Tarskian variations
Dynamic parameters in classical semantics

J. van Benthem, G. Cepparello

Computer Science/Department of Software Technology

Report CS-R9419 March 1994

CWI is the National Research Institute for Mathematics and Computer Science. CWI is part of the Stichting Mathematisch Centrum (SMC), the Dutch foundation for promotion of mathematics and computer science and their applications.
SMC is sponsored by the Netherlands Organization for Scientific Research (NWO). CWI is a member of ERCIM, the European Research Consortium for Informatics and Mathematics.

P.O. Box 94079, 1090 GB Amsterdam (NL)
Kruislaan 413, 1098 SJ Amsterdam (NL)
Telephone +31 20 592 9333
Telefax +31 20 592 4199

Tarskian Variations
Dynamic parameters in classical semantics

Johan van Benthem[1] and Giovanna Cepparello[2,3]

[1] ILLC, Plantage Muidergracht 24, 1018 TV Amsterdam, The Netherlands

[2] CWI, P.O. Box 94079, 1090 GB Amsterdam, The Netherlands

[3] Scuola Normale Superiore, Piazza dei Cavalieri 7, 56100 Pisa, Italy

Abstract

In this paper we propose a general perspective of 'Tarskian Variations' providing standard logics uniformly with a dynamic semantics. Many systems from the current literature can be viewed as different implementations of this same basic idea. Given that, we investigate two main technical issues. First, we develop our general dynamic semantics against the background of 'static' logic. In particular, we briefly sketch an axiomatization of our system and a correspondence theory for its models, while also providing an explicit translation of the system into a static one. Next, we study some questions of 'design' of dynamic languages: in particular, how to combine updates with non-eliminative programs, and how to choose a suitable procedural repertoire. Our answer to this second point consists in giving a number of criteria, suggested by other research areas, including *logicality*, *bisimulation safety*, as well as further *denotational constraints* from the theory of Generalized Quantifiers. As a case study we take dynamic negation, as it shows in various systems, providing some technical results on how to characterize its behaviour (using, for instance, inverse logic reasoning in Relational Algebra and Update Logic).

AMS Subject Classification (1991): 03B60, 03B65, 03B70, 68Q55.
CR Subject Classification (1991): F.3.0, F.3.1, I.2.4.
Keywords and Phrases: formal semantics, dynamic logic, generalized quantifier theory, relational algebra.

1 Dynamization

1.1 Motivation

The interest in 'dynamic semantics' is rapidly growing, both in philosophical and linguistic circles, but also in AI and cognitive science. Its underlying shift from knowledge representation to information processing fits very well with the 'cognitivist' flavour of many current trends in the analysis of natural language and of human reasoning. Examples from linguistics include Heim [25] and Kamp [27] in the field of Semantics and Stalnaker [33] in the field of Pragmatics. More recently, the dynamic tendency has influenced the more philosophically oriented study of natural language as well, witness the works by Groenendijk and Stokhof ([21]) and by Veltman ([34]). Finally, as for dynamic views on knowledge representation, we mention the theory of 'knowledge in flux' by Gärdenfors ([19]) (for a more detailed survey cf. [13]).

The dynamic approach displays the usual features of a young, à la mode research area: there are many different systems, alternative notations, and divergent points of view. This is stimulating in a way (witness the diversity of competing theories in computer science, say, of concurrent computation), but still the need seems clear for more general perspectives.

Report CS-R9419
ISSN 0169-118X
CWI
P.O. Box 94079, 1090 GB Amsterdam, The Netherlands

The present report on 'Tarskian Variations' proposes a uniform strategy of 'dynamization' of existing logical semantics. Our aim here is not to develop a whole new theory, but rather discussing and systematizing what is already on the market. Thus, we suggest a taxonomic criterion for the current literature in the area, which actually provides new formal possibilities, throwing light on existing philosophical and linguistic discussions. Moreover, starting from this general strategy, we suggest some possible technical research lines, demonstrating their viability by a number of 'pilot' results.

1.2 Tarskian Variations

Our uniform strategy of dynamization can be illustrated most easily for the case of standard first-order logic (cf. [8]). There, the well-known Tarskian notion of truth governs semantic interpretation:

$$D, I, A \models \varphi$$

A formula φ is evaluated in a model $M = \langle D, I, A \rangle$ on the basis of a set of what may be called semantic *parameters*: the Domain D , the Interpretation I of the signature, and the Assignment A to the variables[1]. Now, the essence of our dynamic story consists in the systematic *variation* of these parameters. The motivation for doing this lies in actual semantics, where we often want to move from one picture of the world (= first order model) to another. What we are after now is a means of bringing out such moves explicitly, as part of the logic.

Changing the parameter of assignment is a well-known device in computer science (cf. [24]) and also more recently, in linguistic semantics. Thus, the 'dynamic predicate logic' of Groenendijk and Stokhof ([21]) transforms the parameter A by means of 'random assignment' η. A program ηx then denotes the following instruction:

ηx = replace the value assigned to x by an arbitrary new value

A nice consequence of this 'shifting the register' for x is that it permits a compositional treatment of anaphoric binding in natural language.

But variation of the other parameters is also possible and, in fact, it makes sense. For instance, consider the interpretation of the signature. A description of the world can be changed by expanding or modifying the interpretation of one or more predicates (with given objects). Such a change occurs, for instance, with the acceptance of an answer to a question of the form "who (or what) has the property P?", where we want to learn more about the extension of the predicate being queried. Another possible application of the variation of the parameter I would be the semantic modelling of the imperative mode in natural language: 'put the block on the table' leads to a new state of the world where (at least) the interpretation of the predicate 'being on the table' has changed. Finally, 'dynamic variation' of I allows us to formalize (compositionally) natural language texts containing so-called 'VP-ellipsis' (like 'Lucas studies Latin. Marco does it too'), which involve reference to changing predicates, analogous to individual anaphora.

Variation of domains occurs naturally too in linguistic interpretation. For instance, different quantifiers in an expression may range over contextually changing domains (witness the 'context sets' for quantifiers of Westerståhl [37]), and the precise mechanism of change here should be brought out explicitly in our calculus.

Summing up, Tarskian Variations involve a dynamic view of the semantics for expressions π, as effecting changes from one Tarskian triple to another

$$D, I, A \xrightarrow{\pi} D', I', A'$$

Accordingly, expressions in our language become programs giving instructions for such transitions. What these instructions are may still depend on which particular dynamic task one has in mind (model checking, model construction, etc.). In general, these instructions will be parametrized, affecting only one parameter at a time. Of course, there are differences between these parameters, which may lead

[1]Note that the $\models$ relation itself can be viewed as a further parameter.

to specific constraints on their variation. For instance, assignments will only change (in general) by successive changes in their register values, while domain changes consist in adding or removing individual objects. Moreover, changes of special interest may obey further constraints. In many areas of semantics, in fact, we are only interested in so-called 'persistence' under extension of a domain (cf. [2]).

1.3 Kripkean Variations

The style of thinking illustrated above for the case of first-order logic applies to virtually all logical truth definitions. For instance, possible worlds semantics for intensional logics affords many further examples of semantic parameters that can be naturally varied. Examples of the assignment-type are changing temporal reference points in temporal logic (cf. [16]), examples of the interpretation type might be changing accessibility or preference relations (cf. [12], [34], [14]), while domain changes also occur naturally in moving from one possible worlds perspective to another (for instance, in the 'update semantics' to be discussed below). We shall refer to this extended view as 'Kripkean Variations'.

1.4 Architectural Issues

The main features of our approach have been as follows. We take an existing classical semantics, and analyze its dynamic potential, both in its models and in its language. In this process, the old 'static' language need not disappear: in fact, there are reasons for letting it co-exist with its dynamic version. For instance, with Tarskian Variations, the 'old' first order language can be retained for describing intermediate states of information processing driven by the new dynamic component. Moreover, we may want to use pre- and postconditions in the standard computational style, by designing a two-level Pratt-style dynamic system with suitable 'switching' mechanisms between static and dynamic components (cf. [7], [8]).

Besides the duality static/dynamic language, many further architectural points come up in the actual design of Tarskian or Kripkean variations. In this report we will be mainly concerned with three:

- the choice of models
- the choice of operators and of their semantic interpretation
- the combination of different dynamic operators within one calculus.

These points do not exhaust all issues in 'general dynamic logic'. For instance, there is still the matter of suitable notions of *dynamic inference*. Several options and results to this effect are discussed in [9], [7] and [34]. We shall not pursue this line here, even though it is certainly congenial to the present perspective.

2 Tarskian Variations in Dynamic Logic

2.1 A Formal System

A concrete system TV for Tarskian Variations will use a two-level language, producing a multimodal system in the usual style (cf. [24] and [20]). Here we have a set of programs $(\pi_1, \pi_2 \ldots)$ and a set of (first-order) formulas $(\varphi, \psi \ldots)$, given some standard first-order similarity type. The procedural repertoire will include boolean connectives for formulas and regular operations for programs, plus two 'switching operators' (one 'test mode' from atomic formulas to programs and one 'modal projection' from programs to formulas).
More precisely, we have:

$$\textbf{TV Formulas} :: = \; Pt_1 \ldots t_n \mid \neg\varphi \mid \varphi \wedge \psi \mid \langle\pi\rangle\varphi$$

$$\textbf{TV Programs} :: = \; (\varphi)? \mid \eta x \mid \mu \mid \delta \mid \pi_1;\pi_2 \mid \pi_1 \cup \pi_2 \mid \neg\pi$$

The reasons for this particular choice of operators will become clear in section 4.

As for the semantics, we have:

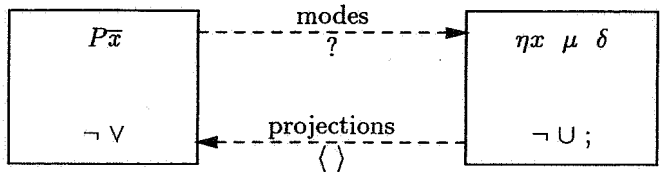

Figure 1: Modes and Projections

Definition 1 (TV model) *A* **TV model** W *consists of a family of models or 'states'* $\langle D, I, A\rangle$*, where* A *is an assignment from variables into the domain* D *and* I *is an interpretation function from predicate letters into denotations over the domain. Notation for states:* $w, v, u, \ldots$*. These models carry the following 'shift relations':*

1. $w =_x v$: *w differs from v at most in its A-value for x*
2. $w =_I v$: *w differs from v at most in its values for I*
3. $w =_D v$: *w differs from v at most in its domain D*

In this definition, we are only using standard tarskian models. We could also allow more general domain shifts, however, which would give us the effects of quantification over 'non-existent objects'.

Next, the truth clauses must be given by a simultaneous induction on programs and formulas. Programs are to be interpreted as sets of state transitions over the set W, while formulas are evaluated at single states.

Definition 2 (Interpretation of TV programs) *A TV program* π *is interpreted on a TV model* W *as a binary relation on* W*, whose graph is as follows:*

1. $[\![\varphi?]\!]_W = \{\langle w, v\rangle \mid w = v \text{ and } v \in [Pt_1 \ldots t_n]_W\}$
2. $[\![\eta x]\!]_W = \{\langle w, v\rangle \mid w =_x v\}$
3. $[\![\mu]\!]_W = \{\langle w, v\rangle \mid w =_I v\}$
4. $[\![\delta]\!]_W = \{\langle w, v\rangle \mid w =_D v\}$
5. $[\![\neg\pi]\!]_W = \{\langle w, v\rangle \mid w = v \text{ and } \neg\exists u \in W : \langle w, u\rangle \in [\![\pi]\!]_W\}$
6. $[\![\pi_1; \pi_2]\!]_W = \{\langle w, v\rangle \mid \exists u : \langle w, u\rangle \in [\![\pi_1]\!]_W \text{ and } \langle u, v\rangle \in [\![\pi_2]\!]_W\}$
7. $[\![\pi_1 \cup \pi_2]\!]_W = [\![\pi_1]\!]_W \cup [\![\pi_2]\!]_W$

These clauses are recursively intertwined with the following ones for TV formulas:

Definition 3 (Interpretation of TV formulas) *A TV formula* φ *is interpreted on a TV model* W *as a subset of* W *according to the following clauses:*

1. $[Px_1 \ldots x_n]_W = \{w = \langle D, I, A\rangle \in W \mid (A(x_1) \ldots A(x_n)) \in I(P)\}$
2. $[\neg\varphi]_W = \{w \in W \mid \text{not } w \models \varphi\}$
3. $[\varphi \vee \psi]_W = \{w \in W \mid w \models \varphi \text{ or } w \models \psi\}$
4. $[\langle\pi\rangle\varphi]_W = \{w \in W \mid \text{there is a } v \in W \text{ such that } \langle w, v\rangle \in [\![\pi]\!]_W \text{ and } v \in [\varphi]_W\}$

2.2 Semantic Fine-Structure

Local and Global Variations This simple system demonstrates some interesting general phenomena. First, consider the striking difference between ηx and μ: ηx modifies the assignment parameter only 'locally', in the variable x, while μ affects the whole interpretation function. Thus, the variational view suggests a new conceptual distinction, between 'local change' and 'global change'. But then, we can also vary the parameter A globally, shifting all registers at once, as well as change the parameter I in a single predicate only. Technically, of course, global variation of a parameter can be decomposed into local changes. For the case of assignment, e.g., we have, using a Kleene star for iteration:

$$\eta = (\cup_{x \in VAR}\ \eta x)^*$$

(but see section 2.3 for a more delicate discussion). The same can be done for global shifts in interpretation. In practice, it is easy to find motivations supporting these finer distinctions. Global assignment change occurs in logic with so-called 'universal' or 'existential closures' of complete formulas, affecting all free variables in them. And similar phenomena occur in natural language, witness the 'unselective binding' of Lewis [29]. Likewise, local change of interpretations occurred naturally in the above example of querying (we usually do not query all predicates at once). Moreover, even in the case of possible worlds for intensional expressions, where one traditionally shifts from one global interpretation for the language to another, there might be a case for more 'controlled' local versions. Thus a modal expression "Mary might win" need only refer to changing the actual world in the single respect of varying the interpretation for the predicate "win", while leaving all other things equal.

Special Constraints on Variations Various interesting subsystems of TV arise from focusing on semantic peculiarities of specific components. For instance, consider the δ operator: it makes sense to split it in two parts, and constrain its variation along the two relations of 'being a submodel' and 'being a model extension':

- $[\![\downarrow \delta]\!] = \{\langle w, v\rangle \mid w \supseteq v\}$
- $[\![\uparrow \delta]\!] = \{\langle w, v\rangle \mid w \subseteq v\}$

This specialization of domain change gives us more expressive possibilities in talking about individuals. For instance, we can now distinguish between quantifying over available objects and over new objects: the latter would be encoded by the formula:

$$\uparrow \delta; \eta x; \varphi$$

expressing the search for an object to instantiate the register x within a possibly bigger domain. Such specializations will make the set of TV validities grow, witness section 2.5.

Going in the opposite direction, one can also move to more abstract versions of our concrete shift relations. For instance, instead of $=_x$, there might be just any abstract relation R_x, whose properties might then be explicitly constrained by special requirements on the behaviour of atomic programs ηx (see section 2.6).

2.3 TV Models as Generalized Tarski Models

TV models can be seen as broader model structures, whose virtue is that they allow us to see more semantic processes going on than would be visible in isolated Tarski models. In this process, we may even come to re-think the standard static semantics that we took for granted until now. Let us see how, following the dynamic re-analysis of [10].

The key clause of Tarski semantics reads:

$$D, I, A \models \exists x\varphi \quad \text{iff there exists an } x\text{-variant } A[x] \text{ of } A \text{ such that } \quad D, I, A[x] \models \varphi$$

This can be seen as a special case of the following general notion, which brings us directly to TV models:

$$D, I, A \models \exists x\varphi \quad \text{iff for some } A',\ R_x A, A' \ \text{ and } \ D, I, A' \models \varphi$$

In our TV models so far, the 'accessibility relation' R_x is simply $=_x$. Given this basic idea, we can say that TV models make the set of relevant parameters grow, in that they provide a context of 'accessible triples'. Thus, ordinary predicate logic can be read against the background of TV models:

$$TV(\text{accessible tarskian triples}), w \text{ (actual tarskian triple)} \models \varphi$$

This TV context may play a role, for instance, in the interpretation of existential quantifiers: $\exists x$ would only look at x-variants within the ambit of available tarskian triples. Note that prescribring a set of 'available' assignments is precisely what has been done by I. Németi in order to isolate decidable fragments of first-order logic. In fact, the set of formulas valid in these 'generalized first-order models', with an arbitrary set of available assignments, is decidable (cf. [30]).

So far, no particular assumption has been made on the available tarskian triples in our TV models. Thus, TV models need not 'have enough states' in the sense of Goldblatt (cf. [20]). This requirement says that for all variables x, for all TV models W, for all tarskian triples $w \in W$ and all objects $d \in D^w$, there exists a $w' \in W$ such that $w =_x w'$, and $A^{w'}(x) = d$. This situation has interesting consequences for our earlier distinction between local and global variations. Notably, the random assignment η in TV (which was interpreted by a global shift) is not equivalent to a standard existential quantifier, or sequence thereof:

$$\langle \eta x \rangle \varphi \not\leftrightarrow \exists x_1 \ldots \exists x_n \varphi$$

Rather, η is a sort of 'inner' quantifier, looking at available tarskian triples, while $\exists$ is a 'outer' quantifier, ranging over the whole logical space. Of course, if we interpret classical formulas directly on TV models, as we have suggested above, then the 'inner' quantifier *is* just the standard one, no matter which assignments are available.

Finally, we would like to point out that TV models are a more general version of Németi's models, in that they do not just model generalized assignments but do the same with interpretations and domains.

2.4 Translation Lore

In this section we look at the question of translating our TV system into a static standard formalism. In principle, a dynamic semantics can always be embedded in a classical one ([8]). The point of such translations, beside their intrinsic interest, is that they allow us, at least in principle, to apply the available theory of standard systems to the dynamic set-up.

What we want to demonstrate here are the following facts: first, the choice of a translation is not unique (we will show two different translations for TV), and special semantic assumptions are eventually necessary in order to guarantee that the translation 'works'. Moreover, looking at a dynamic system 'from its translation' can suggest new questions.

The first translation we propose takes its cue from standard Dynamic Logic. Our starting point was a set of 'tarskian triples' $\langle D, I, A \rangle$ and a set of binary relations among them, built from an atomic base by a family of operators. In order to 'flatten' this semantics to a classical one, we simply take a first-order language with variables $w, w', \ldots$ ranging over states (= tarskian triples) and with a set of atomic binary relations R_π, one for each atomic program. The semantics will be as usual, with Domains of tarskian triples and the usual Interpretation and Assignment functions.

More concretely, we have, employing some harmless confusion of notation:

Definition 4 (Standard counterpart of a TV model) *The* **standard counterpart W** *of a TV model W is a triple* $\langle \mathbf{D}, \mathbf{I}, \mathbf{A} \rangle$ *such that:*

- $\mathbf{D} = \{w \mid w \text{ is a tarskian triple in } W\}$
- $\mathbf{A}(w) = w$
- $\mathbf{I}(R_\pi) = [\![\pi]\!]_W$

Given that, the translation scheme will be as follows:

Definition 5 (Translation scheme I)

- $\overline{\pi} = R_\pi(w_{in}, w_{out})$ *for atomic programs* π

- $\overline{\neg\pi} = (w_{in} = w_{out} \wedge \neg\exists w(\overline{\pi}(w_{out}, w))$
- $\overline{\pi_1; \pi_2} = (\exists w(\overline{\pi_1}(w_{in}, w) \wedge \overline{\pi_2}(w, w_{out}))$
- $\overline{\pi_1 \cup \pi_2} = (\overline{\pi_1}(w_{in}, w_{out}) \vee \overline{\pi_2}(w_{in}, w_{out}))$

The adequacy of this translation rests on a very simple induction:

$$\langle w, w' \rangle \in [\![\pi]\!]_W \quad \text{iff} \quad \mathbf{W} \models \pi(w, w')$$

This abstract translation, where programs are viewed as 'state changers', is not a translation into the same static predicate logic which gets 'dynamized' in TV. Moreover, it is not completely clear that the logic we get is RE: it focuses in fact on a special subclass of the class of models for the abstract language. Thus, we should make sure that our TV models form an RE-definable special class (which is in fact the case, but by no means trivial to prove).

We now propose a direct translation into the static language underlying TV programs. The basic idea is as follows: a TV program π can change (the interpretation of) tuples of predicates, (the assignment to) tuples of variables, and domains. We shall then describe programs via second-order formulas which explicitly encode the change induced by them. In order to express the effect of a program π, its translation φ will duplicate all predicates (including a predicate for 'being in the domain') and all variables affected by the program, tracing the input and the output of π.

This is implemented in the following translation τ taking dynamic formulas $\varphi(\overline{x}, \overline{P}, D)$ to second-order formulas $\tau_\varphi(\overline{x_{in}}, \overline{x_{out}}, \overline{P_{in}}, \overline{P_{out}}, D_{in}, D_{out})$:

Definition 6 (Translation scheme II)

- $\tau_{P(x_1 \ldots x_n)}$ *is a conjunction of the formulas:*
  1. $x_{in} = x_{out}$ *for all variables*
  2. $\forall \overline{x}(P_{in}\overline{x} \leftrightarrow P_{out}\overline{x})$ *for all predicates*
  3. $\forall x(D_{in}x \leftrightarrow D_{out}x)$ *for the domain predicate* D
  4. $P(x_{1,in} \ldots x_{n,in})$
- $\tau_{\eta x}$ *is a conjunction of the formulas:*
  1. $y_{in} = y_{out}$ *for all* $y \neq x$
  2. $\forall \overline{x}(P_{in}\overline{x} \leftrightarrow P_{out}\overline{x})$ *for all predicates*
  3. $\forall x(D_{in}x \leftrightarrow D_{out}x)$ *for the domain predicate* D
- τ_μ, τ_δ *similarly*
- $\tau_{\pi_1;\pi_2} = \exists \overline{u}, \overline{Q}, D : \tau_{\pi_1}(\overline{x_{in}}, \overline{u}, \overline{P_{in}}, \overline{Q}, D_{in}, D) \wedge \tau_{\pi_2}(\overline{u}, \overline{x_{out}}, \overline{Q}, \overline{P_{out}}, D, D_{out})$
- $\tau_{\neg\pi}$ *and* $\tau_{\pi_1 \cup \pi_2}$ *similarly.*

It should be clear how this translation is intended to work. Nevertheless, there are some subtleties. On full models, containing all possible variations of a given tarskian model, an equivalence is easily proved between a TV transition for φ and its τ description. On our general models, however, there may be a mismatch, because the existential quantifiers in the above definition refer to intermediate state transitions that might not be represented in our model (see for instance the clause for a sequential composition). On the other hand, this technical complication may have its benefits, in that we are essentially using a general model semantics in Henkin's sense for the second-order predicate quantifiers, which will allow for axiomatizability (and even decidability in the earlier mentioned generalized perspective).

2.5 Axiomatization

Despite the existence of the above translations, the TV system (which we will consider from now on in its broad version with 'local' and 'global' η and μ) also retains its intrinsic interest, and we certainly want to understand the explicit principles of dynamic reasoning supported by it. We shall not provide a detailed completeness proof here (cf. [15] for one possible strategy), but merely discuss some salient components of the complete logic.

First of all, we need a standard part, including a classical kit plus the minimal modal logic of our semantics:

- The usual axioms and rules of Predicate Logic
- K schemata for $\langle\eta\rangle, \langle\eta x\rangle, \langle\mu\rangle, \langle\mu P\rangle, \langle\delta\rangle$, with x a variable and P a predicate
- S5 schemata for $\langle\eta\rangle, \langle\eta x\rangle, \langle\mu\rangle, \langle\mu P\rangle, \langle\delta\rangle$, with x a variable and P a predicate

Next, we need principles reducing complex programs:

- $\langle\pi_1;\pi_2\rangle\varphi \leftrightarrow \langle\pi_1\rangle\langle\pi_2\rangle\varphi$
- $\langle\neg\pi\rangle\varphi \leftrightarrow \varphi \wedge \neg\langle\pi\rangle\top$
- $\langle\pi_1 \cup \pi_2\rangle\varphi \leftrightarrow \langle\pi_1\rangle\varphi \vee \langle\pi_2\rangle\varphi$

The interaction between 'local' and 'global' operators will be expressed by the following schemata:

- $\langle\eta x\rangle\varphi \rightarrow \langle\eta\rangle\varphi$
- $\langle\mu P\rangle\varphi \rightarrow \langle\mu\rangle\varphi$

Rigidity of interpretation is expressed by the following schema:

- for $\pi = \eta, \eta x, \delta$:

  $$Pa \rightarrow [\pi]Pa$$

  where a is a constant and [ ] denotes the universal modality.

In the presence of Identity, we need some rigidness postulates:

- Rigid Variables schema for $\pi = \mu, \mu P, \delta$:

  $$(x = y) \rightarrow [\pi](x = y)$$

- Rigid Terms schema for δ:

  $$(t_1 = t_2) \rightarrow [\delta](t_1 = t_2)$$

In the presence of fixed domains, Barcan Formulas will be needed:

- for $\pi = \eta, \eta x, \mu, \mu P$, with x a variable and P a predicate:

  $$\langle\pi\rangle\exists x\varphi \rightarrow \exists x\langle\pi\rangle\varphi$$

This concludes our survey of universal principles. Obviously, further specific constraints on variations will have their effects here. For instance, if we opt for splitting the δ operator, S5 principles become too strong, since Symmetry is lost. On the other hand the specific submodel relation also induces new laws, such as the partial order properties of inclusion. On full models, one would even get the existence of 'end-points' (being singleton submdels). An axiomatic characterization in the latter case would therefore include at least:

- S4.1 schemata for $\langle\downarrow\delta\rangle$
- S4 schemata for $\langle\uparrow\delta\rangle$

2.6 Correspondence

The effects of additional principles on TV models can actually be studied more systematically. In general, a 'Correspondence Theory' for TV (and for dynamic) models could be developed, along the lines of Correspondence Theory for Modal Logic (cf. [3]). Here we just want to give some relevant examples:

$\eta x; \eta x = \eta x$ This principle, calling into play only one modality, is already encoded in modal logic correspondence; it expresses in fact the usual S4 principles of Density (from right to left) and Transitivity (from left to right).

$\eta x; \eta y = \eta y; \eta x$ This is a multimodal principle, expressing the following form of 'confluence': $\forall A, A', A'' : (R_x(A, A') \wedge R_y(A', A'')) \rightarrow \exists A'''(R_y(A, A''') \wedge R_x(A''', A''))$. Note that this condition is satisfied by TV models with 'enough states'.

$\eta x; \mu P = \mu P; \eta x$ Analogous to the principle above; notably, it demonstrates the 'independence' of variable and predicate shifts.

$\eta = (\cup_{x \in Var} \eta x)^*$ This principle, which we have already encountered in our discussion, expresses the fact that global shifts must be decomposable into finite sequences of local shifts.

Note that among these 'correspondences', some are universally valid in TV models (like, for instance, Reflexivity and Transitivity of the R_x relation), while those involving existential quantifiers call for specific constraints. A general theory of 'multimodal correspondence' is available from the recent literature (cf. [32] and [35]).

3 Informational Updates

3.1 Options for Update Semantics

The basic dynamic strategy advocated so far consists in shifts in semantic perspective, no matter what these shifts amount to in terms of transmission of information. In other words, TV models are not originally concerned with knowledge representation and information processing, but rather with a formal general policy of dynamization. Nevertheless, it should be noticed that some fragments of the system above do actually have epistemic potentialities. In particular, the operator μ (both in its local and global version) implements some process of 'knowledge updating', at least if we consider tarskian triples as stages of information processing, or as 'databases' (sets of formulas). From this point of view, learning new facts about the world would just amount to changing the interpretation of one or more predicates. In this sense, TV models resemble knowledge representation devices which can be called 'constructive', in that they view information states as 'pictures of the world'. Examples are Discourse Representation Theory (cf. [27]), which updates discourse representation structures by adding data, the recent 'Dynamics of Theory Extension' (cf. [18]), which updates deductively closed theories, and more generally, systems of databases updating (cf. [19]).

Another approach would be to represent information flow using our earlier kripkean variations, via transitions between information states in Kripke models for intuitionistic or similar constructive logics. An example in this spirit is the Dynamic Modal Logic of de Rijke ([31]) and van Benthem ([9]), which has been well-researched using analogies from Dynamic Logic. We shall not follow this approach here, but rather concentrate on yet another - less mathematically explored - viewpoint, which views the process of 'updating' an information state as eliminating successive possibilities from an epistemic horizon. This eliminative view on information processing finds a clear expression in the Update Logic of Frank Veltman (cf. [34]), which we will take as our running example henceforth.

3.2 Eliminative Update Semantics

Consider a possible worlds model for propositional S5. The relevant parameters in the truth definition for a formula φ include a set of possible worlds W, an accessibility relation R and a valuation V for the proposition letters:

$$W, R, V \models \varphi$$

For present purposes, we keep R and V fixed and we modify W, reading 'classical' propositional formulas as instructions for updating the universe of possibilities. Consider the above-mentioned system of 'Update Logic'. Its language consists of the following programs, starting from a set of propositional variables $p \in P$:

UL Programs $:: = p \mid \pi_1; \pi_2 \mid \pi_1 \cup \pi_2 \mid \neg\pi \mid \Diamond\pi$

This language is evaluated in the following models:

Definition 7 (UL model) *A standard* **UL model U**, *given a set of propositional variables P, is the set of possible worlds $\wp(P)$ (ordered by the total accessibility relation), or alternatively the set of all bivalued valuations of the proposition letters.*

In analogy with TV models, one can also define 'generalized' UL models as subsets of the set $\wp(P)$. In this way, we could encode the validity of 'universal constraints' ruling out certain possibilities beforehand. (On this issue of encoding, as well as on the general connection between eliminative models and kripkean variations, see [26]). We will not pursue this issue here.

Over the above models, formulas will be interpreted as 'updating functions':

Definition 8 (Interpretation of UL programs) *A UL program π is interpreted on a model* **U** *as a function from $\wp(\mathbf{U}) \longrightarrow \wp(\mathbf{U})$ satisfying:*

1. $[\![p]\!]_U(U) = \{w \in U \mid p \in w\}$
2. $[\![\neg\pi]\!]_U(U) = U - ([\![\pi]\!]_U(U))$
3. $[\![\Diamond\pi]\!]_U(U)$

$$= \begin{cases} U & \text{if } [\![\pi]\!]_U(U) \neq \emptyset \\ \emptyset & \text{otherwise.} \end{cases}$$

4. $[\![\pi_1; \pi_2]\!]_U(U) = [\![\pi_2]\!]_U([\![\pi_1]\!]_U(U))$
5. $[\![\pi_1 \cup \pi_2]\!]_U(U) = [\![\pi_1]\!]_U(U) \cup [\![\pi_2]\!]_U(U)$

From now on, we will mostly omit the index **U**. The idea of this definition is clear: 'classical information' discards possibilities, whereas the modality $\Diamond$ is an epistemic test of the current horizon. (This close connection with modal S5 may already be seen from the 'monadic translation' first given in [5]). Thus, the general picture of information processing arising here is that of factual updates interleaved with transient tests on successive states of this process. We shall return to various technical aspects of this distinction later.

3.3 Combining Tarskian Variations With Updates I

In the literature, there has been a good deal of attention to the problem of combining variations of tarskian parameters with eliminative updates. Technically, this is highlighted by the question of combining Veltman's UL with DPL (i.e., the dynamic predicate logic of Groenendijk and Stokhof [21], considered here as a fragment of our TV system). Proposed solutions include [17], [16], [23]. We shall briefly discuss some salient points.

The syntax of the combined system simply amounts to DPL syntax (where the only atomic programs are classic atoms and ηx) plus the UL modality $\Diamond$. The difficulty in giving an appropriate semantics is due to a different 'granularity': DPL programs modify single first-order models, whereas UL programs modify modal models, i.e. sets of (propositional or first order) possible worlds. Here, 'lowering' the granularity of UL semantics does not make sense, because its behaviour on sets of tarskian triples cannot be reduced in general to behaviour on individual triples. The obvious policy then consists in 'lifting' the interpretation of DPL programs to the set level. The problems that arise here may be demonstrated vividly by examining the behaviour of negation.

One obvious 'lifted' negation for DPL would be:

$$[\![\neg\pi]\!]W = \{w \in W \mid [\![\pi]\!]\{w\} = \emptyset\}$$

This negation only looks at the 'pointwise' behaviour of programs which would not work in an updating context. As a consequence, the program $\neg\Diamond p$ would become equivalent to $\neg p$. Thus, the following would be true:

$$[\![\Diamond p; \neg\Diamond p]\!]W \neq \emptyset$$

On the other hand, the UL interpretation of negation as 'complement',

$$[\![\neg\pi]\!]W = W - [\![\pi]\!]W$$

does not fit with DPL programs, because of its purely eliminative character. This would produce undesirable effects when applied to 'dynamic quantifiers', as in:

$$[\![\neg(\eta x; Px); (\eta x; Px)]\!]W \neq \emptyset$$

The Dynamic Modal Predicate Logic of van Eijck and Cepparello (cf. [17]) solves this problem by using a two-level semantics, assigning to each program π a 'parametrized' updating function $[\![\pi]\!]^u_s$, where s, u are assignments, which maps index sets I to new index sets. The following definition illustrates how the two relevant semantic dimensions can be kept separate.

Definition 9 (Semantics of DMPL)

1. $[\![Rt_1 \cdots t_n]\!]^s_u(I) = \{i \in I \mid s = u \text{ and } M, i \models_s Rt_1 \cdots t_n\}$.
2. $[\![\pi_1; \pi_2]\!]^s_u(I) =$ *standard*
3. $[\![\pi_1 \cup \pi_2]\!]^s_u(I) =$ *standard*
4. $[\![\neg\pi]\!]^s_u(I) = \{i \in I \mid s = u \text{ and there is no } r \text{ with } i \in [\![\pi]\!]^s_r(I)\}$.
5. $[\![\eta x]\!]^s_u(I) = \{i \in I \mid u = s(x|d) \text{ for some } d \in M\}$

   $$= \begin{cases} I & \text{if } u = s(x|d) \text{ for some } d \in M, \\ \emptyset & \text{otherwise.} \end{cases}$$
6. $[\![\Diamond\pi]\!]^s_u(I) = \{i \in I \mid s = u \text{ and there is an } r \text{ with } [\![\pi]\!]^s_r(I) \neq \emptyset\}$

   $$= \begin{cases} I & \text{if } s = u \text{ and there is an } r \text{ with } [\![\pi]\!]^s_r(I) \neq \emptyset, \\ \emptyset & \text{otherwise.} \end{cases}$$

Thus the puzzle of negation is solved by reading negation as 'complement' along the index set dimension, and, at the same time, as a 'test' on single assignments.

Another approach to this problem has been proposed by Groenendijk, Stokhof and Veltman (cf. [23]). In their system, which we shall call GSV, the modality $\Diamond$ ranges over interpretations and assignments at the same time. The syntax is the same as in DMPL, except for two facts. First, in GSV the random assignment η is not a program by itself, but a 'dynamic quantifier', which then needs to be restricted by a formula (concretely, ηx is not a well formed program, while $\eta x : \pi$ is). Second, the operation $\cup$ of boolean choice is not allowed. The reasons why it is so will be clear in a while. As for the semantics, states are here 'partial': given a fixed domain D, a GSV (atomic) state will include an interpretation over D and a partial assignment A to the variables. Again, the semantics is 'lifted', with sets of partial states as basic units. As a consequence, the problems above (e.g. in defining negation) arise in this context too. But, since - as we said - the GSV-modality is supposed to have a wide spectrum (including alternative assignments), the trick of setting apart two levels in the semantics will not do. Alternatively, the idea is to 'trace' programs effects on single states by introducing an ordering on them (following an idea from [36]). Let us see how.

Definition 10 (Referent system) *A* **referent system** *is a pair $\langle n, r\rangle$ where n is a natural number and r is a partial injective function from Variables into n.*

Definition 11 (atomic state, state, model) *Given a Domain D, an* **atomic state** *s is a quadruple $\langle n, r, A, I\rangle$, where $\langle n, r\rangle$ is a referent system, A is a function from n to D and I is an interpretation over D. A* **state** *S is a set of atomic states. A* **model** *M is a set of states.*

Definition 12 (Extension ordering) *Given two atomic states $s = \langle n, r, A, I\rangle$ and $s' = \langle n', r', A', I'\rangle$, s' is an* **extension** *of s (in symbols $s \leq s'$) if the following conditions hold:*

1. $n \leq n'$
2. $Dom(r) \subseteq Dom(r')$
3. $\forall x \in Dom(r)(r(x) \leq r'(x))$
4. $\forall n \in Dom(A)(A(n) = A(n'))$
5. $I = I'$

Notation: if $s = \langle n, r, A, I\rangle$, $s[x/d]$ is the atomic state $\langle n+1, r', A', I\rangle$ where $r'(y) = r(y)$ for all $y \neq x$, $r'(x) = n$, $A' = A \cup \{\langle n, d\rangle\}$. Similarly, $S[x/d] = \{s[x/d] \mid s \in S\}$. Given that, here are the relevant semantic clauses:

Definition 13 (GSV semantics) *A GSV program π is interpreted on a GSV model M as a function from M to M, satisfying the following clauses:*

1. $[\![P(t_1 \ldots t_n)]\!]S = \{s \in S \mid \langle A(r(t_1)) \ldots A(r(t_1))\rangle \in I(P)\}$
2. $[\![\eta x : \pi]\!]S = \cup_{d \in D}[\![Pt_1 \ldots P_n]\!]S[x/d]$
3. $[\![\Diamond\pi]\!]S = \begin{cases} S & \text{if } [\![\pi]\!]S \neq \emptyset \\ \emptyset & \text{otherwise.} \end{cases}$
4. $[\![\neg\pi]\!]S = \{s \in S \mid \neg\exists s' : s \leq s' \text{ and } s' \in [\![\pi]\!]S\}$

We only make two short comments. First, the reason why $\eta x : Pt_1 \ldots t_n$ cannot be read as a sequential composition is that the interpretation of η is 'piecemeal' in the sense of processing the 'shift' to x object by object. Take then the program:

$$\eta x : \Diamond\pi$$

Its output is supposed to give all possible values of x which are possibly π, and not all the possible values of x if x can be instantiated with an object which is π. In other words, in GSV the following continuity property is lost:

$$\cup_{d \in D}([x/d]; \pi) = \cup_{d \in D}([x/d]); \pi$$

(where by $[x/d]$ we mean the program: $\lambda S.\ S[x/d]$) and, consequently:

$$[\![\eta x : \pi]\!] = \cup_{d \in D}([x/d]; \pi) \neq [\![\eta x; \pi]\!] = (\cup_{d \in D}[x/d]); \pi$$

Note that this continuity holds in DMPL, because its modality ranges over alternative interpretations only, leaving the system distributive as far as the assignment-dimension is concerned.

Our second comment on GSV concerns boolean choice. It is not possible to define it here because it would produce atomic states with different carrying referent systems within the same state, which would make the clause for negation fail, for instance, for atomic first-order formulas.

3.4 Combining Tarskian Variations With Updates II

In this section we make a third attempt, designing a uniform variational system TKV (Tarskian Kripkean Variations), with global and local modifiers for each tarskian parameter and 'parametrized' modalities. Our syntax runs as follows:

TKV Programs :: $= Pt_1 \ldots t_n \mid \neg\pi \mid \pi_1;\pi_2 \mid \pi_1 \cup \pi_2 \mid \eta \mid \eta x \mid \mu \mid \mu P \mid \downarrow\delta \mid \uparrow\delta \mid \Diamond^{\eta}\pi \mid \Diamond^{\eta x}\pi \mid \Diamond^{\mu}\pi \mid \Diamond^{\mu P}\pi \mid \Diamond^{\downarrow\delta}\pi \mid \Diamond^{\uparrow\delta}\pi$

TKV models are just like TV models, but we shall exploit both the individual states and sets of these. On these models, programs will be interpreted as updating functions. In order to keep a double granularity in the semantics, we introduce a program-dependent ordering on single states, encoding the core behaviour of distributive programs:

Definition 14 (Pointwise ordering)

1. *for* $\pi = P(x_1 \ldots x_n)$, $w \succ_\pi w'$ *iff* $w = w'$ *and* $w' \models P(x_1 \ldots x_n)$
2. *for* $\pi = \eta x$, $w \succ_\pi w'$ *iff* $w' =_x w$
3. *similarly for* $\eta, \mu, \mu P, \uparrow\delta, \downarrow\delta$ *with their corresponding shift relations*
4. *for* $\pi = \pi_1;\pi_2$, $w \succ_\pi w'$ *iff there is a* w'' *such that* $w \succ_{\pi_1} w''$ *and* $w'' \succ_{\pi_2} w'$
5. *for* $\pi = \pi_1 \cup \pi_2$, $w \succ_\pi w'$ *iff* $w \succ_{\pi_1} w'$ *or* $w \succ_{\pi_2} w'$.
6. *for all other programs* π, *in particular modal tests,* $\succ_\pi$ *is the identity relation.*

Note that this basically mimicks our earlier TV evaluation. In the general case, our lifted update conditions will assist in evaluating non-distributive constructions. A notational convention: $|w|_A^W = \{v \in W \mid v =_A w\}$, and similarly for all other tarskian parameters.

Definition 15 (Interpretation of TKV Programs) *A TKV program* π *is interpreted on a TKV Model* $\mathbf{W}$ *as a function from* $\wp(\mathbf{W})$ *to* $\wp(\mathbf{W})$, *satisfying the following clauses:*

1. $[\![Pt_1 \ldots t_n]\!]W = \{w \mid \exists w' \in W : w \succ_{Pt_1 \ldots t_n} w'\} = \{w \in W \mid w \models P(x_1 \ldots x_n)\}$
2. $[\![\eta x]\!]W = \{w \mid \exists w' \in W : w' \succ_{\eta x} w\}$
   (Note that this amounts to the image of W *under the earlier relation* $\succ_{\eta x}$*)*
3. *Similarly for* η, μ *etcetera*
4. $[\![\Diamond^{\eta}\pi]\!]W = \{w \mid \exists v \in |w|_A^W \ (\exists u(v \succ_\pi u \ \wedge \ u \in [\![\pi]\!]W))\}$
5. *Similarly for the other modalities*
6. $[\![\neg\pi]\!]W = \{w \mid \neg\exists v \ (w \succ_\pi v \ \wedge \ v \in [\![\pi]\!]W)\}$
7. $[\![\pi_1;\pi_2]\!]W = [\![\pi_2]\!]_W([\![\pi_1]\!]_W)$
8. $[\![\pi_1 \cup \pi_2]\!]W = [\![\pi_1]\!]_W \cup [\![\pi_2]\!]_W$

This definition does not present obvious problems for the negation clause. In particular,

- $[\![\Diamond p;\neg\Diamond p]\!]W = \emptyset$ for every TKV modality
- $[\![\neg(\eta x;Px);(\eta x;Px)]\!]W = \emptyset$

Moreover, this interpretation of TKV respects some peculiarities of the UL modality. For instance, $[\![\neg\pi;\Diamond\pi]\!]W = \emptyset$ is valid, whereas $[\![\Diamond\pi;\neg\pi]\!]W = \emptyset$ is not.

We do not think that this particular system has solved all outstanding problems in the area - not even of negation, which remains a tricky notion. What we propose to do next is attack these issues from another side, by surveying general logical features which should be obeyed by any dynamic system of this kind.

4 Logical Issues

4.1 The logical space of dynamic operators

In this part, we turn to general perspectives on dynamic logical operators. We will be inspired, to some extent, by the theory of Generalized Quantifiers (cf. [4], [37]). Thus, on the one hand we will try to impose conditions on possible operators 'from the outside', guided by basic mathematical intuitions - while on the other, we will start from intuitive desiderata on the behaviour of some specific operator, say negation, and search for their impact. The two perspectives occur interlinked in several definability results to be proved below.

To investigate the logical space of dynamic operators, we need to determine a suitable type for them. Several options are available for their arguments, of which the following two were encountered above. Given a set W of 'atomic states' (possible worlds, tarskian triples), we have:

- Relational algebra type : Relations $\subseteq W \times W$
- Update Semantics type : Functions $\in \wp(W)^{\wp(W)}$

Of these two, the 'relational algebra' type has been investigated at greater length already (cf. [9]). Therefore, our analysis will mainly concentrate on programs of the second type, which are also somewhat more flexible - using results concerning the relational type occasionally to guide us. In general, programs of the update type will be defined as follows:

$$[\![\pi]\!] = \lambda W.\ \{w \mid \varphi(w, W, \ldots)\}$$

where φ is a condition in some suitable formal language.

4.2 Choice of Operators: Logicality

What are legitimate 'logical' dynamic operators? Intuitively, 'logical' expressions in any language are not involved with any content, but only with 'formal structure' (see e.g. [9]). A well-known technical implementation of this intuition is preservation under permutations of individuals in the underlying models.

To understand the idea, here is an example from set theory. An n-ary operation O on sets is called 'permutation-invariant' if, given any n-tuple $S_1 \ldots S_n$ of sets, and any permutation α of the universe:

$$O(\alpha S_1 \ldots \alpha S_n) = \alpha(O(S_1 \ldots S_n))$$

Such operations must behave in a 'uniform' manner, as can be shown by some simple mathematical analysis (cf. the general range of examples in [6]). For instance, it can be proved that an n-ary operation O is permutation invariant iff it can be defined, for each n-tuple, by some combination of the 'Boolean zones' in the Venn diagram formed by all sets in the n-tuple. For dynamic operations, the obvious definition of 'permutation invariance' will employ permutations of programs induced by permutations of the underlying states, as follows:

Definition 16 (Induced permutation) *Given a permutation α of the basic state set, the* **permutation of a program π induced by** *α, in symbols $\alpha(\pi)$, is a program defined as follows:*

$$[\![\alpha(\pi)]\!] = \{\langle \alpha W, \alpha W' \rangle \mid \langle W, W' \rangle \in [\![\pi]\!]\}$$

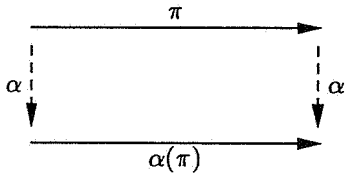

Figure 2: permutation of a program

The picture above suggests another obvious way of describing the permutation of a program π:

$[\![\alpha(\pi)]\!]W =$
$[\![\alpha(\pi)]\!]\alpha\alpha^{-1}W =$
$\alpha(\pi\alpha^{-1}W) =$
$[\![\alpha^{-1};\pi;\alpha]\!]W$

Now, we can define a consonant notion of *permutation invariance* for a dynamic program operation @:

Definition 17 (Permutation invariance for a program operation) *An n-ary program operation @ is* **permutation invariant** *iff, for all programs $\pi_1 \ldots \pi_n$ and permutations α, the following holds:*

$$[\![@(\alpha(\pi_1)\ldots\alpha(\pi_n))]\!] = [\![(\alpha(@(\pi_1\ldots\pi_n)))]\!]$$

or, in other words, iff $[\![\alpha^{-1};@(\pi_1\ldots\pi_n);\alpha]\!] = [\![@((\alpha^{-1};\pi_1;\alpha)\ldots(\alpha^{-1};\pi_n;\alpha))]\!]$, for all $\pi_1\ldots\pi_n$ and all α.

We can now generate many dynamic 'logical' operations, by staying inside some safe linguistic format of definition:

Theorem 1 *If the n-ary program operator @ is defined as follows:*

$$[\![@(\pi_1\ldots\pi_n)]\!] = \lambda W.\ \{w \mid \varphi(\pi_1,\ldots,\pi_n,W,w)\}$$

where φ is a set-theoretic defining condition, then @ is permutation invariant.

Proof: Take an n-tuple of programs $\pi_1\ldots\pi_n$ and a program permutation α. We have that:
$[\![\alpha^{-1};@(\pi_1,\ldots\pi_n);\alpha]\!] =$
$\lambda W.\ \alpha([\![@(\pi_1,\ldots\pi_n)]\!](\alpha^{-1}(W)) =$
$\lambda W.\ \{\alpha(w) \mid \varphi(\pi_1,\ldots\pi_n,\alpha^{-1}(W),w)\} =$
by invariance of set-theoretic statements for permutations of individual domains
$\lambda W.\ \{\alpha(w) \mid \varphi(\alpha(\pi_1),\ldots,\alpha(\pi_n),\alpha\alpha^{-1}(W),\alpha(w))\} =$
by general properties of permutations
$\lambda W.\ \{w \mid \varphi(\alpha(\pi_1),\ldots,\alpha(\pi_n),W,w)\} =$
$\lambda W.\ \{w \mid \varphi(\alpha^{-1};\pi_1;\alpha,\ldots,\alpha^{-1};\pi_n;\alpha,W,w)\} =$
$[\![@((\alpha^{-1};\pi_1;\alpha),\ldots,(\alpha^{-1};\pi_n;\alpha))]\!]$. ∎

There are also partial converses of this result: under favourable circumstances, all invariant operators over a given universe of states are definable in some suitable logical formalism (cf. [9], [6]). We shall return to this issue for a special case later on.

4.3 Further Notions of Logicality

The standard 'procedural equipment' that we have been using for our system TV ('counterdomain negation' $\neg$, 'boolean choice' $\cup$, sequential composition ; and test ?) is of course much more constrained than the very general logical space mapped out in the preceding subsection. The reason is that it satisfies a much stronger criterion for 'logicality' of dynamic operators. We can think roughly of permutation invariance as saying that, whenever arguments of an operator are connected by some isomorphisms, then so are its values, via some canonical modification of those isomorphisms. But this idea can be generalized to 'respect' for other notions of 'simulation' between argument values. Notably, one crucial notion of process equivalence from the computational literature is the following:

Definition 18 (Bisimulation) *A relation f between states in two TV models W and W' is a* **bisimulation for a program** *π if, whenever wfw', the following conditions hold:*

- *w and w' verify the same 'static' atomic formulas*
- *if $\langle w,v\rangle \in [\![\pi]\!]_W$, then there exists a $v' \in W'$ such that $\langle w',v'\rangle \in [\![\pi]\!]$ and vfv', and vice versa.*

Definition 19 (Bisimulation Safety) *An operation $@(\pi_1\ldots\pi_n)$ on programs is* **safe for bisimulation** *if every bisimulation f for $\pi_1\ldots\pi_n$ is also a bisimulation for $@(\pi_1\ldots\pi_n)$.*

Now, instead of full set theory, consider just the obvious first-order formalism for defining operations over binary relations. Here, we can see what makes our earlier repertoire uniquely distinguished, at least over all multi-modal models of the TV kind (cf. [11]):

Theorem 2 *A first-order program operation $@\pi_1 \ldots \pi_n$ is safe for bisimulation iff it can be defined from $\pi_1 \ldots \pi_n$ using atomic tests ? as well as only the three operations $\cup, ;, \neg$.*

Proof: cf. [11] ∎

The earlier notion of permutation invariance can be connected up with bisimulation invariance. Note that f is a bisimulation with respect to π, essentially, if $f; \pi = \pi; f$. Now, this also makes sense for the earlier permutations α. Then we have the following connection.

Theorem 3 *Each permutation-invariant program operator $@(\pi_1 \ldots \pi_n)$ is safe for permutations that bisimulate all its arguments.*

Proof: Suppose that $\alpha; \pi_i = \pi_i; \alpha$, for $1 \leq i \leq n$; i.e., $\pi_i = \alpha^{-1}; \pi_i; \alpha = \alpha(\pi_i)$. By the permutation invariance of @, we have:
$\alpha(@(\pi_1 \ldots \pi_n)) = @(\alpha(\pi_1) \ldots \alpha(\pi_n)) = @(\alpha(\pi_1 \ldots \pi_n)$
Moreover:
$\alpha(@(\pi_1 \ldots \pi_n)) = \alpha^{-1}; @(\pi_1 \ldots \pi_n); \alpha$
Combining these two facts, we get:
$@(\pi_1 \ldots \pi_n); \alpha = \alpha; @(\pi_1 \ldots \pi_n)$ ∎

4.4 Denotational Constraints in Update Semantics

We will now continue the analogy with Generalized Quantifier Theory, focusing on Update Semantics. Let us recall some basic notions from GQT. Quantifiers are functions Q which, to any universe E, assign a binary relation Q_E on $\wp(E)$. Notation:

$$Q_E AB.$$

Given this very general type of semantic object, one now searches for constraints, going from generally plausible intuitions to special-purpose mathematical conditions. Well-known examples from this tradition are:

- Extension EXT (context independence): if $A, B \subseteq E \subseteq E'$, then $Q_E AB$ iff $Q_{E'} AB$.
- Conservativity CONS: $Q_E AB$ iff $Q_E A(B \cap A)$.
- Variety VAR: if $A \subseteq E$ is non-empty, then there exist B, B' such that $Q_E AB$ and not $Q_E AB'$.
- Quantity QUANT (permutation invariance): if α is a bijection between E and E' and $A, B \subseteq E$, then $Q_E AB$ iff $Q_{E'} \alpha(A)\alpha(B)$.

Moreover, here are some other important special properties, which are not generally valid for all quantifiers:

- Upward-Monotonicity MON↑: if $Q_E AB$ and $B \subseteq B'$ then $Q_E AB'$.
- Downward-Monotonicity MON↓: if $Q_E AB$ and $B' \subseteq B$ then $Q_E AB'$.
- Upward-Persistence PERS↑: if $Q_E AB$ and $A \subseteq A'$ then $Q_E A'B$.
- Downward-Persistence PERS↓: if $Q_E AB$ and $A' \subseteq A$ then $Q_E A'B$.

As an example of how these properties can characterize subclasses of quantifiers, here is a typical result:

Theorem 4 *A generalized quantifier satisfies EXT, CONS, VAR, QUANT and PERS iff it is in the* Square of Opposition *(namely iff it is one of the following:* all, some, no *and* not all).

Proof: cf. [4]. ∎

Within reason, this style of analysis also applies to dynamic operators. Here, we are not after developing a whole GQT-style theory of dynamic semantics. Rather, we give a sample, demonstrating its viability by isolating a subclass of dynamic operators which can be treated as generalized quantifiers. This illustrates the possibility of 'exporting' results from GQT to Dynamics.

First, we need to further analyse the type of dynamic operators. As we have already seen, a dynamic operator @ will usually be defined by a set-theoretic condition of the following form:

$$\varphi(\pi_1 \ldots \pi_n, w, W)$$

Examples of such formulas φ are:

$\neg$	$w \in W \wedge w \notin \pi W$
$\vee$	$w \in \pi_1 W \vee w \in \pi_2 W$
;	$w \in \pi_2(\pi_1 W)$
$\Diamond$	$\exists v \in \pi W \wedge w \in W$
$\triangle$	$u \in W \vee u \in \pi(1)$

where '$\triangle$' is a revision modality ('unless π') proposed in [5], which updates the current state with the result of processing π from the initial information state ('1'). Note that the type of these conditions φ is still rather complex: they take into account a set of functions (the programs), an input set W and a 'reference' point w. It turns out that this type gets lowered if we restrict our attention to a special, but rather natural class of program operators, which we will call 'extensional':

- Extensionality: if $\pi_i(W) = \pi_i'(W)$ for $1 \leq i \leq n$, then $@(\pi_1 \ldots \pi_n)(W) = @(\pi_1' \ldots \pi_n')(W)$.

Intuitively, a n-ary program operator is extensional if its defining condition only uses the sets $\pi_i(W)$, with W the input state and $1 \leq i \leq n$. Non-examples are the above sequential composition (it refers to $\pi_1(\pi_2(W))$) and $\triangle$ (it refers to $\pi(1)$). This property allows us to re-type defining conditions for extensional operators. They will have the form:

$$\varphi(A_1 \ldots A_n, w, W)$$

where $A_1 \ldots A_n$ are not functions but sets $(\pi_1(W) \ldots \pi_n(W))$. Another important special feature of program operators, which often occurs in practice, is their possible *continuity*. This can happen both with program arguments and with input state arguments (cf. section 4.7):

- Program Continuity in π_i : the defining condition φ satisfies $\varphi(\pi_1 \ldots \cup_i \pi_i \ldots \pi_n, W, w)$ iff $\bigvee_i \varphi(\pi_1 \ldots \pi_i \ldots \pi_n, W, w)$.
- State Continuity: the defining condition φ satisfies $\varphi(\pi_1 \ldots \pi_n, \cup_i W_i, w)$ iff $\bigvee_i \varphi(\pi_1 \ldots \pi_n, W_i, w)$.

For instance, in UL, the parallel composition '$\wedge$' is continuous in both its program arguments, whereas the sequential composition ; is only continuous in its right-hand program argument:

$$(\pi_1 \cup \pi_2); \pi_3 \neq (\pi_1; \pi_3) \cup (\pi_2; \pi_3)$$
$$\pi_1; (\pi_2 \cup \pi_3) = (\pi_1; \pi_2) \cup (\pi_1; \pi_3)$$

This is reminiscent of the non-homogeneity encountered in Process Algebra, which is due to different 'moments of choice', although there, judgments of validity go the other way round (cf. [1]).

Finally, we mention a property of program (operators) which will return at greater length in section 4.7. A program operator @ is a *test* if, for all programs π and input states W, the output of $@\pi(W)$ can only be W itself or the empty set $\emptyset$ (for example, think of the UL modality):

- Test Property: no reference point occurs in the defining condition (whence it has the following form: $\varphi(\pi_1 \ldots \pi_n, W)$).

Obviously, tests allow one more type-lowering: from $\varphi(\pi_1 \ldots \pi_n, W, w)$ to $\varphi(\pi_1 \ldots \pi_n, W)$. Combining this with extensionality, we get the reduced type: $\varphi(A_1 \ldots A_n, W)$ of a condition on sets, or, in other words, of a relation among sets. But this is just the standard type for generalized quantifiers, with the parameter E for the total universe now read as our 'initial information state', or 'total model' (which played a role, e.g., in our previous Δ).

As a consequence, GQT results now apply without further ado to extensional program operators which are tests. Thus, the earlier 'square of oppositions' for quantifiers fits with dynamic existential and universal modalities ($\Box$, $\neg\Box$, $\Diamond$, $\neg\Diamond$):

Theorem 5 *An extensional logical dynamic test has a defining condition in the* Square of Opposition *iff its associated quantifier satisfies EXT, CONS, VAR and PERS*

Note that these GQT conditions on quantifiers assume a rather different meaning in the dynamic set-up: e.g., CONS will express a form of 'Eliminativity' for update operators.

4.5 Analyzing Negation

Now we shall start from the other end, looking for intuitive semantic constraints by examining a specific basic operator. Intuitively, the *negation* of a program should lead us, from any given stage of information processing to a new stage where the negated program would fail if run. Our first more concrete move is to impose further constraints on this behaviour, so that we can classify suitable candidates. For instance, a plausible requirement is a form of 'dynamic excluded middle' for all information states W:

- (1) $[\![\pi; \neg\pi]\!]W = \emptyset$
- (2) $[\![\neg\pi; \pi]\!]W = \emptyset$

What these conditions tell us is that, once something has been accepted, a processor cannot change its mind any longer. This requirement would of course be dropped in a revisionistic perspective. But in fact, it is even too strong in UL and DPL, as it is shown by the following examples:

- $[\![(\Diamond\pi; \neg\pi); \neg(\Diamond\pi; \neg\pi)]\!]W \neq \emptyset$
- $[\![(Px; \eta x; \neg Px); \neg(Px; \eta x; \neg Px)]\!] \neq \emptyset$

Using double negations, the first counter-example to (1) also becomes one to principle (2). We will discuss this situation in more detail in sections 4.6 and 4.7:

	UL	DPL
(1)	-	-
(2)	-	+

Further reasonable assumptions abound. Given a total set of such desiderata, the art is to design a dynamic system fulfilling them - or even, to find a complete mathematical characterization of the options for achieving this. This is analogous to the 'inverse logic' of Generalized Quantifier Theory. For a start, we record the special conditions on UL and DPL negation expressed by (1) and (2).

1. UL negation validates condition (1) for just those programs π that are 'idempotent', i.e. such that, for all information states W:

   $$[\![\pi]\!]([\![\pi]\!]W) = [\![\pi]\!]W.$$

2. UL negation makes condition (2) true for all programs π that are 'progressive', namely such that, for all information states W:

   $$[\![\pi]\!](W - [\![\pi]\!]W) = \emptyset$$

The proof of these fact is by some straightforward calculations, using eliminativity at some stage.

Here are some corresponding facts about DPL negation:

1. DPL negation validates condition (1) only for programs π with the 'weak update property':

$$\forall w, w'(\langle w, w' \rangle \in [\![\pi]\!] \rightarrow \exists w''(\langle w', w'' \rangle \in [\![\pi]\!]))$$

2. DPL negation satisfies condition (2) for all programs.

Concerning condition (1), there is a stronger update principle that might be reasonable, namely that

$$\forall w, w'(\langle w, w' \rangle \in [\![\pi]\!] \rightarrow \langle w', w' \rangle \in [\![\pi]\!])$$

. This would correspond to the condition that $[\![\pi; \neg(\pi \wedge Id)]\!] = \emptyset$.

4.6 Inverse Logic I: DPL Negation

In this section, we characterize DPL negation in Relational Algebra. Again, this is merely meant as a sample of a feasible general style of analysis. More precisely, we prove:

Theorem 6 *DPL negation is the only permutation-invariant operator in Relational Algebra satisfying the following conditions:*

1. $\neg\, 0 = Id$
2. $\neg(\cup_i \pi_i) = \cap_i \neg\pi_i$
3. $\neg\neg\pi \leq \pi; 1$
4. $\neg\pi; \pi = 0$

Proof: We start with an auxiliary observation.

Lemma 7 2 *implies that* $\neg\pi \leq Id$.

Proof: We know (from Relational Algebra) that $\pi = \pi \cup 0$. From this (again by RA) it follows that $\neg\pi = \neg(\pi \cup 0) = \neg\pi \cap \neg 0$ (by 2), whence $\neg\pi \leq \neg 0 = Id$. ∎

Now, here is the main argument. Given any relation π, the relation $\neg\pi$ can be retrieved from the values $\neg(\{\langle x, y \rangle\})$, for $\langle x, y \rangle \in \pi$. This is because $\pi = \cup_{\pi xy}\{\langle x, y \rangle\}$. Using 2 then, $\neg\pi = \cap_{\pi xy}\neg(\{\langle x, y \rangle\})$.

Now, we have seen that $\neg\pi \leq$ Id. Hence, by the permutation invariance of $\neg$, $\neg(\{\langle x, y \rangle\}) = \{\langle z, z \rangle \mid \varphi(x, y, z)\}$ can only refer to the 'Venn zones' consisting of $\{x\}$, $\{y\}$ and their Boolean combinations. The argument is then case by case.

Case 1: $x = y$. Here are the options for z:

- $z = 1 - \{x\}$. This is what we want.
- $z = 0$. Here we need to distinguish two further cases. If the domain contains one object only, then this is our previous case, and we are done. If the domain contains more than one object, then we obtain a contradiction as follows. Suppose that $\neg(\{\langle x, x \rangle\}) = 0$ and that the domain contains some $y \neq x$. From $\neg(\{\langle x, x \rangle\}) = 0$ it follows, by 1, that $\neg\neg(\{\langle x, x \rangle\}) =$ Id. But by 3, $\neg\neg(\{\langle x, x \rangle\}) \leq (\{\langle x, x \rangle\}); 1$, , and hence Id $\leq (\{\langle x, x \rangle\}); 1$. But this cannot be true, because the domain of Id is larger than $\{x\}$).
- $z = \{x\}$. This is in conflict with 4: as we would have $\neg\{\langle x, x \rangle\}; \{\langle x, x \rangle\} \neq 0$.
- $z = 1$. This is again in conflict with 4: for the same reason.

Case 2: $x \neq y$. Here are the options for z:

- $z = \{x\}$. This is in conflict with 4.
- $z = 0$. This can be disposed of by the same argument as above.
- $z = 1 - \{x\}$ is our intended choice.

- $z = \{y\}$. If x, y are the only objects, then this outcome falls under the previous case. Otherwise, we know, by (4), that x cannot occur in the outcome set. Now, suppose that $\neg(\{\langle x, y\rangle\}) = (\{\langle y, y\rangle\})$. Then, $\neg\neg(\{\langle x, y\rangle\}) = \neg(\{\langle y, y\rangle\}) = 1 - \{y\}$ (cf. case 1). By 3, then, $1 - \{y\} \leq (\{\langle x, y\rangle\}); 1$, which is not possible, since the domain contains at least one $z \neq x, y$.

- $z = 1 - \{x, y\}$. If x, y are the only objects, then this outcome falls under the case $z = 0$. Otherwise, our case amounts to assuming that $\neg(\{\langle x, y\rangle\}) = \mathrm{Id} - (\{\langle x, x\rangle, \langle y, y\rangle\})$. Then, $\neg(\{\langle x, y\rangle\}) = \cup_{z \neq x,y}(\{\langle z, z\rangle\})$. As a consequence, by (2), $\neg\neg(\{\langle x, y\rangle\}) = \neg \cup_{z \neq x,y} (\{\langle z, z\rangle\}) = \cap_{z \neq x,y} \neg(\{\langle z, z\rangle\})$ which equals $\{\langle x, x\rangle, \langle y, y\rangle\}$. But this is again in conflict with 3.

■

This result appears to be the best possible, in that all conditions stated are necessary. For instance, without imposing (4), we could satisfy all other conditions simultaneously via a 'non-standard negation', namely '$Id - \pi$'.

Unique definability results do not necessarily supply complete axiomatizations. nevertheless, it makes sense to try and derive other important properties of $\neg$ from the above set.

Fact 8 *2 and 3 imply that* $\neg\pi \wedge \pi = 0$.

Proof: By lemma 4.6 $\neg\pi \leq Id$. From this we have, using valid principles from Relational Algebra: $\neg\pi \wedge \pi = \neg\pi \wedge \pi \wedge Id = (\neg\pi \wedge \pi) \wedge (\pi \wedge Id) = (\neg\pi \wedge Id); (\pi \wedge Id) \leq \neg\pi; \pi = 0$. ■

In fact, one may observe that all non-validities for negation that we found in relational algebra seem to be refutable even within the special domain of DPL. This, and similar experiences, motivate the following more general conjecture:

Conjecture 9 *Universal validity in DPL is complete with respect to all validities in Relational Set Algebra.*

4.7 Inverse Logic II: UL Negation

We continue with inverse logic for UL. In particular, we shall isolate what may be called a 'progressive' fragment of update logic. But first we introduce some basic notions of independent interest.

Recall that 'dynamic updates' and 'static tests' were the basic ingredients of update semantics as explained in section 3.2. Here is a precise definition:

Definition 20 (tests) *A UL program π is a* **test** *iff for all information states W (sets of states):*

$$[\![\pi]\!]W = W \quad or \quad [\![\pi]\!]W = \emptyset.$$

By contrast, 'pure updates' are the following state-continuous programs:

Definition 21 (pure updates) *A UL program π is a* **pure update** *if for all states W:*

$$[\![\pi]\!]W = \cup_{w \in W} [\![\pi]\!]\{w\}$$

Alternatively, this is the 'state continuity' of section 4.4. Together with Eliminativity for UL programs, this even says that, for all information states W, $[\![\pi]\!]W$ is just $W \cap P$, for some fixed set P (cf. [5]).

Interestingly, these two key properties can be characterized by the way the affect the sequential composition of dynamic operators:

Proposition 10 *A UL program π is a test iff for all UL programs π':*

$$[\![\pi; \pi']\!] = [\![\pi]\!] \cap [\![\pi']\!].$$

Proof It is easy to check that tests satisfy this equation. As for the opposite direction, suppose π is not a test, i.e., for some W, $[\![\pi]\!]W \neq \emptyset$ and $[\![\pi]\!]W \neq W$. Then, take the program '$W - \pi$', and consider: $\pi^* = \Diamond(W - \pi); \pi$. Obviously, $[\![\pi; \pi^*]\!]W = \emptyset \neq [\![\pi]\!]W = [\![\pi]\!]W \cap [\![\pi^*]\!]W$. ■

On the other hand, we have:

Proposition 11 *A UL program π is a pure update iff for all UL programs π':*

$$[\![\pi';\pi]\!] = [\![\pi']\!] \cap [\![\pi]\!] \ .$$

Proof It is easy to check that pure updates satisfy this equation. Next, suppose that the equation holds. Then, for all π' and all information states W: $[\![\pi';\pi]\!]W = [\![\pi']\!]W \cap [\![\pi]\!]W$. Consider then the total initial information state 1, and some program π_W such that $[\![\pi_W]\!]1 = W$. Then, $[\![\pi]\!]W = [\![\pi]\!]([\![\pi_W]\!]1) = [\![\pi_W;\pi]\!]1 = [\![\pi]\!]1 \cap [\![\pi_W]\!]W = [\![\pi]\!] \cap 1$. ■

Let us now turn to progressive forms in UL. First, observe that:

Proposition 12 *Tests are progressive.*

Proof: If π is a test, there are two cases, given a state W. First, suppose $[\![\pi]\!]W = \emptyset$. Then $[\![\pi]\!](W - [\![\pi]\!]W) = [\![\pi]\!]W = \emptyset$. Next, if $[\![\pi]\!]W = W$, then $[\![\pi]\!](W - [\![\pi]\!]W) = [\![\pi]\!]\emptyset = \emptyset$. (We use the eliminativity of update programs here.) ■

Proposition 13 *Pure updates are progressive.*

Proof: If π is state-continuous, then, for any information state W, $[\![\neg\pi;\pi]\!]W = (W - (W \cap P)) \cap P = \emptyset$ (where P is the fixed characteristic set for π). ■

Now, note the following facts:

Fact 14 *Tests are closed under the operations* $\neg$, $\cup$, ;

Fact 15 *Pure updates are closed under the operations* $\neg$, $\cup$, ;

Thus, the basic 'blocks' for test programs may be thought of as modalities closed under the above operations, and something similar holds for pure updates, starting from atomic propositions. Now we can characterize a large class of progressive UL programs, using the following observation:

Proposition 16 *For all progressive UL programs* π_1 *and all tests* π_2, *the program* $\pi_1;\pi_2$ *is progressive.*

Proof: Given an information state W, $[\![\neg(\pi_1;\pi_2);(\pi_1;\pi_2)]\!]W = [\![\pi_2]\!]([\![\pi_1]\!](W - [\![\pi_2]\!]([\![\pi_1]\!]W)))$. Now, since π_2 is a test, there are only two possible cases:
Case 1: $[\![\pi_2]\!]([\![\pi_1]\!]W) = \emptyset$. Then: $[\![\pi_2]\!]([\![\pi_1]\!](W - [\![\pi_2]\!]([\![\pi_1]\!]W))) = [\![\pi_2]\!]([\![\pi_1]\!]W) = \emptyset$.
Case 2: $[\![\pi_2]\!]([\![\pi_1]\!]W) = [\![\pi_1]\!]W$. Then: $[\![\pi_2]\!]([\![\pi_1]\!](W - [\![\pi_2]\!]([\![\pi_1]\!]W))) = [\![\pi_2]\!]([\![\pi_1]\!](W - [\![\pi_1]\!]W)) = [\![\pi_2]\!]\emptyset = \emptyset$, since π_1 is progressive. ■

Corollary 17 *The* ;*-free fragment of UL is progressive.*

Proof: Obvious, from proposition 4.7 and facts 14 and 15. ■

Corollary 18 *If* π *has the form:* $\pi_1;\ldots;\pi_n;\pi_{n+1};\ldots;\pi_m$, *where* π_i *is a pure update for* $1 \leq i \leq n$ *and* π_i *is a test for* $n < i \leq m$, *then it is progressive.*

Proof: Immediate from proposition 4.7 and facts 14 and 15. ■

We must leave the issue of the dynamic behaviour of more elaborate S5-style normal forms in update logic open here.

4.8 A New Dynamic Operator: Local Truth

Our dynamic systems so far suggest a uniform strategy for interpreting dynamic negation: their semantic clauses for $\neg\pi$ basically curtail the input state. Let us call the eliminated part, for an input state W and a program π, the 'local truth' for π in W. We can select this 'local truth' part via an operation $\top$ on programs, which supports the following scheme of definition:

$$[\![\neg\pi]\!]W = W - [\![\top\pi]\!]W.$$

The new operator turns out to have interesting peculiarities. Here are the semantic clauses for $\top$ in TV, UL, GSV and TKV, extracted from the respective clauses for negation:

TV $[\![\top\pi]\!]W = \{w \in W \mid [\![\pi]\!]\{w\} \neq \emptyset\}$
UL $[\![\top\pi]\!]W = [\![\pi]\!]W$
GSV $[\![\top\pi]\!]W = \{w \in W \mid \exists w' : w \leq w' \text{ and } w' \in [\![\pi]\!]W\}$
TKV $[\![\top\pi]\!]W = \{w \in W \mid \exists w' : w \succ_\pi w' \text{ and } w' \in [\![\pi]\!]W\}$

But note that there are other points in the above where the same operator is implicitely at work. For instance, all dynamic modalities so far boil down to the following [2]:

$$[\![\Diamond\pi]\!]W = \begin{cases} W & \text{if there is a } w' \in W \text{ such that } w' \in [\![\top\pi]\!]W, \\ \emptyset & \text{otherwise.} \end{cases}$$

Moreover, the notion also comes up in certain kinds of dynamic inference (cf. [21], [34]), in particular:

'Process all premises successively, then see if the conclusion can run from the resulting state',

which amounts in practice to the following:

$\pi_1 \models \pi_2$ iff, for all states W, $[\![\top\pi_2]\!]([\![\pi_1]\!]W) = [\![\pi_1]\!]W$.

Thus, it becomes of interest to pursue some characteristic semantic properties of the local truth operator $\top$ by itself.

Definition 22 (Weak Normality) *The operator $\top$ is* **weakly normal** *iff for all programs π, for all information states W, the following conditions hold:*

- $[\![\pi]\!]([\![\top\pi]\!]W) = [\![\pi]\!]W$
- $[\![\pi]\!](W - [\![\top\pi]\!]W) = \emptyset$.

Definition 23 (Normality) *The operator $\top$ is* **normal** *iff for all programs π, for all information states W, the following conditions hold:*

- *$\top$ is weakly normal*
- $[\![\top\pi]\!]W = \cap\{W' \subseteq W \mid [\![\pi]\!]W' = [\![\pi]\!]S \text{ and } [\![\pi]\!](W - W') = \emptyset\}$.

Here is a stronger version of the property above:

Definition 24 (Strong Normality) *The operator $\top$ is* **strongly normal** *iff for all programs π and all information states W, the following conditions hold:*

- *$\top$ is normal*
- $[\![\top\pi]\!]W \subseteq W'$ *only if* $[\![\pi]\!]W' = [\![\pi]\!]W$ *and* $[\![\pi]\!](W - W') = \emptyset$.

Intuitively, these degrees of 'normality' tell us to which extent a system of dynamic semantics approximates, modulo 'local truth', a boolean structure. More concretely, if $\top$ is normal or strongly normal and '$\emptyset$-preserving' (see definition below), then it defines a complement-like negation on the class of 'localized' programs, at least for a large class of programs, which we will call '$\emptyset$-continuous' (see definition below).

Definition 25 ($\emptyset$-preservation) *$\top$ is* **$\emptyset$-preserving** *if for all states W and programs π:*

if $[\![\pi]\!]W = \emptyset$ *then* $[\![\top\pi]\!]W = \emptyset$.

Definition 26 ($\emptyset$-continuity) *π is* **$\emptyset$-continuous** *if for all W, W':*

if $[\![\pi]\!]W = \emptyset \wedge [\![\pi]\!]W' = \emptyset$ *then* $[\![\pi]\!](W \cup W') = \emptyset$.

Proposition 19 *If $\top$ is $\emptyset$-preserving and strongly normal (normal), then the following holds about the matching negation (defined by the clause $[\![\neg\pi]\!]W = W - [\![\top\pi]\!]W$), for all $\emptyset$-continuous programs π and all information states W:*

[2] which gets parametrized in TVK

$$[\![\top\neg\pi]\!]W = W - [\![\top\pi]\!]W = [\![\neg\pi]\!]W.$$

Proof: Suppose $\top$ is normal and $\emptyset$-preserving. We show that, for all π and all information states W: $[\![\top\neg\pi]\!]W = W - [\![\top\pi]\!]W$. Let $[\![\top\pi]\!]W = W'$. Obviously:
$[\![\neg\pi]\!]W' = W' - [\![\top\pi]\!]W'$. But $\top$ is Idempotent (this follows from normality plus $\emptyset$-continuity). Hence we have:
$[\![\top\pi]\!]W' = W'$ and $[\![\neg\pi]\!]W' = \emptyset$.
Moreover, $[\![\neg\pi]\!]W - W' = (W - W') - [\![\top\pi]\!](W - W')$. But $[\![\top\pi]\!](W - W') = \emptyset$, because $[\![\pi]\!](W - W') = \emptyset$ and $\top$ is $\emptyset$-preserving. Then: $[\![\neg\pi]\!](W - W') = (W - W')$ namely $[\![\neg\pi]\!](W - [\![\top\pi]\!]W) = W - [\![\top\pi]\!]W = [\![\neg\pi]\!]W$
From which it follows that $W - [\![\top\pi]\!]W = [\![\top\neg]\!]W$. ∎

The following facts confirm, in a sense, how ;-free fragments of the systems above stay close to classical negation:

Proposition 20 *The ;-free fragment of TV is strongly normal.*

Proof: Easy induction on ;-free formulas. ∎

Proposition 21 *The ;-free fragment of UL is normal.*

Proof: Easy induction on ;-free formulas. ∎

Investigating properties of $\top$ may also produce new points of view on negation. For example, it is easy to prove that:

Proposition 22 *If $\top$ is normal and 1-preserving, then its matching $\neg$ satisfies:*

$$[\![\pi;\neg\pi]\!]W = [\![\neg\pi;\pi]\!]W = \emptyset \quad \textit{iff} \quad [\![\top\pi]\!]([\![\pi]\!]W) = [\![\pi]\!]W.$$

Finally we shortly discuss a uniform strategy for 'safely' constructing dynamic systems. Here is the simple idea. A normal 'local truth' operator is logical (i.e. permutation invariant). This is so because a normal $\top$ gives, for an input set W, the intersection of all the subsets of W which behave in the required way; in other words, a normal $\top$ has a set-theoretic definition, which is enough in order to guarantee its logicality (cf. 1). Then, from a logical 'local truth' operator we can obtain a logical negation and logical modalities:

Proposition 23 *If $\top$ is permutation invariant then the matching $\neg$ is permutation invariant.*

Proof: Suppose $\top$ is permutation invariant. Then, for any program π and state W:
$[\![\top(\alpha;\pi;\alpha^{-1})]\!]W = [\![\alpha;\top\pi;\alpha^{-1}]\!]W$. The argument runs as follows:
$[\![\neg(\alpha;\pi;\alpha^{-1})]\!]W =$
$[\![\alpha^{-1}]\!]([\![\neg\pi]\!]([\![\alpha]\!]W)) =$
$[\![\alpha^{-1}]\!]([\![\alpha]\!]W - [\![\top\pi]\!]([\![\alpha]\!]W)) =$
$W - [\![\alpha;\top\pi;\alpha^{-1}]\!]W =$
$W - [\![\top(\alpha;\pi;\alpha^{-1})]\!]W =$
$[\![\neg(\alpha;\pi;\alpha^{-1})]\!]W$ ∎

Proposition 24 *If $\top$ is permutation invariant then the matching modality is permutation invariant.*

Proof: Analogous. ∎

The general reason here and above is in fact as follows: operations which are set-theoretically definable from logical ones are themselves logical.

4.9 Back to Classical Logic

In this section we investigate one more type of dynamic operator, being projections taking dynamic propositions to classical ones. One such operator DPL local truth $\top$, assigning to each program π its 'domain':

$$\pi \longmapsto \{w \mid \pi(\{w\}) \neq \emptyset\}$$

We now move to eliminative update programs π, where a corresponding operation $\top^*$ would be:

$$\pi \longmapsto \{w \mid \pi(\{w\}) = \{w\}\}$$

First note the following:

Fact 25 $\top^*$ *is permutation invariant.*

(This follows from its set-theoretic definition; cf. 1). Consider now the Boolean algebra of all eliminative dynamic propositions, given a set of states S. Note that this is not the full function space $\wp(S)^{\wp(S)}$, but a relativized space obtained by taking only all functions $\pi \leq Id$ (e.g., $\neg\pi$ in UL is "$Id - \pi$"). Then, the following key behaviour may be seen by some simple calculation:

Fact 26 $\top^*$ *is a boolean homomorphism from eliminative updates to classical propositions:*

- $\top^*(\neg\pi) = -\top^*(\pi)$
- $\top^*(\cup_i \pi_i) = \cup_i \top^*(\pi_i)$

Thus, $\top^*$ is a logical projection respecting Boolean structure. Our main result is that, conversely, only two functions have this behaviour:

Theorem 27 *There are only two logical Boolean homomorphisms from eliminative dynamic propositions to classical ones.*

Proof: Suppose that F is such a logical Boolean homomorphism. The action of F is completely determined by its values on the Boolean atoms in the 'update algebra', being all functions '$\alpha_{W,w}$', with $w \in W$, such that:

- $\alpha_{W,w}(X) = \begin{cases} \{w\} & \text{if } X = W \\ \emptyset & \text{otherwise} \end{cases}$

This is so because $F(\pi) = \cup_{\alpha \leq \pi} F(\alpha)$, for all π and all atoms α (by the second homomorphism clause). We now need a subclaim:

Proposition 28 *The values $F(\alpha)$ for all atoms form a complete partition of S*

Proof: Distinct atoms have $\alpha_1 \wedge \alpha_2 = 0$, whence $F(\alpha_1 \wedge \alpha_2) = F(\alpha_1) \wedge F(\alpha_2) = \emptyset$, so they are disjoint. Moreover, $\bigvee_i \alpha_i = 1$, whence $F(1) = W = \cup_i F(\alpha_i)$. ∎

Now, define a map F^* from S to such atoms, by setting $F^*(s)$ as the unique atom α such that $s \in F(\alpha)$. Note that $F(\alpha) = F^{*-1}(\{\alpha\})$. Moreover, we can show that:

Fact 29 F^* *is logical.*

Now, our classification problem is easier. It is enough to prove the following:

Proposition 30 *There are only two logical maps F^* sending states to atoms in the update algebra.*

Proof: We have $s \overset{F^*}{\longmapsto} \alpha_{W,w}$, with $w \in U$. If F^* is logical, then the familiar reasoning about permutations tells us that w can only be s itself; while U could be in principle one of the four sets $\{s\}, W - \{s\}, \emptyset, W$. But the requirement '$w \in U$' rules out two possibilities, and we only have:

$$F_1^*(s) = \alpha_{\{s\},s}$$
$$F_2^*(s) = \alpha_{S,s}$$

■

Now we can calculate backwards, and see which maps F are induced by these two functions:

- $F_1(\pi) = \{w \mid \exists \alpha \leq \pi : w \in F_1(\alpha)\} =$
  $\{w \mid \exists W, w : \alpha = \alpha_{W,w} \wedge w \in \pi(W) \wedge \alpha = \alpha_{\{w\},w}\} =$
  $\{w \mid \pi\{w\} = \{w\}\}$
- $F_2(\pi) = \{w \mid w \in \pi(S)\} = \pi S$

Conversely, it is easy to check that both of these are indeed logical Boolean homomorphisms. ■

5 Conclusion

To conclude, we briefly point out some possible advantages of the 'variational' approach we have been proposing.

First, it is very fine-grained. Parametrizing knowledge updating (in Kripkean Variations) and information processing (in general Tarskian Variations) provides us with a very flexible set-up, with great expressive potential. For instance, we expect that parametrized modalities can efficiently formalize and clarify many puzzles about 'individuals across different possible worlds', rigid designation, de re vs de dicto modalization, referential vs descriptive naming, and other well-known questions (cf., for instance, [28]). We hope to turn to these philosophical repercussions in subsequent publications. Moreover, further possible semantic levels of analysis can be distinguished inside our dynamic logic. For example, one can further parametrize the notion of truth in order to distinguish between cognitive and physical change, as suggested in [38].

On the other hand, we have seen that comparing and mixing different variations can suggest interesting questions and show hidden peculiarities of concrete systems. In our particular case, the different behaviour of Kripkean and Tarskian variations has brought to the fore many issues concerning dynamic negation, which we have investigated using various alternative points of view from the literature. The same line of thinking could be applied to other delicate dynamic notions, such as dynamic consequence.

Acknowledgements

We would like to thank Jan van Eijck for his helpful comments.

References

[1] J.C.M. Baeten and W. P. Weijland (1990), *Process Algebra*, Cambridge Tracts in Theoretical Computer Science 18, Cambridge University Press, Cambridge.

[2] J. Barwise and J. Perry (1983), *Situations and Attitudes*, Bradford Books, MIT Press, Cambridge, Massachussets.

[3] J. van Benthem (1982), *Modal Logic and Classical Logic*, Bibliopolis, Napoli.

[4] J. van Benthem (1986), *Essays in Logical Semantics*, Reidel, Dordrecth.

[5] J. van Benthem (1989), *Semantic Parallels in Natural Languages and Computation*, in: H-D. Ebbinghaus et al. (eds.), *Logic Colloquium. Granada 1987*, North-Holland, Amsterdam.

[6] J. van Benthem (1989), *Logical Constants across Varying Types*, in: Notre Dame Journal of Formal Logic, 3.

[7] J. van Benthem (1991), *Logic and the Flow of Information*, in: D. Prawitz, B. Skyrms and D. Westerståhl (eds.), Proceedings of the 9th International Conference of Logic, Methodology and Philosophy of Science, Uppsala 1991, Elsevier Science Publishers, Dordrecht.

[8] J. van Benthem (1991), *General Dynamics*, in: Theoretical Linguistics, 17 (special issue on 'Complexity in Natural Language').

[9] J. van Benthem (1991), *Language in Action*, Elsevier Science Publishers, Amsterdam.

[10] J. van Benthem (1993), *Modal State Semantics*, manuscript, ILLC, Amsterdam.

[11] J. van Benthem (1993), *Program Constructions that Are Safe for Bisimulation*, Report CSLI-93-179 of the Centre for the Study of Language and Information, Stanford.

[12] J. van Benthem, J. van Eijck and A. Frolova (1993), *Changing Preferences*, Report CS-R9310 of the Centre for Mathematics and Computer Science (CWI), Amsterdam.

[13] J. van Benthem, R. Muskens and A. Visser (1993), *Dynamics*, to appear in: J. van Benthem and A. Ter Meulen (eds.), *Handbook of Logic and Language*, Elsevier Science Publishers, Amsterdam.

[14] C. Boutilier and M. Goldszmidt (1993), *Revision by Conditional Beliefs*, in: Proceedings 11th National Conference on Artificial Intelligence, Washington D.C.

[15] G. Cepparello (1993), *Tarskian Variations, a complete system*, manuscript, Scuola Normale Superiore, Pisa.

[16] P. Dekker (1993), *Existential Disclosure*, in: Linguistics and Philosophy, 16.

[17] J. van Eijck and G. Cepparello (1992), *Dynamic Modal Predicate Logic*, to appear in: M. Kanazawa and C.J. Piñon (eds.), *Dynamics, Polarity and Quantification*, CSLI, Stanford.

[18] J. van Eijck (1993), *The Dynamics of Theory Extension*, to appear in: Proceedings of the 9th Amsterdam Colloquium, Amsterdam.

[19] P. Gärdenfors (1988), *Knowledge in flux. Modelling the Dynamics of Epistemic States*, Bradford Books, MIT Press, Cambridge, Massachussets.

[20] R. Goldblatt (1987), *Logics of Time and Computation*, CSLI Lecture Notes, 7, Stanford.

[21] J. Groenendijk and M. Stokhof (1991), *Dynamic Predicate Logic*, in: Linguistics and Philosophy, 14.

[22] J. Groenendijk and M. Stokhof (1991), *Two theories of Dynamic Semantics*, in: J. van Eijck (ed.), *Logics in AI*, Proceedings of the European Workshop JELIA 90, Springer-Verlag.

[23] J. Groenendijk, M. Stokhof and F. Veltman (1993), *Coreference and Modality*, handout for the 'Workshop on Dynamic Semantics', Fifth European Summer School in Logic, Language and Information, August 1993, Lisbon.

[24] D. Harel (1994), *Dynamic Logic*, in: D. Gabbay and F. Guenthner (eds.), *Handbook of Philosophical Logic* vol II, Reidel, Dordrecht.

[25] I. Heim (1982), *The Semantics of Definite and Indefinite Noun Phrases*, Department of Linguistics, University of Massachussets, Amherst.

[26] J. Jaspars (1994), *Calculi for Constructive Communication* (provisional title), Dissertation, Institute for Language and Knowledge Technology, Katholieke Universiteit Brabant, Tilburg.

[27] H. Kamp (1984), *A Theory of Truth and Semantic Representation*, in: J. Groenendijk et al. (eds.), *Truth, Interpretation and Information*, Foris, Dordrecht.

[28] S. Kripke (1972), *Naming and Necessity*, in: Harman and Davidson (eds.), *Semantics of Natural Language*, Reidel Publishing Co., Dordrecht.

[29] D. Lewis (1975), *Adverbs of Quantification*, in: E. Keenan (ed.), *Formal Semantics*, Cambridge University Press.

[30] I. Németi (1992), *Decidability of weakened versions of first order logic*, in: Proceedings of the Applied Logic Conference 'Logic at Work', Amsterdam.

[31] M. de Rijke (1992), *A System of Dynamic Modal Logic*, Report LP-92-08 of the Institute for Logic, Language and Information, University of Amsterdam.

[32] M. de Rijke (1993), *Extended Modal Logic*, Dissertation, Institute for Language, Logic and Computation, Amsterdam.

[33] Stalnaker (1972), *Pragmatics*, in: D. Davidson and G. Harman (eds.), *Semantics of Natural Language*, Reidel, Dordrecht.

[34] F. Veltman (1989), *Defaults in Update Semantics*, in: H. Kamp (ed.), *Conditionals Defaults and Belief Revision*, Edinburgh.

[35] Y. Venema (1992), *Many-dimensional Modal Logic*, Dissertation, Institute for Language, Logic and Information, Amsterdam.

[36] K. Vermeulen (1993), *Merging without Mistery*, to appear in: Journal of Philosophical Logic.

[37] D. Westerståhl (1989), *Quantifiers in Formal and Natural Languages*, in: D. Gabbay and F. Guenthner (eds.), *Handbook of Philosophical Logic* vol IV, Reidel, Dordrecht.

[38] E. Zalta (1993), *A Philosophical Conception of Propositional Modal Logic*, to appear in: C. Hill (ed.), *Philosophical Topics.*