Dynamic Interpretation and Hoare Deduction
Extended Abstract

Jan van Eijck[1,2] & Fer-Jan de Vries[1]

[1]CWI, Amsterdam

[2]OTS, Utrecht

1 Introduction

This paper presents a dynamic assignment language (called *DAL*) with η and ι assignments and generalized quantifiers, in the style of dynamic predicate logic [8]. The constructs for η and ι assignments allow us a straightforward analysis of indefinite and definite descriptions in natural language. The addition of quantifiers permit us to treat a wide variety of 'donkey' sentences.

Given a translation π of a natural language sentence S that maps pronouns to dynamically bound variables (see [2], [8]), we use ideas from Hoare's logic to calculate the static meaning of S as the weakest precondition for which program π can succeed. Here are some simple examples we claim we can treat correctly.

(1) *If a girl has a boyfriend, she teases him.*

(2) *Every girl who has a boyfriend teases him.*

(3) *Most girls who have a boyfried tease him.*

Our account gives (2) and (3) two possible readings, (these coincide with so-called *weak* and *strong* readings known from the literature for such examples). The strong reading of (2) is equivalent to the reading we get for (1). Our treatment of (3) does not suffer from the so-called *proportion problem* (see [10]). We can handle non-conservative quantifiers, provided they do not have internal dynamic effects. We cannot at present handle (4), because of the presence of the donkey anaphor in a non-conservative quantifier context.

(4) *Only girls who do tease a boyfriend lose him.*

Our proposal for the treatment of quantifiers can be viewed as a compositional reformulation (and quite possibly a correction) of previous proposals for the definition of dynamic generalized quantifiers (see e.g. [13]).

After the introduction of the language and a discussion of its semantics, we present a proof system for it in terms of Hoare's logic. In the full paper ([6]) it is proved that this deduction system is sound and complete with respect to the dynamic semantics.

2 Dynamic Assignment Language: Syntax

We first define the set of programs of *DAL* and the set av of *assignment variables* of a *DAL* program. For simplicity's sake we take the terms of *DAL* to be a set of individual variables V (one might want to add constants and deictic parameters to this, but we will not do so here).

Given a set of terms and a set of relation symbols, the set of *DAL* programs is the smallest set such that the following hold.

1. $\bot$ is a program.
2. If t_1, t_2 are terms, then $t_1 = t_2$ is a program.
3. If R is an n-place relation symbol and $t_1, \ldots, t_n$ are terms, then $R(t_1 \cdots t_n)$ is a program.
4. If π_1 and π_2 are programs then $(\pi_1; \pi_2)$ is a program.
5. If π_1 and π_2 are programs then $(\pi_1 \Rightarrow \pi_2)$ is a program.
6. If π is a program, then $\neg\pi$ is a program.
7. If π is a program and x is a variable, then $\eta x : \pi$ is a program.
8. If π is a program and x is a variable, then $\iota x : \pi$ is a program.
9. If π_1 and π_2 are programs and Q is a quantifier symbol, then $Q^w x(\pi_1, \pi_2)$ and $Q^s x(\pi_1, \pi_2)$ are programs.

The operator ; is used for program composition. $\Rightarrow$ denotes dynamic implication between programs. ηx is used for indefinite assignment, while ιx denotes definite assignment. The quantifier symbols Q are supposed to denote binary generalized quantifiers. Q^w and Q^s distinguish between the weak and strong readings of these. The discussion of the semantic clauses below will make the difference clear.

We will follow the usual predicate logical convention of omitting outermost parentheses for readability. Also, it will become evident from the semantic clause for sequential composition that the ; operator is associative. Therefore, we will often take the liberty to write $\pi_1; \pi_2; \pi_3$ instead of $(\pi_1; \pi_2); \pi_3$ or $\pi_1; (\pi_2; \pi_3)$. Also, we use $\top$ as an abbreviation for $\neg\bot$.

Here are the *DAL* translations of (1), (2) (strong reading), and (3) (weak reading).

(5) $(\eta x : \mathit{girl}\ x;\ \eta y : \mathit{boyfriend}\ y;\ \mathit{has}\ (x,y)) \Rightarrow \mathit{tease}\ (x,y).$

(6) $Q^s_\forall x(\mathit{girl}\ x;\ \eta y : \mathit{boyfriend}\ y;\ \mathit{has}\ (x,y), \mathit{tease}\ (x,y)).$

(7) $Q^w_M x(\mathit{girl}\ x;\ \eta y : \mathit{boyfriend}\ y;\ \mathit{has}\ (x,y), \mathit{tease}\ (x,y)).$

In these translations, $\eta v : \pi$ is a command to assign to variable v an object d for which *DAL* program π succeeds; $\Rightarrow$ denotes dynamic implication, $Q^s_\forall$ is the generalized universal quantifier (in its strong reading), and Q^w_M is the generalized quantifier *most* (in its weak reading).

3 Informal Semantics

This section is devoted to an informal account of the semantics of atomic test predicates, implication, negation and union of *DAL* programs, and η and ι assignment. Section 4 will give the formal semantics. The semantics of quantifiers will be presented in section 5.

The semantic objects of our prime interest are *states*, functions from the set of *DAL* variables to individuals in a model. Semantically, *DAL* programs act as state transformers: a *DAL* program takes an input state and either indicates success by producing an output state or it indicates failure by not producing anything at all. Equivalently, we can view the meaning of a program as a function mapping any input state to the set of all possible outputs the program can produce for that input. A program which is a test will on input A either produce output set $\{A\}$ (in case the test succeeds) or output set $\emptyset$ (in case the test fails). Programs which may produce non-singleton sets are non-deterministic; for some inputs there is more than one possible output state. Examples of non-deterministic programs are η assignment programs; the program $\eta x : \pi$ has, on input A, the set of all states which may differ from A in the fact that they have another x value, namely some value that satisfies π.

The program $\bot$ expresses a test which always fails; it is meant to express the same as **if** *true* **then** *fail* **else** *skip* **fi**. In other words: for every input state A, $\bot$ will produce output state $\emptyset$. As was mentioned above, we use $\top$ as an abbreviation for $\neg\bot$. The program $\top$ is a test which always succeeds; in other words, it is meant to express the same as the ALGOL style statement **if** *true* **then** *skip* **else** *fail* **fi**. In other words, for every input state A, $\top$ will produce output set $\{A\}$. Atomic predicates like $t_1 = t_2$ or $R(t_1 \cdots t_n)$ are meant to express tests which may fail; in ALGOL style notation: **if** $R(t_1 \cdots t_n)$ **then** *skip* **else** *fail* **fi**. Again in terms of input output behaviour: If $R(t_1 \cdots t_n)$ evaluates to true in state A, the predicate will have output set $\{A\}$, otherwise the output set will be $\emptyset$.

Programs of the form $(\pi_1 \Rightarrow \pi_2)$ are intended to treat the interplay of natural language implication and descriptions, as in the translation of example (1)

which was given above as (5). To get the semantics right, one has to assume that (5) is true if and only if every output state for the antecedent ηx : *girl* x; ηy : *boyfriend* y; *has*(x, y) will be an appropriate input state for the consequent *tease*(x, y) (see [2] or [8]).

Negation should allow one to treat examples like the following, where the negation has scope over an indefinite (for convenience we use indices to force the translation to corresponding variables).

(8) *The manager*^{1} *does not use a PC*^{2}.

This example can be translated into *DAL* as follows:

(9) $\iota v_1 : (\textit{manager}\ v_1);\ \neg(\eta v_2 : \textit{pc}\ v_2;\ \textit{use}(v_1, v_2))$.

To get the semantics right, a negated program should act as a test: $\neg\pi$ should accept (without change) all variable states which cannot serve as input for π, and reject all others. In fact, it will turn out that $\neg\pi$ is definable in terms of $\Rightarrow$ and $\bot$, as $\pi \Rightarrow \bot$.

Definite descriptions can act as anaphors and antecedents at the same time. Discourse (10) provides an example.

(10) *A customer*^{1} *entered. The woman*$^2_1$ *sat down. She*$_2$ *smiled.*

The indices indicate that *the woman* has *a customer* as its antecedent, while at the same time acting itself as antecedent for *she* in the next sentence (and constraining the gender of the pronoun). A *DAL* translation of (10) is given in (11).

(11) $\eta v_1 : \textit{customer}\ v_1;\ \textit{enter}\ v_1;\ \iota v_2 : (v_2 = v_1;\ \textit{woman}\ v_2);$
$\textit{sit-down}\ v_2;\ \textit{smile}\ v_2$.

The ι assignment in (11) is dependent on the η assignment to variable v_1. With reference to a particular assignment for v_1, the description is unique. Note that the ι assignment to v_2 does indirectly act as a test on the previous η assignment to v_1: this test will weed out η assignments that are inappropriate in the light of the subsequent discourse.

Definite descriptions can also be dependent on each other. Consider the string of characters in (12).

(12) $a\ \hat{A}\hat{}b\ C$.

Suppose just for an instant that (12) is a state of affairs one is talking about. The state of affairs involves characters and hat symbols (*hats* for short). With reference to (12), it does make sense to talk about *the character with the hat*, although (12) neither has a unique character nor a unique hat. We can, for instance, truthfully assert (13) about (12).

(13) *The character with the hat is a capital.*

The translation into *DAL* is straightforward:

(14) $\iota v_1 : (\textit{character}\ v_1;\ \iota v_2 : (\textit{hat}\ v_2;\ \textit{with}(v_1, v_2)));\ \textit{capital}\ v_1.$

Intuitively, the first ι assignment 'tries out' individual characters C until it finds the unique C with the property that a unique hat H for C can be found.

4 Semantics: Formal Definitions

The semantics of *DAL* programs is given in terms of input-output behaviour. Given a model $\mathcal{M}$ to interpret the basic vocabulary of a program π, the interpretation of program π, notation $[\![\pi]\!]$, is a function from variable states A for the language—mappings of (subsets of) the set of variables of the language to the domain of the model—to sets of variable states (the possible output states for variable state A). If for a given input state A, $[\![\pi]\!](A) \neq \emptyset$, then π succeeds on A, otherwise π fails on A.

Assume a model $\mathcal{M} = \langle U, I \rangle$, with U a universe of individuals and I an interpretation function for the first order relation symbols of the language. We consider the set $\mathbf{S}$ of all functions $A : V \to U$. This is the set of *states* for $\mathcal{M}$.

A state A for $\mathcal{M} = \langle U, I \rangle$ determines a valuation $\mathbf{V}_A$ for the terms of the language as follows: if $t \in V$ then $\mathbf{V}_A(t) = A(t)$ (as we take all our terms to be variables, this is all there is to the definition of $\mathbf{V}_A$). If A is a state for $\mathcal{M}$, x a variable and d an element of the universe or $\mathcal{M}$, then $A[x := d]$ is the state for $\mathcal{M}$ which is just like A except for the possible difference that x is mapped to d.

We define a function $[\![\pi]\!]_{\mathcal{M}} : \mathbf{S} \to \wp\mathbf{S}$ by recursion. A, B, C are used as metavariables over states. The function $[\![\pi]\!]_{\mathcal{M}}$ depends on the model $\mathcal{M}$, but for convenience we will often write $[\![\pi]\!]$ rather than $[\![\pi]\!]_{\mathcal{M}}$. The function should be read as: on input state A, π may produce any of the outputs in output state set $[\![\pi]\!](A)$.

1. $[\![\bot]\!](A) = \emptyset$.
2. $[\![R(t_1 \cdots t_n)]\!](A) = \begin{cases} \{A\} & \text{if } \langle \mathbf{V}_A(t_1), \ldots, \mathbf{V}_A(t_n) \rangle \in I(R) \\ \emptyset & \text{otherwise.} \end{cases}$
3. $[\![t_1 = t_2]\!](A) = \begin{cases} \{A\} & \text{if } \mathbf{V}_A(t_1) = \mathbf{V}_A(t_2) \\ \emptyset & \text{otherwise.} \end{cases}$
4. $[\![(\pi_1;\ \pi_2)]\!](A) = \bigcup\{[\![\pi_2]\!](B) \mid B \in [\![\pi_1]\!](A)\}$.
5. $[\![(\pi_1 \Rightarrow \pi_2)]\!](A) = \begin{cases} \{A\} & \text{if for all } B \in [\![\pi_1]\!](A) \text{ it holds that } [\![\pi_2]\!](B) \neq \emptyset \\ \emptyset & \text{otherwise.} \end{cases}$
6. $[\![\neg\pi]\!](A) = \begin{cases} \{A\} & \text{if } [\![\pi]\!](A) = \emptyset \\ \emptyset & \text{otherwise.} \end{cases}$
7. $[\![\eta x : \pi]\!](A) = \bigcup\{[\![\pi]\!](A[x := d]) \mid d \in U\}$.

8. $[\![\iota x : \pi]\!](A) = \begin{cases} [\![\pi]\!](A[x := d]) \text{ for the unique } d \in U & \\ \text{for which } [\![\pi]\!](A[x := d]) \neq \emptyset & \text{if such a } d \text{ exists} \\ \emptyset & \text{otherwise.} \end{cases}$

9. The rules for the quantifiers will be given in the next section.

Truth is defined in terms of input-output behaviour: π is true relative to model $\mathcal{M}$ if there are states A, B for $\mathcal{M}$ such that $B \in [\![\pi]\!]_{\mathcal{M}}(A)$. Two programs π_1, π_2 are equivalent if for every model $\mathcal{M}$ and every state A for $\mathcal{M}$, $[\![\pi_1]\!]_{\mathcal{M}}(A) = [\![\pi_2]\!]_{\mathcal{M}}(A)$.

Dynamic consequence is defined as follows: $\pi_1 \models \pi_2$ if for every model $\mathcal{M}$ and for all states A, B for $\mathcal{M}$: if $B \in [\![\pi_1]\!]_{\mathcal{M}}(A)$ then there is a state C with $C \in [\![\pi_2]\!]_{\mathcal{M}}(B)$.

The statement $\eta x : \pi$ performs a non-deterministic action, for it sanctions any assignment to x of an individual satisfying π. The statement acts as a test at the same time: in case there are no individuals satisfying π the set of output states for any given input state will be empty. In fact, the meaning of $\eta x : \pi$ can be thought of as a random assignment followed by a test, for $\eta x : \pi$ is equivalent to $\eta x : \top; \pi$, or in more standard notation, $x := ?; \pi$. It follows immediately from this explanation plus the dynamic meaning of sequential composition that $\eta x : (\pi_1); \pi_2$ is equivalent with $\eta x : (\pi_1; \pi_2)$.

The interpretation conditions for ι assignment make clear how the uniqueness condition is handled dynamically. The statement $\iota x : \pi$ consists of a test followed by a deterministic action in case the test succeeds: first it is checked whether there is a unique π; if so, this individual is assigned to x; otherwise the program fails (in other words, the set of output states is empty). Thus we see that the two programs $\iota x : (\pi_1); \pi_2$ and $\iota x : (\pi_1; \pi_2)$ are not equivalent. The program $\iota x : (\pi_1; \pi_2)$ succeeds if there is a unique object d satisfying $\pi_1; \pi_2$, while the requirement for $\iota x : (\pi_1); \pi_2$ is stronger: there has to be a unique individual d satisfying π_1, and d must also satisfy π_2.

5 The Semantics of Quantification

Quantifiers are treated as binary operators on programs that form test programs. The semantic clauses given below ensure that the quantifier variable receives its values in a dynamic way. The quantifier symbol itself has its usual meaning of a relation between sets.

Let $Q_1, Q_2, \ldots$ be a list of binary quantifier symbols. Assume that the interpretation functions of the models $\mathcal{M} = \langle \mathcal{D}, \mathcal{I} \rangle$ are extended with suitable interpretations for these. That is to say, for every Q_i, $I(Q_i)$ is a binary quantifier relation on $\wp(D)$, i.e. a relation satisfying the constraints of *extension*, *isomorphy* and *conservativity* (see for example [16]).

A quantifier relation $I(Q)$ is *conservative* (or: lives on its first argument) if $I(Q)(A, B)$ iff $I(Q)(A, A \cap B)$. A quantifier relation satisfies *extension* if adding or deleting individuals from the part of the universe which is outside the extension of

the arguments does not affect the relation, i.e., if the relation satisfies $I(Q)_E(A,B)$ iff $I(Q)_{E'}(A,B)$, for all E, E' with $E, E' \supseteq A \cup B$.

The semantic clauses for quantifier programs, in their weak and strong readings, respectively, now run as follows:

Weak readings of quantifiers

$$[\![Q^w x(\pi_1,\pi_2)]\!](A) = \begin{cases} \{A\} & \text{if} \\ & \langle \{d \in U \mid [\![\pi_1]\!](A[x:=d]) \neq \emptyset\}, \\ & \{d \in U \mid [\![\pi_1;\pi_2]\!](A[x:=d]) \neq \emptyset\}\rangle \\ & \qquad \in I(Q) \\ \emptyset & \text{otherwise.} \end{cases}$$

Strong readings of quantifiers

$$[\![Q^s x(\pi_1,\pi_2)]\!](A) = \begin{cases} \{A\} & \text{if} \\ & \langle \{d \in U \mid [\![\pi_1]\!](A[x:=d]) \neq \emptyset\}, \\ & \{d \in U \mid [\![\pi_1 \Rightarrow \pi_2]\!](A[x:=d]) \neq \emptyset\}\rangle \\ & \qquad \in I(Q) \\ \emptyset & \text{otherwise.} \end{cases}$$

The first of these semantic clauses ensures that example program (7) will succeed in all models where the majority of girls with boyfriends are girls who tease at least one of their boyfriends. The second semantic clause takes care of the strong reading of this example. In this reading, the program will succeed in all models where the majority of girls with boyfriends are girls who tease *all* their boyfriends.

Note that quantifier programs behave as tests: in case the test succeeds the set of output states has the input state as its only member. Externally, quantifier programs, as defined here, do not change assignments. This means that external dynamic effects of quantification are not yet taken into account in this proposal. In fact, they are beyond the scope of this paper.

To see how the semantics of quantification works in the simplest possible case, let us walk through example program (15).

(15) $Q^w x(Sx, Tx)$.

According to the semantic clause, on input state A this program gives $\{A\}$ iff the sets (16) and (17) are in the relation $I(Q)$.

(16) $\{d \in U \mid [\![Sx]\!](A[x:=d]) \neq \emptyset\}$.

(17) $\{d \in U \mid [\![Sx; Tx]\!](A[x:=d]) \neq \emptyset\}$.

According to the semantic clause for atomic tests and the definition of the valuation function for terms, the set (16) can be rewritten as (18).

(18) $\{d \in U \mid d \in I(S)\}$.

In the same way, and using the semantic clause for ; , the set (17) can be rewritten as (19).

(19) $\{d \in U \mid x \in I(S) \text{ and } x \in I(T)\}$.

Because the Q is assumed to denote a conservative quantifier with first argument as given by (18), (19) can be replaced by (20).

(20) $\{d \in U \mid x \in I(T)\}$.

Thus we find that (15) is true iff the sets given in (18) and (20) are in the relation denoted by the quantifier. This is the expected result.

Nothing out of the ordinary yet. In fact, it is easy to show that as long as the antecedent program π_1 does not have an external dynamic effect, there is nothing to choose between the weak and the strong readings of a quantifier.

Proposition 1 *If Q is a quantifier satisfying extension and conservativity and π_1 is a test, then the programs $Q^w x(\pi_1, \pi_2)$ and $Q^s x(\pi_1, \pi_2)$ are equivalent.*

Proof: Omitted. ∎

It is not difficult to show (in fact, it follows immediately from the proof of proposition 1, which is given in the full paper) that the strong readings of quantifiers do not depend on their conservativity at all. Thus, we will get the right results for non-conservative quantifiers provided they satisfy *extension* and provided their first argument is a test. It turns out that we can handle examples like *Only men are chauvinists*, because in this case the first argument is a test and *only* does satisfy extension. The recipe is simply to rely on the strong reading of the quantifier *only*.

The internal dynamic effect of a quantifier comes into play when its first argument is not a test. Typically, this is the case if the first argument contains a definite or indefinite which is not screened off by a test. The traditional 'donkey' examples such as (2), repeated here as (21), are cases in point (see [7]).

(21) *Every girl[1] who has a boyfriend[2] teases him$_2$.*

The *DAL* translation for the strong reading of example (21) is repeated here as (22).

(22) $Q^s_\forall v_1(Gv_1;\ \eta v_2 : Bv_2;\ Hv_1v_2, Tv_1v_2)$.

Here $Q^s_\forall$ denotes the generalized universal quantifier, i.e. the relation of inclusion, in its strong reading. Establishing the meaning of examples like (21) by direct reasoning about the operational semantics is awkward, so we will guide the reader through the thicket once more.

According to the semantic clause for quantifier programs, on input state A this program gives $\{A\}$ iff the sets (23) and (24) are in the inclusion relation.

(23) $\{d \in U \mid [\![Gv_1;\ \eta v_2 : Bv_2;\ Hv_1v_2]\!](A[v_1 := d]) \neq \emptyset\}$.

(24) $\{d \in U \mid [\![(Gv_1;\ \eta v_2 : Bv_2;\ Hv_1v_2) \Rightarrow Tv_1v_2]\!](A[v_1 := d]) \neq \emptyset\}$.

First we reduce (23). Applying the semantic clauses for ; and η assignment and for atomic tests makes clear that (23) describes the same set as (25).

(25) $\{d \in U \mid$ there is a $d' \in U$ such that $d \in I(G), d' \in I(B),$
$\langle d, d' \rangle \in I(H)\}$.

Similarly, application of the semantic clauses for ;, for η assignment, for $\Rightarrow$ and for atomic tests makes clear that (24) describes the same set as (26).

(26) $\{d \in U \mid$ for all $d' \in U$ such that $d \in I(G), d' \in I(B), \langle d, d' \rangle \in I(H),$
it holds that $\langle d, d' \rangle \in I(T)\}$.

Paraphrasing this, we see that the semantic clause for quantified programs entails that translation (22) of (21) is true iff the set of girls who have a boyfriend is included in the set of girls who tease all their boyfriends. Thus we see that the strong reading for (2) is equivalent to the reading we got for (1).

We invite the reader to look more closely at these examples to check our claim that the weak readings do depend on conservativity plus extension while the strong readings depend only on extension.

Our story about the dynamics of quantification does not work for the quantifier relation interpreting *only*, in those cases where there is an internal dynamic effect. *Only P Q* is true iff the set of non-Ps is included in the set of non-Qs, or equivalently, iff the set of Qs is included in the set of Ps. Because *only* is not conservative, (27) does not mean the same as (28).

(27) *Only girls^1 who tease a boyfriend^2 lose him$_2$.*

(28) *Only girls^1 who tease a boyfriend^2 tease a boyfriend and lose him$_2$.*

Neither does it mean the same as (29), although this paraphrase comes a bit closer.

(29) *Only girls^1 who tease a boyfriend^2 lose all the boyfriends that they$_1$ tease.*

Rather, (27) means something like (30).

(30) *Only girls^1 who tease a boyfriend^2 lose their boyfriend$_2$.*

This suggests that in this case the pronoun is a pronoun of laziness rather than a genuine donkey pronoun. But pronouns in the context of non-conservative quantifiers pose difficult problems, as is also borne out by the following example.

(31) *Only girls^1 who have a boyfriend^2 bring him$_2$ to the party.*

On its most salient reading, (31) is true as a matter of course, because it can be paraphrased as (32).

(32) *No girls^1 who don't have a boyfriend^2 will bring him$_2$ to the party.*

Interestingly, the paraphrase (32) poses a difficulty for our framework too. The problem is that the variable for *a boyfriend*^{2} is screened off by the negation operator, so that it is not available anymore at the level where *him*$_2$ looks for an antecedent. To deal with (32) one would again need externally dynamic negation. We leave the problem of non-conservative quantifiers that are internally dynamic with the remark that it merits further investigation.

Example (3), repeated here as (33), gives rise to the so-called *proportion problem* in traditional discourse representation theory (see [12] for the basics of discourse representation theory, [13] for an up-to-date formulation, and [10] for details on the proportion problem).

(33) *Most girls*^{1} *who have a boyfriend*^{2} *tease him*$_2$.

The proportion problem may arise in connection with (33) in case there are girls who are naughty enough to have a large number of boyfriends and to tease them all. Accounts which give rise to the proportion problem would handle (33) as a case of quantification over girl-boyfriend pairs. To see how the present proposal fares, consider the translation of (33) in *DAL*, under its weak reading, repeated here as (34).

(34) $Q^w_M v_1((girl\ v_1;\ \eta v_2 : boyfriend\ v_2;\ have\ (v_1, v_2)), tease\ (v_1, v_2))$.

This is true in state A if there are states B, C such that the sets given in (35) and (36) are in the Q_M-relation.

(35) $\{d \in U \mid B \in [girl\ v_1;\ \eta v_2 : boyfr\ v_2;\ have\ (v_1, v_2)](A[v_1 := d])\}$

(36) $\{d \in U \mid B \in [girl\ v_1;\ \eta v_2 : boyfr\ v_2;\ have\ (v_1, v_2);\ tease\ (v_1, v_2)](A[v_1 := d])\}$

The set given by (35) is the set of all girls who have a boyfriend, while the set given by (36) is the set of all girls who have a boyfriend and tease him. The quantification is over girls, as it should be, and not over girl–boyfriend pairs, as in the accounts which give rise to the proportion problem. In other words, this spells out a reading that does not suffer from the proportion problem. This is not the only possible reading; the strong reading can be had by replacing Q^w_M in the translation by Q^s_M and applying the semantic clause for strong readings of quantifiers.

To show that the treatment of quantification proposed here is different from the treatment proposed in [8], it is enough to show that the approach advocated there suffers from the proportion problem. One of the examples discussed in [8] (p.81) is, essentially, (37), in the reading which can be paraphrased as (38).

(37) *If a girl*^{1} *has a boyfriend*^{2} *she usually teases him*$_2$.

(38) *In most cases in which a girl*^{1} *has a boyfriend*^{2} *she teases him*$_2$.

To treat this example, Groenendijk and Stokhof reconstruct Lewis' adverb of quantification approach in dynamic predicate logic, by reading the quantifier as a relation

between sets of states. Dynamic implication, $\Rightarrow$, would then correspond to $\rightarrow_\forall$, to be interpreted as: for *all* output states A of the antecedent, applying the consequent to A will produce an output. Similarly, the examples with *usually* or *in most cases* are analyzed with $\rightarrow_M$, to be interpreted as: for *most* output states A of the antecedent, applying the consequent to A will produce an output. To see that this account does give rise to the proportion problem, observe that output states of the antecedent *a girl has a boyfriend* where Mary has John as a boyfriend and where the same Mary has Fred as a boyfriend will have to count as different states. The example sentences may have a reading where these should indeed count as different, but the point is that quantification over states makes it impossible to express readings where they should count as the same, as in the reading of (37) which is equivalent to the most salient reading of (33). For such cases, quantification over states does simply lead to incorrect results.

Our approach differs from the approach in [8] precisely in that quantification is always over individuals and never over states. Our reconstruction of (38) would be as follows. Because the quantification is over cases or occasions, we have to add an occasion parameter to the predicates used for translating verb phrases, so *have(x,y,o)* and *tease(x,y,o)* for *x has y at occasion o* and *x teases y at occasion o*, respectively. The *DAL* translation of (38) now becomes:

(39) $Q^w_M o_1(\eta v_1(\mathit{girl}\ v_1;\ \eta v_2 : \mathit{boyfriend}\ v_2;\ \mathit{have}\ (v_1, v_2, o_1)), \mathit{tease}\ (v_1, v_2, o_1))$.

To make this true, on our account, the set of occasions at which a girl has a boyfriend and the set of occasions at which a girl has a boyfriend which she teases must be in the Q_M-relation.

Of course, on our account there is still a fair amount of latitude as to how (37), (38) and (40) are interpreted.

(40) *If a girl has several boyfriends, she usually teases them.*

But the latitude resides where it belongs, for a margin of uncertainty remains as long as it is unclear what counts as an *occasion*, and it disappears as soon as this is resolved. As soon as we have a model where occasions are fully individuated, our quantificational analysis gives the right meanings. The discussion summarised in [10] of the meanings of 'donkey' examples with *usually* should therefore in our view be re-interpreted as a discussion of factors that might be involved in the individuation of occasions.

6 A Calculus for Dynamic Interpretation

Discussions about the correctness or incorrectness of proposals for dynamic interpretation of language have been hampered in the past by the difficulty of seeing through the ramifications of the dynamic semantic clauses in non-trivial cases. (Incidentally,

this is why we cannot be sure if our proposal is a correction of the proposal for dynamic treatment of generalized quantifiers in [13].) To remedy this, we supplement the dynamic semantics of our representation language with a calculus in the style of Hoare (see [1] for an overview of this approach). The axioms and proof rules we propose form a deduction system allowing us to prove statements about *DAL* programs.

Our deductive system for dynamic logic is a hybrid calculus, with statements characterizing variable states, plus two kinds of correctness statements, which we call ***universal*** and ***existential*** correctness statements. Thus, the system has three kinds of statements: (i) formulae of a language of first order predicate logic with the same sets of variables and predicate letters as the *DAL* language under consideration, and extended with the same set of generalized quantifiers (call this language L), (ii) triples of the form $\{\varphi\}\ \pi\ \{\psi\}$, where φ, ψ are L-formulae, and π is a *DAL*-program, and (iii) triples of the form $\langle\varphi\rangle\ \pi\ \langle\psi\rangle$, where again φ, ψ are L-formulae, and π is a *DAL*-program.

The statements of the form φ are used for making assertions about variable states A for L with respect to models $\mathcal{M}$ for L. Because the *DAL* language and the assertion language L have the same set of variables, variable states for the *DAL* language are variable states for L. The relation $\mathcal{M} \models \varphi[A]$, for state A verifies φ in $\mathcal{M}$, is defined in the standard way.

The statements of the form $\{\varphi\}\ \pi\ \{\psi\}$ are ***universal*** correctness statements. In the terminology of Hoare's logic, they express partial correctness. The statement $\{\varphi\}\ \pi\ \{\psi\}$ expresses that all variable states A (for an arbitrary model $\mathcal{M}$) such that $\mathcal{M} \models \varphi[A]$ have the property that if some variable state B is an output state of π for input state A, then $\mathcal{M} \models \psi[B]$.

The statements of the form $\langle\varphi\rangle\ \pi\ \langle\psi\rangle$ are ***existential*** correctness statements. In terms of Hoare's logic, they represent the bits one has to add to partial correctness statements to ensure total correctness. The statement $\langle\varphi\rangle\ \pi\ \langle\psi\rangle$ expresses that for all input variable states A (for an arbitrary model $\mathcal{M}$) such that $\mathcal{M} \models \varphi[A]$ there is some variable state B satisfying $\mathcal{M} \models \psi[B]$ in the set of output states of π.

Because our intuitions about static meaning seem to be much better developed than our intuitions about dynamic meaning, we can, for a large class of natural language sentences, check whether the intuitive meaning of a sentence S corresponds to the meaning of its *DAL* translation π in the following precise sense. Does the intuitive meaning of S precisely describe the set of states for which π terminates successfully? In terms of Hoare's logic, we can describe this set of states by the weakest existential precondition of π with respect to $\top$. What we are looking for is the weakest φ for which the statement $\langle\varphi\rangle\ \pi\ \langle\top\rangle$ is still true. The φ we are looking for has to satisfy the additional condition that it does not contain free occurrences of the assignment variables of π (the variables for which π may have changed the values between input and output state; the full paper makes this precise); φ gives the ***static meaning*** of the program π.

Our calculus allows us to find weakest preconditions, as follows. Start with the conclusion $\langle ? \rangle\ \pi\ \langle \top \rangle$, and apply the rules of the calculus to work backwards, thus decomposing π. This will eventually produce a formula φ to fill the ? slot. In the full paper we prove that φ does indeed express the weakest precondition for existential correctness of π relative to $\top$.

It may seem that our intention to use the calculus to get from dynamic to static meaning will allow us to get by with just existential correctness statements. To see that this is not so, note that such statements do not allow us to express failure of a program for a given sets of input states. The statement $\langle \varphi \rangle\ \pi\ \langle \bot \rangle$ does *not* express failure of π on input states satisfying φ. Rather, it expresses the fact that for all inputs satisfying φ the program π is guaranteed to produce an output satisfying $\bot$, a statement which is absurd for all non-contradictory φ. Failure of a *DAL* program π on the set of inputs specified by φ, is readily expressed in terms of universal correctness, namely by $\{\varphi\}\ \pi\ \{\bot\}$.

It is clear that in order to treat negation of programs and dynamic implication between programs, both universal and existential correctness statements are needed in the calculus. Here are the axioms and rules of the deduction system for *DAL*.

Test Axioms

$\{\top\}\ \bot\ \{\bot\}$.

$\langle \bot \rangle\ \bot\ \langle \bot \rangle$.

$\{R(t_1 \cdots t_n) \rightarrow \varphi\}\ R(t_1 \cdots t_n)\ \{\varphi\}$.

$\langle R(t_1 \cdots t_n) \wedge \varphi \rangle\ R(t_1 \cdots t_n)\ \langle \varphi \rangle$.

$\{t_1 = t_2 \rightarrow \varphi\}\ t_1 = t_2\ \{\varphi\}$.

$\langle t_1 = t_2 \wedge \varphi \rangle\ t_1 = t_2\ \langle \varphi \rangle$.

For purposes of reasoning with the system one needs an oracle rule for the class $\mathcal{K}$ of models that one is interested in (for natural language applications such a class will generally be given by specifying a set of meaning postulates that all members of $\mathcal{K}$ should satisfy).

$\mathcal{K}$ Oracle Rule

Every assertion valid in $\mathcal{K}$ is an axiom.

The well-known consequence rule holds for universal and existential correctness.

Consequence Rules

$$\frac{\varphi \rightarrow \psi \qquad \{\psi\}\ \pi\ \{\chi\} \qquad \chi \rightarrow \xi}{\{\varphi\}\ \pi\ \{\xi\}.} \qquad \frac{\varphi \rightarrow \psi \qquad \langle \psi \rangle\ \pi\ \langle \chi \rangle \qquad \chi \rightarrow \xi}{\langle \varphi \rangle\ \pi\ \langle \xi \rangle.}$$

Next, one needs to provide rules for complex programs.

Rules of Composition

$$\frac{\{\varphi\}\ \pi_1\ \{\psi\} \qquad \{\psi\}\ \pi_2\ \{\chi\}}{\{\varphi\}\ (\pi_1; \pi_2)\ \{\chi\}.} \qquad \frac{\langle \varphi \rangle\ \pi_1\ \langle \psi \rangle \qquad \langle \psi \rangle\ \pi_2\ \langle \chi \rangle}{\langle \varphi \rangle\ (\pi_1; \pi_2)\ \langle \chi \rangle.}$$

Rules of Negation

$$\frac{\{\varphi\}\ \pi\ \{\bot\}}{\langle\varphi\wedge\psi\rangle\ \neg\pi\ \langle\psi\rangle}. \qquad \frac{\langle\varphi\rangle\ \pi\ \langle\top\rangle}{\{\varphi\vee\psi\}\ \neg\pi\ \{\psi\}}.$$

Rules of Implication

$$\frac{\{\varphi\}\ \pi_1\ \{\psi\} \qquad \langle\psi\rangle\ \pi_2\ \langle\top\rangle}{\langle\varphi\wedge\chi\rangle\ (\pi_1\Rightarrow\pi_2)\ \langle\chi\rangle}. \qquad \frac{\langle\varphi\rangle\ \pi_1\ \langle\psi\rangle \qquad \{\psi\}\ \pi_2\ \{\bot\}}{\{\varphi\vee\chi\}\ (\pi_1\Rightarrow\pi_2)\ \{\chi\}}.$$

Rules of η Assignment

$$\frac{\{\varphi\}\ \pi\ \{\psi\}}{\{\forall x\varphi\}\ \eta x:\pi\ \{\psi\}}. \qquad \frac{\langle\varphi\rangle\ \pi\ \langle\psi\rangle}{\langle\exists x\varphi\rangle\ \eta x:\pi\ \langle\psi\rangle}.$$

In the rules of ι assignment it is convenient to use $\exists!x\varphi$ as an abbreviation for $\exists x\forall y([y/x]\varphi\leftrightarrow y=x)$, where y is a variable which is free for x in φ.

Rules of ι Assignment

$$\frac{\langle\varphi\rangle\ \pi\ \langle\psi\rangle \qquad \{\neg\varphi\}\ \pi\ \{\bot\}}{\langle\exists!x\varphi\rangle\ \iota x:\pi\ \langle\psi\rangle}.$$

$$\frac{\langle\varphi\rangle\ \pi\ \langle\top\rangle \qquad \{\neg\varphi\}\ \pi\ \{\bot\} \qquad \{\psi\}\ \pi\ \{\chi\}}{\{\forall x(\forall y([y/x]\varphi\leftrightarrow y=x)\rightarrow\psi)\}\ \iota x:\pi\ \{\chi\}}.$$

Note that in the static description logic the η and ι operators from the dynamic assignment logic are contextually eliminated.

Rules of Quantification: Weak Readings

$$\frac{\langle\varphi\rangle\ \pi_1\ \langle\top\rangle \qquad \{\neg\varphi\}\ \pi_1\ \{\bot\} \qquad \langle\psi\rangle\ \pi_1;\pi_2\ \langle\top\rangle \qquad \{\neg\psi\}\ \pi_1;\pi_2\ \{\bot\}}{\langle Qx(\varphi,\psi)\wedge\chi\rangle\ Q^{w}x(\pi_1,\pi_2)\ \langle\chi\rangle}.$$

$$\frac{\{\neg\varphi\}\ \pi_1\ \{\bot\} \qquad \langle\varphi\rangle\ \pi_1\ \langle\top\rangle \qquad \{\neg\psi\}\ \pi_1;\pi_2\ \{\bot\} \qquad \langle\psi\rangle\ \pi_1;\pi_2\ \langle\top\rangle}{\{Qx(\varphi,\psi)\rightarrow\chi\}\ Q^{w}x(\pi_1,\pi_2)\ \{\chi\}}.$$

Rules of Quantification: Strong Readings

$$\frac{\langle\varphi\rangle\ \pi_1\ \langle\top\rangle \qquad \{\neg\varphi\}\ \pi_1\ \{\bot\} \qquad \langle\psi\rangle\ \pi_1\Rightarrow\pi_2\ \langle\top\rangle \qquad \{\neg\psi\}\ \pi_1\Rightarrow\pi_2\ \{\bot\}}{\langle Qx(\varphi,\psi)\wedge\chi\rangle\ Q^{s}x(\pi_1,\pi_2)\ \langle\chi\rangle}.$$

$$\frac{\{\neg\varphi\}\ \pi_1\ \{\bot\} \qquad \langle\varphi\rangle\ \pi_1\ \langle\top\rangle \qquad \{\neg\psi\}\ \pi_1\Rightarrow\pi_2\ \{\bot\} \qquad \langle\psi\rangle\ \pi_1\Rightarrow\pi_2\ \langle\top\rangle}{\{Qx(\varphi,\psi)\rightarrow\chi\}\ Q^{s}x(\pi_1,\pi_2)\ \{\chi\}}.$$

If we know Q to be $\downarrow$MON, $\uparrow$MON, MON$\downarrow$ or MON$\uparrow$, then in the rules of quantification the first, second, third or fourth premiss, respectively, can be omitted.

7 Soundness of the Calculus

The above axioms and rules engender a notion of $\mathcal{K}$-derivation, as follows. A $\mathcal{K}$-derivation is a finite sequence of correctness formulae $F_1,\ldots,F_n$ such that for every

i, $1 \leq i \leq n$, F_i is a test axiom or a an axiom according to the $\mathcal{K}$ oracle rule, or F_i is the conclusion of an instance of one of the inference rules while the premisses of that rule occur among $F_1, \ldots, F_{i-1}$. A $\mathcal{K}$-derivation $F_1, \ldots, F_n$ is said to be a $\mathcal{K}$-derivation of F_n. F is called *$\mathcal{K}$-derivable* in the proof system if there is a $\mathcal{K}$-derivation of F. Notation: $\mathcal{K} \vdash F$. In the next section, the soundness of this proof system relative to $\mathcal{K}$ will be proved.

An inference from premisses $F_1, \ldots, F_n$ to conclusion F is called $\mathcal{K}$-valid if $\mathcal{K}$ validity of the premisses implies $\mathcal{K}$ validity of the conclusion. We will now show that the proof system given in the previous section is correct relative to $\mathcal{K}$, i.e. for every correctness statement F:

$\mathcal{K} \vdash F$ implies $\mathcal{K} \models F$.

To prove this, we first show that the axioms are $\mathcal{K}$-valid, and next that the inference rules preserve $\mathcal{K}$-validity. The soundness result then follows by induction on the length of derivations.

Theorem 2 (Soundness) *If $\mathcal{K} \vdash F$ then $\mathcal{K} \models F$.*

Proof: Omitted. ∎

8 Completeness of the Calculus

Suppose we establish $\mathcal{K} \vdash \langle\varphi\rangle\, \pi_1 \Rightarrow \pi_2\, \langle\varphi\rangle$ for some φ with $\mathcal{K} \models \varphi$. Then it follows by the soundness of the calculus that $\mathcal{K} \models \langle\varphi\rangle\, \pi_1 \Rightarrow \pi_2\, \langle\varphi\rangle$, and by the $\mathcal{K}$ validity of φ that $\mathcal{M} \models \pi_1[A]$ implies $\mathcal{M} \models \pi_2[A]$, for all $\mathcal{M} \in \mathcal{K}$ and all states A for $\mathcal{M}$ defined for φ. In other words, the proof system can be considered as an axiomatisation of the notion of dynamic consequence, relative to classes of models $\mathcal{K}$. To see that the proof system is powerful enough we also have to establish its completeness relative to $\mathcal{K}$. For this we need the concepts of the weakest universal precondition and the weakest existential precondition of a *DAL* program and a formula of the assertion language.

The weakest universal precondition of a *DAL* program π and an L formula ψ is the L formula φ for which the following holds: $\mathcal{M} \models \varphi[A]$ iff for all $B \in [\![\pi]\!]_{\mathcal{M}}(A)$, it holds that $\mathcal{M} \models \psi[B]$ (for arbitrary $\mathcal{M}$). The weakest existential precondition of a *DAL* program π and an L formula ψ is the L formula φ for which the following holds: $\mathcal{M} \models \varphi[A]$ iff there is a $B \in [\![\pi]\!]_{\mathcal{M}}(A)$ with $\mathcal{M} \models \psi[B]$ (for arbitrary $\mathcal{M}$).

Note that it follows immediately from these definitions that the weakest universal precondition of a program π and an L formula ψ equals the negation of the weakest existential precondition of π and $\neg\psi$. This is because for all $B \in [\![\pi]\!]_{\mathcal{M}}(A)$ it holds that $\mathcal{M} \models \psi[B]$ is equivalent to: there is no $B \in [\![\pi]\!]_{\mathcal{M}}(A)$ for which $\mathcal{M} \models \neg\psi[B]$. This equivalence means that either of the two notions would suffice for what follows. For practical purposes, however, it is convenient to use both weakest universal and weakest existential preconditions, so we will define functions for both.

It is not obvious at first sight that the weakest universal and existential precondition of a *DAL* program and an L formula always exist (as formulas of L), so we have to show that this is indeed the case. In the full paper, we inductively define functions $\mathbf{wup}(\pi, \psi)$ and $\mathbf{wep}(\pi, \psi)$ of which we then show that they express the weakest universal precondition, respectively the weakest existential precondition of π and ψ. The proof of this uses a case by case check; the cases of atomic programs are checked directly, and induction is used to check the cases of complex programs. Thus we arrive at the following lemma.

Lemma 3 (wep/wup adequacy) *For all $\mathcal{M} \in \mathcal{K}$, all states A for $\mathcal{M}$, all $\psi \in L$, and all $\pi \in DAL$:*

$\mathcal{M} \models \mathbf{wup}(\pi, \psi)[A]$ *iff it holds for all* $B \in [\![\pi]\!]_{\mathcal{M}}(A)$ *that* $\mathcal{M} \models \psi[B]$.

$\mathcal{M} \models \mathbf{wep}(\pi, \psi)[A]$ *iff there is a* $B \in [\![\pi]\!]_{\mathcal{M}}(A)$ *with* $\mathcal{M} \models \psi[B]$.

Proof: Omitted. ∎

Next, in the full paper, we prove the following result.

Lemma 4 (wep/wup derivability) *For all $\pi \in DAL$ and for all $\psi \in L$,*

$\mathcal{K} \vdash \langle \mathbf{wep}(\pi, \psi) \rangle\ \pi\ \langle \psi \rangle$ *and* $\mathcal{K} \vdash \{\mathbf{wup}(\pi, \psi)\}\ \pi\ \{\psi\}$.

Proof: Omitted. ∎

The rest of the proof of the completeness result is now very easy. We want to show that $\mathcal{K} \models F$ implies $\mathcal{K} \vdash F$. In case F equals φ for some $\varphi \in L$, $\mathcal{K} \models \varphi$ implies $\mathcal{K} \vdash \varphi$ by the $\mathcal{K}$ oracle rule. For the case where F equals $\langle \varphi \rangle\ \pi\ \langle \psi \rangle$ the reasoning is as follows. Assume (41).

(41) $\mathcal{K} \models \langle \varphi \rangle\ \pi\ \langle \psi \rangle$.

By the **wep** adequacy lemma it follows from (41) that (42).

(42) $\mathcal{K} \models \varphi \rightarrow \mathbf{wep}(\pi, \psi)$.

From this, by the $\mathcal{K}$ oracle rule, (43).

(43) $\mathcal{K} \vdash \varphi \rightarrow \mathbf{wep}(\pi, \psi)$.

From the wep/wup derivability lemma we have (44).

(44) $\mathcal{K} \vdash \langle \mathbf{wep}(\pi, \psi) \rangle\ \pi\ \langle \psi \rangle$.

From (43) and (44) by an application of the existential consequence rule, (45).

(45) $\mathcal{K} \vdash \langle \varphi \rangle\ \pi\ \langle \psi \rangle$.

By similar reasoning we derive from (46) that (47).

(46) $\mathcal{K} \models \{\varphi\}\ \pi\ \{\psi\}$.

(47) $\mathcal{K} \vdash \{\varphi\}\ \pi\ \{\psi\}$.

This completes the proof of the final result:

Theorem 5 (Completeness) *If* $\mathcal{K} \models F$ *then* $\mathcal{K} \vdash F$.

9 Use of the Calculus

The calculus allows us to derive the static meanings of *DAL* programs. As an example, we calculate the static meaning of the conditional donkey example in (5). We want to find the weakest precondition φ such that (48).

(48) $\langle\varphi\rangle\ (\eta x : Gx;\ \eta y : By;\ Hxy) \Rightarrow Txy\ \langle\top\rangle$.

According to the rule for implication, we are done if we can find the weakest ψ such that $\langle\psi\rangle\ Txy\ \langle\top\rangle$ and then calculate the weakest φ such that (49).

(49) $\{\varphi\}\ \eta x : Gx;\ \eta y : By;\ Hxy\ \{\psi\}$.

It follows from one of the test axioms plus the existential consequence rule that ψ equals Txy. Thus, we are done if we can find the weakest φ such that (50).

(50) $\{\varphi\}\ \eta x : Gx;\ \eta y : By;\ Hxy\ \{Txy\}$.

Two applications of the universal rule for composition and two applications of the universal rule for η assignment give the end result.

(51) $\{\forall x(Gx \rightarrow \forall y(By \rightarrow (Hxy \rightarrow Txy)))\}\ \eta x : Gx;\ \eta y : By;\ Hxy\ \{Txy\}$.

As we have not used the consequence rules for presupposition strengthening, we have indeed calculated the weakest precondition. From the fact that the procedure calculates the weakest precondition under which (5) can succeed it follows that $\forall x(Gx \rightarrow \forall y(By \rightarrow (Hxy \rightarrow Txy)))$ is the static meaning of this program.

Next, we derive the static meaning of (3) (under its weak reading), the translation of which is repeated again for convenience.

(52) $Q^w_M v_1(Gv_1;\ \eta v_2 : Bv_2;\ Hv_1v_2, Tv_1v_2)$.

We want to find the weakest precondition φ for which (53).

(53) $\langle\varphi\rangle\ Q^w_M v_1(Gv_1;\ \eta v_2 : Bv_2;\ Hv_1v_2, Tv_1v_2)\ \langle\top\rangle$.

In view of the first quantifier rule, we know that φ equals $Q^w_M v_1(\psi, \chi)$, where ψ and χ are given by (54), (55), (56), and (57).

(54) $\langle\psi\rangle\ Gv_1;\ \eta v_2 : Bv_2;\ Hv_1v_2\ \langle\top\rangle$.

(55) $\{\neg\psi\}\ Gv_1;\ \eta v_2 : Bv_2;\ Hv_1v_2\ \{\bot\}$.

(56) $\langle\chi\rangle\ Gv_1;\ \eta v_2 : Bv_2;\ Hv_1v_2;\ Tv_1v_2\ \langle\top\rangle$

(57) $\{\neg\chi\}\ Gv_1;\ \eta v_2 : Bv_2;\ Hv_1v_2;\ Tv_1v_2\ \{\bot\}$.

Note that (55) and (57) are only there to guarantee that ψ and χ are *weakest* existential preconditions of the given programs with respect to $\top$. The quantifier

rule says that the rule in this case would still hold if we omit (57) (by virtue of the fact that Q_M is MON↑), but of course then there is no guarantee anymore that $Q_M v_1(\psi, \chi)$ expresses the *weakest* existential precondition of (52) with respect to $\top$. However, if we take care not to use the consequence rules we calculate weakest preconditions anyway, so then we can omit (55) and (57) and still arrive at the weakest existential precondition of (52) with respect to $\top$.

Application of the rules for η assignment and composition give the following value for ψ:

(58) $Gv_1 \wedge \exists v_2(Bv_2 \wedge Hv_1v_2)$.

(59) $Gv_1 \wedge \exists v_2(Bv_2 \wedge Hv_1v_2 \wedge Tv_1v_2)$.

Thus, we arrive at the following static meaning for (52):

(60) $Q_M v_1(Gv_1 \wedge \exists v_2(Bv_2 \wedge Hv_1v_2), Gv_1 \wedge \exists v_2(Bv_2 \wedge Hv_1v_2 \wedge Tv_1v_2))$.

One final remark on the fact that our calculus is geared to finding preconditions, given a program and an output condition. We hope to have demonstrated the usefulness of this in the above examples. However, one might also be interested in calculating postconditions (which of course is possible with the calculus). It is straightforward to check the following. For a program π which is a test, calculating the weakest existential precondition of π with respect to $\top$ is equivalent to calculating the strongest universal postcondition of π with respect to $\top$. For programs which are not tests, this equivalence breaks down, but in such cases there is a different reason for being interested in postconditions. The strongest universal postcondition with respect to $\top$ of a program π which is not a test will have free occurrences of precisely those variables that are available for external dynamic binding in programs π' following π.

10 Conclusion

In this paper we have demonstrated the potential of the use of tools from programming language semantics for the semantics of natural language. While η and ι assignment can in principle be decomposed in random assignment with subsequent testing, we have two reasons for preferring the treatment we gave. In the first place, extending the treatment of definite descriptions with an account of their presuppositions (which is an obvious next move in the framework we have presented) would make a decomposition of ι assignment impossible or at least very impractical. Secondly, and more importantly, ι and η assignments are to be preferred over a decomposition in terms of random assignment plus subsequent testing because the introduction of an individual by a definite or an indefinite noun phrase gets an exact counterpart in the dynamic translation language. In other words, the real merit of ι and η assignments is that they allow faithfulness to linguistic form.

Instead of the universal and existential Hoare-style correctness statements that we employed we might have used the toolkit of dynamic logic (cf. [9]). Replacing $\{\varphi\}\ \pi\ \{\psi\}$ by $\varphi \rightarrow [\pi]\psi$ and $\langle\varphi\rangle\ \pi\ \langle\psi\rangle$ by $\varphi \rightarrow \langle\pi\rangle\psi$ is all there is to such a change. Still, we prefer our notation, for several reasons. In the first place, it is less cluttered than the dynamic logic notation. Next, the full expressive power of dynamic logic is not needed for our purposes, so it seems wiser to choose a tool that fits the requirements more precisely. Finally, the static $\{\varphi\}$ and $\langle\varphi\rangle$ statements can be used as comments to annotate *DAL* programs, thus providing proof outlines for deriving the static meanings of programs.

Acknowledgement

This paper has benefitted from helpful comments by Krzysztof Apt, Reinhard Muskens, Martin Stokhof, Johan van Benthem and Albert Visser.

References

[1] K.R. Apt, 'Ten Years of Hoare's Logic: A Survey—Part I', *ACM Transactions on Programming Languages and Systems*, Vol. 3, No. 4, October 1981, 431–483.

[2] J. Barwise, 'Noun Phrases, Generalized Quantifiers and Anaphora', in P. Gärdenfors (ed.), *Generalized Quantifiers / Linguistic and Logical Approaches*, 1–29, Reidel, Dordrecht, 1987.

[3] J. van Benthem, 'General Dynamics', ITLI report, Amsterdam, 1990 (to appear in *Theoretical Linguistics*).

[4] Gennaro Chierchia, 'Anaphora and Dynamic Logic', in M. Stokhof, J. Groenendijk & D. Beaver (eds.), *Quantification and Anaphora I*, Edinburgh 1991 (DYANA deliverable R2.2.A), 37–78.

[5] G. Evans, 'Pronouns', in: *Linguistic Inquiry*, 11, 337-362.

[6] J. van Eijck & F.J. de Vries, 'Dynamic Interpretation and Hoare Deduction', CWI Technical Report CS-R9115, Amsterdam 1991 (submitted for publication to the *Journal of Logic, Language and Information*).

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

[8] J. Groenendijk & M. Stokhof, 'Dynamic Predicate Logic', *Linguistics and Philosophy*, 14, 1991, 39–100.

[9] D. Harel, 'Dynamic Logic', in D. Gabbay & F. Guenthner, *Handbook of Philosophical Logic, Vol. II*, Reidel, Dordrecht, 1984, 497–604.

[10] I. Heim, 'E-Type Pronouns and Donkey Anaphora', *Linguistics and Philosophy*, 13, 1990, 137–177.

[11] C.A.R. Hoare, 'An Axiomatic Basis for Computer Programming', *Communications of the ACM*, Vol. 12, no. 10, 1969, 567–580, 583.

[12] H. Kamp, 'A Theory of Truth and Semantic Representation', in Groenendijk e.a. (eds.), *Formal Methods in the Study of Language*, Mathematisch Centrum, Amsterdam, 1981.

[13] H. Kamp & U. Reyle, *From Discourse to Logic*, Manuscript, Institute for Computational Linguistics, University of Stuttgart, 1990.

[14] C. Roberts, 'Modal Subordination and Pronominal Anaphora in Discourse', *Linguistics and Philosophy*, 12, 1989, 683–721.

[15] B. Russell, 'On Denoting', *Mind*, 14, 1905, 479–493.

[16] D. Westerståhl, 'Quantifiers in Formal and Natural Languages', in Gabbay & Guenthner (eds.), *Handbook of Philosophical Logic*, Vol. IV, Reidel, Dordrecht, 1989, 1–131.

Jan van Eijck — jve@cwi.nl
Fer-Jan de Vries — ferjan@cwi.nl

CWI, Kruislaan 413, 1098 SJ Amsterdam — The Netherlands