The donkey strikes back

Extending the dynamic interpretation "constructively"

Tim Fernando
fernando@cwi.nl
Centre for Mathematics and Computer Science
P.O. Box 4079, 1009 AB Amsterdam, The Netherlands

Abstract

The dynamic interpretation of a formula as a binary relation (inducing transitions) on states is extended by alternative treatments of implication, universal quantification, negation and disjunction that are more "dynamic" (in a precise sense) than the usual reductions to tests from quantified dynamic logic (which, nonetheless, can be recovered from the new connectives). An analysis of the "donkey" sentence followed by the assertion "It will kick back" is provided.

1 Introduction

The line

If a farmer owns a donkey he beats it (1)

from Geach [6] is often cited as one of the success stories of the so-called "dynamic" approach to natural language semantics (by which is meant Kamp [12], Heim [9], Barwise [1], and Groenendijk and Stokhof [7], among others). But add the note

It will kick back (2)

and the picture turns sour: processing (1) may leave no beaten donkey active. Accordingly, providing a referent for the pronoun it in (2) would appear to call for some non-compositional surgery (that may upset many a squeamish linguist). The present paper offers, as a preventive, a "dynamic" form of implication $\Rightarrow$ applied to (1). Based on a "constructive" conception of discourse analysis, an overhaul of Groenendijk and Stokhof [7]'s *Dynamic Predicate Logic* (DPL) is suggested, although $\Rightarrow$ can also be introduced less destructively so as to extend DPL conservatively. Thus, the reader who prefers the old "static" interpretation of (1) can still make that choice, and declare the continuation (2) to be "semantically ill-formed." On the other hand, Groenendijk and Stokhof [7] themselves concede that "at least in certain contexts, we need alternative externally dynamic interpretations of universal quantification, implication and negation; a both internally and externally dynamic treatment of disjunction." A proposal for such connectives is made below, extending the dynamic interpretation in a manner analogous to the extension of classical logic by constructive logic (with its richer collection of primitive connectives), through a certain conjunctive notion of parallelism.

To put the problem in a somewhat general perspective, let us step back a bit and note that in assigning a natural language utterance a *meaning*, it is convenient to isolate an intermediate notion of (say) a *formula*. By taking for granted a translation of the utterance to a formula, certain complexities in natural language can be abstracted away, and *semantics* can be understood rigorously as a map from formulas to meanings. Characteristic of the dynamic approach mentioned above is the identification of the meaning of a formula A with a binary relation on *states* (or contexts) describing transitions A induces, rather than with a set of states validating A. In the present paper, formulas are given by first-order formulas, and the target binary relations given by programs. To provide an account of anaphora in natural language, DPL translates first-order formulas A to programs A^{DPL} from (quantified) dynamic logic (see, for example, Harel [8]) as follows

$$A^{\mathsf{DPL}} = A? \qquad \text{for atomic } A$$

$$(A\&B)^{\mathsf{DPL}} = A^{\mathsf{DPL}} ; B^{\mathsf{DPL}}$$
$$(\neg A)^{\mathsf{DPL}} = \neg (A^{\mathsf{DPL}})$$
$$(\exists x\ A)^{\mathsf{DPL}} = x :=? ; A^{\mathsf{DPL}} .$$

The negation $\neg p$ of a program p is the dynamic logic test

$$([p]\ \bot)\ ?$$

with universal and static features (indicated respectively by $[p]$ and ?),[1] neither of which is intrinsic to the concept of negation. Whereas some notion of universality is essential to universal quantification and implication (which are formulated through negation

$$\forall x\ A = \neg\exists x\ \neg A$$
$$A \supset B = \neg(A\&\neg B)$$

and accordingly inherit some properties of negation), our treatment of (2) will be based on a *dynamic* (rather than static) form $\Rightarrow$ of implication. Dynamic forms of negation $\sim$, universal quantification and disjunction will also be proposed, but first we focus on implication.

2 The idea in brief

The semantics $[\![A]\!]$ assigned to a first-order formula A is that given to the program A^{DPL} — i.e., a binary relation on *states*. In dynamic logic, states are *valuations*; more precisely, the set of states is defined, relative to a fixed first-order model M and a set X of variables (from which the free variables of formulas A are drawn), as the set $|M|^X$ of functions $f, g, \ldots$ from X to the universe $|M|$ of M. Atomic programs come in two flavors: tests $A?$ where A is a formula in the signature of M with free variables from X, and random assignments $x :=?$ where $x \in X$. These are analyzed semantically by a function ρ taking a program p to a binary relation $\rho(p) \subseteq |M|^X \times |M|^X$ according to

$$f\rho(A?)g \quad \text{iff} \quad f = g \text{ and } M \models A[f]$$
$$f\rho(x :=?)g \quad \text{iff} \quad f = g \text{ except possibly at } x .$$

The programs are then closed under sequential composition (interpreted as relational composition)

$$f\rho(p;p')g \quad \text{iff} \quad f\rho(p)h \text{ and } h\rho(p')g \text{ for some } h ,$$

non-deterministic choice (interpreted as union)

$$f\rho(p + p')g \quad \text{iff} \quad f\rho(p)g \text{ or } h\rho(p')g ,$$

and Kleene star (interpreted as the reflexive transive closure). Rather than extending $\models$ simultaneously to formulas built from modalites $[p]$ and $\langle p\rangle$ labelled by programs p, it is sufficient to close the programs under a negation operation interpreted semantically as follows

$$f\rho(\neg p)g \quad \text{iff} \quad f = g \text{ and } f\rho(p)h \text{ for no } h .$$

As previously noted, $\neg p$ is equivalent to $([p]\bot)?$.

Returning to DPL, an implication $A \supset B$ between formulas is interpreted in DPL by equating it with $\neg\ (A\ \&\ \neg B)$, which is in turn translated into the dynamic logic program

$$\neg\ \ (A^{\mathsf{DPL}} ; \neg(B^{\mathsf{DPL}})) .$$

Applying the semantic function ρ to this then yields

$$s[\![A \supset B]\!]t \quad \text{iff} \quad t = s \text{ and } (\forall s' \text{ such that } s[\![A]\!]s')\ \exists t'\ s'[\![B]\!]t' . \qquad (3)$$

Now, given that a state is a single function from X to $|M|$, it is hardly odd that implication is static (in the sense that the input and output states s and t must be the same), as any number of instantiations of s' (and t') may be relevant to the right hand side of (3). That is, in terms of (1), the difficulty is that there may be several farmer/donkey couples, whereas a state can accomodate only one such pair, rendering an interpretation of (2) problematic. To overcome this predicament, the collection of states can be extended in at least two ways.

(P1) Borrowing and modifying an idea from Kleene [14] (and Brouwer, Kolmogorov,...), incorporate into the final state t a functional witness f to the $\forall\exists$-clause in the right hand side of (3) to obtain

$$s[\![A \Rightarrow B]\!]t \quad \text{iff} \quad t = (s, f) \text{ and } f \text{ is a function with domain } \{s' \mid s[\![A]\!]s'\} , \text{ and } (\forall s' \in \mathrm{dom}(f))\ s'[\![B]\!]f(s') .$$

Or, to simplify the state t slightly, break the condition (in the righthand side) up into two mutually exclusive clauses depending on whether or not the domain of f is empty

$$s[\![A \Rightarrow B]\!]t \quad \text{iff} \quad (t \text{ is a function with non-empty domain } \{s' \mid s[\![A]\!]s'\} \text{ and } (\forall s' \in \mathrm{dom}(t))\ s'[\![B]\!]t(s')) \text{ or } (t = s \text{ and } \neg\exists s'\ s[\![A]\!]s') ,$$

so that closing the notion of a state under a partial function space construct becomes sufficient.

[1] The semantics of dynamic logic is reviewed in the next section, where what exactly is meant, for example, by "static" is explained.

(P2) Keep only the image of a functional witness so that the new (expanded) set of states consists simply of the old states (i.e, valuations) together with sets of valuations. More precisely, define

$$s[\![A \Rightarrow B]\!]t \quad \text{iff} \quad (\exists \text{ a function } f \text{ with non-empty domain } \{s' \mid s[\![A]\!]s'\} \text{ where } t \text{ is the collapsed image of } f \text{ and } (\forall s' \in \mathrm{dom}(f)) \; s'[\![B]\!]f(s')) \text{ or } (t = s \text{ and } \neg\exists s' \; s[\![A]\!]s') \; . \qquad (4)$$

The "collapsed image of f",

$$\{t' \in |M|^X \mid \exists s' \; f(s') = t'\} \; \cup \; \bigcup\{t' \subseteq |M|^X \mid \exists s' \; f(s') = t'\}) \; ,$$

is simply the image of f except that the sets of valuations in the image are "collapsed", so that the resulting set has only valuations as elements. (The collapsing is "justified" by the associativity of conjunction.)

Observe that, in either case, DPL's negation can be derived

$$\neg A \quad = \quad A \Rightarrow \bot$$

(whence $\supset$ is also definable from $\Rightarrow$ and &). The first proposal, (P1), yields a dizzying tower of higher-order functions, in comparison to which, the second proposal is considerably simpler. Behind the step from (3) to either proposal is the idea that implication can spawn processes running in parallel. (Buried in (3) is the possibility of the input state s branching off to a multiplicity of states t'.) The parallelism here is "conjunctive" in that a family of parallel processes proceeds along happily so long as every member of the family is well; all is lost as soon as one fails.[2] More precisely, observe that, under (P2), a natural clause for $s[\![A]\!]t$, where s is a set of valuations and A is an atomic formula, is[3]

$$s[\![A]\!]t \quad \text{iff} \quad \exists \text{ a function } f : s \rightarrow_{onto} t \text{ such that } (\forall s' \in s) \; s'[\![A]\!]f(s') \; .$$

[2]The notion of parallelism is thus not unlike that of concurrent dynamic logic (Peleg [19]). By contrast, the (non-empty) sets of valuations used (e.g., in Fernando [4]) to bring out the eliminative character of information growth induced by tests A? live disjunctively (and die conjunctively).

[3]A (non-equivalent) alternative is

$$s[\![A]\!]t \quad \text{iff} \quad (\forall s' \in s) \; (\exists t' \in t) \; s'[\![A]\!]t' \text{ and } (\forall t' \in t) \; (\exists s' \in s) \; s'[\![A]\!]t' \; ,$$

yielding a more promiscuous ontology. This is studied in Fernando [5], concerning which, the reader is referred to the next footnote.

(That is, in the case of (2), *every* donkey that a farmer beats according to (1) must kick back.) A similar clause must be added to (P1), although to make the details for (P1) obvious, it should be sufficient to focus (as we will) on the case of (P2), where the states are structurally simpler. But then, a few words justifying the structural simplification in (P2) relative to (P1) might be in order.[4]

3 A digression: forgetfulness and information growth

If semantic analysis amounts abstractly to a mapping from syntactic objects (or formulas) to other mathematical objects (that we choose to call meanings), then what (speaking in the same abstract terms) is gained by the translation? Beyond some vague hope that the meanings have more illuminating structure than have the formulas, a reason for carrying out the semantic analysis is to abstract away inessential syntactic detail (with a view towards isolating the essential "core"). Thus, one might expect the semantic function not to be 1-1. The more general point is that an essential feature of semantic analysis is the process of *forgetting* what can be forgotten.

More concretely, turning to dynamic logic and its semantic function ρ, observe that after executing a random assignment $x :=?$, the previous (=input state) value of x is overwritten (i.e., forgotten) in the output state.[5] Perhaps an even more helpful example is the semantic definition of a sequential composition $p;p'$. The intermediate state arising after p but before p' is forgotten by $\rho(p;p')$ (tracking, as it does, only input/output states). Should such information be stored? No doubt, recording state histories would *not* decrease the scope of the account that can then be developed. It would almost surely increase it, but at what cost? The simpler the semantic framework, the better — all other things being equal, that is (chief among which is explanatory power). Otherwise, a delicate balance must be struck between the complexity of the framework and its scope. Now, part of the computational intuition underlying dynamic logic is that at any point in time, a state (i.e., valuation) embodies all that is relevant about the past to what can happen in the future. (In other words, the meaning of a program is specified simply by pairs of input/output states.) This same intuition underlies (P2), discarding (as it does) the wit-

[4]The discussion here will be confined to a somewhat intuitive and informal level. A somewhat more technical mathematical account is developed at length in Fernando [5], where (P2) is presented as a reduction of (P1) to a disjunctive normal form (in the sense of the "conjunctive" and "disjunctive" notions of parallelism already mentioned).

[5]It should, in fairness, be pointed out that Vermeulen [22] presents a variant of dynamic logic directed towards revising this very feature.

ness function tracing processes back to their "roots." (Forgetting that spawning record would seem to be akin to forgetting the intermediate state in a sequential composition $p;p'$.) Furthermore, for applications to natural language discourse, forgetfulness would appear quite innocuous if the information content of a state increases in the course of interpreting discourse (so that all past states have no more information content than has the current state). And it is quite natural in discourse analysis to assume that information does grow.

Consider the following claim in an early paper (Karttunen [13]) pre-occupied with a problem (viz., that of presuppositions) that may appear peripheral to (1) or (2), but is nonetheless fundamental to the "constructive" outlook on which $\Rightarrow$ is based

> There are definitions of pragmatic presupposition ... which suggest that there is something amiss in a discourse that does not proceed in [an] ideal orderly fashion. ... *All things considered, this is an unreasonable view.* ... People do make leaps and shortcuts by using sentences whose presuppositions are not satisfied in the conversational context. This is the rule rather than the exception, and we should not base our notion of presupposition on the false premiss that it does not or should not happen. But granting that ordinary discourse is not always fully explicit in the above sense, I think we can maintain that a *sentence is always taken to be an increment to a context that satisfies its presuppositions.* [p. 191, italics added]

To bring out an important dimension of "increment to a context", and at the same time get around the destruction of information in DPL by a random assignment, we will modify the translation $\cdot^{\mathsf{DPL}}$ (mapping first-order formulas into programs) slightly into a translation $\cdot^{e}$, over which (P2) will be worked out (though the reader should afterwards have no difficulty carrying out the similar extension to DPL). The modification is based (following Fernando [4], and, further back, Barwise [1]) on (i) a switch from valuations defined on all variables to valuations defined on only finitely many variables, and on (ii) the use of *guarded assignments* $x := *$ (in place of random assignments), given by

$$x = x? \quad + \quad \neg(x = x?)\ ;\ x :=? \ ,$$

which has the effect of assigning a value to x precisely when initially x is unbound (in which case the test $x = x?$ fails). Note that (i) spoils bivalence, which is to say that certain presuppositions may fail.[6] Accordingly, our translation $R(\overline{x})^e$ of an atomic formula $R(\overline{x})$ to a program must first attend to presuppositions by plugging truth gaps through guarded assignments, before testing $R(\overline{x})$

$$R(\overline{x})^e \quad = \quad \overline{x} := *\ ;\ R(\overline{x})? \qquad (5)$$

(where $\overline{x} := *$ abbreviates $x_1 := *; \ldots; x_k := *$ for $\overline{x} = x_1, \ldots, x_k$). To avoid clashes with variables bound by quantifiers, the latter variables might be marked

$$(\exists x\ A)^e \quad = \quad y_{A,x} := *\ ;\ A[y_{A,x}/x]^e \ , \qquad (6)$$

the idea being to sharpen (5) by translating atomic formulas $R(\overline{x},\overline{y},\overline{z})$ with unmarked variables $\overline{x}$, and marked variables $\overline{y},\overline{z}$ (for $\exists$ and $\forall$ respectively) as follows

$$R(\overline{x},\overline{y},\overline{z})^e \quad = \quad \overline{x} := *\ ;\ R(\overline{x},\overline{y},\overline{z})? \qquad (7)$$

Note that to assert a formula A is not simply to test A, but also to establish A (if this is at all possible). Establishing *not* A is (intuitively) different from testing (as in DPL) that A *cannot* be established.[7] A negation $\sim$ reflecting the former is described next, avoiding an appeal to a modal notion (hidden in $\neg$ by writing $\neg p$ instead of $([p]\bot)?$).

4 Working out the idea formally

Starting over and proceeding a bit more rigorously now, given a first-order signature $\mathbf{L}$, throw in, for every n-ary predicate symbol $R \in \mathbf{L}$, a fresh n-ary predicate symbol $\hat{R}$ and extend the map $\hat{\cdot}$ to these symbols by setting $\hat{\hat{R}} = R$. Then, interpret $\hat{R}$ in an $\mathbf{L}$-structure M as the complement of R

$$\hat{R}^M \quad = \quad |M|^n - R^M \ .$$

So, without loss of generality, assume that we are working with a signature $\mathbf{L}$ equipped with such a map $\hat{\cdot}$, and let M be an $\mathbf{L}$-model obeying the complementarity condition above (readily expressible in the first-order language). Fix a countable set X_0 of variables, and define two fresh (disjoint) sets Y and Z of "marked" variables inductively simultaneously with a set Φ of $\mathbf{L}$-formulas (built from $\&, \vee, \forall, \exists$ and $\Rightarrow$) as follows

(i) $\top, \bot$ and every atomic $\mathbf{L}$-formula with free variables from $X_0 \cup Y \cup Z$ is in Φ

(ii) if A and B are in Φ, then so are $A\&B$, $A \vee B$ and $A \Rightarrow B$

(iii) for every ("unmarked") $x \in X_0$, if $A \in \Phi$, then $\forall x\ A$ and $\exists x\ A$ belong to Φ

[6] To what extent an account of presuppositions can be based on the break down in bivalence resulting from uninitialized variables will not be taken up here. The interested reader is referred to Fernando [4] for an internal notion of proposition as an initial step towards this end.

[7] As detailed in Fernando [4], this distinction can be exploited to provide an account of Veltman [21]'s might operator as $\neg\neg\cdot$ relative to an internal notion of proposition.

(iv) for every $x \in X_0$, if $A \in \Phi$, then the fresh ("marked") variables $y_{A,x}$ and $z_{A,x}$ belong to Y and Z respectively.

Next, define a "negation" map $\sim \cdot$ on Φ by

$$\begin{aligned} \sim\top &= \bot \\ \sim\bot &= \top \\ \sim R(\overline{x},\overline{y},\overline{z}) &= \hat{R}(\overline{x},\overline{y},\overline{z}) \\ \sim(A\&B) &= \sim A \vee \sim B \\ \sim(A \vee B) &= \sim A \,\&\, \sim B \\ \sim(\forall x\ A) &= \exists x\ \sim A \\ \sim(\exists x\ A) &= \forall x\ \sim A \\ \sim(A \Rightarrow B) &= A \,\&\, \sim B\ . \end{aligned}$$

This approach, going back at least to Nelson [17] (a particularly appropriate reference, given its connection with Kleene [14]), treats positive and negative information in a nearly symmetric fashion; on formulas in Φ without an occurrence of $\Rightarrow$, the function $\sim\sim\cdot$ is the identity. Furthermore, were it not for $\Rightarrow$, our translation $\cdot^e$ would map formulas in Φ to programs interpreted as binary relations on

$$S_0 = \{s \mid s \text{ is a function from a finite subset of } X \text{ to } |M|\}\ ,$$

where X is the full set of marked an unmarked variables

$$X = X_0 \cup Y \cup Z\ .$$

All the same, the clauses for $s[\![A]\!]t$ can be formulated uniformly whether or not $s \in S_0$, so long as it is understood that for a set s of valuations, $u \in X$, and atomic A,

$$\begin{aligned} s\rho(u := *)t \quad &\text{iff} \quad \exists \text{ a function } f : s \to_{onto} t \text{ such that } (\forall s' \in s)\ s'\rho(u := *)f(s') \\ s\rho(A?)t \quad &\text{iff} \quad t = s \text{ and } (\forall s' \in s)\ s'\rho(A?)s'\ . \end{aligned}$$

(These clauses are consistent with the intuition described in section 2 of a "conjunctive" family of processes running in parallel.) The translation $\cdot^e$ is then given by (7),

$$\begin{aligned} (A\&B)^e &= A^e; B^e \\ (A \vee B)^e &= A^e + B^e\ , \end{aligned}$$

(6) and (4), with $|M|^X$ replaced by S_0. All that is missing is the clause for universal quantification $\forall x\ A$, which (following Kleene [14]) can be interpreted essentially as $z_{A,x} = z_{A,x} \Rightarrow A[z_{A,x}/x]$, except that in the antecedent, $z_{A,x}$ is treated as unmarked

$$\begin{aligned} s[\![\forall x\ A]\!]t \quad \text{iff} \quad & t \text{ is the collapsed image of a function } f \text{ with domain} \\ & \{s' \mid s\rho(z_{A,x} := *)s'\} \text{ such that } (\forall s' \in \mathrm{dom}(f)) \\ & \qquad s'[\![A[z_{A,x}/x]]\!]f(s')\ . \end{aligned}$$

The reader seeking the definition of $[\![A]\!]$ spelled out in full is referred to the appendix.

Observe that non-deterministic choice $+$ (for which DPL has no use) is essential for defining $\sim$. Strong negation $\sim$ is different from $\neg$, and lacks the universal force necessary to interpret implication (either as $\sim(\cdot\&\sim\cdot)$) or as $\cdot\vee\sim\cdot$). On the other hand, $\neg A$ can be recovered as $A \Rightarrow \bot$, whence static implication $\supset$ is also derivable. Note also that an element s of S_0 can be identified with $\{s\}$, yielding states of a homogeneous form.

5 A few examples

The present work does *not* rest on the claim that the disorderly character of discourse mentioned above by Karttunen [13] admits a compositional translation to a first-order formula. The problem of translating a natural language utterance to a first-order formula (e.g., assigning a variable to a discourse marker) is essentially taken for granted, falling (as it does) outside the scope of formal semantics (conceived as a function from formulas to meanings). This affords us considerable freedom to accomodate various interpretations. The donkey sentence (1) can be formulated as

$$\begin{gathered} \mathtt{farmer}(x)\ \&\ \mathtt{owns}(x,y)\ \&\ \mathtt{donkey}(y) \\ \Rightarrow \\ \mathtt{beats}(x,y) \end{gathered}$$

or given an alternative "weak" reading

$$\begin{gathered} \mathtt{farmer}(x)\ \&\ \mathtt{owns}(x,z)\ \&\ \mathtt{donkey}(z) \\ \Rightarrow \\ \mathtt{owns}(x,y)\ \&\ \mathtt{donkey}(y)\ \&\ \mathtt{beats}(x,y) \end{gathered}$$

so that not every donkey owned by a farmer need be beaten (Chierchia [2]). In either case, the pay back (2) can be formulated as

$$\mathtt{kicks\text{-}back}(y,x)\ .$$

A further alternative that avoids presupposing the existence of a donkey is to formulate (1) and (2) as

$$\begin{gathered} \mathtt{farmer}(x)\ \&\ \mathtt{owns}(x,y)\ \&\ \mathtt{donkey}(y) \\ \Rightarrow \\ \mathtt{beats}(x,y)\ \&\ \mathtt{kicks\text{-}back}(y,x)\ , \end{gathered}$$

observing that

$$[\![(A \Rightarrow B)\&C]\!] \neq [\![A \Rightarrow (B\&C)]\!]\ .$$

Next, we consider a few examples from Groenendijk and Stokhof [7]

If a client turns up, you treat him politely. You offer him a cup of coffee and ask him to wait. (8)

Every player chooses a pawn. He puts it

on square one. (9)

It is not true that John doesn't own a car.

It is red, and it is parked in front of his house. (10)

Either there is no bathroom here, or it is a funny place. In any case, it is not on the first floor. (11)

Example (8) can be formulated as

$$\texttt{client}(x)\ \&\ \texttt{turns-up}(x) \Rightarrow \texttt{treat-politely}(y, x)$$

followed by

$$\texttt{offer-coffee}(y, x)\ \ \&\ \ \texttt{ask-to-wait}(y, x)\ ,$$

and (9) as

$$\texttt{player}(x)\ \Rightarrow\ \texttt{choose}(x, y)\ \&\ \texttt{pawn}(y)$$

followed by

$$\texttt{put-on-square-one}(x, y)\ .$$

The double negation in (10) can be analyzed dynamically using $\sim\!\sim\cdot$, and (11) can be treated as

$$\texttt{bathroom}(x)\ \Rightarrow\ \neg\texttt{here}(x) \vee \texttt{funny-place}$$

followed by

$$\neg\texttt{on-first-floor}(x)\ ,$$

where, in this case, the difference between $\sim$ and $\neg$ is immaterial.

Groenendijk and Stokhof [7] suggest equating (*not* A) *implies* B, in its dynamic form, with $A \vee B$. To allow *not* A to be dynamic, *not* should not be interpreted as $\neg$. But even $(\sim A) \Rightarrow B$ is different from $A \vee B$, as the non-determinism in $A \vee B$ is lost in $(\sim A) \Rightarrow B$, and $\Rightarrow$ may lead to structurally more complex states ($\notin S_0$). What is true is that

$$\begin{aligned} \sim\!\sim((\sim A) \Rightarrow B) &= \sim((\sim A)\ \&\ \sim B) \\ &= (\sim\!\sim A) \vee \sim\!\sim B \end{aligned}$$

which reduces to $A \vee B$ if $\Rightarrow$ occurs neither in A nor B. Whereas the translation $\neg\neg\cdot$ yields a static approximation, the translation $\sim\!\sim\cdot$, applied recursively, projects to an approximation that is a binary relation on S_0.

Notice that quantifers do not appear in the translations above of natural language utterances into first-order formulas. The necessary quantification is built into the semantic analysis of quantifier-free formulas, following the spirit (if not the letter) of Pagin and Westerståhl [18]. (A crucial difference, of course, is that the universal quantification above arises from a dynamic $\Rightarrow$.) The reader interested in compositionality should be pleased by this feature, insofar as quantifer-free formulas avoid the non-compositional relabelling of variables bound by quantifiers (in the semantic analysis above of quantified formulas).

6 Concerning certain points

The present paper is admittedly short on linguistic examples — a defect that the author hopes some sympathetic reader (better qualified than he) will correct. Towards this end, it may be helpful to take up specific points (beyond the need for linguistic examples) raised in the review of the work (in the form it was originally submitted to EACL).

Referee 1. *What are the advantages over explanations of the anaphoric phenomenon in question in terms of discourse structure which do not require a change of the formal semantics apparatus?*

The "anaphoric phenomenon in question" amounts, under the analysis of first-order formulas as programs, to the treatment of variables across sentential boundaries. A variable can have existential force, as does the farmer in

A farmer owns a donkey,

or universal force, as does the farmer in

Every farmer owns a donkey.

Taking the "the formal semantics apparatus" to be dynamic logic, DPL treats existential variables through random assignments. The advantage of the proposal(s) above is the treatment of universal variables across sentential variables, based on an extension of dynamic logic with an implication connective (defined by (4), if A and B are understood as programs). (Note that negation and disjunction can be analyzed dynamically already within dynamic logic.)

Referee 2. *Suggestions for choosing between the static/dynamic versions would enhance the usefulness of the framework.*

Choose the dynamic version. Matching discourse items with variables is, afterall, done by magic, falling (as it does) outside the scope of DPL or *Discourse Representation Theory* (DRT, Kamp [12]). But the reader may have good reason to object.

Programme Committee. *A comparison to a DRT-style semantics should be added.*

Yes, the author would like to describe the *discourse representation structures* (DRS's) for the extension to higher-order states above. Unfortunately, he does not (at present) know how to.[8] Short of that, it may be helpful to present the passage to states that are conjunctive sets of valuations in a different light. Given a state that is a set s of valuations $s_1, s_2, \ldots$, let X_s be the set of variables in the domain of some $s_i \in s$

$$X_s\ =\ \bigcup_{s_i \in s} \mathrm{dom}(s_i)\ .$$

[8] Some steps (related to footnote 4) towards that direction are taken in Fernando [5]. Another approach, somewhat more syntactic in spirit, would be to build on K. Fine's arbitrary objects (Meyer Viol [15]).

Now, s can be viewed as a set F_s of functions f^x labelled by variables $x \in X_s$ as follows. Let f^x be the map with domain $\{s_i \in s \mid x \in \mathrm{dom}(s_i)\}$ that sends such an s_i to $s_i(x)$. In pictures, we pass from

$$s = \left\{ \begin{array}{c} s_1 : d_1 \to c_1 \\ s_2 : d_2 \to c_2 \\ \vdots \end{array} \right\}$$

to

$$F_s = \left\{ \begin{array}{c} f^{x_1} : \{s_i \in s \mid x_1 \in d_i\} \to c_1 \\ f^{x_2} : \{s_i \in s \mid x_2 \in d_i\} \to c_2 \\ \vdots \end{array} \right\},$$

so that the step from states $s_1, s_2, \ldots$ in S_0 to the more complicated states s in $\mathrm{Power}(S_0)$ amounts to a semantic analysis of variables as functions, rather than as fixed values from the underlying first-order model. (But now what is the domain of such a function?) The shift in point of view here is essentially the "ingenious little trick" that Muskens [16] (p. 418) traces back to Janssen [11] of swapping rows with columns. We should be careful to note, however, that the preceding analysis of variables was carried out relative to a fixed state s — a state s that is to be supplied as an argument to the partial binary functions globally representing the variables.

Finally, A. Visser and J. van Eijck have suggested that *a comparison with type-theoretic and game-theoretical semantics (e.g., Ranta [20] and Hintikka and Kulas [10]) is in order.*

This again is no simple matter to discuss, and (alas) falls somewhat beyond the scope of the present paper. For now, suffice it to say that (i) the translation $\cdot^e$ above starts from first-order formulas, on which (according to Ranta [20], p. 378) the game-theoretic "truth definition is equivalent to the traditional Tarskian one", and that (ii) the use of constructive logic in Ranta [20] renders the reduction from the proposal (P1) to (P2) (described in section 2) implausible inasmuch as that represents a (constructively unsound) transformation to a disjunctive normal form (referred to in footnote 4). But what about constructiveness?

7 Between construction and truth

Having passed somewhat hastily from (P1) to (P2), the reader is entitled to ask why the present author has bothered mentioning realizability (alluding somewhat fashionably or unfashionably to "constructiveness") and has said nothing about (classical) modal logic-style formalizations (e.g., Van Eijck and De Vries [3]), building say on concurrent dynamic logic (Peleg [19]). A short answer is that the connection with so-called and/or computations came to the author only after trying to understand the interpretation of implication in Kleene [14] (interpreting implication as a program construct being nowhere suggested in Peleg [19], which instead introduces a "conjunction" $\cap$ on programs). A more serious answer would bring up his attitude towards the more interesting question

does all talk about so-called dynamic semantics come to modal logic?

The crazy appeal dynamic semantics exerts on the author is the claim that a formula (normally conceived statically) is a program (i.e., something dynamic); showing how a program can be understood statically is less exciting. Some may, of course, find the possibility of "going static" as well as "going dynamic" comforting (if not pleasing). But if reducing dynamic semantics to static truth conditions is to complete that circle, then formulas must first be translated to programs. And that step ought not to be taken completely for granted (or else why bother talking about "dynamic semantics"). Understanding a computer program in a precise (say "mathematical") sense is, in principle, to be expected insofar as the states through which the computer program evolves can be examined. If a program can be implemented in a machine, then it has a well-defined operational semantics that, moreover, is subject (in some sense or another) to Church's thesis. In that sense, understanding a computer *program* relative to a mathematical world of eternal truths and static formulas is not too problematic. Not too problematic, that is, when compared to natural language, for which nothing like Church's thesis has gained acceptance. To say that

natural language is a programming language

is outrageous (— perhaps deliberately so —), and those of us laboring under this slogan must admit that we do not know how to translate an English sentence into a FORTRAN program (whatever that may mean). Nor, allowing for certain abstractions, formulas into programs. Furthermore, a favorite toy translation, DPL, goes beyond ordinary computability (and FORTRAN) when interpreted over the natural numbers. (The culprit is $\neg$.) *Not* that the idea of a program must necessarily be understood in the strict sense of ordinary recursion theory. But some sensitivity to matters relating to computation ("broadly construed") is surely in order when speaking of programs.

It was the uncomputable character of DPL's negation and implication that, in fact, drove the present work. Strong negation $\sim$ is, from this standpoint, a mild improvement, but it would appear that the situation for implication has only been made more complicated. This complication can be seen, however, as only a first step towards getting a handle on the computational character of the programs used in interpreting formulas dynamically. Whether more effective forms of realizability (incorporating, as was

originally conceived, some notion of construction or proof into the witnessing by functions) can shed any helpful light on the idea of dynamic semantics is an open question. That realizability should, crazily enough, have anything to say whatsoever about a linguistic problem might hearten those of us inclined to investigate the matter. (Of course, one might take the easy way out, and simply restrict $\Rightarrow$ to finite models.)

Making certain features explicit that are typically buried in classical logic (such as the witness to the $\forall\exists$-clause in $\Rightarrow$) is a characteristic practice of constructive mathematics that just might prove fruitful in natural language semantics. A feature that would seem particularly relevant to the intuition that discourse interpretation amounts to the construction of a context is information growth.[9] The extension of the domain of a finite valuation is an important aspect of that growth (as shown in Fernando [4], appealing to Henkin witnesses, back-and-forth constructions, ...). The custom in dynamic logic of reducing a finite valuation to the set of its total extensions (relative to which a static notion of truth is then defined) would appear to run roughshod over this feature — a feature carefully employed above to draw a distinction between establishing and testing a formula (mentioned back at the end of section 3).

But returning to the dynamic implication $\Rightarrow$ introduced above, observe that beyond the loss of structure (and information) in the step from (P1) to (P2), it is possible within (P2) (or, for that matter, within (P1)) to approximate $\Rightarrow$ by more modest extensions. There is, for instance, the translation $\sim\!\sim\cdot$ (not to be confused with $\neg\neg\cdot$) which (in general) abstracts away structure with each application. The interpretation of implication can be simplified further by noting that $\neg\pi$ can be recovered as $\pi \Rightarrow \bot$, and thus the static implication $\supset$ of DPL can be derived from $\Rightarrow$. Reflecting on these simplifications, it is natural to ask what structure can dynamic semantics afford to forget?

Is there more structure lurking behind construction than concerns truth?

With the benefit of the discussion above about the dual (establishing/testing) nature of asserting a proposition — or perhaps even without being subjected to all that babble —, surely we can agree that

Story-telling requires more imagination than verifying facts.

[9] The idea that information grows during the run of a typical computer program is, by comparison, not so clear. One difference is that whereas guarded assignments would seem sufficient for natural language applications, a typical computer program will repeatedly assign different values to the same variable. To pursue the matter further, the reader may wish to (again) consult Vermeulen [22].

Acknowledgments

My thanks to J. van Eijck and J. Ginzburg for criticisms of a draft, to K. Vermeulen, W. Meyer-Viol, A. Visser, P. Blackburn D. Beaver, and M. Kanazawa for helpful discussions, and to the conference's anonymous referees for various suggestions.

Appendix: (P2) fleshed out without prose

Fix a first-order model M and a set X of variables partitioned between the unmarked $(x, \ldots)$ and marked $(y, \ldots$ and $z, \ldots$ for existential and universal quantification, respectively). (It may be advisable to ignore the marking of variables, and quantified formulas; see section 5 for some examples.) Let S_0 be the set of functions defined on a finite subset of X, ranging over the universe of M. Given a sequence $\overline{u}$ of variables $u_1, \ldots, u_n$ in X, define the binary relation $\rho(\overline{u} := *)$ on s and $t \in S_0 \cup \mathrm{Power}(S_0)$ by

$$
\begin{aligned}
s\rho(\overline{u} := *)t \quad \text{iff} \quad & (s \in S_0\ ,\ t \in S_0\ ,\ t \supseteq s \text{ and} \\
& \mathrm{dom}(t) = \mathrm{dom}(s) \cup \{u_1, \ldots, u_n\}) \\
& \text{or} \\
& (s \notin S_0 \text{ and} \\
& \exists \text{ a function } f : s \rightarrow_{onto} t \text{ such} \\
& \text{that } (\forall s' \in s)\ s'\rho(\overline{u} := *)f(s'))\ .
\end{aligned}
$$

L-formulas A from the set Φ defined in section 3 are interpreted semantically by binary relations

$$
[\![A]\!] \subseteq (S_0 \cup \mathrm{Power}(S_0)) \times (S_0 \cup \mathrm{Power}(S_0))
$$

according to the following clauses, understood inductively

$$
\begin{aligned}
s[\![R(\overline{x}, \overline{y}, \overline{z})]\!]t \quad \text{iff} \quad & (s \in S_0\ ,\ s\rho(\overline{x} := *)t \\
& \text{and } M \models R[t]) \\
& \text{or} \\
& (\exists \text{ a function } f \text{ from} \\
& s \text{ onto } t \text{ such that} \\
& (\forall s' \in s) \\
& \quad s'[\![R(\overline{x}, \overline{y}, \overline{z}]\!]f(s')) \\
s[\![A \& B]\!]t \quad \text{iff} \quad & s[\![A]\!]u \text{ and } u[\![B]\!]t \text{ for} \\
& \text{some } u \\
s[\![A \vee B]\!]t \quad \text{iff} \quad & s[\![A]\!]t \text{ or } s[\![B]\!]t \\
s[\![\forall x\ A]\!]t \quad \text{iff} \quad & t \text{ is the collapsed image} \\
& \text{of a function } f \text{ with} \\
& \text{domain} \\
& \{s' \mid s\rho(z_{A,x} := *)s'\} \\
& \text{such that} \\
& (\forall s' \in \mathrm{dom}(f)) \\
& \quad s'[\![A[z_{A,x}/x]]\!]f(s') \\
s[\![\exists x\ A]\!]t \quad \text{iff} \quad & s\rho(y_{A,x} := *)u \text{ and}
\end{aligned}
$$

$$
\begin{array}{lll}
 & & u[\![A[y_{A,x}/x]]\!]t \text{ for} \\
 & & \text{some } u \\
s[\![A \Rightarrow B]\!]t & \text{iff} & (\exists \text{ a function } f \text{ with} \\
 & & \text{non-empty domain} \\
 & & \{s' \mid s[\![A]\!]s'\} \text{ where} \\
 & & t \text{ is the collapsed} \\
 & & \text{image of } f \text{ and} \\
 & & (\forall s' \in \mathrm{dom}(f)) \\
 & & \qquad s'[\![B]\!]f(s')) \\
 & & \text{or} \\
 & & (t = s \text{ and} \\
 & & \neg\exists s'\ s[\![A]\!]s') \ ,
\end{array}
$$

and, not to forget negation,

$$
\begin{array}{lll}
s[\![\top]\!]t & \text{iff} & s = t \\
s[\![\bot]\!]t & \text{iff} & \text{you're a donkey}
\end{array}
$$

(in which case you are free to derive anything).

References

[1] Jon Barwise. Noun phrases, generalized quantifiers and anaphora. In E. Engdahl and P. Gärdenfors, editors, *Generalized Quantifiers*, Studies in Language and Philosophy. Dordrecht: Rediel, 1987.

[2] G. Chierchia. Anaphora and dynamic logic. ITLI Prepublication, University of Amsterdam, 1990.

[3] J. van Eijck and F.J. de Vries. Dynamic interpretation and Hoare deduction. *J. Logic, Language and Information*, 1, 1992.

[4] Tim Fernando. Transition systems and dynamic semantics. In D. Pearce and G. Wagner, editors, *Logics in AI*, LNCS 633 (subseries LNAI). Springer-Verlag, Berlin, 1992. A slightly corrected version has appeared as CWI Report CS-R9217, June 1992.

[5] Tim Fernando. A higher-order extension of constraint programming in discourse analysis. Position paper for the First Workshop on Principles and Practice of Constraint Programming (Rhode Island, April 1993).

[6] P.T. Geach. *Reference and Generality: an Examination of Some Medieval and Modern Theories.* Cornell University Press, Ithaca, 1962.

[7] J. Groenendijk and M. Stokhof. Dynamic predicate logic. *Linguistics and Philosophy*, 14, 1991.

[8] David Harel. Dynamic logic. In D. Gabbay and F. Guenthner, editors, *Handbook of Philosophical Logic, Volume 2.* D. Reidel, 1984.

[9] Irene Heim. The semantics of definite and indefinite noun phrases. Dissertation, University of Massachusetts, Amherst, 1982.

[10] J. Hintikka and J. Kulas. *The Game of Language.* D. Reidel, Dordrecht, 1983.

[11] Theo Janssen. *Foundations and Applications of Montague Grammar.* Dissertation, University of Amsterdam (published in 1986 by CWI, Amsterdam), 1983.

[12] J.A.W. Kamp. A theory of truth and semantic representation. In J. Groenendijk et. al., editors, *Formal Methods in the Study of Language.* Mathematical Centre Tracts 135, Amsterdam, 1981.

[13] Lauri Karttunen. Presupposition and linguistic context. *Theoretical Linguistics*, pages 181–194, 1973.

[14] S.C. Kleene. On the interpretation of intuitionistic number theory. *J. Symbolic Logic*, 10, 1945.

[15] W.P.M. Meyer Viol. Partial objects and DRT. In P. Dekker and M. Stokhof, editors, *Proceedings of the Eighth Amsterdam Colloquium.* Institute for Logic, Language and Computation, Amsterdam, 1992.

[16] Reinhard Muskens. Anaphora and the logic of change. In J. van Eijck, editor, *Logics in AI: Proc. European Workshop JELIA '90.* Springer-Verlag, 1991.

[17] David Nelson. Constructible falsity. *J. Symbolic Logic*, 14, 1949.

[18] P. Pagin and D. Westerståhl. Predicate logic with flexibly binding operators and natural language semantics. Preprint.

[19] David Peleg. Concurrent dynamic logic. *J. Assoc. Computing Machinery*, 34(2), 1987.

[20] Aarne Ranta. Propositions as games as types. *Synthese*, 76, 1988.

[21] Frank Veltman. Defaults in update semantics. In J.A.W. Kamp, editor, *Conditionals, Defaults and Belief Revision.* Edinburgh, Dyana deliverable R2.5.A, 1990.

[22] C.F.M. Vermeulen. Sequence semantics for dynamic logic. Technical report, Philosophy Department, Utrecht, 1991. To appear in *J. Logic, Language and Information.*