The Dynamics of Description

Jan van Eijck

CWI, Kruislaan 413, 1098 SJ Amsterdam, The Netherlands

&

OTS, Trans 10, 3512 JK Utrecht, The Netherlands

Abstract

In static semantics of natural language, the use of definite and indefinite descriptions encounters certain difficulties. This paper shows that these problems can to a large extent be overcome by switching to a dynamic perspective, and demonstrates how description operators acquire a new lustre and attractiveness in a dynamic setup.

The paper first presents a dynamic assignment language with η and ι assignment in the style of dynamic predicate logic [10]. The constructs for η and ι assignment allow a very straightforward analysis of indefinite and definite descriptions in natural language. It is shown how the standard [5, 10] way of defining the semantics for these leads to a Russellian treatment of definite descriptions. A Hoare style calculus for this system (a subset of the calculus from [5]) is introduced and briefly discussed. It is shown how the Hoare style rules allow us to calculate static truth conditions, or equivalently, static falsity conditions of dynamic representation structures.

Next, the dynamic semantics is enriched with error states, intended to monitor failure of uniqueness presuppositions for definite descriptions. The rules for ι assignment can now take the presuppositions of the use of definite descriptions into account, which gets us a Strawsonian treatment of definites. It is briefly indicated how a Hoare calculus for the error state semantics might be devised.

1985 Mathematics Subject Classification: 03B65, 68Q55, 68S10.
CR Categories: F.3.1, F.3.2, I.2.4, I.2.7.
Keywords and Phrases: semantics of natural language, dynamic interpretation, Hoare logic, knowledge representation languages.

1 Descriptions in Static Logic

Descriptions are used in logic in a variety of ways. As an example, consider the way in which Hilbert and Bernays use them. In Hilbert & Bernays [13], a definite descriptor $\iota x : \varphi(x)$ is introduced, and it is proposed that a ι term may be used after it has been proved that there is a referent and that the referent is unique.

(1) $$\frac{\exists x \varphi(x). \quad \forall x \forall y((\varphi(x) \wedge \varphi(y)) \rightarrow x = y).}{\varphi(\iota x : \varphi(x)).}$$

The proposal is meant to capture mathematic usage only, but it is a fairly close approximation of the use of definite descriptions in natural language, as *intended* by the language user. If someone uses a definite description in a given context, he or she intends the phrase to refer uniquely in that context. Because of the unicity requirement, the definite descriptor $\iota x : \varphi(x)$ is rather difficult to handle.

Interestingly, Hilbert and Bernays also introduce a notation for indefinite descriptions. They use $\eta x : \varphi(x)$ for an arbitrary member from the set of things satisfying φ. The use of η terms for natural language analysis is advocated in Reichenbach [17]. As in the case of ι terms, it is necessary to ensure that a presupposition is fulfilled. The indefinite descriptor is introduced by the following schema.

(2) $$\frac{\exists x \varphi(x).}{\varphi(\eta x : \varphi(x)).}$$

Hilbert and Bernays remark that the indefinite descriptor $\eta x : \varphi(x)$ is difficult to handle because of the requirement that there be φs. However, unless in the case of the unicity requirement for definite descriptions, this defect can easily be remedied.

To improve on the concept of η terms, Hilbert and Bernays introduce an operator ϵ to refer to arbitrary objects satisfying a predicate φ, or to an arbitrary member of the universe in case there are no φs. In case there are φs, $\epsilon x : \varphi(x)$ denotes an arbitrary φ, in case there are no φs, $\epsilon x : \varphi(x)$ denotes an arbitrary non-φ. Hilbert and Bernays are not concerned with the semantics of ϵ-terms; the ϵ-terms are introduced as a proof theoretic device, and it is proved that if $\Gamma \vdash_\epsilon \varphi$ where Γ and φ are ϵ-free, then $\Gamma \vdash \varphi$. In other words: a deduction of an ϵ-free formula from ϵ-free hypotheses that uses ϵ-terms can always be replaced by an ϵ-free deduction. This means that it is unnecessary to define truth conditions for arbitrary formulae with epsilon terms. Such ϵ-terms are introduced by axioms of the form (3).

(3) $$\varphi(t) \rightarrow \varphi(\epsilon v : \varphi(v)/t).$$

Here φ can be any formula and t can be any term.

In this paper I will demonstrate, among other things, that ι terms and η terms can be made to behave very nicely in dynamic logic, and that in a dynamic setting there is no need for ϵ terms at all.

2 The Dynamics of Picking an Arbitrary φ

In static logic, the command 'Pick an arbitrary φ' is awkward, because there may not be such a φ. The introduction of ϵ terms to remedy this defect does not really help, because the semantics of ϵ terms is unintuitive. A semantic account (see Leisenring [16]) involves a choice function Φ that picks out a member of the universe for every definable subset of the universe. Given a particular choice function Φ, the interpretation of $\epsilon x : \varphi(x)$ is the object d that Φ assigns to the set $[\![\varphi(x)]\!]$. In case there are no φs, the term $\epsilon x : \varphi(x)$ will refer to the individual that is the value under Φ of the empty set.

Suppose we apply this to natural language, and translate an indefinite noun phrase such as *a man* as an epsilon term. Suppose we use the noun phrase twice. The second use of *a man* will then refer to the same individual as the first. This is hardly ever what we want. In some cases this bug can be fixed by using a translation $\epsilon y : (man\ y \wedge y \neq (\epsilon x : man\ x))$, a description that refers to a different individual than $\epsilon x : man\ x$, but this ploy can never be a systematic remedy. What we want, instead, is to employ different choice functions as we go along, and to let the interpretation process fail in case no appropriate choice of φ is possible because there are no φs. It turns out that dynamic logic for natural language in the spirit of Barwise and Groenendijk & Stokhof [2, 10] gives roughly the results we want, on condition that one disregards the presuppositions of the use of descriptions.

Consider mini discourse (4), where the hearer is asked to take two different individuals in mind.

(4) *A man walked in. He sat down. Another man walked in.*

Since we intend the reading where *he* is anaphorically linked to the earlier indefinite and *another* is anaphorically constrained (to borrow a term from Barwise [2]) by that same indefinite, we may use indices to indicate the intention. I follow Barwise [2] in using superscript indices for antecedents and subscripts for anaphors.

(5) *A man*^{1} *walked in. He*$_1$ *sat down. Another*$_1$ *man*^{2} *walked in.*

What intuitively happens when one processes discourse (5) can be described as follows. First one is invited to focus on an arbitrary man. Then one is asked to consider a choice of man where that man walked in. Furthermore one is asked to focus on a choice of man where that man sat down as well. Next one is assumed to keep this choice of man in mind, and again to pick a reference to an arbitrary

man, but in such a way that that man is different from the first man. Finally one is to consider a second choice of man where that second man walked in.

This account sounds like a piece of imperative programming, which suggests that its meaning can be given in terms of a translation into a programming language. Here is such a translation (tense is ignored), in a language which has the same expressive power as the dynamic predicate logic of Groenendijk & Stokhof [10], but which reveals its imperative programming nature a bit more clearly.

(6) $\eta v_1 : \mathit{man}\ v_1;\ \mathit{walk\text{-}in}\ v_1;\ \mathit{sit\text{-}down}\ v_1;$
$\eta v_2 : (v_2 \neq v_1;\ \mathit{man}\ v_2);\ \mathit{walk\text{-}in}\ v_2.$

The programming language employed in (6) has two kinds of basic statements: assignments and tests. Sequences of statements are formed with the sequencing operator ;. The assignments are non-deterministic, which means that semantically the program is not a function from states to states, but a relation between states (or equivalently, a function from states to sets of states). Test statements narrow down the set of output states. A test relates an input state that satisfies it to itself, and a input state that does not satisfy it to nothing at all.

I will fix the meaning for this assign-and-test mini-language by giving a dynamic semantics for it. For good measure, I will also provide a set of Hoare style rules for the representation language. The advantage of this addition is that the Hoare style rules provide a link to notions of static semantics. This allows us to take snapshots of truth conditions at various stages in the discourse processing, so to speak. Stated otherwise, the Hoare style rules allow us to consider projections from dynamic logic to static logic, in the sense of Van Benthem [3].

3 Dynamic Assignment Logic: Syntax and Informal Semantics

This section is meant as a brief introduction to dynamic predicate logic in its undisguised form as an imperative programming language. We have already encountered atomic tests, sequential program composition, and indefinite assignment; in the syntax description of *DAL* (Dynamic Assignment Logic) below we add implication, negation and definite assignment.

In natural language, one does not engage in explicitly bookkeeping with regard to the 'slots' used for keeping track of individuals mentioned in discourse. One just keeps them in mind, and does not confuse them, that is all. One could make sure that in *DAL* the slots do not get confused by stipulating that new assignments to variables which are already 'active' are forbidden (see Van Eijck & De Vries [5]), but it turns out that such scrupulousness is unnecessary.

The set of programs of *DAL* has as its terms a set $C \cup V$, where C is a set of individual constants and V a set of individual variables. Individual constants

are needed in the translation of proper names. Individual variables will be used in the translation of indefinite and definite descriptions and in the translations of anaphoric pronouns.

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.

I 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, I 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, $t_1 \neq t_2$ will be used to abbreviate $\neg t_1 = t_2$ (cf. example (6)).

The remainder of this section is devoted to an informal account of the dynamic semantics of atomic test predicates, implication and negation of *DAL* programs, and η and ι assignment. The next section will give the formal dynamic semantics.

Semantically, what we are interested in is *states*, functions from the set of *DAL* variables to individuals in a model. Semantically, *DAL* programs act as state transformers: a *DAL* program maps input states to sets of output states. A program maps an 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.

I will 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 $\{A\}$. 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 $\emptyset$. 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 following example.

(7) *If a man*^{1} *admires the king*^{2}*,* *he*$_1$ *cheers him*$_2$*.*

To get the semantics (roughly) right, one has to assume that (7) is true if and only if every output state for the antecedent will be an appropriate input state for the consequent (see Barwise [2] or Groenendijk & Stokhof [10]).

Negation should allow one to treat examples like the following, where the negation has scope over an indefinite.

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

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

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

To get the semantics right (again, roughly), 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 turns out that $\neg\pi$ is definable in terms of $\Rightarrow$ and $\bot$, as $\pi \Rightarrow \bot$.

Definite descriptions can be used as anaphors, while at the same time acting as antecedents. 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) $\eta v_1 : customer\ v_1;\ enter\ v_1;$
$\iota v_2 : (v_2 = v_1;\ woman\ v_2);\ sit\text{-}down\ v_2;\ smile\ \ v_2$.

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) $\iota v_1 : (character\ v_1;\ \iota v_2 : (hat\ v_2;\ with(v_1, v_2)));\ capital\ v_1.$

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.

The semantic picture sketched above is still in need of one extra touch. So far I have said nothing about presuppositions of the use of descriptions. In fact, I will refrain from doing so for the moment, and first work out a semantics that treats descriptions in the Russellian (two valued) way [18].

4 Proper State Semantics

In this section I will spell out the formal details of the standard semantics for dynamic predicate logic with ι assignment. I refer to the standard semantics as *proper state semantics* to emphasise that the definition is couched in terms of proper states only. Given a model $\mathcal{M} = \langle U, I \rangle$, U a universe of individuals and I an interpretation function for the individual constants $C = \{c_0, c_1, \dots\}$ and the first order relation symbols of the language, a proper state for $\mathcal{M}$ is a function in $V \rightarrow U$. I will refer to the set of proper states for $\mathcal{M}$ as $\mathbf{S}_\mathcal{M}$.

A proper 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 $\mathbf{V}_A(t) = A(t)$, if $t \in C$, then $\mathbf{V}_A(t) = I(c)$. If A is a proper state for $\mathcal{M}$, x a variable and d an element of the universe or $\mathcal{M}$, then $A[x := d]$ is the proper state for $\mathcal{M}$ which is just like A except for the possible difference that x is mapped to d.

I define a function $[\![\pi]\!]_\mathcal{M} : \mathbf{S}_\mathcal{M} \rightarrow \mathcal{P}\mathbf{S}_\mathcal{M}$ by recursion. A, B, C are used as metavariables over (proper) states. The function $[\![\pi]\!]_\mathcal{M}$ depends on the model $\mathcal{M}$, but for convenience I 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. $[\![\iota x : \pi]\!](A) = \begin{cases} [\![\pi]\!](A[x := d]) \text{ for the unique } d \in U & \\ \qquad \text{for which } [\![\pi]\!](A[x := d]) \neq \emptyset & \text{if } d \text{ exists,} \\ \emptyset & \text{otherwise.} \end{cases}$

Truth is defined in terms of input-output behaviour: π is true relative to model $\mathcal{M}$ if there are proper 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 proper states A, B for $\mathcal{M}$: if $B \in [\![\pi_1]\!]_{\mathcal{M}}(A)$ then there is a proper 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 and π is performed; otherwise the program fails (in other words, the set of output states is empty). It is not difficult to see that this results in the Russell treatment for definite descriptions. Also, 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.

The clause for dynamic implication should take care of the proper treatment of the definite description *his wife* in example (15).

(15) *If John is married, his wife will be cross with him.*

To get this translated into *DAL*, under the intended reading that the possessive pronoun *his* is anaphorically linked to *John*, we have to decide what to do with proper names. The trouble is that proper names do not observe the same anaphoric constraints as definite or indefinite descriptions.

It seems to me that the anaphoric behaviour of proper names can be accounted for by assuming that they are assimilated to descriptions, but this will only work if descriptions are treated as carrying uniqueness presuppositions. Since we are now exploring a Russellian treatment of definites, this road is not yet open to us, however. Therefore I will sidestep the issue for now, and just treat anaphoric links to names by translating the anaphor as the constant which also translates its name antecedent. An indexing for example (15) using constants as indices for antecedent and anaphors, as in (16), paves the way for this.

(16) *If John^j is married then [his$_j$ wife] will be cross with him$_j$.*

A suitable *DAL* translation for the example now runs as follows:

(17) $(\mathit{married}\ j) \Rightarrow (\iota x : \mathit{wife\text{-}of}\ (x, j);\ \mathit{cross\text{-}with}\ (x, j))$.

Now either the program for *John is married* will not complete successfully, and then the program for the consequent *his wife will be cross with him* will not be executed at all, or it will indeed give precisely one output (this is because the antecedent program is a test). But then the fact that there is an output guarantees that there will be a unique referent for ι assignment in the consequent, so the program for *his wife will be cross with him* will only fail if the person who is in fact John's wife is *not* cross with John.

5 Hoare Statements for Proper State Semantics

The proper state semantics of our representation language will now be supplemented with an axiom system in the style of Hoare (see Apt [1] for an overview of this approach, and Hoare [14] for the original proposal). The axioms and proof rules I propose form a deduction system allowing us to prove statements about *DAL* programs. I will merely present and illustrate the system here. The soundness and completeness of a calculus for a dynamic language which has the present language as a subset, with respect to the proper state semantics, is proved in Van Eijck & De Vries [5].

Our deductive system for dynamic logic contains statements characterizing variable states, plus two kinds of correctness statements, which I call *universal* and *existential* correctness statements. Thus, the system has three kinds of statements:

1. formulae of a language of first order predicate logic with the same sets of variables and predicate letters as the *DAL* language under consideration (call this assertion language L),

2. triples of the form $\{\varphi\}\ \pi\ \{\psi\}$, where φ, ψ are L-formulae, and π is a *DAL*-program,

3. 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 proper 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. If φ is a formula of the assertion language L and x, y are variables then $[y/x]\varphi$ is the result of the substitution of y for all free occurrences of x in φ.

The meanings of $\{\varphi\}\ \pi\ \{\psi\}$ and $\langle\varphi\rangle\ \pi\ \langle\psi\rangle$ are formally specified in terms of the dynamic interpretation function $[\![\cdot]\!]_{\mathcal{M}}$ that was given above plus the satisfaction relation $\mathcal{M} \models \varphi[A]$. The notion of $\mathcal{K}$-validity for correctness statements is defined as follows.

$\mathcal{K}$-validity of Correctness Statements

> If F has the form φ, where φ is a formula of the assertion language, then $\mathcal{K} \models F$ if $\mathcal{M} \models F[A]$ for all models $\mathcal{M} \in \mathcal{K}$ and all states A for $\mathcal{M}$.
>
> If F has the form $\{\varphi\}\ \pi\ \{\psi\}$, then $\mathcal{K} \models F$ if the following holds. For all models $\mathcal{M} \in \mathcal{K}$ and for all states A for $\mathcal{M}$, if $\mathcal{M} \models \varphi[A]$ then for all states $B \in [\![\pi]\!]_{\mathcal{M}}(A)$ it is the case that $\mathcal{M} \models \psi[B]$.
>
> If F has the form $\langle\varphi\rangle\ \pi\ \langle\psi\rangle$, then $\mathcal{K} \models F$ if the following holds. For all models $\mathcal{M} \in \mathcal{K}$ and for all states A for $\mathcal{M}$, if $\mathcal{M} \models \varphi[A]$ then there is at least one state $B \in [\![\pi]\!]_{\mathcal{M}}(A)$ with $\mathcal{M} \models \psi[B]$.

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.

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.

6 A Calculus for Proper State Semantics

This section gives a proof system for dynamic interpretation with proper state semantics.

The atomic predicates of *DAL* act as tests. The following test axioms account for their behaviour.

Test Axioms

$$\frac{}{\langle\bot\rangle\ \bot\ \langle\bot\rangle}.$$

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

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

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

$$\frac{}{\langle t_1=t_2\wedge\varphi\rangle\ t_1=t_2\ \langle\varphi\rangle}.$$

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

The axioms for the program $\bot$ express that $\bot$ always fails. The existential axioms for atomic predicates and identities give the preconditions which guarantee successful termination, with the postconditions guaranteed by the test. The universal axioms for atomic predicates and identities give the preconditions under which, if the program succeeds, all output states will satisfy φ. The reader is invited to convince her- or himself that these axioms are sound.

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 existential and universal correctness.

Consequence Rules

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

$$\frac{\varphi \to \psi \qquad \{\psi\}\ \pi\ \{\chi\} \qquad \chi \to \xi}{\{\varphi\}\ \pi\ \{\xi\}.}$$

It is instructive to reflect on these rules in order to convince oneself that they are indeed sound.

The remaining rules specify the meanings of complex programs in the axiomatic framework.

Rules of Composition

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

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

Again, the proof of the soundness of these rules is left to the reader.

Rules of Negation

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

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

Recall that negation is a test which gives the input states as its only output state just in case the unnegated program would fail for the input. To see that the first rule is sound, notice that what is says is: if a program π fails for all inputs satisfying φ, then the program $\neg\pi$ will succeed for all inputs satisfying φ. If moreover these inputs satisfy ψ as well, then there is a guarantee that at least one output state (in fact, the one and only output there is) will satisfy ψ. As for the second rule, it says that if a program π succeeds for all φ states, then if you give it a state satisfying $\varphi \vee \psi$, it will only accept that state if it is a $\neg\varphi$ state. In other words, all output states will satisfy ψ.

Rules of Implication

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

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

The reader is invited to check the soundness of these rules for her- or himself.

Rules of η Assignment

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

$$\frac{\{\varphi\}\ \pi\ \{\psi\}}{\{\forall x\varphi\}\ \eta x : \pi\ \{\psi\}.}$$

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

To convince ourselves that the rules for η assignment are sound with respect to the semantics, let us consider the rules one by one. For the first rule, suppose you know that for all φ inputs the program π will produce at least one ψ output. Now give the program $\eta x : \pi$ an input satisfying $\exists x\varphi$. Recall that what the program does is: first assign a random value to x, and then perform π on the resulting state. We know that the input of $\eta x : \pi$ satisfies $\exists x\varphi$, so there must be some d in the universe of the model under consideration that satisfies φ. One of the random choices for x that will be considered is the assignment of this very d to x. This results in a state $A[x := d]$ which will satisfy φ. Apply the premiss to see that at least one output of π will satisfy ψ.

Consider the second rule. Suppose you know that when the program π acts on an input state satisfying φ, then every output state will satisfy ψ. This is what the premiss says. Now take an input state which satisfies the stronger condition $\forall x\varphi$ and run the program $\eta x : \pi$. What $\eta x : \pi$ does is: first assign an arbitrary value to x, and then perform π on the resulting state. Because of the fact that $\forall x\varphi$ holds, any choice for the value of x will result in a state where φ holds. Then apply the premiss to see that any output state of $\eta x : \pi$ will satisfy ψ.

We will study the rules for ι assignment a bit closer. In discussing these rules 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. I will first give a version which almost works, and then remedy it later in the light of further discussion. The rules will contextually eliminate the ι operator.

Rules of ι Assignment (first attempt)

$$\frac{\langle\varphi\rangle\ \pi\ \langle\top\rangle \qquad \langle\psi\rangle\ \pi\ \langle\chi\rangle}{\langle\exists!x\varphi \wedge \exists x\psi\rangle\ \iota x : \pi\ \langle\chi\rangle.}$$

$$\frac{\langle\varphi\rangle\ \pi\ \langle\top\rangle \qquad \{\psi\}\ \pi\ \{\chi\}}{\{\exists!x\varphi \rightarrow \forall x\psi\}\ \iota x : \pi\ \{\chi\}.}$$

The premisses of the first rule state that π succeeds for any input which satisfies φ, and moreover, that if it is run on an input satisfying ψ then at least one output state will satisfy χ. For ease of discussion, assume for a moment that π is the atomic test program $king\ (x)$. Then $\langle king\ (x)\rangle\ king\ (x)\ \langle\top\rangle$ will certainly be true. Suppose we are in a state with a unique king, i.e., a state in which $\exists!x\,king\ (x)$ holds. Then the rule allows us to conclude that the program $\iota x : king\ (x)$ will succeed in this state.

The first premiss of the second rule states that π succeeds on an input satisfying φ, while the second premiss says that if an input of π satisfies ψ, then then all outputs will satisfy χ. Again assume that π is the program $king\ (x)$, which tests whether x is a king. Then the premisses $\langle king\ (x)\rangle\ king\ (x)\ \langle\top\rangle$ and $\{king\ (x) \rightarrow monarch\ (x)\}\ king\ (x)\ \{monarch\ (x)\}$ both satisfy the test axioms. The conclusion that can be drawn from this:

(18) $\{\exists!x\,king\ (x) \rightarrow \forall x(king\ (x) \rightarrow monarch\ (x))\}$
$\iota x : king\ (x)\ \{monarch\ (x)\}.$

If in state A it holds that if there is a unique king then every king is a monarch, then all output states of the king-test will satisfy $monarch\ (x)$. This is of course correct for the present Russellian analysis.

Does this means that all is well with the rules? Not at all, for one should bear in mind that in this example $\langle king\ (x)\rangle\ king\ (x)\ \langle\top\rangle$ gives the weakest precondition of success. But the rules as they are stated are more general than that. $\langle king\ (x) \wedge beggar\ (x)\rangle\ king\ (x)\ \langle\top\rangle$ is also a true correctness statement. It can be got from $\langle king\ (x)\rangle\ king\ (x)\ \langle\top\rangle$ by existential precondition weakening, using the existential consequence rule. Using this as premiss, the first rule would allow us to conclude that (19).

(19) $\langle\exists!x(king\ (x) \wedge beggar\ (x))\rangle\ \iota x : king\ (x)\ \langle\top\rangle.$

To see that this is wrong, imagine a situation where there are two kings, with one of them, incongruous as it may seem, being also a beggar. Then the test $\iota x : king\ (x)$ will fail because there is no unique king, thus contradicting (19).

To see that the second rule is also wrong, take as first premiss again the true correctness statement (20).

(20) $\langle king\ (x) \wedge beggar\ (x)\rangle\ king\ (x)\ \langle\top\rangle.$

Take as second premiss the following correctness statement.

(21) $\{whatever\ (x)\}\ king\ (x)\ \{whatever\ (x)\}.$

Note that (21) is trivially true. To see that it is also derivable in the calculus note that it can be got from one of the universal test axioms, plus presupposition weakening using the universal consequence rule. The second rule of ι assignment would license drawing the following conclusion from (20) and (21).

(22) $\{\exists!x(\mathit{king}\ (x) \wedge \mathit{beggar}\ (x)) \rightarrow \forall x\ \mathit{whatever}\ (x)\}$
$\iota x : \mathit{king}\ (x)\ \{\mathit{whatever}\ (x)\}.$

Consider a situation where there is precisely one king, but this king is a regular one, not a beggar. Then the program $\iota x : \mathit{king}\ (x)$ will succeed in this situation. Because the unique king is not a beggar, in the situation under consideration there is no unique individual combining the properties of being a king and a beggar. Therefore the statement $\exists!x(\mathit{king}\ (x) \wedge \mathit{beggar}\ (x)) \rightarrow \forall x\ \mathit{whatever}\ (x)$ is trivially true. Still, the conclusion drawn in (22) that the king can have any property whatsoever is of course not warranted.

The problem with the rules as they stand is that in both cases the first premiss does not specify the *weakest* precondition for a given outcome of π. To enforce that φ is the weakest precondition for success, one has to add the premiss that $\neg\varphi$ guarantees failure. This leads to the following version of the rules.

Rules of ι Assignment

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

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

It is left to the reader to check that these rules are indeed sound with respect to the proper state semantics of ι assignment.

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: $\mathcal{K} \vdash F$. For proofs of the soundness and completeness of this calculus with respect to the proper state semantics the reader is referred to Van Eijck & De Vries [5].

It may be enlightening to work out some examples of derivations of static meanings using the calculus. I will concentrate on very simple sentences.

(23) *The King of France is bald.*

Example (23) can be translated into *DAL* as $\iota x : Kx; Bx$. Here is a derivation of the weakest precondition for success of this program, i.e., a derivation of its static truth conditions.

$$\dfrac{\dfrac{\langle Kx\rangle\ Kx\ \langle\top\rangle \qquad \{\neg Kx\}\ Kx\ \{\bot\} \qquad \langle Kx \wedge Bx\rangle\ Kx\ \langle Bx\rangle}{\langle \exists!x Kx \wedge \exists x(Kx \wedge Bx)\rangle\ \iota x : Kx\ \langle Bx\rangle} \qquad \langle Bx\rangle\ Bx\ \langle\top\rangle}{\langle \exists!x Kx \wedge \exists x(Kx \wedge Bx)\rangle\ \iota x : Kx; Bx\ \langle\top\rangle.}$$

Strictly speaking one should check that the derived precondition is indeed the weakest precondition that works. In fact this follows from a general property of the calculus:

Fact As long as the two consequence rules are not used, all preconditions derived in the calculus are **weakest** preconditions.

Here is a derivation of the static falsity conditions of the Russell example.

$$\dfrac{\dfrac{\langle Kx\rangle\ Kx\ \langle\top\rangle \qquad \{\neg Kx\}\ Kx\ \{\bot\} \qquad \{Kx \rightarrow \neg Bx\}\ Kx\ \{\neg Bx\}}{\{\exists!x Kx \rightarrow \forall x(Kx \rightarrow \neg Bx)\}\ \iota x : Kx\ \{\neg Bx\}} \qquad \{\neg Bx\}\ Bx\ \{\bot\}}{\{\exists!x Kx \rightarrow \forall x(Kx \rightarrow \neg Bx)\}\ \iota x : Kx; Bx\ \{\bot\}.}$$

Again, the consequence rules were not used, so we have derived the weakest universal precondition of the program with respect to $\bot$. And in fact $\exists!x Kx \rightarrow \forall x(Kx \rightarrow \neg Bx)$ is the negation of $\exists!x Kx \wedge \exists x(Kx \wedge Bx)$, the weakest existential precondition of the program with respect to $\top$.

Now consider example (24).

(24) *The King of France is not bald.*

Following Russell, I will suppose that there are two readings, with different scope for the negation operator. It is left to the reader to verify in the calculus that the weakest existential precondition with respect to $\top$ of the program $\iota x : Kx; \neg Bx$ is given by: $\exists!x Kx \wedge \exists x(Kx \wedge \neg Bx)$. The weakest existential precondition with respect to $\top$ for the other reading is given by the following derivation.

$$\dfrac{\dfrac{\dfrac{\langle Kx\rangle\ Kx\ \langle\top\rangle \quad \{Kx\}\ Kx\ \{\bot\} \quad \{Kx \rightarrow \neg Bx\}\ Kx\ \{\neg Bx\}}{\{\exists!x Kx \rightarrow \forall x(Kx \rightarrow \neg Bx)\}\ \iota x : Kx\ \{\neg Bx\}} \qquad \{\neg Bx\}\ Bx\ \{\bot\}}{\{\exists!x Kx \rightarrow \forall x(Kx \rightarrow \neg Bx)\}\ \iota x : Kx; Bx\ \{\bot\}}}{\langle \exists!x Kx \rightarrow \forall x(Kx \rightarrow \neg Bx)\rangle\ \neg(\iota x : Kx; Bx)\ \langle\top\rangle.}$$

This is indeed the result one would expect.

Finally, let us look at an example where the definite is part of the consequent of an implication, with the antecedent setting up the requirements for uniqueness of reference.

(25) *If a woman is married, her husband will look after her.*

I will derive the static truth conditions in two stages. The existential rule for dynamic implication tells us that the formula we are looking for is the weakest φ such that we can find a ψ for which the following holds:

$$\frac{\begin{array}{cc}\vdots & \vdots \\ \{\varphi\}\ \eta x : Wx; Mx\ \{\psi\} & \langle\psi\rangle\ \iota y : Hyx; Lyx\ \langle\top\rangle\end{array}}{\langle\varphi\rangle\ (\eta x : Wx; Mx) \Rightarrow (\iota y : Hyx; Lyx)\ \langle\top\rangle.}$$

For the first subderivation we find:

$$\frac{\dfrac{\{Wx \rightarrow (Mx \rightarrow \psi)\}\ Wx\ \{Mx \rightarrow \psi\}}{\{\forall x(Wx \rightarrow (Mx \rightarrow \psi))\}\ \eta x : Wx\ \{Mx \rightarrow \psi\}\ \{Mx \rightarrow \psi\}\ Mx\ \{\psi\}}}{\{\forall x(Wx \rightarrow (Mx \rightarrow \psi))\}\ \eta x : Wx; Mx\ \{\psi\}.}$$

The second subderivation computes ψ:

$$\frac{\dfrac{\langle Hyx\rangle\ Hyx\ \langle\top\rangle \quad \{\neg Hyx\}\ Hyx\ \{\bot\} \quad \langle Hyx \wedge Lyx\rangle\ Hyx\ \langle Lyx\rangle}{\langle \exists! y Hyx \wedge \exists y(Hyx \wedge Lyx)\rangle\ \ \iota y : Hyx\ \langle Lyx\rangle} \qquad \langle Lyx\rangle\ Lyx\ \langle\top\rangle}{\langle \exists! y Hyx \wedge \exists y(Hyx \wedge Lyx)\rangle\ \iota y : Hyx; Lyx\ \langle\top\rangle}$$

Combining these two results we get that formula (26) expresses the static truth conditions of the example sentence.

(26) $\forall x(Wx \rightarrow (Mx \rightarrow (\exists! y Hyx \wedge \exists y(Hyx \wedge Lyx))))$.

This is again what one would expect under the present regime. Deriving the weakest universal precondition φ for which the program fails we find the negation of (26). Again, the uniqueness presupposition of the definite is swallowed up by the truth conditions, so to speak: no distinction is made between falsity and failure of presupposition. In the next section I will propose a means of separating out the cases of presupposition failure.

7 Error State Semantics: Informal Discussion

Nothing we have said so far makes clear how the dynamic treatment of definite descriptions is meant to deal with their uniqueness presuppositions. Consider situation (27) and assertion (28) about this situation.

(27) $a\ \hat{a}\ \hat{b}\hat{}\,b\ C$.

(28) *The character with the hat is not a capital.*

On a Russellian analysis [18], example (28) is false with respect to (27) if the description is taken to have scope over the negation operator, as in *DAL* translation (29).

(29) $\iota v_1 : (character\ v_1;\ \iota v_2 : (hat\ v_2;\ with(v_1, v_2)));\ \neg capital\ v_1$.

This is not quite what we want if we intend to follow Frege [7, 8], Hilbert & Bernays [13] and Strawson [19] rather than Russell: we intend to preserve the distinction between falsity and failure of presupposition. In the Frege view, which is shared by Hilbert & Bernays and by Strawson, (29), rather than being false, suffers from presupposition failure.

Our way to work out this distinction will be as follows. I assume that *DAL* programs can execute in either of three ways when acting on a given input state:

1. they report success by producing at least one proper output state,
2. they report failure by not producing an output state at all,
3. they report error by producing a special error state as only output.

The first case indicates truth of the information contained in the program in the context of the current input state. The second case indicates falsity of the information contained in the program in the context of the current input state. The third case, finally, corresponds to failure of presupposition in the context of the current input state. In formulating dynamic meaning, one must specify the (mutually exclusive) conditions for the three cases.

Interestingly, in many cases uniqueness presuppositions for definite descriptions are carried along in the discourse. Consider example (25) again, with its translation (30).

(30) $(\eta x : Wx;\ Mx) \Rightarrow (\iota y : Hyx;\ Lyx)$.

The presupposition of the consequent of (30), namely that $\iota y : \mathit{husband\text{-}of}(y, x)$ has a unique referent relative to an assignment of a married woman to x, is nicely taken care of by the antecedent of the implication. Strictly speaking, one also has to assume a meaning postulate expressing that married men do have one and only one wife. But then an account that handles presuppositions should ensure that the presupposition of the consequent program will be cancelled in the larger context of the dynamic implication.

Given the distinction between program failure and program error abortion, one can also easily implement the view, found in Hilbert & Bernays [13], that indefinites are also loaded with a presupposition. This view entails that (31) has no truth value relative to situation (27), because the situation does not contain a capital with a hat.

(31) *A capital with a hat precedes a capital.*

It is not clear to me whether this is intuitively acceptable, but possibly my intuitions are blurred by the fact that I find it hard to forget about other situations which do contain characters with hats. I take it, however, that example (32), makes clear that in general a presuppositional treatment of indefinites *à la* Hilbert & Bernays cannot be correct for natural language analysis.

(32) *There is a capital with a hat.*

The most reasonable assumption seems to be that (32) is simply *false* with respect to (27). I will therefore not treat indefinites as presupposition-loaded.

8 Error State Semantics: Formal Definitions

Assume a model $\mathcal{M} = \langle U, I \rangle$ as before. I use $\mathbf{S}_\mathcal{M}$ for the set of proper states for $\mathcal{M}$, i.e., for the set of functions $U \rightharpoonup V$.

As before, a proper state A for $\mathcal{M} = \langle U, I \rangle$ determines a valuation $\mathbf{V}_A$ for the terms of the language as follows:

1. If $t \in C$ then $\mathbf{V}_A(t) = I(t)$.
2. If $t \in V$ then $\mathbf{V}_A(t) = A(t)$.

The symbol ϵ will be used to designate a special error state. If $\mathbf{S}_\mathcal{M}$ is the set of proper states for $\mathcal{M} = \langle U, I \rangle$, then $\mathbf{S}_\mathcal{M} \cup \{\epsilon\}$ is the set of *states* for $\mathcal{M}$. There is no connection between the error state ϵ and the epsilon terms from the introduction.

We define a function $[\![\pi]\!]_\mathcal{M} : \mathbf{S}_\mathcal{M} \to \mathcal{P}(\mathbf{S}_\mathcal{M} \cup \{\epsilon\})$ by recursion. A is used as a metavariable over proper states, and B as a metavariable over states. The function $[\![\pi]\!]_\mathcal{M}$ depends on the model $\mathcal{M}$, but for convenience I will again write $[\![\pi]\!]$ rather than $[\![\pi]\!]_\mathcal{M}$.

The meaning conditions are rather involved because of the need to specify error abortion behaviour. The reader should keep in mind that the only case where error arises is the case where a ι assignment aborts because no unique referent can be found in the context of the current input state. The mention of error states in the conditions of the other program constructs is meant to take care of error propagation.

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. $[\![(\pi_1;\, \pi_2)]\!](A) = \begin{cases} \{\epsilon\} & \text{if } [\![\pi_1]\!](A) = \{\epsilon\}, \\ \bigcup\{[\![\pi_2]\!](B) \mid B \in [\![\pi_1]\!](A), B \neq \epsilon\} & \text{otherwise.} \end{cases}$
5. $[\![(\pi_1 \Rightarrow \pi_2)]\!](A) = \begin{cases} \{\epsilon\} & \text{if } \epsilon \in [\![\pi_1]\!](A), \\ & \text{or for some proper state } B \in [\![\pi_1]\!](A) \\ & \text{it holds that } [\![\pi_2]\!](B) = \{\epsilon\}, \\ \{A\} & \text{if for all } B \in [\![\pi_1]\!](A) \\ & \text{it holds that } B \neq \epsilon \text{ and } [\![\pi_2]\!](B) \not\subseteq \{\epsilon\}, \\ \emptyset & \text{otherwise.} \end{cases}$

6. $[\![\neg\pi]\!](A) = \begin{cases} \{\epsilon\} & \text{if } [\![\pi]\!](A) = \{\epsilon\}, \\ \{A\} & \text{if } [\![\pi]\!](A) = \emptyset, \\ \emptyset & \text{otherwise.} \end{cases}$

7. $[\![\eta x : \pi]\!](A) = \bigcup\{[\![\pi]\!](A[x := d]) \mid d \in U\}$.

8. $[\![\iota x : \pi]\!](A) = \begin{cases} [\![\pi]\!](A[x := d]) \text{ for the unique } d \in U & \\ \quad \text{with } [\![\pi]\!](A[x := d]) \not\subseteq \{\epsilon\} & \text{if } d \text{ exists,} \\ \{\epsilon\} & \text{otherwise.} \end{cases}$

Truth is defined in terms of input-output behaviour: π is true relative to model $\mathcal{M}$ if there are proper states A, B for $\mathcal{M}$ such that $B \in [\![\pi]\!](A)$. As there is more to meaning than just truth I must say a bit more. Because of the possibility of presupposition failure it makes sense to define the following three sets for any program π and model $\mathcal{M}$.

- $1_{\pi,\mathcal{M}} \stackrel{\text{def}}{=} \{A \in \mathbf{S}_{\mathcal{M}} \mid [\![\pi]\!]_{\mathcal{M}}(A) \cap \mathbf{S}_{\mathcal{M}} \neq \emptyset\}$.
- $0_{\pi,\mathcal{M}} \stackrel{\text{def}}{=} \{A \in \mathbf{S}_{\mathcal{M}} \mid [\![\pi]\!]_{\mathcal{M}}(A) = \emptyset\}$.
- $*_{\pi,\mathcal{M}} \stackrel{\text{def}}{=} \{A \in \mathbf{S}_{\mathcal{M}} \mid [\![\pi]\!]_{\mathcal{M}}(A) = \{\epsilon\}\}$.

Note that it follows immediately from these definitions that we have the following:

- $1_{\pi,\mathcal{M}} = \mathbf{S}_{\mathcal{M}} - (0_{\pi,\mathcal{M}} \cup *_{\pi,\mathcal{M}})$.
- $0_{\pi,\mathcal{M}} = \mathbf{S}_{\mathcal{M}} - (1_{\pi,\mathcal{M}} \cup *_{\pi,\mathcal{M}})$.
- $*_{\pi,\mathcal{M}} = \mathbf{S}_{\mathcal{M}} - (1_{\pi,\mathcal{M}} \cup 0_{\pi,\mathcal{M}})$.

Two programs π, π' will be called truth equivalent in case for all models $\mathcal{M}$ it holds that $1_{\pi,\mathcal{M}} = 1_{\pi',\mathcal{M}}$, falsity equivalent in case for all models $\mathcal{M}$ it holds that $0_{\pi,\mathcal{M}} = 0_{\pi',\mathcal{M}}$, and error equivalent in case for all models $\mathcal{M}$ it holds that $*_{\pi,\mathcal{M}} = *_{\pi',\mathcal{M}}$. Because of the possibility of error, truth equivalent programs need not be falsity equivalent, for there may be a proper state for which one of the programs gives error and the other falsity. Likewise, falsity equivalent programs need not be truth equivalent, for there may be a proper state for which one of the programs gives error and the other truth.

Just for the record, the dynamic consequence notion does not change. Dynamic consequence is still defined as follows: $\pi_1 \models \pi_2$ if for every model $\mathcal{M}$ and for all proper states A, B for $\mathcal{M}$: if $B \in [\![\pi_1]\!]_{\mathcal{M}}(A)$ then there is a proper state C with $C \in [\![\pi_2]\!]_{\mathcal{M}}(B)$. Note that the fact that ϵ may be among the outputs for π_1 does not matter; the only thing that matters is that program π_2 succeeds for all *proper* outputs of π_1. This reflects the fact that dynamic consequence preserves presupposition, so to speak, in the sense that the program π_2 is processed on the assumption that the presuppositions of π_1 are fulfilled. Note however that,

although the *definition* of the dynamic consequence has not changed, the relation itself has, for the relation depends on the semantic interpretation function of error state semantics.

The interpretation conditions for ι assignment make clear how the uniqueness presupposition are now handled dynamically. The statement $\iota x : \pi$ consists of a test followed by a deterministic action in case the test succeeds, followed again by the program π. First it is checked whether there is a unique π; if so, this individual is assigned to x and π is executed in the result state; otherwise, the only possible output state is the error state. In other words, if the uniqueness presupposition is not met, the program will not fail, as it did under the regime of proper state semantics, but instead it will abort with error. It is not difficult to see that this implements the Frege, Hilbert & Bernays, Strawson view of the behaviour of definite descriptions.

Also, we see that the two programs $\iota x : (\pi_1); \pi_2$ and $\iota x : (\pi_1; \pi_2)$ again are not equivalent. More precisely, they are neither truth equivalent nor falsity 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. The programs are not truth equivalent because $\iota x : (\pi_1; \pi_2)$ may succeed while $\iota x : (\pi_1); \pi_2$ aborts with error. This happens in case there is a unique object satisfying $\pi_1; \pi_2$, but more than one d satisfies π_1. They are not falsity equivalent because $\iota x : (\pi_1; \pi_2)$ may abort while $\iota x : (\pi_1); \pi_2$ fails. This happens in case there is no unique object satisfying $\pi_1; \pi_2$, but there is a unique object satisfying π_1, and this object does not satisfy π_2.

9 A Calculus for Error State Semantics

Because of the presence of the error state ϵ, the semantics for *DAL* that was presented in the previous section is even more awkward to deal with directly than the proper state semantics presented earlier. An obvious next move is the development of an extended Hoare calculus that is sound and complete for this interpretation.

Such a Hoare style deduction system has been developed and is presented in Van Eijck [4], but here I will limit myself to a brief discussion of the new rules for ι assignment, to get the flavour of the approach across.

The set of Hoare correctness statements is extended with two new kinds of assertions, namely triples of the forms $\{\varphi\}\ \pi\ \epsilon!$ and $\{\varphi\}\ \pi\ \epsilon?$. The Hoare correctness statements are now to be interpreted as follows.

$\mathcal{K}$-validity of Correctness Statements

> If F has the form φ, where φ is a formula of the assertion language, then $\mathcal{K} \models F$ if $\mathcal{M} \models F[A]$ for all models $\mathcal{M} \in \mathcal{K}$ and all proper states A for $\mathcal{M}$.

If F has the form $\{\varphi\}\ \pi\ \{\psi\}$, then $\mathcal{K} \models F$ if the following holds. For all models $\mathcal{M} \in \mathcal{K}$ and all proper states A for $\mathcal{M}$, if $\mathcal{M} \models \varphi[A]$ then $\epsilon \notin [\![\pi]\!]_{\mathcal{M}}(A)$ and for all (proper) states $B \in [\![\pi]\!]_{\mathcal{M}}(A)$ it is the case that $\mathcal{M} \models \psi[B]$.

If F has the form $\langle\varphi\rangle\ \pi\ \langle\psi\rangle$, then $\mathcal{K} \models F$ if the following holds. For all models $\mathcal{M} \in \mathcal{K}$ and for all proper states A for $\mathcal{M}$, if $\mathcal{M} \models \varphi[A]$ then there is at least one proper state $B \in [\![\pi]\!]_{\mathcal{M}}(A)$ with $\mathcal{M} \models \psi[B]$.

If F has the form $\{\varphi\}\ \pi\ \epsilon!$, then $\mathcal{K} \models F$ if the following holds. For all models $\mathcal{M} \in \mathcal{K}$ and for all proper states A for $\mathcal{M}$, if $\mathcal{M} \models \varphi[A]$ then $[\![\pi]\!]_{\mathcal{M}}(A) = \{\epsilon\}$.

If F has the form $\{\varphi\}\ \pi\ \epsilon?$, then $\mathcal{K} \models F$ if the following holds. For all models $\mathcal{M} \in \mathcal{K}$ and for all proper states A for $\mathcal{M}$, if $\mathcal{M} \models \varphi[A]$ then $[\![\pi]\!]_{\mathcal{M}}(A) \subseteq \{\epsilon\}$.

In the new calculus we have three rules for ι assignment instead of two. The first rule allows us to calculate truth conditions, the second rule is for calculating falsity conditions, and the third rule describes the conditions of presupposition failure. The new rules for ι assignment look as follows.

Rules of ι Assignment for Error State Semantics

$$\frac{\langle\varphi\rangle\ \pi\ \langle\top\rangle \qquad \{\neg\varphi\}\ \pi\ \epsilon? \qquad \langle\psi\rangle\ \pi\ \langle\chi\rangle}{\langle\exists!x\varphi \wedge \exists x\psi\rangle\ \iota x : \pi\ \langle\chi\rangle.}$$

$$\frac{\langle\varphi\rangle\ \pi\ \langle\top\rangle \qquad \{\neg\varphi\}\ \pi\ \epsilon? \qquad \{\psi\}\ \pi\ \{\chi\}}{\{\exists!x\varphi \wedge \forall x\psi\}\ \iota x : \pi\ \{\chi\}.}$$

$$\frac{\langle\varphi\rangle\ \pi\ \langle\top\rangle \qquad \{\neg\varphi\}\ \pi\ \epsilon?}{\{\neg\exists!x\varphi\}\ \iota x : \pi\ \epsilon!}$$

To see what these rules do, first note that the combination of the premisses $\langle\varphi\rangle\ \pi\ \langle\top\rangle$ and $\{\neg\varphi\}\ \pi\ \epsilon?$ ensures that φ is the weakest precondition for success of π. The first premiss gives that the program will succeed in all φ states, while the second yields that in all $\neg\varphi$ states it will either abort with error or fail.

To see that the first rule is sound, assume the premisses are true, and consider a state A where $\exists!x\varphi \wedge \exists x\psi$ is true. In other words, there is an object d which is the unique φ and moreover, there is an object d' which satisfies ψ. We know that φ is the weakest precondition for success of π, and that ψ is a precondition for success of π, so $\psi \rightarrow \varphi$, and therefore $\forall x(\psi \rightarrow \varphi)$, will hold in state A. It follows that $d' = d$. By the rule for ι assignment, this object d is assigned to x, and the program π is then made to act on input $A[x := d]$. Because d satisfies ψ we know that $A[x := d]$ is a ψ state, so the third premiss applies. This guarantees that there is an output state which satisfies χ. For the second

rule the reasoning is similar: the third universal premiss now guarantees that all output states of $[\![\pi]\!](A[x := d])$ will satisfy χ. The third rule states that the program $\iota x : \pi$ will abort in precisely the cases where the uniqueness condition for $\varphi(x)$ is violated. This is of course correct.

One last thing to call attention to is the switch from $\{\neg\varphi\}\ \pi\ \{\bot\}$ to $\{\neg\varphi\}\ \pi\ \epsilon?$ in the premisses. The reason for this is that we want to be able to treat definites which have other definites inside them, such as *the man who mistook his wife for a hat*. The program which translates *man who mistook his wife for a hat* may abort with error because no unique referent for *his wife* can be found. The premiss $\{\neg\varphi\}\ \pi\ \epsilon?$ covers this case, while under the error state semantics the premiss $\{\neg\varphi\}\ \pi\ \{\bot\}$ does not.

This discussion shows, by the way, that there should be a rule in the calculus which relates the notions $\epsilon?$ and $\epsilon!$. Such a rule can be formulated as follows.

Failure or Error Rule

$$\frac{\{\varphi\}\ \pi\ \{\bot\} \qquad \{\psi\}\ \pi\ \epsilon!}{\{\varphi \vee \psi\}\ \pi\ \epsilon?}$$

It is clear that this describes the relation between $\epsilon!$ and $\epsilon?$ correctly, so the rule is obviously sound. It is also clear that statement (33) should be derivable in a complete calculus, as it is obviously true:

(33) $\{\bot\}\ \pi\ \epsilon!$

From the failure or error rule and (33) we get the following derived rule:

Derived Failure or Error Rule

$$\frac{\{\varphi\}\ \pi\ \{\bot\}}{\{\varphi\}\ \pi\ \epsilon?}$$

Although we have not given a full calculus, we can already demonstrate the rules we have introduced so far on a very simple case such as Russell's King of France example, repeated here for convenience.

(34) *The king of France is bald.*

Here is the derivation of its static truth conditions:

$$\dfrac{\dfrac{\langle Kx\rangle\ Kx\ \langle\top\rangle \qquad \dfrac{\{\neg Kx\}\ Kx\ \{\bot\}}{\{\neg Kx\}\ Kx\ \epsilon?} \qquad \langle Kx \wedge Bx\rangle\ Kx\ \langle Bx\rangle}{\langle \exists!x Kx \wedge \exists x(Kx \wedge Bx)\rangle\ \iota x : Kx\ \langle Bx\rangle} \qquad \langle Bx\rangle\ Bx\ \langle\top\rangle}{\langle \exists!x Kx \wedge \exists x(Kx \wedge Bx)\rangle\ \iota x : Kx; Bx\ \langle\top\rangle}$$

The rules that have been used are the derived failure or error rule, the existential ι rule, and the existential composition rule.

Here is a derivation of the static falsity conditions of the Russell example. This derivation uses the derived failure or error rule, the universal ι rule, and the universal composition rule.

$$\dfrac{\dfrac{\dfrac{}{\langle Kx\rangle\ Kx\ \langle\top\rangle} \qquad \dfrac{\dfrac{}{\{\neg Kx\}\ Kx\ \{\bot\}}}{\{\neg Kx\}\ Kx\ \epsilon?}}{\{\exists!xKx \wedge \forall x(Kx \rightarrow \neg Bx)\}\ \iota x : Kx\ \{\neg Bx\}} \qquad \dfrac{}{\{Kx \rightarrow \neg Bx\}\ Kx\ \{\neg Bx\}} \qquad \dfrac{}{\{\neg Bx\}\ Bx\ \{\bot\}}}{\{\exists!xKx \wedge \forall x(Kx \rightarrow \neg Bx)\}\ \iota x : Kx; Bx\ \{\bot\}}$$

The derivation of the error conditions for this example also uses the following error rule for sequential composition (in the full calculus, this is a derived rule):

Derived Error Rule for Sequential Composition

$$\dfrac{\{\varphi\}\ \pi_1\ \ \epsilon!}{\{\varphi\}\ \pi_1; \pi_2\ \epsilon!}$$

Here is the derivation of the error conditions for (34).

$$\dfrac{\dfrac{\dfrac{}{\langle Kx\rangle\ Kx\ \langle\top\rangle} \qquad \dfrac{\dfrac{}{\{\neg Kx\}\ Kx\ \{\bot\}}}{\{\neg Kx\}\ Kx\ \epsilon?}}{\{\neg\exists!xKx\}\ \iota x : Kx\ \ \epsilon!}}{\{\neg\exists!xKx\}\ \iota x : Kx; Bx\ \ \epsilon!}$$

Hopefully the above hints have made clear how one should go about developing a Hoare style calculus for error state semantics. The full development of the system, with a proof that it is sound and complete for the envisaged semantics, and a demonstration of how it can be used as a calculus of presupposition failure is the topic of another paper [4].

10 Conclusion

In this paper I have explored two varieties of dynamic semantics, proper state semantics and error state semantics, while focussing on the constructs of η and ι assignment for the treatment of indefinite and definite descriptions. It turned out that an error state semantics is well suited to take the uniqueness presuppositions of the use of definite descriptions into account.

If we adopt a proper state semantics, then both η assignment and ι assignment ccan in principle be decomposed. The process of η assignment is decomposable in random assignment with subsequent testing, while that for ι assignment is only slightly more complicated. In fact, we can read $\iota x : \pi$ as an abbreviation of $\eta x : \pi; \neg(\eta y : [y/x]\pi; y \neq x)$. Under an error state semantics η assignment is still decomposable, but a decomposition of ι assignment has become impossible, because of the way in which the uniqueness presuppositions

are handled. It is interesting to speculate about the addition of special error handling constructions to the representation language. In this connection, an obvious question one might ask is: what is the minimal enhancement of the representation language that would make $\iota x : \pi$ decomposable again? But irrespective of the answer to this question, the nice thing about the constructs for η and ι assignment is that they allow us to remain faithful to linguistic form.

For proper state semantics I have spelled out a Hoare calculus, while for error state semantics I have hinted at one. In fact, the second calculus, which I have not given in full detail, is much more versatile than the first. A full calculus for error state semantics is given in Van Eijck [4]. This paper also demonstrates how the new calculus is used for calculating static truth conditions, static falsity conditions and static error conditions. The latter describe the class of model/state pairs where the presuppositions of a program are not fulfilled.

Acknowledgement

This paper has benefitted from helpful comments by Krzysztof Apt, Reinhard Muskens, Martin Stokhof and Fer-Jan de Vries.

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.

[4] J. van Eijck, 'Presupposition Failure — A Comedy of Errors', manuscript, CWI, Amsterdam, 1991.

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

[6] G. Evans, 'Pronouns', in: *Linguistic Inquiry*, 11, 1980 337-362.

[7] G. Frege, 'Funktion und Begriff', 1891, translated as 'Function and Concept' in P. Geach & M. Black (eds.), *Translations from the Philosophical Writings of Gottlob Frege*, Basil Blackwell, Oxford 1952.

[8] G. Frege, 'Ueber Sinn und Bedeutung', 1892, translated as 'On Sense and Reference' in P. Geach & M. Black (eds.), *Translations from the Philosophical Writings of Gottlob Frege*, Basil Blackwell, Oxford 1952.

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

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

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

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

[13] D. Hilbert & P. Bernays, *Grundlagen der Mathematik*, Göttingen 1939 (Second edition: Berlin etc. 1970).

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

[15] 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.

[16] A.C. Leisenring, *Mathematical Logic and Hilbert's ϵ-symbol*, Gordon and Breach Science Publishers, New York, 1969.

[17] H. Reichenbach, *Elements of Symbolic Logic*, The Macmillan Company, New York, 1947.

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

[19] P.F. Strawson, 'On Referring', *Mind*, 59, 1950, 320-344.

[20] J.V. Tucker & J.I. Zucker, *Program Correctness over Abstract Data Types, with Error State Semantics*, North Holland, Amsterdam 1988.