjunct in $\phi \vee \psi$ is officially embedded under a negation that has scope only over it.

[22] We are taking it for granted that $\exists x Fx \rightarrow Gx$ does, invariably, say that all Fs are Gs. This is a simplification. Consider weak *donkey*- or *parking meter*-conditionals (the examples are from Schubert and Pelletier 1988):

(i) a. If Pedro owns a donkey, he will ride it to town tomorrow.

b. If I have a quarter in my pocket, I will put it in the parking meter.

There are readings of these that don't require Pedro to ride all of his donkeys to town tomorrow and don't require that I pump the meter with every quarter in my pocket. The hard-wired universal force that DPL delivers isn't as hard-wired as it seems, though: weak-conditional variants are available as amendments. This is taken up (in somewhat different ways) in Section 4.2 and Section 4.3 of DPL (pp. 81–2, 89). (Hard-wiring of either sort seems incorrect since, as pointed out in Dekker (1996), simply negating the consequent of a weak-donkey conditional brings the strong reading back to center stage.)

The test-making behavior of negation has empirical predictions for the derived operators. We will briefly touch on them.

Since conditionals are tests, and so externally static, we get an explanation for why pronouns beyond the scope of the conditional aren't bound by the indefinite in the conditional's antecedent.

(4) a. If a farmer owns a donkey, he feeds it hay. #It is happy with that meal.

b. A farmer walked in the field or he whistled.

While the *he* and the *it* in the consequent of the conditional in (4a) get bound by *a farmer* and *a donkey*, that is where it ends. And in (4b) it is hard to hear *he* as bound by *a farmer*.[23]

Similarly:

(5) a. It's not the case that a farmer walks in the field. #He whistles.

b. No farmer owns a donkey. #He's happy about it.

c. Every farmer walks in the field. #He whistles.

There aren't bound readings of *he* in these sorts of discourses.[24] This is explained naturally in DPL: the first sentence of (5a) is a test and so doesn't change the values of variables. Hence, it doesn't constrain the *he* in the second sentence. Similarly for (5b) and, for exactly the same reason, the universal (5c).

There are (at least) two related things to keep in mind. First, this isn't meant to be the last word. Second, this is a different means of generating these predictions from the way the very same data gets explained in DRT and FCS. In DRT, for example, the behavior is explained by the rules that govern when discourse referents (officially, DRSs that contain them) are and aren't *accessible* from a given DRS. Here the explanation instead follows from the DPL behavior of the basic connectives ($\exists$, $\wedge$, and $\neg$) and the completely intro-logic-textbook definitions that introduce $\rightarrow$, $\vee$, and $\forall$.

[23] Here, too, things are trickier. Groenendijk and Stokhof discuss discourses like this (DPL, p. 91):

(i) a. If a client turns up, you treat him politely. You offer him a cup of coffee and ask him to wait.

b. Either there's no bathroom here or it is in a funny place.

Here *him* seems to talk about the client who hypothetically turned up (and *it* the bathroom). The lesson they suggest is that static, non-dynamic meanings might be recoverable on-demand from the default dynamic meanings and that certain contexts or situations trigger that recovery. For another approach (especially to discourses like (ia)) see Poesio and Zucchi (1992).

[24] But see footnote 23.

Truth, Entailment, Equivalence

Meaning in DPL is richer than truth conditions. So when it comes to thinking about logical concepts (truth, entailment, equivalence) we have options: seemingly different ways of thinking that collapse or coincide in a classical universe may well mark out different regions in a dynamic universe.[25]

In DPL truth in g is a matter of whether it is *possible to execute* the program $[\![\phi]\!]$ in g.[26]

Definition 3 (DPL Truth) For any M, g and ϕ:

1. ϕ is true (in M w.r.t. g) iff $g\ [\![\phi]\!]\ h$ for some h.
   Notation: $g \models_M \phi$
2. ϕ's satisfaction set is the set of assignments at which it is true.
   Notation: $\|\phi\|_s = \{g\colon\ g \models_M \phi\}$

Thus ϕ is true w.r.t. g iff g is in ϕ's satisfaction set. For the simplest tests, the classical set-up comes out as a special case. Suppose $g \in \|Fx\|_s$. So $g\ [\![Fx]\!]\ h$ for some h. Since Fx is a test, $g = h$ and so $g(x) \in I(F)$. This is what it takes for g to classically satisfy Fx in M.

But DPL truth conditions go beyond classical truth conditions. Take our earlier example $\exists x Fx \wedge Gx$. For this to be true w.r.t. an information state g there must be something that is both an F and a G. These are the same as the truth conditions for the wide-scope existential $\exists x(Fx \wedge Gx)$. Summing up:

Fact 3 *For any M:* $\|\exists x Fx \wedge Gx\|_s = \|\exists x(Fx \wedge Gx)\|_s$

Similarly, the donkey-like conditional $\exists x Fx \rightarrow Gx$ is true w.r.t. g iff every x-alternative k to g such that $k(x)$ is an F is such that $k(x)$ is a G, too. These are the same as the truth conditions for $\forall x(Fx \rightarrow Gx)$. Hence:

Fact 4 *For any M:* $\|\exists x Fx \rightarrow Gx\|_s = \|\forall x(Fx \rightarrow Gx)\|_s$

Something strictly stronger is in fact true: each of these pairs is equivalent.

Classically satisfaction, entailment, and equivalence are all tightly constrained: entailment is preservation of satisfaction, sameness of satisfaction sets is equiva-

[25] We will only look at things from a semantic perspective. This raises the question: what does proof theory (say, a sequent system or a natural deduction system) for DPL look like? The answer is: interesting but complicated! The only completeness proofs I am aware of are in Veltman (2000) and (in a more general setting) (Dekker, 2016).

[26] Compare this to truth conditions for atomic DRSs in DRT. A DRS like

$$[x\colon \texttt{farmer}(x); \texttt{walked-in-field}(x)]$$

is true in M iff *there is* a (partial) assignment g such that $g(x)$ is a farmer in M and $g(x)$ walked in a field in M. The existential force of indefinites comes from the existential truth conditions, not from quantificational meaning of indefinites.

lence, equivalence is mutual entailment. In DPL there is more room. DPL meanings are fine grained, satisfaction is derivative and doesn't determine entailment, and two-way entailment isn't enough for equivalence.

Definition 4 (DPL Entailment, Equivalence) For any $\phi, \phi_1, \ldots, \phi_n$ and ψ:

1. ϕ and ψ are equivalent ($\phi \simeq \psi$) iff: for any M

$$[\![\phi]\!]^M = [\![\psi]\!]^M$$

2. $\phi_1, \ldots, \phi_n$ entail ψ ($\phi_1, \ldots, \phi_n \models_M \psi$) iff: for any M

$$g([\![\phi_1]\!]^M \circ \cdots \circ [\![\phi_n]\!]^M)h \text{ implies } h \models_M \psi$$

Equivalence asymmetrically implies sameness of satisfaction sets.

Fact 5 *If $\phi \simeq \psi$ then $\|\phi\|_s = \|\psi\|_s$ but $\|\phi\|_s = \|\psi\|_s$ does not imply $\phi \simeq \psi$.*

In computing the semantic value of $\exists x Fx \wedge Gx$ we actually showed this:

$$\exists x Fx \wedge Gx \simeq \exists x(Fx \wedge Gx) \tag{8.24}$$

This equivalence is sometimes called *Egli's theorem*.[27] Similarly, we did enough to show this:

$$\exists x Fx \rightarrow Gx \simeq \forall x(Fx \rightarrow Gx) \tag{8.25}$$

This equivalence is sometimes called *Egli's corollary*.

To see that sameness of satisfaction sets is not generally sufficient for sameness of meaning, consider $\exists x Fx$ and $\exists y Fy$. Note that $g \models \exists x Fx$ iff something in D is an F. Same for $\exists y Fy$. Thus:

$$\|\exists x Fx\|_s = S = \|\exists y Fy\|_s \tag{8.26}$$

These sentences prime different variables for downstream binding: $[\![\exists x Fx]\!]$ takes a g to an h_1 that maps x to an F while $[\![\exists y Fy]\!]$ takes g to an h_2 that maps y to an F. Those are different outputs reflecting their different meanings. Thus:

$$\exists x Fx \not\simeq \exists y Fy \tag{8.27}$$

In canonical DPL the terminology is that $\exists x Fx$ and $\exists y Fy$ differ in their *production sets* (DPL, p. 56).[28]

[27] See Egli 1979.

[28] Define
$\|\phi\|_p = \{h\colon\ g\ [\![\phi]\!]\ h \text{ for some } g\}$

When it comes to entailment in DPL, it is sequences (not sets) of premises that matter, so you might expect the order and repetition of premises to matter. And since entailment involves composing the programs denoted by the premises, you might expect that the interesting dynamics associated with interpreting indefinites and conjunctions matter for entailment. Putting those together we'll see that there are entailments of classical logic that are not entailments in DPL and the other way around, too.

DPL entailment has many good-making features. Here is one: it validates the deduction theorem.

Fact 6 $\phi_1, \ldots, \phi_n \models \psi$ *iff* $\models (\phi_1 \wedge \cdots \wedge \phi_n) \rightarrow \psi$.

DPL entailment also goes beyond its classical counterpart. Whether ψ is entailed by $\phi_1, \ldots, \phi_n$ depends on what is true in a state after executing the composite program $[\![\phi_1]\!] \circ \cdots \circ [\![\phi_n]\!]$. This means that discourse information can get passed from the premises to a conclusion.

(6) a. A woman came in and ordered a sandwich. So: she ordered a sandwich.

b. A woman came to the department. So: she came to campus.

Like the examples of donkey anaphora, these mini-arguments can get straightforward representations in L_{FO}:

$$\exists x(\texttt{woman}\, x \wedge \texttt{order-sandwich}\, x) \therefore \texttt{order-sandwich}\, x \tag{8.28}$$

$$\exists x(\texttt{student}\, x \wedge \texttt{came-}to\texttt{-department}\, x) \therefore \texttt{came-to-campus}\, x \tag{8.29}$$

The common element is that an indefinite seems to reach from premises to an anaphoric pronoun in the conclusion.

Fact 7 $\exists x(Fx \wedge Gx) \models Gx$ *and* $\forall x(Gx \rightarrow Hx), \exists x(Fx \wedge Gx) \models Hx$

This is all due to the way discourse information gets introduced and passed through externally dynamic operators.

DPL entailment is about how a *sequence* of premises changes an information state. Neither repetition nor order matters when it comes to classical entailment, but both can be relevant here.

Fact 8 $\phi_1, \ldots, \phi_{n-1}, \phi_n \models \psi$ *does not imply* $\phi_1, \ldots, \phi_n, \phi_{n-1} \models \psi$.

Similar facts hold about production sets: if $\phi \simeq \psi$ then $\|\phi\|_p = \|\psi\|_p$ but not in general the other way around. Satisfaction sets pool the states where you can get somewhere from whereas production sets pool states gotten to from somewhere. Still, it is not in general true that $\|\phi\|_s = \|\psi\|_s$ and $\|\phi\|_p = \|\psi\|_p$ together entail that $\phi \simeq \psi$. Counterexamples are things like $Fx \vee \neg Fx$ and $\exists x(Fx \vee \neg Fx)$. Exactly one is a test!

Counterexamples involve existential quantifiers with matching variables competing over a free variable in the conclusion.

$$\exists x Fx, \exists x Gx \models Gx \text{ but } \exists x Gx, \exists x Fx \not\models Gx \tag{8.30}$$

Since existential quantifiers can bind across entailment, interloping existential quantifiers bearing the same variable can interfere with this. This is because $\exists x$ resets the value we have associated with x.[29]

This is also why right monotonicity fails for DPL entailment.

Fact 9 *$\phi \models \psi$ implies $\chi, \phi \models \psi$ but $\phi \models \psi$ does not imply $\phi, \chi \models \psi$.*

Counterexamples involve things like

$$\exists x Fx \models Fx \text{ but } \exists x Fx, \exists x Gx \not\models Fx \tag{8.31}$$

A somewhat natural instance of this pattern (using singular *they*):

(7) a. Someone is at home. So: they are at home.

b. Someone is at home. Someone is at work. ?So: they are at home.

Again we have quantifiers competing over the downstream pronoun.

This is also why idempotence and repetition are not structural properties of DPL entailment.

Fact 10 *$\phi \not\models \phi$ and $\phi, \psi, \chi \not\models \psi$.*

By now you will see how to concoct the right kind of counterexample.

So the effects of dynamic entailment show up in lots of places, but a common thread is simply that $\exists x Fx \wedge Gx$ doesn't mean the same thing as $Gx \wedge \exists x Fx$.

Fact 11 $\phi \wedge \psi \not\simeq \psi \wedge \phi$

DPL entailment is not unruly because it lacks some classical property or other. There are systematic, precise and well-understood boundaries to when instances of classical patterns are DPL-valid and when they aren't.[30] That said, it may be pushing luck somewhat to mention that DPL entailment isn't transitive, either.[31]

Fact 12 *$\phi \models \psi$ and $\psi \models \chi$ do not imply that $\phi \models \chi$.*

[29] This is sometimes called "destructive update" or the "downdating problem" in DPL. I see it as more feature than bug, an interesting property that arises because of the way existential quantifiers reset values to variables. We will see another reflection of this feature later. That said, this is a place where FCS and DPL differ (see, e.g., Dekker 1996).

[30] See in particular the discussion in Section 3.5 (DPL, pp. 68–70).

[31] Also, no cut: $\exists x Fx, \exists x \neg Fx \models \exists x Fx$ and $\exists x Fx, \exists x \neg Fx, \exists x Fx \models Fx$ but $\exists x Fx, \exists x \neg Fx \not\models Fx$.

Here the counterexamples still have to do with reaching down to conclusions, but the blocking is achieved not by a competing quantifier but by negation.

$$\neg\neg\exists x\, Fx \models \exists x\, Fx \text{ and } \exists x\, Fx \models Fx, \text{ but } \neg\neg\exists x\, Fx \not\models Fx \tag{8.32}$$

Note that $\exists x\, Fx \rightleftharpoons \neg\neg\exists x\, Fx$. But they are not equivalent.[32]

Fact 13 $\neg\neg\phi \rightleftharpoons \phi$ *but* $\neg\neg\phi \not\simeq \phi$.

We have been stressing the way in which the richer universe of meanings that DPL opens up interacts with how we think about things like truth and entailment and equivalence and how those things relate to each other. This is different from the more restricted classical set-up where different ways of thinking collapse so quickly we barely notice. To finish off this section, we'll see one more example of this richness.

Return to the choice of basic versus derived connectives. Classically, this is largely a matter of taste and pragmatic considerations. Not so in DPL. For instance, it matters that $\exists$ is basic and $\forall$ is introduced in terms of it. One way to see that:

Fact 14 *While* $\neg\exists x\phi \simeq \forall x\neg\phi$ *it is not in general true that* $\exists x\phi \simeq \neg\forall x\neg\phi$. *However,* $\neg\neg\exists x\phi \simeq \neg\forall x\neg\phi$

Thus if you instead start with $\forall$ and introduce $\exists$ in terms of it, what you'll get will be a test and hence not the same as the DPL meaning for $\exists$ (and not something that would fit together with $\wedge$ very well). For exactly the same reasons it is important that $\wedge$ is basic in DPL and $\rightarrow$ and $\vee$ are introduced in terms of it. These facts percolate to familiar logical patterns: for example, while contraposition and DeMorgan's equivalences don't in general hold, restricted forms with well-placed closure operators do.

Updating Information States

We have focused on canonical, relational DPL. This has pluses and minuses. Among the pluses: the connection to dynamic logic is clearer. Among the minuses: there is some mismatch between the mechanics of DPL and the idea that the meaning of a sentence is "the way it changes (the representation of) the information of the interpreter" (DPL, p. 43).

[32] Add the one-place closure operator $\blacktriangleright$ and interpret it this way:

$$[\![\blacktriangleright\phi]\!] = \{\langle g, h\rangle : \ g = h \text{ and } \langle h, k\rangle \in [\![\phi]\!] \text{ for some } k\}$$

This takes a sentence and (only) strips it of its dynamic potential (this is sometimes symbolized by $\Diamond$ (DPL, p. 63)). You should show that $\blacktriangleright\phi \simeq \neg\neg\phi$.

Here's the mismatch. DPL meanings are relations between information states: what links a prior information state to a posterior state is the meaning of the sentence whose information is being taken on board. But hearers do not in general have complete information and so the "information of the interpreter" should be partial. In relational DPL information states are total assignment functions. Thus the picture that relational DPL leaves us with is one in which interpreters lurch from one complete state of information to another.

Fortunately, this is an artifact. We will explore an implementation of DPL that that doesn't involve such lurching. It will also bring out different ways that meanings can be "dynamic" and it will be good preparation for tackling the issues that arise when the dynamics of discourse information and world information mix (in particular, for tackling the system explored in Groenendijk et al. (1996)).

Let's start with a change in subject: update systems. Both the general set-up and the specific system we'll look at are from Veltman (1996).

Definition 5 (Update Systems) Given a language L, an update system for L is a pair $\langle \mathfrak{S}, [\cdot] \rangle$ where $\mathfrak{S}$ is the set of information states (ordered by $\leq$) with minimal state $\mathbf{1}$ and absurd state $\mathbf{0}$, and $[\cdot]\colon\ \mathfrak{S} \to \mathfrak{S}$ is an interpretation function for L.

Intuitively, $s \leq t$ iff t contains at least as much information as s does.

Fact 15 *Define $s + t = t$ iff $s \leq t$. Then $+$ is a join in the lattice $\langle \mathfrak{S}, \mathbf{1}, + \rangle$.*[33]

We will look at a simple example of an update system for a simple propositional modal language $L_\Diamond$ (Definition 17). The resulting dynamic interpretation of it is Update Semantics (US).[34]

Definition 6 (Information States in US) The set I of information states is the set:

$$s \in I \text{ iff } s \subseteq W$$

where $W = 2^A$ (and A is the set of atomic sentences in $L_\Diamond$).

This takes an operationalist view on possible worlds: they are what sorts out which atomic sentences are true and which are false. Two limiting cases of information states: W plays the $\mathbf{1}$ role, $\emptyset$ plays the $\mathbf{0}$ role. Note that $\subseteq$ is the natural ordering of states and $\cap$ is the join. These information states, unlike DPL information states, represent genuine uncertainty: it is not in general true that every information state is opinionated about every issue.

As in DPL, sentences express programs. Here, though, they are deterministic: meanings in US are functions from information states to information states.

[33] So: (i) $\mathbf{1} + s = s$, (ii) $s + s = s$, (iii) $s + t = t + s$, and (iv) $(s + t) + r = s + (t + r)$.

[34] This simple modal system is given as a warm-up exercise in Veltman (1996) before more sophisticated systems are developed. (The version here is different in small ways; e.g., we are allowing the modals to be out-scoped by $\neg$ and $\wedge$.) The disclaimer at the beginning of this paper holds for US, too: what we'll do isn't — couldn't be — an improvement on the original and you are hereby invited to spend quality time with it.

Definition 7 (Meanings in US) For any ϕ:

$$[\phi]_{\text{US}} : I \to I$$

The convention is to write the function post-fix (and omit the subscript when unambiguous): the result of applying $[\phi]$ to state s is $s[\phi]$.

This doesn't say which function $[\phi]$ denotes. We'll look at some characteristic examples; the official version is in the appendix (Definition 21).

Often, updating is a matter of eliminating worlds from contention. Consider a simple negated sentence like

(8) It's not raining.
$\neg p$

Adding the information carried by this to an information state s goes by two steps; the clauses for $\neg$ and atomic sentences conspire to achieve the intuitively right result. Negation works by set complementation:

$$s[\neg p] = s \setminus s[p] \tag{8.33}$$

So, first, we need to hypothetically update s with p:

$$s[p] = \{w \in s : w(p) = 1\} \tag{8.34}$$

Thus $s \setminus s[p]$ is the state t which contains all and only the worlds in s in which p is false. That is: $\neg p$ expresses the program or instruction to eliminate p-worlds from consideration.

Conjunction in US is *functional* composition. So updating s first with $\neg p$ and then with q further refines things: $s[\neg p]$ is the state in which $[q]$ is executed.

$$s[\neg p \wedge q] = s[\neg p][q] = (s \setminus s[p])[q] \tag{8.35}$$

The resulting state is the state t that contains all and only the worlds from s in which p is false and in which q is true.

The existential modal claim $\Diamond\phi$ is a different sort of program. It is a test, checking whether ϕ is compatible with s.[35]

$$s[\Diamond p] = \{w \in s : s[p] \neq \emptyset\} \tag{8.36}$$

Notice that this test behavior means that either $s[\Diamond\phi] = s$ or $s[\Diamond\phi] = \emptyset$ for any state s and any prejacent ϕ (Fig. 8.4).

[35] There's a surprising ambiguity in "compatibility" (Gillies, 2020).

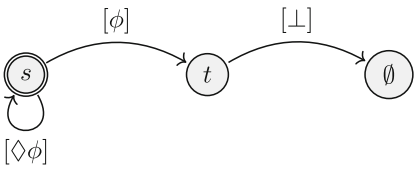

Fig. 8.4 Compatibility test

Definition 8 (US Truth, Entailment, Equivalence) For any $\phi, \phi_1, \ldots, \phi_n, \psi$ and any state s:

1. ϕ is true in s iff $s[\phi] = s$
   Notation: $s \Vdash\!\!\!\!=_{\text{US}} \phi$
2. $\phi_1, \ldots, \phi_n$ entails ψ iff for any s: $s[\phi_1] \ldots [\phi_n] \Vdash \psi$
   Notation: $\phi_1, \ldots, \phi_n \Vdash\!\!\!\!=_{\text{US}} \psi$
3. ϕ and ψ are equivalent iff $[\phi] = [\psi]$
   Notation: $\phi \simeq_{\text{US}} \psi$

We are, again, in a rich universe: truth is what you might expect, but entailment is more than preservation of truth at a state, and equivalence is stronger than co-entailingness.[36]

For the descriptive, non-modal fragment things aren't really that different from the classical interpretation of propositional logic.

Fact 16 *For non-modal ϕ, ψ and state s: $W[\phi]$ behaves like a (classical) proposition, updating s is intersection with a proposition, and truth is propositional containment. That is:*

1. $W[\neg\phi] = W \setminus W[\phi]$ and $W[\phi \wedge \psi] = W[\phi] \cap W[\psi]$;
2. $s[\phi] = s \cap W[\phi]$; and
3. $s \Vdash \phi$ iff $s \subseteq W[\phi]$.

Things are different with $\Diamond$ and this leads to some similar-to-DPL-looking logical facts: non-commutativity, non-idempotence, non-permutation, etc.

Fact 17 *$\phi \wedge \psi \not\simeq \psi \wedge \phi$; $\phi \not\Vdash \phi$; and $\phi_1, \ldots, \phi_{n-1}, \phi_n \Vdash \psi$ does not imply $\phi_1, \ldots, \phi_n, \phi_{n-1} \Vdash \psi$.*

The relevant counterexamples all involve $\Diamond$.[37]

[36] There are other sensible choices for entailment: see van Benthem 1996; Veltman 1996 for menus of options. The interesting thing is that various ways of thinking about entailment that tend to collapse in a classical set-up can come apart in a dynamic one.

[37] For instance: $\Diamond p \wedge \neg p \not\simeq \neg p \wedge \Diamond p$. *Proof:* let $s = \{w, v\}$ where $w(p) = 1$ and $v(p) = 0$ and note that $s[\Diamond p \wedge \neg p] = \{v\}$ while $s[\neg p \wedge \Diamond p] = \emptyset$. Using this information, you can construct counterexamples for idempotence and permutation.

The point for now is to see US as implementing a natural way of updating an information state where those states are partial pictures of the way things are.

DPL Updates

We want to implement the core of DPL in an update system that traffics in partial information states. In DPL the only information is discourse information so we don't want every information state to take a stand on the value of every variable. To achieve this, take sets of variable assignments as our states.

Definition 9 (Information States in Functional DPL) Let $M = \langle D, I \rangle$ be a first-order model and $S = \wp(D^V)$ the set of variable assignments. The set $\mathbf{S}$ of (functional) DPL information states is the set:

$$s \in \mathbf{S} \text{ iff } s \subseteq S$$

That is: information states are *sets* of relational DPL information states. Note that $\mathbf{S}$ plays the $\mathbf{1}$ role, $\emptyset$ the $\mathbf{0}$ role, and s contains at least as much information as t iff $s \subseteq t$.

Definition 10 (Meanings in Functional DPL) For any ϕ:

$$[\phi]^M_{\text{DPL}} : \mathbf{S} \to \mathbf{S}$$

The conventions from US carry-over here. We'll look at characteristic examples of the interesting clauses, saving the official version of $[\cdot]_{\text{DPL}}$ for Definition 22.

The move from $[\![\cdot]\!]^M_{\text{DPL}}$ to $[\cdot]^M_{\text{DPL}}$ is mostly straightforward: atomic formulas and negations operate on the assignments that make up a state and conjunction becomes functional composition.

So, for instance, applying the meaning $[Fx]$ to a state s:

$$s[Fx] = \{g \in s : \ g(x) \in I(F)\} \tag{8.37}$$

The instruction: figure out which individual assignments in s map x to something in $I(F)$, and keep those (and only those). Likewise, in applying $[\neg\phi]$ to s, test the individual assignments in s to see if they permit updating by $[\phi]$. Extending our example:

$$\begin{aligned} s[\neg Fx] &= s \setminus \{g : \ \{g\}[Fx] \neq \emptyset\} \\ &= \{g \in s : \ g(x) \notin I(F)\} \end{aligned} \tag{8.38}$$

The singleton sets $\{g\}$ on the right-hand side are a giveaway that DPL negation, unlike its US cousin, is entirely point-wise: $s[\neg Fx]$ rides on $\{g\}[Fx]$ for each $g \in s$.

Updating a state s with $[\exists x Fx]$ should output a state that only contains assignments mapping x to an F. There are multiple ways to achieve this.

Definition 11 Fix a model $M = \langle D, I \rangle$ and let s be any state, x any variable, and d any object in D.

$$s[x/d] = \{h\colon\ h = g[x/d] \text{ for some } g \in s\}$$

This is the set of assignments that differ from an assignment in s at most in that they map x to d. This achieves a global and specific rejiggering of the information so that all assignments have x pointing to this particular d.[38]

The official, full-generality, clause for updating with an existentially quantified formula is this:

$$s[\exists x\phi] = \bigcup_{d \in D} (s[x/d][\phi]) \tag{8.39}$$

The intermediate, rejiggered, state $s[x/d]$ is key. What we are doing is taking an object d from D and re-assigning x to map to it, and then updating the resulting $s[x/d]$ with ϕ, and then repeating this process for every object in D.

An example may help. First, given a $d \in D$, we take the state $s[x/d]$ and update it with Fx:

$$g \in s[x/d][Fx] \text{ iff } g \in s[x/d] \text{ and } g(x) \in I(F) \tag{8.40}$$

Next, collect up the results from doing this for all the ways of choosing such a d:

$$g \in \bigcup_{d \in D} (s[x/d][Fx]) \text{ iff for some } d : g \in s[x/d] \text{ and } g(x) \in I(F) \tag{8.41}$$

Thus $[\exists x Fx]$ instructs us to randomly reassign a value to x and then insists that whatever x points to is an F.

Fact 18 *Let* $s[x] = \{g[x/d]\colon\ g \in s \text{ and } d \in D\}$. *Then* $s[\exists x\phi] = s[x][Fx]$.

So we could have defined $s[\exists x\phi]$ in terms of $s[x]$ and $[\phi]$.[39]

Truth, entailment, and equivalence undergo similar changes: we just take everything up a level to sets of assignments.

[38] Note that it is not in general true that $s[x/d] \subseteq s$. This is not surprising but important: it will come up again in the next section.

[39] Notice that this amounts to treating $\exists x$ as categorematically expressing the function $[x]$ and thus $\exists x\phi$ (abusing our object language formation rules) as the conjunction $\exists x \wedge \phi$. These two formulations agree but can come apart once modals are added to the mix. Some (canonical) discussions: van Eijck and Cepparello (1994); Groenendijk et al. (1996); Beaver (2001).

Definition 12 (DPL Truth, Entailment and Equivalence, Again) For any $\phi, \phi_1, \ldots, \phi_n$, and ψ:

1. ϕ is true in s w.r.t M ($s \Vdash \phi$) iff:

$$\text{for every } g \in s\colon \ \{g\}[\phi] \neq \emptyset$$

2. $\phi_1, \ldots, \phi_n$ entail ψ ($\phi_1, \ldots, \phi_n \Vdash \psi$) iff:

$$\text{for every } M, s\colon \ s[\phi_1]\ldots[\phi_n] \Vdash \psi$$

3. ϕ and ψ are equivalent ($\phi \simeq \psi$) iff:

$$\text{for every } M\colon \ [\phi]^M = [\psi]^M$$

Truth at an information state (set of assignments) is just truth with respect to each point (assignment) in that state. Similarly for entailment. Equivalence is sameness of semantic value.

This really is DPL in an update system's clothing. Here are two ways to see that.

Fact 19 *$g \llbracket\phi\rrbracket h$ iff $h \in \{g\}[\phi]$.*

Fact 20 *$s \Vdash \phi$ iff for every $g \in s\colon g \models \phi$. Similarly, $\phi_1, \ldots, \phi_n \Vdash \psi$ iff $\phi_1, \ldots, \phi_n \models \psi$.*

Thus all of the characteristic predictions and patterns of (non-)entailment and (non-)equivalence of relational DPL hold for the functional implementation. From that perspective, there is no difference.

Dynamic Differences

Why bother with the update reformulation then? A few reasons. First: it shows that the essential insights of DPL are independent of the relational implementation. Second: it sits better with the idea that contexts are characterized by states of uncertainty and that meanings are ways of updating that uncertainty. Third: it makes for a better comparison with the other dynamic system on the table (US). We turn to this comparison now.

DPL and US have different targets: one cares only about discourse information (level of granularity: which pronouns pick out which objects) and one cares only about world information (level of granularity: which worlds are compatible with the information gathered so far). Even so there are similarities. In both set-ups semantic values are functions linking information states to information states and key concepts like truth and entailment are relativized to information states not to

individual unanalyzed points of evaluation. Both say that equivalence is more than mutual entailment and both exhibit the similar non-classical entailment behavior.

For all this similarity, there are deep differences. We will touch on two of them. The first difference is that in US information always accumulates but DPL permits information "loss".[40] The second difference: everything in US semantics happens *set-wise*: the information states are key players *as* states. In DPL at key points the call to information states gets flattened out to quantification over what happens at the assignments that make them up. The result is a system in which, in a precise sense, there is no essentially set-wise behavior.

The updates in US are aptly named: posterior states contain at least as much information as their prior states do.

Definition 13 (Eliminativity) An update system $\langle \mathfrak{S}, [\cdot] \rangle$ for L with ordering $\leq$ is eliminative iff: for every $s \in \mathfrak{S}$ and $\phi \in L$:

$$s \leq s[\phi]$$

Eliminative update systems are ones in which information accumulates: $s[\phi]$ contains at least the information that s does.

Fact 21 *$\langle I, [\cdot]_{\text{US}} \rangle$ is eliminative and $\langle \mathbf{S}, [\cdot]_{\text{DPL}} \rangle$ is not eliminative.*

You can see why $s[\phi]_{\text{US}}$ is always included in s: each characteristic clause either eliminates possibilities or leaves the prior state unchanged. Not so in DPL: updating a set of assignments s with $\exists x$ globally resets the value associated with x, forgetting the prior value. The way it achieves such resetting is by admitting into $s[x]$ assignments not in s. Hence $s[\exists x \phi]$ will not in general be included in s.

We have already seen how the resetting behavior makes for empirical and structural differences in DPL: it is involved in why conjunctions don't commute and in failures of right monotonicity, idempotence, and repetition. There is another place where it comes up and where it is alleged to be problematic.

(9) A farmer walked in the field. A farmer whistled.
$\exists x(\texttt{farmer}\ x \wedge \texttt{walked-in-field}\ x) \wedge \exists x(\texttt{farmer}\ x \wedge \texttt{whistled}\ x)$

Suppose we update s so that x is associated with a farmer who walked and then update the resulting state so that x is associated with a farmer who whistled. The result will be the same as had we just changed s so that x is associated with a farmer who whistled. The information carried by the first sentence is gone. This is the destructive update or downdate problem of DPL, again.

It is tricky to say exactly what the problem is, though. It isn't that re-using a variable gets us incorrect predicted truth conditions or entailments. You can check that (9) is true in DPL iff there is a farmer who walked and there is a farmer who

[40] The scare quotes indicate that there is some interpretive controversy about the nature of the loss of information. When we get to it, we'll flag the controversy.

whistled.[41] Instead it must be more conceptual: there is a nagging feeling that re-using variables shouldn't destroy the information that was associated with an upstream use of that variable and, more generally, that information updates should be bona fide updates. But if we are only tracking discourse information, re-setting on re-use is what we should expect.[42]

Now to the second deep difference. In US it is whole information states that count: they get implicated in the recursive definition of $[\cdot]_{\text{US}}$, truth at a state isn't defined as truth at the points that make that state up, and entailment is about whether a whole state is a fixed-point of a sequence of updates. In contrast, DPL meanings live and die on the behavior of updates to "states" that are just the individual assignments that make states up.

Definition 14 (Distributivity) An update system $\langle \mathfrak{S}, [\cdot] \rangle$ for L is distributive (a.k.a. continuous) iff: for every $s \in \mathfrak{S}$ and $\phi \in L$

$$s[\phi] = \bigcup_{x \in s} \{x\}[\phi]$$

In a distributive system $s[\phi]$ is always the same thing as collecting up the results from computing $\{x\}[\phi]$ for each $x \in s$. In such systems there is nothing essentially global or set-wise as far as $[\cdot]$ is concerned.

Fact 22 *$\langle I, [\cdot]_{\text{US}} \rangle$ is not distributive and $\langle \mathbf{S}, [\cdot]_{\text{DPL}} \rangle$ is distributive.*

It is the modal $\Diamond$ that is responsible for non-distributivity in US.[43] Here is a simple example: let $s = \{w, v\}$ where $w(p) = 1$ and $v(p) = 0$.

$$\begin{aligned} \bigcup_{x \in s} \{x\}[\Diamond p] &= \{w\}[\Diamond p] \cup \{v\}[\Diamond p] \qquad (8.42) \\ &= \{w\} \cup \emptyset \\ &\neq s[\Diamond p] = s \end{aligned}$$

On the other hand, you can see tell-tale signs of DPL's distributivity: for example, DPL negation doesn't care about the incoming state *as* a state; $s[\neg Fx]$ is determined by what $\{g\}[Fx]$ is for each assignment $g \in s$.

Again, an example may help. Let $M = \langle D, I \rangle$ be a model with just three objects a, b, and c where $I(F) = \{a, b\}$ and let $s = \{h, k\}$. We will look at $s[\exists x \neg Fx]$ and compare this to $\{h\}[\exists x \neg Fx] \cup \{k\}[\exists x \neg Fx]$.

[41] This same point is made in Charlow (2019).

[42] In a more expressive fragment, we may need a more subtle way of tracking different dimensions of information. For instance, the referent systems in Groenendijk et al. (1996) are designed explicitly to disassociate a variable from the world information associated with it. This suggests there is no downdating problem as such for the tracking of discourse information.

[43] That's why Fact 16 only holds for descriptive (non-modal) sentences.

There are two steps to look at. First, randomly reassign x throughout s as a whole:

$$\begin{aligned} s[\exists x] &= s[x] \\ &= \{h[x/a], h[x/b], h[x/c], k[x/a], k[x/b], k[x/c]\} \end{aligned} \tag{8.43}$$

Compare this to applying it to $\{h\}$ and to $\{k\}$:

$$\begin{aligned} \{h\}[\exists x] &= \{h\}[x] \\ &= \{h[x/a], h[x/b], h[x/c]\} \end{aligned} \tag{8.44}$$

$$\begin{aligned} \{k\}[\exists x] &= \{k\}[x] \\ &= \{k[x/a], k[x/b], k[x/c]\} \end{aligned} \tag{8.45}$$

You can probably already see why in general $s[x] = \bigcup_{g \in s} \{g\}[x]$.

The other step that is instructive to look at is negation.

$$\begin{aligned} s[x][\neg Fx] &= s[x] \setminus \{g\colon\ \{g\}[Fx] \neq \emptyset\} \\ &= \{h[x/c], k[x/c]\} \end{aligned} \tag{8.46}$$

Compare to the point-wise updates:

$$\begin{aligned} \{h\}[x][\neg Fx] &= \{h\}[x] \setminus \{g\colon\ \{g\}[Fx] \neq \emptyset\} \\ &= \{h[x/c]\} \end{aligned} \tag{8.47}$$

$$\begin{aligned} \{k\}[x][\neg Fx] &= \{k\}[x] \setminus \{g\colon\ \{g\}[Fx] \neq \emptyset\} \\ &= \{k[x/c]\} \end{aligned} \tag{8.48}$$

Here you can see why in general $s[\neg\phi] = \bigcup_{g \in s} \{g\}[\neg\phi]$. A full, but entirely routine, inductive proof establishes things aren't trickier than this example suggests: $s[\phi]$ is always the same as taking each $g \in s$ one at a time, figuring out $\{g\}[\phi]$, and collecting the results.

Eliminativity and distributivity aren't two randomly chosen properties. Together they characterize the update systems that are *additive*.[44]

Definition 15 An update system $\langle \mathfrak{S}, [\cdot] \rangle$ for L with join $+$ is additive iff for every $s \in \mathfrak{S}$ and $\phi \in L$:

$$s[\phi] = s + \mathbf{1}[\phi]$$

[44] This basic idea goes back to van Benthem (1986) but its modern formulation can be found in both Groenendijk and Stokhof (1990) and Veltman (1996). See also Rothschild and Yalcin (2016).

Such systems are commutative ($s[\phi][\psi] = s[\psi][\phi]$) and idempotent ($s[\phi][\phi] = s[\phi]$). So there's not much dynamics: $[\phi]$ is less an instruction for how to move from a given state s to a posterior state t and more a description of a state that shares information with both s and $\mathbf{1}[\phi]$.

Fact 23 *An update system* $\langle \mathfrak{S}, [\cdot] \rangle$ *for L with join* $+$ *is additive iff it is both eliminative and distributive.*

So US and DPL are both non-additive but in exactly complementary ways.

Extensions, Amendments, Revisions

We have long since run out of space. So covering all the many ways in which DPL (in spirit if not always in letter) has been subsequently put to use is out of the question. But to close, I want to mention one prominent extension.

The fact that US and DPL depart from additive systems in different directions is part of what makes it interesting, and hard, to think about systems that track the dynamics of both discourse and world information. This is taken up in Groenendijk et al. (1996).[45]

One thing that needs grappling with is how $\exists x$ re-assigns values. Above we breezily slid from talk of distributive re-assignment to global random assignment; the difference didn't matter. To highlight the issue, consider these two options:

$$s[\exists x \phi] = \bigcup_{d \in D} (s[x/d][\phi]) \tag{8.49}$$

$$s[\exists x \phi] = (\bigcup_{d \in D} s[x/d])[\phi] \tag{8.50}$$

In the presence of a consistency-testing modality, there is a difference between these.

(10) a. Someone is hiding in the closet. He might be guilty.
$\exists x Hx \wedge \Diamond Gx$

b. Someone who is hiding in the closet might be guilty.
$\exists x(Hx \wedge \Diamond Gx)$

[45] There are similar issues when the dynamics of quantification and the dynamics of presupposition mix. A dynamic presupposition operator ∂ naturally has a test profile: $s[\partial\phi] = s$ if $s \models \phi$ and is undefined otherwise. This leads to a non-additive system for reasons of non-distributivity. So modeling the dynamics of quantification together with the dynamics of presupposition poses a lot of the same types of issues and requires a lot of the same kinds of innovations. On this score, see Beaver 2001.

These sentences (arguably) don't say the same thing and the distributive, but not the global, way of re-assigning values to variables can predict this.[46]

This, of course, is not the last word on how the dynamics of discourse and world information fit together, but it will in fact be the last word here.[47]

Acknowledgments This paper's main overarching debt is recounted in Footnote 8. More proximately, this paper is based on handouts and lectures I have used to teach DPL over the years; thanks are due to students and colleagues who tolerated and corrected them along the way, especially David Boylan, Jingyi Chen, Sam Carter, Nico Kirk-Giannini, Simon Goldstein, Jeff King, Zack Kofi, Sara Packalan, Paul Pietroski, Lauren Richardson, and Andrew Rubner. Thanks also to Zoltán Gendler Szabó, Louise McNally, and an anonymous referee for comments and suggestions.

Appendix: Some Official Definitions

Definition 16 (First Order Languages) L_{FO} is a first order language iff L_{FO} is built from a set V of variables, a set of predicate symbols, and the operators $\neg$, $\wedge$, and $\exists$ in the usual way.[48] Other operators are introduced as abbreviations: $(\phi \rightarrow \psi) = \neg(\phi \wedge \neg\psi)$, $(\phi \vee \psi) = \neg(\neg\phi \wedge \neg\psi)$, and $\forall x\phi = \neg\exists x\neg\phi$.

Definition 17 (Propositional Modal Languages) A propositional modal language $L_{\Diamond}$ is built on a set $A = \{p, q, \ldots\}$ of atomic sentences, and operators $\neg$, $\wedge$, and $\Diamond$ in the usual way. $\Box$ abbreviates $\neg\Diamond\neg$.

Definition 18 (PDL w/tests) Fix a set $\mathbf{A}$ of atomic programs. The language L_{PDL} is the smallest set of sentences including L_0 (propositional logic) and is closed as follows:

1. L_{PDL} is closed under $\rightarrow$ and $\neg$;
2. Π is the smallest set including $\mathbf{A}$ and is such that: (a) if $\pi_1, \pi_2 \in \Pi$ then so are $\pi_1; \pi_2$ (composition), $\pi_1 \cup \pi_2$ (choice), and π_1^* (iteration); and (b) if $\phi \in L_{\text{PDL}}$ then $\phi? \in \Pi$;
3. If $\phi \in L_{\text{PDL}}$ and $\pi \in \Pi$ then $\langle\pi\rangle\phi \in L_{\text{PDL}}$.

Definition 19 (Semantics for PDL) A PDL (propositional) model is a pair $M = \langle W, V\rangle$ where W is a non-empty set of states and V is a function taking atomic formulas to subsets of W and atomic programs to subsets of $W \times W$. $[\![\cdot]\!]^M$ is the interpretation function based on M iff for any p, a, ϕ, ψ, π:

[46] Suppose you don't know who did it and you don't know who is in the closet. Then (10a) is true. But if you also know that certain people are *not* guilty but you can't rule out that one of them is who is in the closet, then arguably (10b) isn't.

[47] Recommended reading: van Eijck and Cepparello (1994); Groenendijk et al. (1996); Aloni (2005); Dekker (2012).

[48] We focus on languages with variables and no constants. When there is no risk of confusion omit the subscript on L_{FO} (this goes for $L_{\Diamond}$ and L_{PDL}, too).

1. Atomic:
   (a) $[\![p]\!] = V(p)$ for atomic formulas
   (b) $[\![a]\!] = V(a)$ for atomic programs
2. Boolean:
   (a) $[\![\neg\phi]\!] = W \setminus [\![\phi]\!]$
   (b) $[\![\phi \rightarrow \psi]\!] = (W \setminus [\![\phi]\!]) \cup [\![\psi]\!]$
3. Programs:
   (a) $[\![\pi_1; \pi_2]\!] = [\![\pi_1]\!] \circ [\![\pi_2]\!]$
   (b) $[\![\pi_1 \cup \pi_2]\!] = [\![\pi_1]\!] \cup [\![\pi_2]\!]$
   (c) $[\![\pi^*]\!] = ([\![\pi]\!])^*$
   (d) $[\![\phi?]\!] = \{\langle w, w\rangle : \ w \in [\![\phi]\!]\}$
4. Modalities:
   (a) $[\![\langle\pi\rangle\phi]\!] = \{w : \ \text{there is a } v \text{ s.t. } \langle w, v\rangle \in [\![\pi]\!] \text{ and } v \in [\![\phi]\!]\}$
   (b) $[\![[\pi]\phi]\!] = \{w : \ \text{for all } v \text{ if } \langle w, v\rangle \in [\![\pi]\!] \text{ then } v \in [\![\phi]\!]\}$

In this definition, * has a dual role. Attached to a program as in π^* this is the program you get by executing π zero or more times; attached to a program's *meaning* (i.e., a relation on states) as in $([\![\pi]\!])^*$ this is the reflexive and transitive closure of the relation $[\![\pi]\!]$.

Definition 20 (Relational DPL) Let $M = \langle D, I\rangle$ be a first-order model and g an assignment function. Let $g[x]h$ iff for some $d \in D$: $h = g[x/d]$. Then $[\![\cdot]\!]^M_{\text{DPL}}$ is defined as follows:

1. $g\,[\![Pt_1, \ldots, t_n]\!]\,h$ iff $h = g$ and $\langle [t_1]_g, \ldots, [t_n]_g\rangle \in I(P)$
2. $g\,[\![\neg\phi]\!]\,h$ iff $h = g$ and for no k : $g\,[\![\phi]\!]\,k$
3. $g\,[\![\phi \wedge \psi]\!]\,h$ iff there is a k s.t. $g\,[\![\phi]\!]\,k$ and $k\,[\![\psi]\!]\,h$
4. $g\,[\![\exists x\phi]\!]\,h$ iff there is a k s.t. $g[x]k$ and $k\,[\![\phi]\!]\,h$

Definition 21 (Update Semantics) $[\cdot]_{\text{US}} : \ I \rightarrow I$ is defined as follows (where p is any atomic sentence and ϕ, ψ are any sentences of $L_\Diamond$):

1. $s[p] = \{w \in s : \ w(p) = 1\}$
2. $s[\neg\phi] = s \setminus s[\phi]$
3. $s[\phi \wedge \psi] = s[\phi][\psi]$
4. $s[\Diamond\phi] = \{w \in s : \ s[\phi] \neq \emptyset\}$

Definition 22 (Functional DPL) Let $M = \langle D, I\rangle$ be a first-order model. Define the DPL interpretation function $[\cdot]^M_{\text{DPL}} : \wp(D^V) \rightarrow \wp(D^V)$ as follows (super-/subscripts omitted when unambiguous):

1. $s[Px_1 \ldots x_n] = \{g \in s : \langle g(x_1), \ldots, g(x_n)\rangle \in I(P)\}$
2. $s[\neg\phi] = s \setminus \{g : \{g\}\,[\phi] \neq \emptyset\}$
3. $s[\phi \wedge \psi] = s[\phi][\psi]$
4. $s[\exists x\phi] = \bigcup_{d \in D}(s[x/d][\phi])$

References

Aloni, M. (2005). Individual concepts in modal predicate logic. *Journal of Philosophical Logic, 34*, 1–64.

Barwise, J. (1987). Noun phrases, generalized quantifiers and anaphora. In: P. Gärdenfors (Ed.), *Generalized Quantifiers: Linguistic and Logical Approaches* (pp. 1–29). Dordrecht: D. Reidel.

Beaver, D. I. (2001). *Presupposition and Assertion in Dynamic Semantics*. New York: CSLI Press.

Charlow, S. (2019). Where is the destructive update problem? Rutgers: Rutgers University.

Dekker, P. (1996). The values of variables in dynamic semantics. *Linguistics and Philosophy, 19*, 211–257.

Dekker, P. J. E. (2012). *Dynamic Semantics. Studies in Linguistics and Philosophy* (Vol. 91). Berlin: Springer.

Dekker, P. (2016). Exclusively indexical deduction. *The Review of Symbolic Logic, 9*(3), 603–637. https://doi.org/10.1017/S1755020316000125.

Egli, U. (1979). The stoic concept of anaphora. In R. Bäuerle, U. Egli, & A. von Stechow (Eds.) *Semantics from Different Points of View* (pp. 266–283). Berlin: Springer.

Gamut, L. T. F. (1991). *Logic, Language, and Meaning* (Vol. 2). Chicago: University of Chicago Press.

Geach, P. T. (1962/1980). *Reference and Generality* (3rd ed.). New York: Cornell University Press.

Geurts, B. (2019). On Kamp's "A theory of truth and semantic representation". In L. McNally, & Z. G. Szabo (Eds.). *A Reader's Guide to Classic Papers in Formal Semantics, Studies in Linguistics and Philosophy*. Berlin: Springer.

Geurts, B., Beaver, D. I., & Maier, E. (2016). Discourse respresentation theory. In E. Zalta (Ed.) *Stanford Encyclopedia of Philosophy* (Spring 2016 edition). http://plato.stanford.edu/archives/spr2016/entries/discourse-representation-theory/.

Gillies, A. S. (2020). Updating data semantics. *Mind, 129*, 1–41. https://doi.org/10.1093/mind/fzy008.

Groenendijk, J., & Stokhof, M. (1990). Two theories of dynamic semantics. In J. van Eijck (Ed.). *Logics in AI* (pp. 55–64). Berlin: Springer.

Groenendijk, J., & Stokhof, M. (1991). Dynamic predicate logic. *Linguistics and Philosophy, 14*, 39–100.

Groenendijk, J., Stokhof, M., & Veltman, F. (1996). Coreference and modality. In S. Lappin (Ed.). *The Handbook of Contemporary Semantic Theory* (pp. 179–216). Oxford: Blackwell.

Harel, D., Kozen, D., & Tiuryn, J. (2000). *Dynamic Logic*. New York: MIT Press.

Heim, I. (1982). The semantics of definite and indefinite noun phrases, PhD thesis, University of Massachusetts, Amherst. http://semanticsarchive.net/Archive/Tk0ZmYyY/dissertation.pdf.

Heim, I. (1983). File change semantics and the familiarity theory of definites. In R. Bäuerle, C. Schwarze, & A. von Stechow (Eds.). *Meaning, Use and the Interpretation of Laguage* (pp. 164–189). Berlin: de Gruyter.

Kamp, H. (1981). A theory of truth and semantic interpretation. In J. Groenendijk, T. Janssen, & M. Stokhof (Eds.) *Formal Methods in the Study of Language* (pp. 277–322). Amsterdam: Mathematical Centre.

Kamp, H., & Reyle, U. (1993). *From Discourse to Logic*. Dordrecht: Kluwer.

Karttunen, L. (1976). Discourse referents. In J. McCawley (Ed.) *Syntax and Semantics* (Vol. 7, pp. 363–85). New York: Academic Press.

Muskens, R., van Benthem, J., & Visser, A. (1997). Dynamics. In J. van Benthem, & A. ter Meulen (Eds.) *Handbook of Logic and Language* (pp. 587–648). Amsterdam: Elsevier.

Poesio, M., & Zucchi, A. (1992). On telescoping. In C. Barker & D. R. Dowty (Eds.) *Proceedings of SALT 2* (pp. 347–366).

Rothschild, D., & Yalcin, S. (2016). Three notions of dynamicness in language. *Linguistics and Philosophy, 39*(4), 333–355. https://doi.org/10.1007/s10988-016-9188-1.

Schubert, L., & Pelletier, F. J. (1988). Generically speaking, or, using discourse representation theory to interpret generics. In G. Chierchia, B. Partee, & R. Turner (Eds.), *Properties, types, and meaning, II* (pp. 193–268). Dordrecht: Kluwer.

van Benthem, J. (1986). *Essays in Logical Semantics*. Dordrecht: D. Reidel.

van Benthem, J. (1996). *Exploring Logical Dynamics*. New York: CSLI Press.

van Eijck, J., & Cepparello, G. (1994). Dynamic modal predicate logic. In M. Kanazawa, & C. Piñón Non (Eds.) *Dynamics, Polarity, and Quantification* (pp. 251–276). New York: CSLI Publications.

Veltman, F. (1996). Defaults in update semantics. *Journal of Philosophical Logic, 25*, 221–261. https://doi.org/10.1007/BF00248150.

Veltman, F. (2000). Proof systems for Dynamic Predicate Logic. Amsterdam: University of Amsterdam. https://staff.fnwi.uva.nl/f.j.m.m.veltman/papers/DPL2000.pdf.

Yalcin, S. (2012). Dynamic semantics. In G. Russell, & D. G. Fara (Eds.) *The Routledge Companion to Philosophy of Language* (pp. 253–279). London and New York: Routledge.