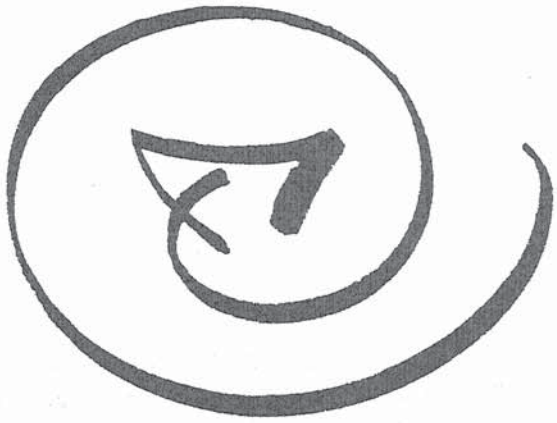
Giovanna Cepparello

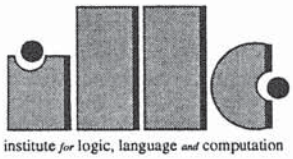
Studies in Dynamic Logic



EDIZIONI ETS

Studies in Dynamic Logic

ILLC Dissertation Series 1995-10



For further information about ILLC-publications, please contact

Institute for Logic, Language and Computation
Universiteit van Amsterdam
Plantage Muidergracht 24
1018 TV Amsterdam
phone: +31-20-5256090
fax: +31-20-5255101
e-mail: illc@wins.uva.nl

Studies in Dynamic Logic

Tesi di Perfezionamento in Filosofia

Scuola Normale Superiore
Classe di Lettere e Filosofia

di

Giovanna Cepparello

Relatori: Prof. J.van Benthem
Prof. M.L. Dalla Chiara
Prof. J. van Eijck
Prof. E. Moriconi

Printed and bound by ETS, Pisa.

ISBN: 90 - 74795 - 30 - 7

al mio babbo

Contents

Acknowledgments

First, I want to thank the persons who made my Ph.D. experience so enjoyable and rich.

On the dutch side, Johan van Benthem was always surprising. Surprising for his deep competence in Logic and in Philosophy, for the enthusiasm towards intellectual work he can communicate. Working under his guidance was extremely instructive, and it was also fun.

Jan van Eijck also played a key role inside my Ph.D. project. He helped me to overcome my fear for paper-writing; moreover, he always contributed to the advancement of my work with witty criticisms and useful hints.

In Pisa, I was supported by Mauro Mariani and Enrico Moriconi. A conversation with Mauro is always worthwile, since his sound knowledge of many cultural areas, including Philosopy and Logic, gives unsuspected depth to the simplest idea.

Enrico helped me a lot. He always read my manuscripts and gave me good suggestions. Besides, and this is a very important contribute, he constantly recalled me in many ways that there is more to Logic than the small part of it I was busy with.

Finally, last but not least, I want to thank prof. Maria Luisa Dalla Chiara. She was extremely cooperative, and I am really grateful to her for accepting to be one of my promotors.

Many other people from the academic world gave me their 'moral' and practical support in the last period, showing a great kindness. I sincerely thank them all.

The Scuola Normale Superiore provided me with many intellectual stimuli and with a pleasant cultural and 'social' environment; its financial support was always generous.

I also want to express my gratitude to the Centre for Mathematics and Computer Science in Amsterdam. It was always an ideal refuge where I could get new ideas, interact with many clever guys and work hard. Concerning my staying in Holland, there is a group of friends - inside and outside my work environment - that made them incredibly pleasant and useful. I want to say thank you to all of them.

Finally, I owe quite a lot of more personal thanks. To my good friends; to my wonderful family; and, most of all, to my mother and to my husband. Thanks!

Pisa
July, 1995.

Giovanna Cepparello

Chapter 1

Introduction

This introductory chapter has two tasks. First, we want to explain the general motivations that underlie the present work. Sections 1 and 2 will cover this. Second, we will give a brief summary of the contents of the thesis, while also sketching the basic technical concepts and relative notations that we will use on the way. This will be done in section 3.

1.1 From Truth to Action

In this section we will try to highlight a general tendency that has emerged in the last forty years (and more dramatically in the last decade) through a budget of disciplines that are concerned with the study of (human) intelligent skills, like reasoning, speaking, commmunicating etc...

Before discussing the single cases, let us sketch the core of this growing tendency: intelligent skills can be better understood (or codified) by using *dynamic* concepts as basic units of analysis. What do we mean by *dynamic* concepts here? We do not want to give a precise definition, rather we point at a number of dynamic concepts, opposed to some *static* notions; this will hopefully light the right intuition, which will get shaped up within the course of the dissertation. Therefore, cognition (= acquisition of knowledge) is dynamic, in contrast with actual knowledge. In the same way, actions, procedures, rules, transitions have a dynamic character, while truth, declarations, assertions, states descriptions are clearly static.

To sum up our first claim: we want to argue that many ideas that we can classify as 'dynamic' pervade a range of disciplines that study Human Intelligence in its various forms, where 'study' can mean represent, formalise or more generally analyse. In doing this, we hope we will underline a cultural trend

that constitutes the grounds of our work. We will start with pointing at some far-off, archaic traces of this dynamic trend, coming from Artificial Intelligence and Psychology. Afterwards we will focus on some more concrete examples, some of them very up-to-date, from Linguistics and Philosophy of Language.

Let us start with Artificial Intelligence. Computers are surely one of the hallmarks of this century, and Artificial Intelligence is maybe the mainspring of this general dynamic cultural shift: developing algorithms for solving intelligent problems inevitably demonstrates how efficiently these problems can be *represented* in terms of rules, instructions, in one word actions, rather than as sequences of plain truth conditional inferences. This matches with the general idea underlying this research area (and already springing up since the dawning of AI): the human reasoning or, in a more informatic fashion, the 'human information processing' can be seen as a more complex and sophisticated version of the automatic information processing. Minds and computers have something in common, and (fragments of) minds' activities can be represented in form of computers' algorithms, namely in dynamic terms.

Concerning Psychology, and in particular that branch of Psychology that deals with acquisition and elaboration of knowledge, we must immediately stress that the influence of AI on it was very strong since the sixties, and it actually contributed to the birth of so-called Cognitive Psychology (see for a survey of its main ideas [61]), which still constitutes the 'mainstream' in the area. Until the end of the fifties, the behaviourist trend was in fact prevailing; according to the behaviourist doctrine (cf. [74]), the task of psychology would consist in measuring, so to speak, human answers to stimuli. In other words, analysing, e.g., a given process of problem solving, should amount to measuring its manifest 'extension' at the behavioural level (the problem will be eventually decomposed in a set of stimuli, and the solution, possibly in several steps, will provide the answers). Against this method, the cognitive approach suggests to investigate precisely what happens between a stimulus and an answer. Around the end of the fifties, psychologists started to take seriously the above metaphor mind/computer, and consequently to study how human beings do acquire and elaborate knowledge in terms of computational models: between a stimulus and an answer there are *stages of the processing*, and successive *instructions* to perform different *operations*. In other words, between a stimulus and an answer a sort of *program* is performed by the human processor. A good example of primitive Cognitive Psychology is the famous unit TOTE (Test - Operate - Test - Exit), suggested in [59] as a suitable tool for codifying intelligent processes (the informatic flavour of this notion should be immediately detectable).

We have thus roughly sketched some of the dynamic ideas pervading both AI and Cognitive Psychology. We should say however that these disciplines are totally outside the scope of the present dissertation; rather, we will often

refer in the following chapters to both Linguistics and Philosophy of Language, since they constitute in a sense the cultural environment in which many of the dynamic notions quoted so far are at the moment playing a major role and acquiring more sharp contours. Therefore, let us now outline some relevant dynamic ideas coming from Philosophy of Language.

First, we want to mention Wittgenstein's *language games*, where the actional character of language comes to the fore, together with the fundamental link meaning-use (cf. [76]). Moreover, as remarqued in [29], there is David Lewis idea that the process underlying Natural Language understanding is somehow similar to the process of score keeping during a game, where the score board keeps track of the many possible changes in the context (cf. [58]).

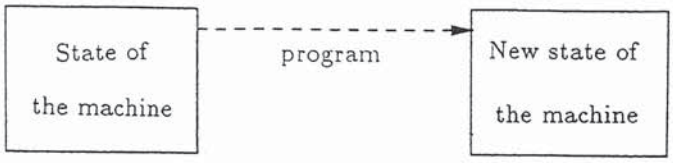
Related in some way to these influential philosophical conceptions, there is a point of view emerging in recent Philosophical Linguistics (maybe the new Philosophy of Language), which we shall refer to as 'the dynamic conception of meaning'. According to this conception, the meaning of a linguistical expression is no longer - phenomenologically - what determines its truth-conditions, but rather it lies 'in the way it changes the (representation of) the information of the interpreter' (cf. [43]). In other words, the meaning of a linguistic bit is now measured in terms of the effects it produces on the intended receptor. If this is so, then linguistic bits are to be read accordingly as rules to change some information stage, or as 'cognitive actions' for moving on the elaboration of knowledge. A well known concrete example of this trend is the interpretation of linguistic *anaphora* it suggests: expressions containing anaphora *mean* - among other things - a change in the instantiation of some variables, that is kept along the discourse (cf. the system of 'Dynamic Predicate Logic' presented in [43], and also [54], [47]). Similarly, the dynamic conception of meaning has affected the more recent literature on presupposition (see e.g. [3], [29], [31]), on temporal reference (see e.g. [24]), on preferences (see e.g. [15]) etc...

As we said, this conception of meaning is now becoming very popular, and it will be often referred to in what follows. Here we want to stress how one can find in it some of the chief ideas that were dominating both in the earliest AI and in Cognitive Psychology; in particular, viewing Natural Language as a set of instructions fits with the above metaphor computer programs/intelligent processes[1].

Therefore, across such different fields as AI, Cognitive Psychology, Philosophy of Language and Linguistics, a common idea has emerged in different forms: Human Intelligence can be fruitfully represented along a dynamic model of some sort. Or, to give a pictorial intuition, human intelligent processes (from

[1]Note that one of the research projects that by no mean fall in within this 'dynamic conception of meaning', run in the last few years, was called 'Structural and Semantic Parallels between Natural Language and Programming Languages'

chess playing to knowledge acquisition to linguistic competence) fall in within some framework of the following form:



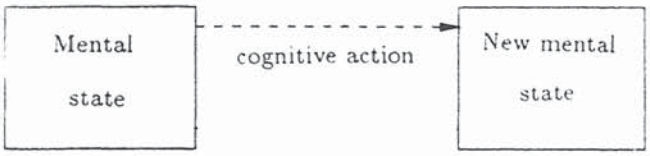
1.2 Dynamic Logic

We can now make a further step toward the specific topic of the present dissertation. We have argued that a cultural shift 'from truth to action' is detectable across many disciplines. In fact, none of these disciplines will constitute our favourite standpoint, so to speak, although we will often refer to both Philosophy of Language and Philosophical Linguistics. Generally speaking, our main concern will consist in discussing, testing and enriching a possible mathematical model for giving a formal account of this dynamic tendency. More specifically, among various possible formal implementations of the dynamic ideas above we will choose the one offered by Dynamic Logic, and do what we said, namely: discuss its potentialities, test it against specific technical problems, and possibly enrich its paradigm when necessary. As we have already implicitly said, other choices would be just as well possible (cf. [13] for an exhaustive discussion); yet, we should stress that our choice is not accidental: Dynamic Logic is now having its 'golden age', and precisely in connection with this general cultural trend.

Therefore, we have two aims in this section. First we want to demonstrate the reasons why Dynamic Logic looks such suitable a choice as a formal paradigm of the dynamic trend, while also giving the reader a rough intuitive notion of how Dynamic Logic looks like. Second, we want to start sketching the line of analysis that we will pursue. As we have already said, a more detailed description of the content of this dissertation will be provided in the next section.

We have seen how influential was the informatic metaphor on the dynamic trend as described in section 1. The idea that knowledge is *processed* and *elaborated* following *instructions* of some sort and along *states* of some sort, clearly has a computational flavour, and it represents in a sense the dynamic core shared by some doctrines in AI, Cognitive Psychology, Linguistics and Philosophy of Language. And, because of this computational flavour pervading the dynamic trend, Dynamic Logic represents a very natural choice when searching for a mathematical paradigm. For Dynamic Logic (cf. [64]) was originally designed

as the Logic for 'founding' computer programs; more precisely, the original task of Dynamic Logic was talking and proving properties of computer programs. The intended models that the dynamic-logical syntax was supposed to describe were just a computational instantiation of the picture above:



Next section will provide a more precise technical definition; for the moment, it is enough to stress that Dynamic Logic is meant to describe the 'flow of information' that a program produces when run. Thus, there will be *formulas* describing the 'states of the machine', and *programs* describing the transitions among them. But there is more to it than that. Since the very moment of its birth, it was immediately clear that the potentialities of Dynamic Logic could go well beyond its literal original task: if the intended model was the computational one, Dynamic Logic was also acknowledged more generally as 'a system of reasoning about action', where the 'action' can be carried out by a computer running a program but also by a human processor. 'Reasoning about action' can mean representing and modeling a wide range of activities, from the automatic to the human intelligent ones. And this generality enforces the thesis that Dynamic Logic is the right mathematical model for the 'dynamic trend', which takes so many different forms.

Summing up, our task was to find a suitable formal paradigm for what we have called the 'dynamic trend', and we opted for Dynamic Logic, the reasons being as follows. The *metaphor* of programs run on a computer constitutes one of the common grounds in the 'dynamic trend'. And Dynamic Logic has been created just as a logical paradigm - natural development of the dominating Hoare Calculus (cf. [49]) - for reasoning about computer programs. Moreover, the dynamic-logical framework can go beyond the computational boundaries, and fit the different instantiations that the informatic metaphor has taken within the 'dynamic trend'.

We now want to conclude with saying a few words on the general line of this work. Although since its beginning it was clear that Dynamic Logic could be more than the 'logic of computer programs', its original design as well as most of the further development heavily reflects its informatic-oriented origin. More precisely, the 'programs' in a standard presentation of Dynamic Logic mostly come from some regular programming language, or, which is the same, from a Hoare Calculus of some sort. But, given the current revival of Dynamic Logic outside Computer Science, a question naturally arises on the possibility of al-

ternative choices in creating actual dynamic systems. If Dynamic Logic has to be the paradigm of a general cultural trend, then the issue of proving meta-properties of actual computer programs becomes marginal. Rather, given the 'core' of Dynamic Logic (a two typed-syntax, including formulas and programs, with a matching states-transitions semantics), the art is now implementing on this frame logics for talking about different actions. Our work lies then in this 'new deal' of Dynamic Logic, that aims to build a broad, many-aimed family of Dynamic Logics along a layered and disciplined strategy.

1.3 Our themes

In this section we want to briefly explain the 'geography' of the whole dissertation (namely which topics we will consider in which chapters). In doing this, we will also introduce - in an intuitive fashion - the very basic technical notions that we will be using on the way while also hinting at some of the concrete problems that we will face.

1.3.1 The format

First, let us single out clearly the general conception of Dynamic Logic that we assume (not a new conception, only a general one extrapolated from the literature!).

1.3.1. DEFINITION. [Dynamic Language] A *Dynamic Language* is a language with two types of expressions: formulas (properties of states, of the type (state, truth-value)), and programs (relations among states, of the type (state, (state, truth-value))).

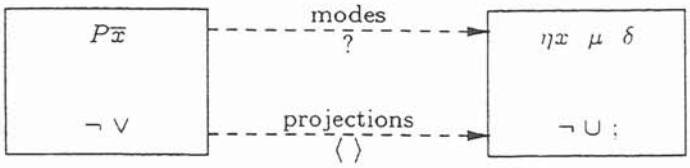
1.3.2. DEFINITION. [Dynamic Model] A *Dynamic Model* is a set S of States s with an interpretation function $[\![\ ]\!]$: Programs $\longrightarrow$ $S \times S$ assigning relations to the programs, and another one $[\ ]$: Formulas $\longrightarrow$ S assigning states to the formulas.

A matter of notation. The interpretation of programs will be expressed alternatively in the following - obviously equivalent - manners:

1. $[\![\pi]\!] = \{\langle s, s' \rangle \mid \ldots\}$
2. $[\![\pi]\!]s = \{s' \mid \ldots\}$

We should remark that the 'states' in the definition can be static models of any kind; this thesis will be mainly concerned with first-order models or possible worlds models -states, but other options would be conceivable. Note that intensional states (i.e. possible worlds models) have a different granularity with respect to first-order states, involving a set of states in their turn.

Figure 1.1: The interface static/dynamic



The last point we want to make in this 'basic tour of dynamics' is of a methodological kind. We have defined a *dynamic language* as a language including two types of expressions: formulas and programs. Often within the course of this dissertation we will focus on the programs-fragment of our systems, and leave out the formulas. This we will do only for the sake of simplicity. Yet we reckon that the interface static-level/dynamic-level is very important, and we will make a heavy use of it for proving our completeness theorems (see Appendices A and B.1). This interface, with its 'switching' mechanisms between the two modes, does pervade the best tradition of Dynamic Logic (cf. for instance [9] and [64]), and thinking in terms of static counterpart of a dynamic program (cf. for instance the notion of pre- and post-condition) can throw light on the 'meaning' of it.

After outlining the general format of Dynamic Logic that we shall take as the basis of our work, we can now briefly hint at the issues that we shall successively take into account in the next chapters.

1.3.2 Dynamization

As we have seen in 1.2, there is a strong need to enlarge the dynamic-logical framework outside the boundaries of Computer Science, in order to meet a general cultural demand. Therefore, one of the issues we will tackle within the present dissertation will concern the possibility of creating interesting programs that do not come necessarily from some programming language. One of the policies we will suggest will consist in fact in elaborating a strategy for building dynamic programs out of static systems such as First-order Logic and Modal Logic (cf. Chapter 2).

1.3.3 Kinds of programs: the issue of combination

We now want to make a point that will play a fundamental role in the present work. Namely: interesting distinctions among classes of programs can be made by looking at their behaviour. We will now shortly discuss a few properties of programs that codify their behaviour; we will run into many others in the following chapters, with two main purposes:

1. the intuitive interpretation of them - mainly along the analogy programs = instructions for carrying on the information processing - will help us finding and testing the applications of programs to the analysis of cognition;

2. at a more technical level, codifying the behaviour of programs will be a way of checking their mathematical legitimacy, so to speak, with respect to certain fixed criteria.

Let us then start seeing two simple properties of programs.

First, note that static formulas could be interpreted along programs: the interpretation function $[\![\ ]\!]$ applied to a formula will assign to it nothing but a subset of the Identity function. This gives us the opportunity of giving the following straightforward definitions:

1.3.3. DEFINITION. [Proper programs] A program π is *proper* if it is not the case that for all dynamic models M of its language, $[\![\pi]\!]_M \subseteq Id$, where Id is the Identity function.

1.3.4. DEFINITION. [Test programs] A program π is a *test* if for all dynamic models M of its language, $[\![\pi]\!]_M \subseteq Id$, where Id is the Identity function.

Test programs amount in fact to formulas.

Let us now spell out two more important properties of dynamic programs.

1.3.5. DEFINITION. [Eliminative programs] A program π is *eliminative* if for all dynamic models M of its language, for all states s of M, $[\![\pi]\!]_M(s) \subseteq s$.

1.3.6. DEFINITION. [Continuous programs] A program π is *continuous* if for all dynamic models M of its language, for all states s of M, $[\![\pi]\!]_M \cup_{i\in I} s_i = \cup_{i\in I}[\![\pi]\!]_M s_i$.

Interestingly, it is possible to prove that (see [6]):

1.3.7. PROPOSITION. *If a program π is eliminative and continuous, then for all dynamic models M of its language and all states s of M, $[\![\pi]\!]_M s$ is just $s \cap P$ for some fixed 'information set' P.*

We now want to make our general point: if so many distinctions can be made among classes of programs, a natural question arises on the possibility of handling different kinds of programs within the same formal system. For instance, combining tests with proper programs will amount in fact to allow the interface static/dynamic advocated above; on the other hand, combining non-eliminative programs with non-continuous programs will turn out to be much more problematic, since, as we will see, these two kinds of programs represent two radically different *modes* of processing the information. Chapter 3 will be in fact devoted to this question. A detailed analysis of the issue will be provided, together with three possible solutions recently proposed, and with a comparison among them.

1.3.4 Intermezzo

We now want to briefly discuss how the properties above can be used to 'measure' the distance of a certain dynamic system from a static counterpart.

First, we have seen how eliminative and continuous programs have a totally 'extensional' character. Therefore, the following opposition is commonly accepted

Eliminative *and* Continuous Programs = Classical Programs

vs

Non-eliminative *or* Non-continuous Programs = Dynamic Programs

Instead of this opposition, that certainly has its sound rationale, we prefer to distinguish among 'Static' programs vs 'Dynamic Programs', as follows:

Test Programs = Static Programs

vs

Proper Programs = Dynamic Programs

And, transverse to this antithesis, we suggest this other, among 'Extensional' and 'Intensional' programs:

Continuous Programs = Extensional Programs

vs

Non-continuous Programs = Intensional Programs

The reason why we suggest this shift of perspective is basically that we think it can be a bit misleading to see the 'dynamicity' of a program as the disjunction of non-eliminativity and non-continuity. Modal Logic is static, still the interpretation of its formulas is not continuous with respect to arbitrary unions of possible-worlds models. In a slogan: one thing is the difference between Classical Logic and Modal Logic, another matter is the difference between Static Logic and Dynamic Logic. This means that there can be, e.g., static-intensional programs. On the other hand, the notion of Eliminativity tends to overlook the granularity of the states in a dynamic model. If one runs a program π from a state s (that can be a propositional or a first-order model, a set of propositional or first-order models or a model of any kind), the first question in order to check the 'potential' of the program is whether it does bring the processor 'somewhere' else (namely to a state $s' \neq s$, regardless its set-theoretic relations with s). No need to say that Eliminativity is most certainly a very useful notion; only, there are others too. This slight change of standpoint will actually play a role at some points in our set-up.

1.3.5 Kinds of dynamic operators

We now want to say a few words on the basic 'procedural kit' we will use, where by 'procedural kit' we mean the set of operators to manipulate programs. Starting from this basis, that we choose for reasons that will become clear later, we will define some possible strategies for *safely* defining other possible program connectives. This will be the topic at stake in Chapter 4.

Basically, the basic repertoire includes the following operators:

1. sequential composition (notation: ;)
2. program negation (notation $\neg$)
3. boolean choice (notation $\bigcup$)

The sequential composition is a binary program connective that, applied to programs $\langle \pi_1, \pi_2 \rangle$ (notation: $\pi_1; \pi_2$) tells us to execute first π_1 and then π_2. Intuitively, this is a harmless operator, but there are some subtleties when giving a precise semantic clause for it. More explicitly, one obvious semantic clause for the sequential composition would be:

$$[\![\pi_1; \pi_2]\!]s = [\![\pi_2]\!]([\![\pi_1]\!]s)$$

that reads sequential composition simply as the functional composition. But this will not always be a possible choice, mainly due to the granularity of the states in the model. We will discuss this issue thoroughly in the following chapters.

On the other hand, the negation is much more problematic. It's not clear what 'negating a program' should mean. In first approximation, it will sometimes mean: 'go to a state where you cannot execute the program π'; alternatively, if π is a bit of information, its negation will be interpreted as a bit of information as well (just 'it's not the case that the information conveyed by π is true'). The discussion of this questions will take a big part of Chapter 4.

Finally, the boolean choice. The program $\pi_1 \bigcup \pi_2$ simply tells the processor to execute alternatively π_1 or π_2. In most cases, then, $\bigcup$ will be interpreted as set-theoretic union.

One last remark; the following trivial facts support the notions introduced in paragraph 1.3.3:

1.3.8. FACT. Tests are closed under ;, $\neg$ and $\cup$.

1.3.9. FACT. Proper programs are closed under ;, $\neg$ and $\cup$.

1.3.10. FACT. Eliminative programs are closed under ;, $\neg$ and $\cup$.

1.3.11. FACT. Continuous programs are closed under ;, $\neg$ and $\cup$.

1.3.6 Philosophical repercussions

In Chapter 5 we will briefly discuss some of the philosophical repercussions of the general dynamic set-up we propose and investigate. In particular, we will move along two directions:

1. first, we will take a philological end, and try to detect if some of the ideas that we have suggested are already detectable in the past 'analytic' literature, e.g. on game-theoretical interpretations of Natural Language;

2. afterwards, we will pursue the 'Russelian' way (= test the potential of a theory against concrete puzzles), and check if and where the dynamic turn can positively contribute to the solution of some relevant riddles in that philosophical area investigating the question of Reference and Modality.

Chapter 2

Semantic Variations

In this chapter we will display a general method for building dynamic programs out of static systems. Moreover, motivations from other disciplines such as Linguistics or Philosophy of Language will be given for carrying out this strategy.

2.1 The idea

We have seen in the introductory chapter the formulas/programs-states/ transitions structure of the dynamic set-up. What we are after now is designing a simple policy to create programs or, which is the same, to define instructions to move from one state to another. In a sense, the strategy we are going to suggest is the simplest one can think of just by looking at the very general definition of Dynamic Logic and its semantics. In that definition, a state is a stage of an information processing of some sort (i.e., it can be seen either as an information state, a picture of the world, or as a state of the memory of a computer, ...). Technically, this means that a state is where formulas (= statements about the world) are interpreted. Or, which is the same, a state is a model of some sort. Thus, we search for instructions to switch from one model to another. As we said, within the course of the present dissertation the states-models we will take into account will basically be either first-order models or possible-worlds models. This is a reasonable choice: we want to keep the static fragment of our dynamic systems within the safe boundaries of either first-order or modal logic. Moreover, first-order models and possible-worlds models represent certainly the most standard choices as formal descriptions of one's partial or total image of the world (which fits with our favourite interpretation of Dynamic Logic as a formalism to talk about the human information processing). Therefore, we are trying to define a kit of instructions to be interpreted as transitions over first

order models or over modal models.

A first-order model is a state of affairs codified in such a way as to make it possible the interpretation of first-order formulas. Thus, it includes a 'substratum' of blank objects without qualities (the Domain) that get shaped up via an Interpretation function plus a provisional mapping of the variables to the objects (Assignment function). More concretely, here is a standard first-order model:

$$M = \langle D, I, A \rangle$$

where D stands for Domain, I for Interpretation, and A is the Assignment function.

On the other hand, a propositional[1] possible-worlds model is meant to represent intensional knowledge (where intensionality can be of the epistemic kind, or generally modal, or deontic, or temporal...). Thus, it will include a set W of indexes (meant as possibilities, or temporal points etc..), one (ore more) accessibility relation(s) R and a valuation V for the proposition letters:

$$M = \langle W, R, V \rangle$$

Given that, here is our straightforward 'recipe' for creating our basic kit of programs (cf. [9]): they will just modify, one at a time, the *semantic parameters* either of first-order models or of possible-worlds models. In a slogan: if states are models of some kind and programs are instructions to modify them, then a first obvious constraint on the possible range of instructions consists in 'parametrizing' them along the states semantic parameters. We will see in the next sections that some relevant dynamic systems on the market fall within this paradigm of Semantic Variations, including Dynamic Predicate Logic (cf. Introduction) with its dynamic treatment of existential quantification.

We will call the variations of the semantic parameters of first-order models 'Tarskian Variations', while variations of the possible-worlds parameters will be referred to as 'Kripkean Variations'. Next section will be devoted to Tarskian Variations (while Kripkean Variations will be the issue of section 2.3).

2.2 First-order Variations

In this section we will present the idea of Semantic Variations for the case of first-order models. The first paragraph will give a general description of Tarskian Variations, without going into technical details. In paragraph 2.2.2 we will illustrate the extra-logical motivations that support this policy, while paragraph 2.2.3 will be devoted to designing a formal system for Tarskian Variations.

[1] Variations of first-order possible-worlds models are much more complex; we will tackle this issue in chapter 3.

2.2.1 General description

As we said, the semantic parameters involved in a first-order model are basically the Domain D, the Interpretation I and the Assignment A. Thus, we are now after defining instructions that change these parameters.

Let us start with the Assignment. We will use the conventional notation η (historically coming from [48]) to designate the assignment-changing program. Intuitively, its semantic clause will look as follows (the reader must wait until paragraph 2.2.3 for a more formal presentation):

> $[\![\eta x]\!]M = \{$ all the models that differs from M at most in the Assignment to the variable $x\}$

It's easy to see that this is nothing but the most orthodox atomic program of standard Dynamic Logic, coming from the informatic tradition: the so-called (random) *assignment*, which is in fact an instruction that changes the semantic parameter of Assignment (of objects to the variables).

A brief remark: what one often finds in the literature is rather a quantifier-like treatment of the η, which is then called to set the scope for a formula to be processed. Typically:

> $[\![\eta x Px]\!]M = \{$all the models that differ from M at most in the Assignment to x and that validate $Px\}$

Now, in this simple case it is easy to guess that:

$$[\![\eta x Px]\!] = [\![\eta x; Px]\!]$$

(for the moment we will use the sequential composition counting only on the intuitive definition of it that we have given in the Introduction. The reader will find a precise definition of sequential composition for this 'tarskian variations' in paragraph 2.2.3). Nevertheless, there are other quantifier-like treatments of η in which this equivalence fails (cf. Chapter 3 for an example). Anyway, thorough the whole dissertation we will consider η as an autonomous program and not primarily as a quantifier. The reasons for this choice will become clear later.

Summing up, starting from a state $M = \langle D, I, A\rangle$, η tells the processor to go to a certain 'storage location' of the memory - say x - and change its content according to the availabilities in the Domain.

Next, let us consider variation of the semantic parameter Interpretation. As for the notation, we will use the greek letter μ to indicate the program that changes the Interpretation of a given language; the semantic clause for μ will intuitively amount to what follows:

> $[\![\mu]\!]M = \{$ all the models that differs from M at most in the Interpretation of the signature$\}$

Therefore, μ changes 'properties' of individuals.

Finally, let us consider the variation of the semantic parameter Domain, for which we will use the notation: δ. Here is the intuitive semantic clause:

> $[\![\delta]\!] = M = \{$ all the models that differs from M at most in the Domain$\}$

δ modifies then the 'context' for the interpretation of formulas.

2.2.2 Linguistic and Philosophical Motivations

Let us now say a few words on the utilisation of this tarskian variations in the formal representation of Natural Language, within the new trend of the dynamic conception of meaning. Recall that our initial claim was that the broadening of Dynamic Logic outside the informatic influence makes sense in as much as it copes with a general cultural demand. It's now time to begin demonstrating it.

We will start again with the random assignment. It has mainly two - linked - applications.

First, it is a suitable tool for formalising indefinite descriptions like 'a woman'. One of the traditional ways of representing such a description would be by using Hilbert's ϵ terms: ϵx : woman$x$ is in fact a way of designating an *arbitrary* woman within the domain (if there is such a thing). The big disadvantage of this approach, however, is that it doesn't work when two equal indefinite descriptions, referring to two different objects, come out within the discourse, like in 'a woman is watching you; another woman is listening to you'. Without going into details, we simply remark that the typical semantics of ϵ terms would give us the same object for the two different women (cf. [27] for an ample discussion). While if we read indefinite descriptions as *processes* of pinpointing an arbitrary something - according to the dynamic ideology -, then the random assignment can be used to represent the process. 'A woman' would then be formalised with this two steps program:

> ηx; woman x

This amounts to the following instruction to the processor: open an 'address' within your memory, scan your universe and select the objects-women you find. And no problem for the 'another woman': the processor is then supposed to open a *different address.*

This very same feature of η, namely that it allows us to give addresses to the thing we pick up in the Domain, plays a crucial role in the second application of it to Natural Language. As we have already hinted in the introductory chapter, η can be in fact successfully used for formalising in a *compositional* way natural language texts containing anaphoras (including the so-called donkey-

sentences)[2]. Historically, this was one of the first applications of Dynamic Logic to a dynamic analysis of meaning (cf. [43]). A trivial example:

A woman walks in. She is blonde

would have the following first-order translation:

$\exists x$ (woman $x \wedge$ walks-in $x \wedge$ blonde x)

which is typically non-compositional, in that the translation of the first sentence of the text above ('a woman walks in') is not a subformula of it, and the same holds for the second sentence 'she is blonde'. On the other hand, using the η assignment and formalising the discourse in point as an instruction to be processed, we get the following - compositional - translation:

ηx; woman x; walks-in x; blonde x

As it is well-remarked in [43], compositionality is first and foremost a methodological principle. Yet, we agree with the prevailing ideology and acknowledge its theoretical importance within a formal analysis of Natural Language. Interestingly, this principle is also playing an important role in the more recent philosophical discussions, in as much as it contrasts another fashionable tendency, namely so-called 'holism of meaning'; cf. for an ample discussion on this issue [22].

Concerning the variation of parameter 'Interpretation', its application to the formalization of Natural Language or, more generally, of pieces of the human information processing, is even more natural and straightforward than in the case of η. The parameter Interpretation assigns properties to objects. Thus, changing the Interpretation will amount to changing the properties of objects; and it is very plausible and intuitive that something like this must happen within the human elaboration of knowledge, or at least that something in the human elaboration of knowledge can be understood or represented in terms of it. This looks easier to digest than the commitment that human reasoning functions by means of 'storage locations addresses' (= variables) and successive changes of their content[3]. By remarking this we simply want to stress that there is a great difference among the two 'variations' we have been considering up to now: η mainly models the use of our reference instruments, giving an account of how we perform certain skills in speaking of things; on the other hand, μ directly embodies a 'real change' in the way we picture the world.

More concretely let us make a few examples of the applications of μ. Generally speaking, a program that changes the parameter Interpretation can be

[2]Roughly speaking, the principle of *compositionality* says that the meaning of a complex expression is a function of the meanings of its parts (cf. [38]).

[3]But note that using the informatic random assignment to represent some human intelligent processes does not imply per se any commitment on the real nature of human mind and cognition.

useful for formalizing all the 'cognitive moves' that produce or are produced by some metamorphosis of the current state of the world (or of the current picture of the world).

So, for instance, μ can be successfully employed for representing the imperative mode of Natural Language, where the processor is supposed to carry out a modification of some sort. The command 'put the block on the table' can be codified as an instruction that leads the processor to a new state where (at least) the interpretation of the predicate 'being on the table' has been changed. But as we said, μ can also be used to account for transitions that are caused by a change in the processor's picture of the world. As an example, we can think of the case when the processor is asked to incorporate into its picture of the world the answer to a question like 'who has the property P? Again, this cognitive instruction can be performed by changing the interpretation of the queried predicate according to the answer to the question. Finally, the dynamic variation of the Interpretation could make it possible to formalize in a compositional manner natural language texts containing 'VP-ellipsis' (like 'Giulia reads Dante; Alessandro does it too'), which involve reference to changing predicates, similarly to individual anaphora.

Let us now consider the variation of the parameter Domain. It is quite easy to imagine that the Domain of reference can change along an information processing. As an example, we recall the 'context sets' for quantifiers as suggested in [75], that give an account of how different quantifiers in a linguistic expression can range over contextually changing domains.

2.2.3 A system of Tarskian Variations

Let us now give a precise description of a dynamic system including the semantic variations we have seen so far (cf. [14]).

The system for Tarskian Variations, TV from now on, will use a two-level language, producing a multimodal system in the usual style (cf. [46] and [42]). Thus, it will include 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. As for the procedural repertoire, we will have the standard 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:

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

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

This particular choice of operators will be discussed at great length in chapter 3.

As one would expect, the semantics will match the formulas/programs feature of the syntax with a states/transitions structure:

2.2.1. DEFINITION. [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'.

The truth clauses must be given by a simultaneous induction on programs and formulas. Obviously, programs will be interpreted as sets of state transitions over the set W (by means of an interpretation function $[\![\ ]\!]$ as in 1.3.1), while formulas will be evaluated at single states (by means of an interpretation function $[\ ]$ as in 1.3.1).

2.2.2. DEFINITION. [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. $[\![\varphi?]\!]_W = \{\langle w, v\rangle \mid w = v \text{ and } v \in [\varphi]_W\}$
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. $[\![\pi_1 \cup \pi_2]\!]_W = [\![\pi_1]\!]_W \cup [\![\pi_2]\!]_W$

The reader should note how these clauses represent a precise technical instantiation of the intuitive description of Tarskian Variations in the previous sections. In particular, we want to draw the attention on the clause for negation, which as we said will be discussed at great length in chapter 4: the DPL negation of a program π takes the states from which π would fail if run. For this reason it is called 'counterdomain' negation.

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

2.2.3. DEFINITION. [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. $[Px_1 \dots x_n]_W = \{w = \langle D, I, A\rangle \in W \mid (A(x_1) \dots A(x_n)) \in I(P)\}$
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. $[\langle\pi\rangle\varphi]_W = \{w \in W \mid \text{there is a } v \in W \text{ such that } \langle w, v\rangle \in [\![\pi]\!]_W \text{ and } v \in [\varphi]_W\}$

The clauses 1-3 are totally standard (from ordinary 'static' first-order logic). The clause 4 is very important, since the formula '$\langle\pi\rangle\varphi$' establishes a link between the properly dynamic level of programs and the static level of formulas. It should be clear that $\langle\pi\rangle$ is here interpreted as an existential modality 'labelled' by the program π: $\langle\pi\rangle\varphi$ is true at a certain state w if running π from w would produce at least a state v where φ holds.

A notion of Validity for TV formulas can be easily defined as follows:

2.2.4. DEFINITION. [Validity for TV formulas] A TV formula φ is ***valid*** if for all TV models W it holds that $[\varphi]_W = W$.

Before switching to the next topic, we would like to give a few examples of what one can express within this TV system. Therefore, we now list a number of interesting formulas, not necessarily valid, that show the 'expressive' power of the TV syntax. We will get back to some of this formulas in section 2.4, and see if and when they are valid.

First of all, it would seem natural to have a principle expressing the relation of the TV η with the static existential quantifier. Here is such a principle:

$$\langle\eta x\rangle\varphi \leftrightarrow \exists x\varphi$$

Concretely, this principle expresses the fact that ordinary first-order logic can be embedded into our TV system. But, as we will see in paragraph 2.4.1, this simple formula is not valid on TV models, but rather it calls for a strong constraint on them.

Similarly, it would make sense to ask ourselves what would it be the static counterpart of such modalities as $\langle\mu\rangle$ or $\langle\delta\rangle$ (this would actually require second-order logic).

Furthermore, one can look at purely modal principles, where the interaction of different modalities is at stake:

$$\langle\eta x\rangle\langle\eta y\rangle\varphi \leftrightarrow \langle\eta y\rangle\langle\eta x\rangle\varphi$$

$$\langle\mu\rangle\langle\eta x\rangle\varphi \leftrightarrow \langle\eta x\rangle\langle\mu\rangle\varphi$$

$$\langle\eta x\rangle\langle\eta x\rangle\varphi \leftrightarrow \langle\eta x\rangle\varphi$$

All these principles have an intrinsic interest, since they talk about relations between first-order models. Thus, it also makes sense to check - again - under which conditions they are valid, which we will do in paragraph 2.4.2.

Finally, there are many trivial principles inspired by the standard 'static' tradition that become intriguing in this dynamic context. For instance, it will turn out that the TV negation does not quite behave as a static negation: in chapter 4 we will see that, e.g., it is not the case that:

$$[\neg\pi;\pi]\bot$$

as one could expect.

2.3 Variations for Knowledge-updating

We will now demonstrate how to apply the policy of Semantic Variations to Propositional Modal Logic. In the first paragraph we will give the background motivations for doing this, while paragraph 2.3.2 will be devoted to the concrete description of a system of Kripkean Variations.

2.3.1 The question of Knowledge-updating

An important question for linguists and in general for knowledge representators is giving a semantic account of the updating of knowledge by a (human) processor. Namely: how to model the fact that the processor learns new facts about the world, and consequently updates or revises its mental picture of it.

Obviously, the dynamic framework in its cognitive interpretation (states = mental states and transitions = epistemic inputs), where the word 'cognition' is taken in its proper meaning of 'acquisition of knowledge', is very appropriate for coping with this question.

In fact, also the Tarskian Variations of the paragraphs above have some potentialities for representing knowledge-updating, even though this was not their original motivation. In particular, the program μ, changing the semantic parameter Interpretation, can express a process of this sort: the tarskian model is a picture of the world at a certain stage of the information processing, and μ updates it by changing the properties of its objects. This representation perfectly fits within the 'constructive' tradition for modeling the knowledge-updating: from a picture of the world - eventually incomplete but 'certain' at that stage - to a different one, that the processor actually builds as a consequence of getting new epistemic inputs. Examples from the literature include the Discourse Representation Theory, that changes its 'discourse representation structures' with new data (cf. [54]), the recent Dynamics of Theory Extension, updating deductively closed theories with new formulas (cf. [28]), and in general systems of databases updating.

An alternative, though based in a sense on the same conception of knowledge-updating, is to represent the flow of information via transitions between information states in Kripke models for intuitionistic or similar constructive logics, eventually using analogies from Dynamic Logic (cf. for example [66] and [10]).

We will not pursue this approach here. Rather, we will focus on yet another tradition, based on a radically different notion of knowledge-updating. According to this tradition, the process of knowledge-updating is primarily a process of decreasing one's degree of uncertainty. Therefore, updating a stage of the information processing amount to eliminating possibilities - that, due to lack of information, are still open - from an epistemic horizon. This alternative to the constructive approach is clearly exemplified - in a dynamic style - by the Update Logic (cf. [70]) that we will examine in the next paragraph as an example of Kripkean Variations.

A brief remark before going into an explanation of what we shall call the 'eliminative' approach to knowledge-updating; we think that both the constructive and the eliminative viewpoints are singly insufficient to model the process of knowledge-updating, since they cope with different aspects of it. Changing one's image of the world is not the same as loosing a bit of uncertainty on it (cf. [52] for an ample discussion of this point). We will see in Chapter 3 how it is actually feasible to intertwine the two perspectives within a single dynamic system.

2.3.2 Kripkean Variations

In section 2.2 we have explained the idea of semantic variations for the case of first-order models. But, as we have said, the same idea is applicable in principle to all truth definitions one can think of. Thus, we now take into account the case of possible worlds models; dynamic variations on them will make it possible to face the question of knowledge- updating.

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:

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

All these parameters can certainly be varied with appropriate instructions. We keep R and V fixed and we modify W, reading 'classical' propositional formulas as instructions for updating the universe of possibilities. This will be just the Update Logic mentioned above.

Its language consists of the following programs, starting from a set of propositional variables $p \in P$:

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

This language is evaluated in the following models:

2.3.1. DEFINITION. [UL model] A standard *UL model* **U**, given a set of propositional variables P, is a set of possible worlds $\subseteq \wp(P)$ (ordered by the total accessibility relation).

Following the general definition of dynamic models that we have given in the introductory chapter, one should define a UL model as a subset of the set of all the possible worlds models over P, namely as the set $\wp(\wp(P))$, carrying the transitions as defined by the semantic clauses in what follows. We opt for the definition above, substantially equivalent, only for the sake of simplicity.

Over the above models, programs will be interpreted as 'updating functions' (epistemic inputs that make the processor go from one state/possible-worlds-model to another):

2.3.2. DEFINITION. [Interpretation of UL programs] A UL program π is interpreted on a model $\mathbf{U}$ as a function from $\wp(\mathbf{U}) \longrightarrow \wp(\mathbf{U})$ satisfying:

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

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

4. $[\![\pi_1;\pi_2]\!]_{\mathbf{U}}(U) = [\![\pi_2]\!]_{\mathbf{U}}([\![\pi_1]\!]_{\mathbf{U}}(U))$
5. $[\![\pi_1 \cup \pi_2]\!]_{\mathbf{U}}(U) = [\![\pi_1]\!]_{\mathbf{U}}(U) \cup [\![\pi_2]\!]_{\mathbf{U}}(U)$

From now on, we will mostly omit the index $\mathbf{U}$. The idea of this definition is clear: 'classical information' varies the semantic parameter 'W'; in doing so, it discards possibilities, decreasing the degree of ignorance of the processor. On the other hand, the modality $\Diamond$ is an epistemic test of the current horizon. A close connection with modal S5 may already be seen from the 'monadic translation' first given in [6]. Thus, the general picture of information processing arising here is that of factual updates interleaved with transient tests on successive states of this process.

Note that in the system UL the 'static' part missing: UL syntax does not have formulas. We have taken this option only because it is the most common. And it is easy to see that the language for describing UL states would be just the language of ordinary Propositional Modal Logic, interpreted in the standard way (we have already pointed out that UL states are just S5 modal models). In other words, UL formulas would simply mimic UL programs.

This gives rises to a natural question: is it possible to define a notion of validity for this 'mutilated' dynamic syntax? Namely: does the notion of 'valid program' make sense? The answer is yes, and we will now give one possible definition of UL validity, in terms of 'fixed points':

2.3.3. DEFINITION. [Validity for UL programs] A program π of UL is said to be *valid* if, for all UL models $\mathbf{U}$, and for all states of U in $\mathbf{U}$, the following holds:

$[\![\pi]\!]_{\mathbf{U}}(U) = U$

Thus, valid programs are the closest dynamic equivalent of static *tautologies*: their peculiarity lies in that they do not carry any new informational content.

Before concluding the present section, we would like to show the UL system 'at work', by demonstrating a few interesting things one can express in it.

Herewith, we will use another notion of validity, defined at a meta-level. The reason why we want to do this is that the principles we want now to discuss are identities, and the set of programs above do not contain propositions identities.

Therefore, we will say that a given identity (say, $\pi_1 = \pi_2$) is 'valid' if, in all models, the graph of π_1 is equal to the graph of π_2 (namely: two programs are equal if they have the same extension).

Thus, the following two validities support UL as a good candidate for modeling knowledge-updating:

$$\Diamond\pi \,;\, \neg\pi \neq \emptyset$$

$$\neg\pi \,;\, \Diamond\pi = \emptyset$$

Suppose π conveys an informational content of some sort (e.g. it stands for 'it rains'), and the plausibility of these principles will become immediately clear!

On the other hand, the behaviour of UL negation can look surprising at first sight. The following two principles are both invalid:

$$\neg\pi \,;\, \pi = \emptyset$$

$$\pi \,;\, \neg\pi = \emptyset$$

We will discuss these principles in Chapter 4.

2.4 Logical themes

We now want to list a number of possibly interesting technical facts and questions on both tarskian and kripkean variations (although we will mainly focus on tarskian variations). In particular, the present section has two aims, that we will pursue in parallel:

1. on the one hand, we will briefly hint at some possible research tracks one can follow along the lines of standard first-order or modal logic;

2. more specifically, we will try to make clear that there are many 'degrees of freedom' in the Semantic Variations policy. In other words, there is no unquestionable choice when changing a semantic parameter, and we will briefly show some alternative options.

We will use the principles that we have mentioned in paragraphs 2.2.3 and 2.3.2 as a guideline for this general discussion.

2.4.1 The choice of models 1: closure constraints

We will now examine one first degree of freedom when designing a TV model. In doing this, we'll take our cue from one of the formula cited in 2.2.3, telling us that first-order logic is embeddable into TV:

(*) $\quad \langle \eta x \rangle \varphi \leftrightarrow \exists x \varphi$

We have already hinted at the fact that this principle is not universally TV valid. The reason is that in the definition of TV models no special assumption is made on the set of states; in particular, nothing is said about its being closed under the relation $=_A$, which means that, given a model W and a state $w = \langle D, I, A \rangle$, there can be x-variants of w (namely w' such that $w' =_A w$) that do not belong to W. Therefore, in order to make (*) valid, a constraint has to be imposed on TV models. They must have *enough states* (this definition comes from [42]):

2.4.1. DEFINITION. [TV models with enough states] A TV model W is said to have *enough states* if it is closed under the relation $=_A$.

Interestingly, one can find in the literature a static counterpart of TV models without enough states (and, consequently, of the dynamic quantifier η interpreted on TV models): in [62], Nemeti proposes what we shall call 'generalized first-order models', in which another semantic parameter comes into play; a generalized first-order model is in fact a quadruple $\langle Ass, D, I, A \rangle$, where D, I, A are the standard first-order parameters, while Ass is the set of 'available assignments', that has a crucial role in the semantic clause for the existential quantifier:

> $Ass, D, I, A \models \exists x \varphi$ iff there is a $A' \in Ass$ such that A' differs from A at most in the object it assigns to x and $Ass, D, I, A' \models \varphi$.

This same idea of reading the existential quantifier can be adapted to the TV set-up, where $\exists x \varphi$ could be interpreted according to the following clause:

$$[\exists x \varphi]_W = \{w \in W \mid \exists w' \in W : w =_A w' \text{ and } w' \models \varphi\}$$

Note that the principle (*) would obviously be true for this particular reading of existential quantification.

Nemeti actually proved that the set of formulas valid in these 'generalized first-order models' is in fact decidable.

The same line of thinking also applies to UL models: in our definition (cf. paragraph 2.3.2) a model for UL was defined, given a set of propositional variables P, as *a subset* of the set of possible worlds $\wp(P)$ (ordered by the total accessibility relation). Thus, an obvious closure constraint would require for a UL model to te *the* set of possible worlds $\wp(P)$. Interestingly, this would not change the set of (fixed-point) validities (cf. Definition 2.3.3).

2.4.2 The choice of models 2: correspondence

In this paragraph we want to briefly hint at the possibility of building a whole 'Correspondence Theory' for this Semantic Variations perspective, following the lines of Correspondence Theory for standard Modal Logic ([4]). More specifically we want to show how TV formulas can impose interesting assumptions on the 'information structure', i.e. on the structure of TV models. We will only give a few examples, taking our clue once more from paragraph 2.2.3:

$\eta x; \eta x = \eta x$ 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):

$$\forall A, A' : (R_x(A, A')) \rightarrow \exists A''(R_x(A, A'') \wedge R_x(A'', A'))$$

and Transitivity (from left to right):

$$\forall A, A', A''(R_x(A, A') \wedge R_x(A', A'')) \rightarrow (R_x(A, A''))$$

$\eta x; \eta y = \eta y; \eta x$ This is a multimodal principle, expressing the following form of 'confluence':

$$\forall A, A', A''(R_x(A, A') \wedge R_y(A', A'')) \rightarrow \exists A'''(R_y(A, A''') \wedge R_x(A''', A''))$$

$\eta x; \mu P = \mu P; \eta x$ 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 on TV models (like, for instance, Reflexivity and Transitivity of the R_x relation), while those involving existential quantifiers call for specific constraints. For a general theory of 'multimodal correspondence' see for instance [67] and [71].

2.4.3 The choice of States: kinds of Assignments

Here we want to focus on yet another range of open options when designing TV models, this time concerning the parameter Assignment. In our definition of TV models the Assignment was just the standard one from first-order logic: a function $Var \longrightarrow D$. But the current literature already offers us some other kinds of dynamic models, that could be easily adapted to the TV set-up, where the Assignments have some peculiarities. We will make two relevant examples.

First, one can follow, e.g., Groenendijk, Stokhof and Veltman (cf. [45]) and opt for partial Assignments. We will see in more details what are the key

consequences of this option (cf. Chapter 3). For the moment, we only want to hint at the fact that partial assignments possibly provide us with a further 'direction' in which the information can grow: getting new information about the world can force the processor to 'light' a new variable which had not been assigned yet.

A second possibility can be found in a recent work by Hollenberg and Vermeulen (see [73]). There, a dynamic system is defined (called DPLE for Dynamic Predicate Logic with Exit Operators), whose syntax corresponds to Dynamic Predicate Logic except for the way it expresses existential quantification. This syntax is interpreted on a particular class of models, that we shall call Stacks models. Roughly speaking, given a Domain D and an Interpretation I, a Stacks model will be a set of 'stacks' assignments over D, where a stacks assignment is a function assigning to each variable a stack, namely a sequence $d_1 \dots d_n$ of objects in D. For a stack $s = d_1 \dots d_n$, the element d_n will be called the top of s. Intuitively, existential quantification over a first order formula φ corresponds there to the following instruction, starting from a stack s: add an arbitrary object at the top of s, check if φ holds of this object, and in this case throw the object away getting back to the original s; fail otherwise. Note that, as the authors remark, ordinary assignments (being them total or partial) are just a special case of stacks assignments. The important point, that shows how the dynamic set-up possibly allows a more fine-grained reading of classical notions, is the following: the formal instantiation of the instruction above makes it possible to de-compose the existential quantifier (which now reads: $[_x\varphi]_x$, where $[_x$ stands for 'add an arbitrary object at the top of the stack assigned to x', and dually $]_x$ stands for 'throw away the object at the top of the stack assigned to x'). This decompositions allows the authors to reduce the number of variables used for performing a program with respect to more standard formalisms (the moral being: the number of necessary variables depends a lot on the choose of programs constructs).

2.4.4 The choice of programs 1: Local vs Global Variations

Herewith we want to demonstrate a further degree of freedom in the design of Semantic Variations, this time concerning the very kit of variational programs.

It's easy to see that there is a striking difference between ηx and μ: the random assignment modifies the parameter Assignment only in the variable x, while μ changes the parameter Interpretation entirely. This suggests us the possibility of making a finer distinction, between 'global' and 'local' semantic variations. Along this distinction, it makes sense to conceive a global variation of the parameter Assignment, that modifies all 'storage locations' at once. This global variation of the Assignment occurs in logic with the so-called 'universal' or 'existential closure' of complete formulas, affecting all free variables in them.

Natural Language provides us with other interesting examples where a similar process goes on (cf. the 'unselective binding' of Lewis [57]). Analogously, one can vary the semantic parameter Interpretation locally, up to a certain predicate. This is very plausible for many representational tasks, including the above example of questions (what has the property P? only needs to take into account P-variations of the current picture of the world).

It should be noticed that global variations of a semantic parameter can be decomposed into local variations. Using the Kleene star (that intuitively tells the processor to iterate a process) we get the following equivalence for the Assignment variation:

$$\eta = (\bigcup_{x \in VAR} \eta x)$$

Notably, the same local/global distinction could be applied to the Kripkean Variations set-up as well; in particular, epistemic tests could be restricted to a single predicate P (in which case they should search for open possibilities only within the P-variants of a certain reference point).

2.4.5 The choice of programs 2: 'up' and 'down' Variations of the Domain

Another degree of freedom comes out in the variation of the parameter Domain. The instruction 'change the Domain of Interpretation' can naturally be split in two sub-instructions, along with the logical relations of 'being a submodel' and 'being a model extension'. Therefore we get:

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

this would allow us to perform a more subtle 'dynamics of contexts', and make the set-up more expressive for talking about individuals.

Interestingly, this specification of the δ operator would also make the set of validities grow. In the Modal Logic for the relation '$\downarrow \delta$', for instance, both S4.1 and S4.2 would become valid, since the submodel with domain $\{w\}$ would be an endpoint.

2.4.6 The choice of operators

We have seen how many options are available when designing TV models as well as TV basic programs. Here we only want to point out that even more options are available when choosing the procedural repertoire, namely the dynamic connectives. We do no want to go into details, since this issue will be pursued at great length in chapter 4.

Chapter 3

Combining Variations

In this chapter we will tackle the question of combining different dynamic systems. First, we will defend the legitimacy, so to speak, of such a question. Afterwards, we will take as our case study the - truly controversial - question of blending within one single system two instantiations respectively of Tarskian Variations and of Kripkean Variations, namely Dynamic Predicate Logic and Update Logic. A few possible approaches to this problem will be discussed.

3.1 The problem

3.1.1 Many things dynamized

In this paragraph we want to defend the legitimacy of the general issue of combining (radically) different dynamic systems. Our point is articulated as follows: many of the dynamic systems that has been recently designed within the 'new deal' of Dynamic Logic (cf. Chapter 1) have specific linguistic (or knowledge-representational) motivations. Often, they were born to model one of the many (virtually infinite) aspects of the 'human information processing'. As we have pointed out in our introductory chapter, this is the key motive that support the current revival of Dynamic Logic. But then, trying to merge different formalisms for tackling different representational problems is a very natural task, just a complementary step to gain a better understanding of cognition: different aspects of the human information processing are bound to interact, bringing to the fore new features and new questions.

Before examining our case study, we want to briefly point out some actual issues - mainly from Linguistics - that have been approached by dynamists (some of these examples we have already met):

representation of anaphora - we have already seen how the system Dynamic Predicate Logic ([43]) makes it possible to give a compositional representation of anaphoric bindings;

representation of epistemic modalities - Update Logic (cf. [70]) accounts for the non-monotonicity of epistemic modalities in Natural Language;

representation of change of epistemic preferences - the change of epistemic preferences has been accounted for in terms of change of certain accessibility relations (note that this is a form of Kripkean Variations), which produces a 'preferential updating' (cf. [15]);

representation of change of 'focus' - the change of the interpretation of pronouns has been formalized in dynamic terms by using dynamic models where states are 'contexts' in which account is given for many factors that play a role in the process of pronouns instantiations - in particular, the attention of focus of the processor (cf. [53]).

3.1.2 Our case study: DPL + UL

We have seen how many dynamic systems on the market account for how many relevant aspects of the human information processing. As a consequence, we have argued that the general issue of combining dynamic systems constitutes an important step toward a possible more efficient formal paradigm of human intelligent skills. We will now focus on a case study that well fits with the general architecture of the present work: we will discuss at great length the issue of blending a fragment of Tarskian Variations (DPL) with a fragment of Kripkean Variations (UL). The present paragraph will be devoted to demonstrating the particular motivations that underlie this fusion. In doing this, we will also try to defend a slight shift with respect to the most commonly accepted standpoint.

In [45], that represents one of the more sophisticated approaches to this blending operation, the authors sort out a distinction between two kinds of information. First, they reckon, there is *information about the world*, namely information on how the world is like. Second, there is *discourse information*, namely information on how to use our 'linguistic resources', so to speak, like anaphoric bindings. Thus, given the fact that there are (at least) two important types of information, the question of fusing DPL and UL becomes crucial in as much as, according to the authors of [45], while random assignment of DPL accounts for an important kind of discourse information (namely, as we have seen, information on how to handle anaphoric links), UL models the updating of information about the world. Thus, combining the two systems would be a way of blending these two fundamental kinds of information within the same formal set-up.

As we said, the point of view we are trying to suggest is slightly different. We certainly acknowledge the distinction between world and discourse information, as well as the fact that they play a central role respectively in DPL and UL. But besides this, we should like to stress that this distinction is transverse to the distinction between tarskian and kripkean variations. In particular, there seems to be a tacit assumption pervading the common opinion on this matter that there is a strict relation between non-eliminativity and discourse information from one hand and non-continuity and information about the world from the other. We do not think this is true; take for instance the tarskian program μ, changing the parameter 'Interpretation': we believe that the informational content it conveys is not of the 'discourse' kind, for it tells the processor to update (yes, update) the description of the (current) world. On the other hand, we will see how it makes sense to have tests on the 'active' assignments within an epistemic horizon, which do not check information about the world, but discourse information.

Our claim is that the two perspectives (namely the tarskian one and the kripkean one) should be intertwined just because they account for two important *modes* of processing the information (of both kinds), namely the extensional mode (tarskian variations) and the intensional mode (kripkean variations). A great deal of classical philosophical literature supports the desirability of the interface extensional/intensional when coping with knowledge-representation problems, therefore we will not defend it any further.

3.1.3 Constraints on the combination

We now want to make more clear what 'combining' DPL and UL should mean. More concretely, we want to single out which constraints are called to 'discipline' the combination. There are basically three kinds of constraints:

1. first, a possible requirement concerns the design of the new system, that should look as much as possible as the 'sum' of DPL and UL; in other words, according to this constraint, the system DPL + UL should truly fuse the architectures of its ingredients;

2. second, several requirements may be imposed on which features of the two components the resulting system should inherit; it is commonly accepted that DPL + UL should inherit both the DPL reading of existential quantification (so as to allow a compositional treatment of anaphora) and the UL 'non-monotonic' account of epistemic modalities. More concretely, the resulting system should keep the DPL semantic clause for η while also interpreting the $\Diamond$ in such a way as to maintain the following UL validities (cf. paragraph 2.3.2):

$$\Diamond\pi \; ; \; \neg\pi \neq \emptyset$$

$$\neg\pi \; ; \; \Diamond\pi \; = \; \emptyset$$

3. furthermore, other constraints can be imposed on which should be the idiosyncratic features of the blended system; for instance, since DPL + UL expresses both quantification and modality, it should maybe say something interesting on the vexata quaestio of individuals and modality.

Among these constraints, only the second one is generally considered to be inescapable[1], so to speak, while the others depend much about one's taste.
We can now examine the technical reasons that make this blending (for the moment only constrained by requirement 2) quite difficult to be carried out.

The syntax of the system DPL + UL will be the syntax of DPL enriched with the UL operator $\Diamond$. Problems arise when trying to define suitable semantic clauses for the well-formed programs of this language (where 'suitable' means conform to the intended meaning of atomic programs and of operators as it emerges from DPL and UL). The standard explanation of these problems (as presented for instance in [44]) is that they are due to the fact that DPL is non-eliminative and continuous while UL is non-continuous and eliminative. In other words, they are dynamic in two different senses. It is easy to check that this is true: the η of DPL is clearly non-eliminative, while the intensional $\Diamond$ of UL gives rise to non-continuous programs.

Alternatively, and according to the shift of perspective we have suggested in paragraph 1.3.4, one can put the matter in the following terms; the problem with DPL + UL consists in that their intended models have a different granularity: while DPL programs perform transitions over first-order models, UL programs must be interpreted over sets of models of any kind (of first-order models in this case). From this point of view, UL is non-eliminative just like DPL, in as much as it is a dynamic system; namely, if we consider the states of a UL model as *intensional states*, or more concretely as possible-worlds models, then processing a UL program p (propositional variable) from any UL state s will lead us to *another* possible-worlds state s', different from s just like a state t of a model for tarskian variations may differ from t' if $\langle t, t' \rangle \in [\![\eta x]\!]_M$. Even so, there is no doubt that UL is 'set-theoretically' eliminative, since its programs transform the states of its models into subsets of them; and this feature plays an important part in the formulation of the semantic clauses for UL programs.

We will stick to the latter perspective and see how this mismatch of granularity can be handled. The problem here is to clarify in which kind of models the programs of UL + DPL can be interpreted. Intuitively, the system we get by fusing UL and DPL should be a (dynamic version of) Modal Predicate Logic.

[1]Nevertheless, it must be noticed that it makes perfect sense, for instance, to design a dynamic version of Modal Predicate Logic that does not read existential quantifiers as random assignments; as an example, think of a quantified version of Update Logic, where the syntax is just the syntax of Modal Predicate Logic, and formulas are read as updates of successive epistemic states... (cf. [30]).

Thus, the reason would call for an interpretation of the programs of this new system as transitions over first-order possible worlds models. In other words, 'lowering' the granularity of UL models goes against the intensional nature of the $\Diamond$ operator. The correct policy seems to consist in 'lifting' the granularity of DPL models, making their states intensional. Unfortunately, this task is not trivial at all. Many expressive complications occur, since some DPL semantic clauses are formulated by looking at the 'pointwise' behaviour of programs. Thus, for instance, it becomes impossible to treat the semantic clause for negation (cf. [14]), and at the same time giving a definition of logical entailment gets problematical (this was in fact the first obstacle in the enterprise to be pointed out in [44]). We will see in chapter 4 that all these troubles depend upon a same basilar point. For the moment, we will give the reader a flavour of the issue by taking into account the question of negation.

Just for the sake of clarity, we stress once more that the models for DPL + UL we are thinking of are sets of first-order possible-worlds models (namely sets of models for Predicate Modal Logic), and we want to interpret programs of DPL + UL as transitions over them.

Let us first of all try to adapt the DPL semantic clause for negation to this new set up. We would get something like this, for M a model for DPL + UL and W a state of M (i.e. a first-order possible-world model):

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

Obviously this semantic clause is purely extensional, in as much as it looks at the 'pointwise' behaviour of programs; in other words, this clause cannot work when intensional operators like the UL $\Diamond$ are involved (its inadequacy is highlighted by the fact that $\neg\Diamond\pi$ would collapse according to it on $\neg\pi$).

Let us then try with the UL clause for negation, formulated in terms of set-theoretic complement, and see if it fits this enlarged frame:

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

It's easy to guess that this is not appropriate either, since it heavily uses the 'set-theoretic eliminativity' of UL programs, that fails in the DPL case. An example of an incongruity that we would get by exporting this clause in DPL + UL is:

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

that does not seem to be acceptable.

Summing up, it seems clear that more structure is needed in the models for this Dynamic Modal Predicate Logic. Next section will be devoted to the examination of two possible solutions that have been recently proposed.

3.2 Dynamic Modal Predicate Logic

3.2.1 General features

Let us start with a brief discussion of the ground motives that support this particular solution to the question of merging DPL and UL.

Clearly, combining DPL and UL amounts in a sense to giving a dynamic semantics for a system of Modal Predicate Logic, generally conceived for the moment as a first-order logic with a modal operator. It is well-known that already the classical semantics for Modal Predicate Logic is at least problematic. More precisely, the design of such a semantics is not uniquely determined, for quite a few alternatives are available (for a detailed presentation of these alternatives cf. for instance [40] and [20]). In particular, one has to choose if the Domain of interpretation must be the same in all possible worlds, and if the Assignment function must behave in the same way in all possible worlds. This latter choice has many interesting consequences; roughly speaking, a fixed Assignment function will treat the variables of a language as 'rigid designators', and as a consequence, the modal operators as *de re* operators, while a non-fixed one (or, as we shall say, a world-relative one) will account for the so-called *de dicto* intensional attribution. This as far as 'static' Modal Predicate Logic is concerned. Now, when delineating a dynamic semantics for MPL, one has to face in a sense with the same dilemmas: if programs are transitions over sets of states = first-order possible worlds models, how are the Domains in this states? and how does the Assignment function behave? Concerning the Domain, it is usually taken to be fixed, mainly for the sake of simplicity. The option on the Assignment remains open. The system we are going to present, DMPL, takes a clear position on this option: its modality will be a 'de re' modality, since it will range over fixed Assignments. Thus, the UL $\Diamond$ will range in DMPL over alternative states of affairs - alternative Interpretations -, given a fixed Assignment of objects to the variables. On the other hand, the DPL η will modify Assignments, according to its original interpretation.

3.2.2 Technical outline

We will give a sketched description of the system DMPL (for Dynamic Modal Predicate Logic), as presented in [35] (see Appendix A for a detailed presentation).

The information states of a model for DMPL should be seen as sets of first-order models, all sharing the same Domain, and when a modal program is processed it will look at the equivalence classes in this sets modulo the semantic parameter Assignment. But let us give a brief description of the system.

First, here is the syntax of DMPL[2] :

[2]for the sake of simplicity, we do not consider the 'static' part of the system; yet, an

DMPL Programs :: $= Pt_1 \dots t_n \mid \eta x \mid \pi_1;\pi_2 \mid \pi_1 \cup \pi_2 \mid \neg\pi \mid \Diamond\pi$

Here is a definition of a DMPL model:

3.2.1. DEFINITION. [DMPL model] A *DMPL model* M consists of a set of first-order possible-worlds models W over a given Domain D (since the domain is fixed, we will sometimes indicate the possible worlds as couples $\langle s, I\rangle$, where s is an Assignment over D and I is an Interpretation). Over this models, DMPL programs will be interpreted as transitions over the Ws, namely as subsets of $M \times M$.

As we have explained, the modal operator $\Diamond$ will test alternative Interpretations for a fixed Assignment, and, consequently, classical first-order formulas will eliminate possibilities, i.e. change the set of active Interpretations for a fixed Assignment. On the other hand, the η will vary Assignments. Thus, the processing of a DMPL program π from a state W should act pointwise on the equivalence classes modulo Assignment in W (that constitute the appropriate context for reading the modality), and transform them into other sets of first-order models. In other words, it should transform the single $\langle U, s\rangle$, where U is a set of Interpretations and s is an Assignment, into other sets of possible worlds.

To solve the expressive difficulties that this raises, we first define a function that, given a couple of Assignments $\langle s, u\rangle$, takes as its argument a set of Interpretations and transforms it into another one. This is a way to show how the two dimensions of variations (variation on sets of Interpretations and on Assignments) are active at the same time. The proper interpretation function for DMPL programs will be easily retrieved from this function.

3.2.2. DEFINITION. [Ass/Int function]

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. $[\![\Diamond\pi]\!]^s_u(I) = \{i \in I \mid s = u \text{ and there is an } r \text{ with } [\![\pi]\!]^s_r(I) \neq \emptyset\}$

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

---

extension to a Dynamic Logic in the standard sense as suggested in paragraph 1.3.1 can be trivially performed.

A program π is then assigned the following transition relation over a DMPL model M:

3.2.3. DEFINITION. [Interpretation of DMPL programs] Given a DMPL model M and a DMPL program π, the following holds, for any $W, W' \in M$:

$$\langle W, W' \rangle \in [\![\pi]\!]_M \quad \text{iff} \quad W' = \{\langle s', I' \rangle \mid \exists \langle s, I \rangle \in W \ : \ [\![\pi]\!]^s_{s'}(I) = I'\}$$

Let us briefly comment on. Given an Assignment s, the interpretation of first-order statements $Rt_1 \cdots t_n$ simply eliminates Interpretations I (or, in other words, possible worlds $\langle s, I \rangle$). Thus, providing the processor with new information about how the world is like, it eliminates wrong possibilities, changing in this way the context for the further knowledge elaboration.

Concerning the sequential composition and the negation, we first remark that they both truly intermix the features of their matches in DPL and in UL. Concretely, the DMPL clause for sequential composition blends the 'functional' character of the UL clause (taking sets of Interpretations as basic entities, and being in this way just function composition as far as Interpretations are concerned) with the 'relational' character of the DPL one (looking at the pointwise behaviour of Assignments, and therefore amounting to relational composition). On the other hand, DMPL negation performs as boolean complement (in the UL style) along the dimension of the parameter Interpretation, once more acting on *sets of* Interpretations, while it looks at the Assignments pointwise.

The clause for η is at heart *the same* as in DPL: it makes the whole machinery (which is now more structured than in DPL, but which still contains as its atoms plain first-order models with plain Assignments to be varied) change the value of a certain register.

The clause for $\Diamond$ checks open possibilities in the submodels of an information state (possible worlds model) with fixed Assignments. Thus, again, it is in a sense *the same* as in UL, since it performs a test on an epistemic space, that is in this case a set of alternative Interpretations.

3.2.3 Pros and Cons

Let us now discuss advantages and disadvantages of DMPL, starting from the advantages.

First, we should stress the very essential feature of DMPL, namely that it is a clear combination of two different systems (representing two different styles of dynamization) that intertwines their respectives hallmarks. In fact, the original motivation of Dynamic Modal Predicate Logic was first and foremost a technical one, since at that time it was not clear if and how a blending of 'eliminative' updates with 'constructive' programs meeting the second constraint above was *formally* possible. From this point of view, DMPL gives a positive answer, by performing this blending in such a way that the two ingredients remain well visible and yet are mixed together. How effectively this blending works can be

measured by looking at the notion of entailment for DMPL, that was originally marked (cf. [44]) as one of the more thorny points in the performance of the fusion. The common schema of the definitions of entailment in UL, DPL and DMPL is as follows:

π_1 entails π_2 if for all states W π_2 is *satisfied* by $[\![\pi_1]\!]W$

where the notion of 'satisfaction' is the variable that gets a different instantiation in every set-up. Our claim is that the DMPL satisfaction clearly combines its counterparts in UL and DPL. Concretely:

3.2.4. DEFINITION. [Satisfaction in UL] A UL program π is *satisfied* by the state W if it holds that:

if $w \in W$ then $w \in [\![\pi]\!]W$

In other words, since UL is eliminative, W satisfies π if $[\![\pi]\!]W = W$. On the other hand:

3.2.5. DEFINITION. [Satisfaction in DPL] A DPL program π is *satisfied* by a state w if it holds that:

$\exists v : v \in [\![\pi]\!]w$

Finally, here is the DMPL definition, that follows the UL format concerning the parameter Interpretation, while conforming to DPL as far as the Assignments are concerned:

3.2.6. DEFINITION. [Satisfaction in DMPL] A DMPL program π is *satisfied* by a state W if it holds that:

if $\langle s, I\rangle \in W$ and $i \in I$, then $\exists u : i \in [\![\pi]\!]^s_u I$

The second advantage of DMPL comes in a sense from static Modal Predicate Logic, in that it lies in the intrinsic interest of 'de re' modalities. Yet we must say that 'de re' modalities possibly acquire a peculiar flavour within the dynamic framework. In particular, the interpretation of dynamic programs as epistemic inputs (where by 'epistemic' we mean conveying information of any sort, being it *about the language* or *about the world*) gives the dynamic de re modality of DMPL its idiosyncrasy, so to speak. For those who know the old querelle on quantified modalities (cf., for instance, [65]), it will be easy to guess that this peculiarity of the DMPL $\Diamond$ will be also the reason of the main disadvantages of this system. But we will deal with this point in a little while, and get back now to the possibly positive features of DMPL. Concretely, we will give an example of how its modality can be used in the representation of human information processing.

As we have seen, the modality of DMPL tests open possibilities (= open alternative Interpretations), given a fixed Assignment. Thus, let us suppose that the variable x has been 'lighted' for talking about a woman: ηx; woman x.

More precisely, let us suppose our processor is trying to guess who this woman is, by getting hints from another speaker. A typical hint could be 'she is blond', that, in our dynamic representation, would be sequentially processed as follows:

$\eta\ x$; woman x; ...; blond x; ...

But one can also imagine a hint like: 'she cannot be angry' (since she is extremely well-tempered), that typically represents a 'de re' modality applied to an indefinite individual. In symbols:

$\eta\ x$; woman x; ...; blond x; $\neg\Diamond$angry x

In this sense, de re modalities are properties of 'individual concepts', and do convey an informational content, being more than simple epistemic tests.

The disadvantages of DMPL we can think of are all linked to this de re nature of its modality. Thus, again, they are not proper of the dynamic context, but rather they come from static Modal Predicate Logic. First of all, the 'de re' modality has the disadvantage of being philosophically controversial (we trust that even our above trivial example on the 'essentially' peaceful woman would drive many philosophers mad..). We will not pursue this issue here (but see Chapter 5 for an ample discussion). Second, also for those who acknowledge their role within Natural Language, 'de re' modalities are certainly more marginal (i.e. statistically less used) than their 'de dicto' counterpart. In particular, as it is well known, fixed assignments do not allow us to cope with identities (since they make the statement $\Diamond s = t$ collapse on its dual $\Box s = t$).

3.3 An alternative: the system GSV

We will now present an alternative recipe for combining DPL and UL: the system recently designed by Groenendijk, Stokhof and Veltman (cf. for instance [45]), which we shall call for short GSV.

3.3.1 General features

The system GSV is more than a simple 'addition' of the UL modality to the set-up of Dynamic Predicate Logic. Rather, it should be seen as a dynamic version of Modal Predicate Logic, that does keep the basic motivations of DPL and UL, but is radically different in its technical structure (in other words, the first of the constraints in paragraph 3.1.3 is not fulfilled here).

The first fundamental feature of GSV that we want to stress is that it makes a different choice concerning the interpretation of its modality, which does range over alternative Assignments too. Therefore, we would be tempted to say that the GSV modality is a 'de dicto' modality; yet, this would be quite imprecise, since the very peculiarity of the system is that it is fine-grained enough to cope

both with a 'de dicto' and with a 'de re' reading of the $\Diamond$. Thus, before describing GSV, we will say a few words on how this double reading is possible, trying to make it clear that the reason does not have much to do with the dynamic character of the semantics. More concretely, we will first briefly describe a static system, that we shall call 'static GSV', that already brings to the fore the feature of (dynamic) GSV that allows a very powerful formalization of epistemic modalities.

The syntax of static GSV is simply the syntax of Modal Predicate Logic. The models of static GSV are the models of Modal Predicate Logic, with fixed Domain and World-relative Assignments (i.e. a model W is a set of first order models w all sharing the same Domain D, but not the same Assignment). The semantic clauses for well-formed formulas of GSV are standard. The peculiarity of the system, linked to the fact that it is interpreted over World-relative Assignments, lies in the clause for the existential quantifier, that reads as follows:

$W, w \models \exists x\varphi$ iff there exists a $d \in D$ such that $W[x/d], w[x/d] \models \varphi(x)$

where $w[x/d]$, for $w = \langle D, I, A\rangle$, is equal to w except for the fact that its Assignment (say, $A[x/d]$) assigns the object d to the variable x, and $W[x]$ is just $\{w[x/d] \mid w \in W\}$.

Obviously, this semantic clause does not differ from the usual one - usual at least when Assignments are fixed - in which the $W[x/d]$ does not show up, if the formula φ in the scope of the quantifier does not contain modal operators. On the other hand, if the formula is a modal one, then the difference becomes clear. In order to show the advocated advantages of this difference when applying the formalism to the analysis of Natural Language, we will just summarize an example from [45]. Consider the following discourses:

1. There is someone hiding in the closet who might have done it.

2. There is someone hiding in the closet. He might have done it.

Intuitively, they *can* mean something different[3]. Shortly, the first discourse seems to call into play a 'de re' modality (saying that a particular individual that has the property of 'hiding in the closet' has also the property of 'possibly having done it'), while the 'might' in the second has a 'de dicto' flavour.

Here are the corresponding standard formalizations:

1. $\exists x(Qx \wedge \Diamond Px)$

2. $\exists xQx \wedge \Diamond Px$

It is easy to see that these two formulas would be equivalent in a standard Modal Predicate Logic set-up with Fixed Assignments models (since the second formula would collapse on the first). On the other hand, the interpretation of

[3]although maybe the difference is not as self-evident as the authors claim in [45].

the existential quantifier as in static GSV makes the logical equivalence between 1 and 2 fail; the reason of this 'success' is that the modality gets a 'de re' reading when inside the scope of an existential quantifier (since the existential quantifier 'resets' the relevant registers, say the xs in all possible worlds, on the same value), and a 'de dicto' reading when outside, according to the free assignments of the model. From another perspective, we can say that static GSV, because of its models with World-relative Assignments, interpretes the terms as *individual concepts* when outside the scope of a quantifier, and as *individuals* when inside.

3.3.2 Technical outline

Let us now examine the system GSV, that as we said, can be seen intuitively as a dynamic counterpart of the static GSV above[4].

The syntax of GSV is basically the same as the syntax of DMPL, except for the fact that it does not allow boolean choice (the reason of this will be soon clear):

$$\textbf{GSV Programs} :: = \; Pt_1 \dots t_n \mid \eta x \mid \pi_1;\pi_2 \mid \neg\pi \mid \Diamond\pi$$

The semantics is more structured than the DMPL semantics. The Domain of a GSV model, again, is fixed. But the atomic states (that will compose proper intensional states) are here partial, including an Interpretation over D (the Domain) and a *partial* Assignments of objects in D to the variables. Since the GSV $\Diamond$ is supposed to have a wider range than its DMPL companion, the expressive difficulties of the blending cannot be solved by distinguishing the level of Interpretation and the level of Assignment. Rather, the GSV stratagem consists in defining an extension ordering on atomic states, that allows one to keep track of the effect of programs on them (cf. the notion of 'local truth' in Chapter 4). Let us see how this stratagem is performed. The basic idea here comes from [72].

3.3.1. DEFINITION. [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.

According to [45], given a referent system $\langle n, r\rangle$, we will refer to the natural numbers smaller than n as the *pegs* of it.

3.3.2. DEFINITION. [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* - or alternatively 'information state' - S is a set of atomic states sharing the same referent system. A *model* M is a set of information states.

[4]the relation between static GSV and GSV could be made precise by using static GSV for expressing pre- and post-conditions of GSV. We will not pursue this topic here.

The use of referent systems permits to define the following extension ordering on atomic states (as the authors point out in [45], this ordering would be difficult to define without using *pegs*, unless one accepts to give up the possibility of re-using a quantifier).

3.3.3. DEFINITION. [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:

3.3.4. DEFINITION. [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. $[\![\pi_1; \pi_2]\!]S = [\![\pi_2]\!]([\![\pi_1]\!]S)$
3. $[\![\eta x : \pi]\!]S = \cup_{d \in D}[\![Pt_1 \ldots P_n]\!]S[x/d]$
4. $[\![\Diamond\pi]\!]S = \begin{cases} S & \text{if } [\![\pi]\!]S \neq \emptyset \\ \emptyset & \text{otherwise.} \end{cases}$
5. $[\![\neg\pi]\!]S = \{s \in S \mid \neg\exists s' : s \leq s' \text{ and } s' \in [\![\pi]\!]S\}$

3.3.3 Pros and cons

As in the case of DMPL, let us now comment on the pros and the cons of this system, starting once more from the pros.

Actually, the reader can guess the merit of GSV, since it already showed up in the 'static GSV': the peculiar interpretation of the existential quantifier makes it possible to merge a 'de re' reading of the modal operator (when existentially bound variables are involved), with a 'de dicto' account. Concretely, the example above on the guy hiding in the closet that might have done it, can be formalized here in two fashions, accordingly to the modality we want to express:

1. $\eta x(Qx; \Diamond Px)$
2. $\eta x Qx; \Diamond Px$

where in 1 $\Diamond P$ is an *individual property* ('de re' reading), while in 2 the $\Diamond$ refers to the 'dictum' Px. Similarly, modalizing identities produces now something more intriguing than in a totally 'de dicto' (or in a totally 'de re') context; consider for instance these two elaborations of the discourses above:

1. Someone has done it. There is someone hiding in the closet who might be the one who has done it.

2. Someone has done it. There is someone hiding in the closet. He might be the one who has done it.

GSV allows us to formalize them as follows:

1. $\eta x Dx; \eta y(Qy; \Diamond y = x)$

2. $\eta x Dx; \eta y Qy; \Diamond y = x$

Note that in 1 the $\Diamond$ is 'de re' only with respect to y, since the modality shows up within the scope of ηy. (i.e. y has the property of being possibly equal to x. This is why $\Diamond y = x$ does not collapse on $\Box y = x$). On the other hand, the reading in 2 is totally 'de dicto'.

Summing up, GSV constitutes an original approach to the standard puzzles on quantified modalities, so as to propose 'a dynamic cut on their solution' (cf. [16]). Actually, we prefer to see this advocated cut as non essentially linked to the dynamic bent of GSV, in as far as we think it rests on the peculiar interpretation of existential quantification, that can be mimicked in static terms (see Chapter 5 for an ample discussion of this theme).

Since we have talked about the features of the GSV η, we should make a short remark on the reason why $\eta x : Pt_1 \ldots t_n$ cannot be read as a sequential composition. As we have seen, 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.

Let us now discuss the disadvantages of the system GSV.

Our main criticism to GSV is of a methodological kind. We have seen how in GSV the reading of the modality depends on its being inside or outside the scope of a quantifier. For it is the quantifier that limits the range of the $\Diamond$ so as to make it a 'de re' operator. This can look elegant from a logical point of view: we recall that the Fixed Assignments semantics for Modal Predicate Logic would make the formulas 1 and 2 of paragraph 3.3.1 equivalent, which surely is limitating. But we wonder if it is really the case that 'de re' modalities only show up as referred to bound variables. More explicitly, here is our main concern: is it always the case that the modal intention of a speaker (meaning by 'modal intention' the intention towards the reading of modalities) is so strictly linked to the binding structure of the discourse? This would imply for instance that when someone utters a discourse like 'there is someone hiding in the closet. He might be the killer' can never mean the same as uttering 'there is someone hiding in the closet who might be the killer'. But suppose you saw who is hiding in the closet, and you don't know him but he looks a very unreliable guy... Then by saying 'there is someone hiding in the closet. He looks scaring.... He might be the killer' you are not using a 'de dicto' modality, but you are rigidly referring to the man you saw, and looking at *him* in all the worlds of your epistemic horizon. Still, it could be reasonable to look for a compositional formulation of that discourse... Moreover, we have seen above that in a formula like:

- $\eta x Dx; \eta y(Qy; \Diamond y = x)$

the reason why $\Diamond y = x$ does not collapse on $\Box y = x$ is that x can be assigned different objects. But take the following formula, where a is a constant (for Alessandro) which is assigned the same object all over the model:

(*) $\eta x(Qx; \Diamond x = a)$

This is the formalization of:

There is someone hiding in the closet who might be Alessandro.

According to [45], this utterance would make sense in case you know something about the guy who is hiding in the closet - for instance you know how his voice looks like - and attribute to him a modal property. But unfortunately, in case you know who Alessandro is (as we said, a has the same interpretation in all states), from (*) you are allowed to infer:

(*) $\eta x(Qx; \Box x = a)$

Another qualm with GSV, of a totally different sort, is that boolean choice cannot be defined in this set-up. The reason is that 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 Tarskian Kripkean Variations

3.4.1 General discussion

In this section we want to design a general combination of Tarskian Variations with the epistemic modalities (cf. [14]). Namely, we will enrich the syntax of DPL + UL with the variations of the others tarskian parameters (and with the matching modalities); this enrichment will actually have intriguing consequences on the expressive potentialities of the combined system. Note that the formal strategy we are going to suggest well applies to the original problem of fusing DPL and UL.

The basic philosophy underlying this attempt is as follows. We have seen how DMPL succeeds in computing, so to speak, the 'addition' of DPL and UL (following the first and the second constraints of paragraph 3.1.3); but, this addition is performed there by enforcing a distinction of levels in the semantics (level of Interpretations vs level of Assignments), and consequently by confining the modality to a total 'de re' reading, certainly insufficient when applied to Natural Language.

On the other hand, GSV manages to account within the same set-up both for a 'de re' and for a 'de dicto' use of its *Might* ($\Diamond$) (meeting in this way constraint 3 of paragraph 3.1.3, together with 2). It does this by means of a particular interpretation of the existential quantifier, that controls the reading of modalities within its scope. Therefore, it basically links the meaning of modal operators to the quantificational structure of discourse.

Thus, taking our clue respectively from DMPL and GSV (in the sense of trying to fulfil both constraint 1 and 3 of paragraph 3.1.3, we will build a dynamic system where:

1. the two 'ingredients' (tarskian updates and kripkean updates) remains well discernible, and keep as much as possible their original mould;

2. account is given for the manifold use of modalities within the human information processing, without making any commitment on the link intensional knowledge/quantificational discourse structure.

Accordingly, our strategy will consist in the following steps, for coping respectively with points 1 and 2 above:

1. taking a hint from GSV, we will define a 'pointwise' ordering on atomic states, which will make it possible to deal with the 'double granularity' of the blended system without requiring a distinction of levels as in DMPL;

2. in the spirit of the 'semantic variations' of Chapter 2, we will 'parametrize' the epistemic modality (to single semantic parameters), hopefully providing a very fine-grained intensional formalism, without calling into play its relation with quantification.

3.4.2 Technical outline

Let us define the syntax of this system, that we shall call TKV (for Tarskian Kripkean Variations):

$$\textbf{TKV Programs} :: = Pt_1 \dots t_n \mid \neg\pi \mid \pi_1;\pi_2 \mid \pi_1 \cup \pi_2 \mid \eta \mid \eta x \mid \mu \mid \mu P \mid \Diamond^{\eta}\pi \mid \Diamond^{\eta x}\pi \mid \Diamond_{\eta}\pi \mid \Diamond_{\eta x}\pi \mid \Diamond^{\mu}\pi \mid \Diamond^{\mu P}\pi \mid \Diamond_{\mu}\pi \mid \Diamond_{\mu P}\pi$$

In this syntax, the reader will recognize the programs of 'generalized' tarskian variations (i.e. both in their local and global version - cf. paragraph 2.4.4 -). The programs that modify the semantic parameter Domain are left out here, partly for the sake of simplicity, partly because the issue of updating the domain of reference is somewhat marginal in the current debate on combining different kinds of updates. The modal operators of TKV are new. Intuitively, given an atomic state w, $\Diamond^{\eta}$ will check open possibilities within the worlds that *differ* from w at most as far as the parameter Assignment is concerned (or, using the notation of Chapter 2, the worlds v such that $w =_A v$). Analogously for $\Diamond^{\eta x}$, $\Diamond^{\mu}$, and $\Diamond^{\mu P}$. Dually, given an atomic state w, $\Diamond_{\eta}$ will check open possibilities within the worlds v that *coincide* with w at least as far as the parameter Assignment is concerned (we will write in this case $w \neq_A v$). And similarly for $\Diamond_{\eta x}$, $\Diamond_{\mu}$, and $\Diamond_{\mu P}$.

TKV models will be just like TV models, with an obvious extension of the 'shift relations' (cf. definition 2.2.1), according to the local/global character of the parametrization and to the coming into play of such modalities as $\Diamond_{\eta}$:

3.4.1. DEFINITION. [TVK model] A *TVK model* consists of a family of states $\langle D, I, A\rangle$, where A is ana assignment from variables into the domain D and I is an interpretation function from predicate letters into denotations over the domain. Notation for states: $w, v, u, \dots$. These models carry the following 'shift relations':

1. $w =_A v$: w differs from v at most in its A-values
2. analogously for $=_x$, $=_I$, $=_P$
3. $w \neq_A v$: w coincides with v at least in its A-values
4. analogously for $\neq_x$, $\neq_I$, $\neq_P$

On these models, programs will be interpreted as updating functions. But in order to give the semantic clauses (that must account for the double granularity of the system), we need to introduce a program-dependent ordering on single states, encoding the core behaviour of distributive programs:

3.4.2. DEFINITION. [Pointwise ordering]

1. for $\pi = P(x_1 \dots x_n)$, $w \succ_\pi w'$ iff $w = w'$ and $w' \models P(x_1 \dots x_n)$
2. for $\pi = \eta x$, $w \succ_\pi w'$ iff $w' =_x w$

3. similarly for $\eta, \mu, \mu P$ with their corresponding relations $=_A$, $=_I$, $=_P$
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.

This pointwise ordering, that basically mimics our earlier TV evaluation, allows us to trace the behaviour of programs on atomic states (similarly to the 'extension ordering' of GSV), and to define our lifted semantic clauses. We shall use two notational conventions:

$$|w|^{A,W} = \{v \in W \mid v =_A w\}$$

$$|w|_{A,W} = \{v \in W \mid v \neq_A w\}$$

which similarly applies to the other (local and global) tarskian parameters.

3.4.3. DEFINITION. [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 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 parameters
6. $[\![\Diamond_\eta \pi]\!]W = \{w \mid \exists v \in |w|_{A,W} \ (\exists u(v \succ_\pi u \ \wedge \ u \in [\![\pi]\!]W))\}$
7. Similarly for the other parameters
8. $[\![\neg\pi]\!]W = \{w \mid \neg\exists v \ (w \succ_\pi v \ \wedge \ v \in [\![\pi]\!]W)\}$
9. $[\![\pi_1;\pi_2]\!]W = [\![\pi_2]\!]_W([\![\pi_1]\!]_W)$
10. $[\![\pi_1 \cup \pi_2]\!]W = [\![\pi_1]\!]_W \cup [\![\pi_2]\!]_W$

3.4.3 Pros and cons

The comment we want to do concerning TKV discloses at the same time its main pro and its main con: the syntax of this system is very fine-grained, which means that it is very 'ugly', while also extremely powerful for the representation of quantified modalities (we leave it to the reader to check how the examples of previous sections can be easily treated within this set-up). In other words, the abundance of possible readings of modal operators in Natural Language is here

dealt with by means of a complex language; of this language we are able to give a uniform semantic treatment, also relatively simple. No commitments are made on the link quantification/reading of modalities. This is an advantage in a sense (for it allows us, e.g., to give a compositional treatment of 'de re' modalities, which, we recall, was not possible in GSV), though we want to stress once more that the way GSV managed to build that link was certainly intriguing. In a slogan, TKV is theoretically weaker than GSV, in that it mimics at the level of the syntax the complexity of Natural Language; on the contrary, GSV positively suggests a 'lowering' of that complexity by means of a commitment on the 'internal grammar' of Natural Language. GSV is stronger, and consequently more 'falsifiable'.

As a passing remark, we should stress how the semantic clause for boolean choice becomes here unproblematic.

Moreover, there are no obvious problems for the negation clause. In particular,

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

Finally, 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.

Chapter 4

The procedural kit

What we were after up to now was defining a large set of instructions for Dynamic Logic(s), that, according to our propositive introductory chapter, do not necessarily lie within the boundaries of programming languages. We accomplished this task by means of a 'variational' policy that, we should say, is totally in line with the dynamic tradition, in as much as it extends to other semantic parameters the fundamental idea underlying the 'random assignment'. Moreover, the strategy of tarskian/kripkean variations simply codifies the atomic moves from a state (say a mental-epistemic state) according to the parameters we have been used in order to describe it. And this, again, fits with the general cognitivist motivation of actual dynamics. In one word, Semantic Variations look a legitimate and harmless 'source' of programs, in line with the current dynamic trend.

Therefore, given this sound set of (atomic) dynamic programs, we want to carry on our possible 'enlargement' of Dynamic Logic at the procedural level, i.e. at the level of program connectives. Here, we will not define a uniform set of dynamic connectives; rather, we aim to give a number of criteria that hopefully will draw up a bit the abundance of conceivable operators to handle programs.

4.1 Invariance criteria

Invariance criteria have a sound mathematical tradition. Intuitively, the idea underlying these criteria is the following: given an operation on a structure of some sort, one can check if this operation would behave *in the same way* on suitably transformed versions of the same structure; namely, invariance criteria check if and to which extent a mathematical operation is linked to particular features of the base structure.

4.1.1 Permutation invariance

The first criterion we want to examine is a very broad one (in that it puts up with many possible connectives): logicality (see e.g. [10]). Let us first expound the intuitive meaning of logicality. Basically, a 'logical' connective of a language must be free from any content, and aimed to confer complex expressions their *formal structure*. This intuition is typically implemented in a very well-know technical notion, namely permutation invariance. More precisely, an expression is said to be 'logical' if it is preserved under permutations of individuals in the underlying domains.

Before giving a precise definition of 'logical dynamic connectives', let us see how logicality applies to set-theoretic operations. Dynamic logicality will be seen afterwards as a lifting of its set-theoretic counterpart.

4.1.1. DEFINITION. [Logical set-theoretic operations] An n-ary operation O on sets is *permutation invariant* if, given any n-tuple $S_1 \dots S_n$ of sets, and any permutation α of the universe, the following holds:

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

The import of this feature shows up, for instance, in the following simple fact (but see the examples in [7]):

4.1.2. PROPOSITION. *An n-ary set-theoretic 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.*

Proof: see [7] ∎

As the logicality of set operations is defined in terms of set permutations, the logicality of program operations will call into play program permutations. Therefore, here is our definition of permutation of a program (cf. [14]), that truly amounts to the permutation of its graph:

4.1.3. DEFINITION. [Program permutation] Given a permutation α of the set of states, the *permutation of a program π induced by α*, in symbols $\alpha(\pi)$, is a program defined as follows:

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

Note that graph permutation can be also described in the following, equivalent, way:

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

By means of this 'program permutation' we can finally define an appropriate notion of logicality for program operations @:

4.1.4. DEFINITION. [Permutation invariance] An n-ary program operation @ is *permutation invariant* if, for all programs $\pi_1 \dots \pi_n$ and permutations α, the following holds:

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

or, in other words, if, for all $\pi_1 \dots \pi_n$ and all α:

$$[\![\alpha^{-1}; @(\pi_1 \dots \pi_n); \alpha]\!] = [\![@((\alpha^{-1}; \pi_1; \alpha) \dots (\alpha^{-1}; \pi_n; \alpha))]\!]$$

Interestingly, it turns out that a certain format of definition already is enough to guarantee logicality (and this demonstrates how broad is this criterion):

4.1.5. THEOREM. *If the n-ary program operator @ is defined as follows:*

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

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

Proof: Take an n-tuple of programs $\pi_1 \dots \pi_n$ and a program permutation α. We have that:

$[\![\alpha^{-1}; @(\pi_1, \dots \pi_n); \alpha]\!]$ =
$\lambda W.\ \alpha([\![@(\pi_1, \dots \pi_n)]\!](\alpha^{-1}(W))$ =
$\lambda W.\ \{\alpha(w) \mid \varphi(\pi_1, \dots \pi_n, \alpha^{-1}(W), w)\}$ =
by invariance of set-theoretic statements for permutations of individual domains
$\lambda W.\ \{\alpha(w) \mid \varphi(\alpha(\pi_1), \dots, \alpha(\pi_n), \alpha\alpha^{-1}(W), \alpha(w))\}$ =
by general properties of permutations
$\lambda W.\ \{w \mid \varphi(\alpha(\pi_1), \dots, \alpha(\pi_n), W, w)\}$ =
$\lambda W.\ \{w \mid \varphi(\alpha^{-1}; \pi_1; \alpha, \dots, \alpha^{-1}; \pi_n; \alpha, W, w)\}$ =
$[\![@((\alpha^{-1}; \pi_1; \alpha), \dots, (\alpha^{-1}; \pi_n; \alpha))]\!]$. ∎

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. [10], [7]).

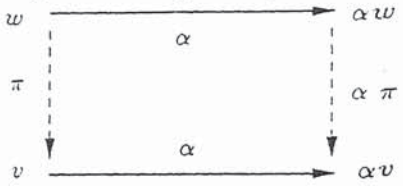
4.1.2 Tightening the logical space: Bisimulation Safety

Now, after mapping out such a general logical space as that of permutation invariant program operators, let us tighten it and characterize the standard procedural kit that has been used, e.g., in our systems for Semantic Variations. Following [12], we are going to demonstrate a stronger criterion for 'logicality', known as *bisimulation safety* and based on a crucial notion of process equivalence from computer science.

We will define such a criterion so as to make it clear how it truly represents a generalization of the idea of permutation invariance.

4.1.6. DEFINITION. [Bisimulation] A relation f between $\langle W, \pi_1 \dots \pi_n \rangle$ and $\langle W', \pi'_1 \dots \pi'_n \rangle$ is a *bisimulation* if, whenever wfw', the following conditions hold:

Figure 4.1: morphisms preservation



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

A bisimulation f between $\langle W, \pi_1 \ldots \pi_n\rangle$ and $\langle W', \pi'_1 \ldots \pi'_n\rangle$ will be also called 'bisimulation *for* $\pi_1 \ldots \pi_n$'.

4.1.7. DEFINITION. [Bisimulation Safety] An operation $@(\pi_1 \ldots \pi_n)$ on programs is *safe for bisimulation* if every bisimulation f between $\langle W, \pi_1 \ldots \pi_n\rangle$ and $\langle W', \pi'_1 \ldots \pi'_n\rangle$ is also a bisimulation between $\langle W, @(\pi_1 \ldots \pi_n)\rangle$ and $\langle W', @(\pi'_1 \ldots \pi'_n)\rangle$.

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. [12]):

4.1.8. THEOREM. *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. [12] ∎

As we have said, this criterion can be seen as a generalization of the permutation invariance above. In fact, they both ask that dynamic connectives inherit, so to speak, certain morphisms among the graphs of their arguments. In order to make this point more clear, we can easily see how permutation invariance is in fact only a particular case of the following 'isomorphism safety':

4.1.9. DEFINITION. [Isomorphism] A relation α between $\langle W, \pi_1 \ldots \pi_n\rangle$ and $\langle W', \pi'_1 \ldots \pi'_n\rangle$ is a *isomorphism* if the following conditions hold:

- α is a bijection
- $\langle w, v\rangle \in [\![\pi_i]\!]_W$ iff $\langle \alpha w, \alpha v\rangle \in [\![\pi_i]\!]_{W'}$

It is easy to see that by imposing the further condition that $W = W'$, α would become a permutation of the base set of states, and $\pi'_1 \ldots \pi'_n$ would be the permutation of $\pi_1 \ldots \pi_n$ induced by α (namely, $W' = \alpha W$ and $\pi'_i = \alpha\pi_i$, for $1 \leq in$).

4.1.10. DEFINITION. [Isomorphism safety] An operation $@(\pi_1 \dots \pi_n)$ on programs is said *isomorphism safe* if every isomorphism α between $\langle W, \pi_1 \dots \pi_n \rangle$ and $\langle W', \pi'_1 \dots \pi'_n \rangle$ is also an isomorphism between $\langle W, @(\pi_1 \dots \pi_n) \rangle$ and $\langle W', @(\pi'_1 \dots \pi'_n) \rangle$.

Note how weaker is isomorphism safety (and consequently permutation invariance) with respect to bisimulation safety: as weaker as 'isomorphism' is stronger than 'bisimulation'.

4.1.3 A link

Finally, we want to point to a further notion of 'logicality', that generalizes the previous permutation invariance (and is even broader than that). This notion will bring to the fore a link between permutation invariance and bisimulation invariance.

4.1.11. DEFINITION. [Permutation for π] A relation α between $\langle W, \pi_1 \dots \pi_n \rangle$ and $\langle W', \pi'_1 \dots \pi'_n \rangle$ is a *permutation for* π_i if the following conditions hold:

- α is a permutation
- $\pi_i = \pi'_i$

In other words, a permutation α of the states set W is a permutation for π if it holds that $\pi W = \alpha \pi W$. Note that in case π is 'functional' (i.e. it has unique values for each input), a permutation α which preserves the static formulas is a permutation *for* π iff it is a bisimulation *for* π from the base domain of states to itself.

Thus, here is our last notion of 'logicality':

4.1.12. DEFINITION. [Permutation safety] An operation $@(\pi_1 \dots \pi_n)$ on programs is *permutation safe* if every permutation α for $\pi_1 \dots \pi_n$ is also a permutation for $@(\pi_1 \dots \pi_n)$.

Therefore, we have the following connection:

4.1.13. THEOREM. *An operation $@(\pi_1 \dots \pi_n)$ on programs is permutation invariant only if it is permutation safe.*

Proof: Take any sequence $\pi_1 \dots \pi_n$. 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:

$\alpha(@(\pi_1 \dots \pi_n)) = @(\alpha(\pi_1) \dots \alpha(\pi_n)) = @(\alpha(\pi_1 \dots \pi_n)$

Moreover:

$\alpha(@(\pi_1 \dots \pi_n)) = \alpha^{-1}; @(\pi_1 \dots \pi_n); \alpha$

Combining these two facts, we get:

$@(\pi_1 \dots \pi_n); \alpha = \alpha; @(\pi_1 \dots \pi_n)$ ∎

4.2 Denotational Constraints

As the reader may have noticed, the style of analysis that we are pursuing in this chapter is inspired, to some extent, by the theory of Generalized Quantifiers (cf. [5] and [75]). At least, we are following one of the common strategies in the Generalized Quantifiers 'foundation' of grammatical types: we are constraining the space of the dynamic operations type by imposing conditions 'from the outside', so to speak (but see next section for an approach 'from the inside'). What we want to do now is following this analogy with GQ theory more closely, and applying the same kind of criteria that are used there to dynamic operators (cf. [14]).

Let us then start with recalling some basic notions from GQ theory. Quantifiers are here seen as 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:

4.2.1. THEOREM. *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. [5]. ∎

Here, we want to demonstrate how this style of analysis also applies to dynamic operators. In particular, we will briefly sketch a possible strategy for developing a GQ-style approach to UL semantics. This we will do by isolating a subclass of UL operators that can be treated just *as* generalized quantifiers.

In order to isolate this GQ-type subclass of dynamic operators, we need to further analyse their format. As we have already seen, a dynamic operator @ will usually be defined by a set-theoretic condition of the following form:

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

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 [6], which updates the current state with the result of processing π from the initial information state ('1').
The type of these conditions φ is still rather complex: they need to take into account a set of functions (the programs), an input set W and a 'reference' point w. It turns out that two type-lowerings are possible, by confining our attention to two special, but rather natural, classes of program operators, meeting respectively the following conditions:

- Extensionality: if $\pi_i(W) = \pi'_i(W)$ for $1 \leq i \leq n$, then $@(\pi_1 \dots \pi_n)(W) = @(\pi'_1 \dots \pi'_n)(W)$.
- Test Property: no reference point occurs in the defining condition (whence it has the following form: $\varphi(W, \pi_1 \dots \pi_n)$).

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:

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

where $A_1 \dots A_n$ are not functions but sets $(\pi_1(W) \dots \pi_n(W))$.

Concerning the test property, note that it is really the same property we have stated in definition 1.3.4: a program operator @ has the test property if, for all programs π and all input states W, the output of $@\pi(W)$ is either W itself or the empty set. It should be clear that this test property makes it possible one more type-lowering: from $\varphi(w, W, \pi_1 \dots \pi_n)$ to:

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

Therefore, extensional program operators with the test property will have the following reduced type:

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

or, more explicitly:

$$@(W, A_1 \ldots A_n)$$

Summing up, extensional test dynamic operators have the type 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 $\triangle$). Thus, an extensional test operators ranging on a set of states will have as its denotation a set of sets of states.

The nice consequence of this type-lowering is that 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 ($\Box$, $\neg\Box$, $\Diamond$, $\neg\Diamond$):

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

It should be noticed how some of these constraints acquire a different meaning in the dynamic set-up; for instance:

- CONS (= $@W, A$ iff $@W, A \cap W$) establishes a form of 'eliminativity', in that the input state already 'sets the stage' for the denotation of the @;

4.3 Case study: the dynamic negation

Let us now focus on a single dynamic operators, namely negation. Instead of imposing constraints 'from the outside' and designing dynamic negations accordingly, we will take into account actual semantic clauses for negation (concretely, the DPL and the UL clause for negation), and try to draw out their essential features by checking them against suitable conditions. In other words, following again the analogy with GQ theory, what we are after now is designing an *inverse logic* for dynamic negation.

4.3.1 Constraints

We will start our 'inverse logic' analysis of DPL and UL negation by referring to a couple of principles we have already encountered in the previous chapters (see paragraphs 2.2.3 and 2.3.2). They suggest us this basic intuition: running

a negated program $\neg\pi$ from any stage of the information processing should lead us to a (new) stage where the program π would fail if run.

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

The meaning of (1) and (2) is that once the processor has accepted the knowledge conveyed by a given program (or once he has performed the instruction expressed by a certain program), he can no longer change his mind.

Again, as we have already hinted, it turns out that these principles are too strong for the two cases we are considering, i.e. for the DPL and for the UL negation. 1 fails in fact in DPL, as it is shown by the following counterexample:

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

On the other hand, both (1) and (2) fail in UL. The following counterexample to (1) transforms into a counterexample to (2) by means of double negation (valid in UL):

$$[\![(\Diamond\pi;\neg\pi);\neg(\Diamond\pi;\neg\pi)]\!] \neq \emptyset$$

Here is a pictorial summary of the situation:

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

Given this, our next move within this inverse logic analysis will consist in trying to isolate the 'normal' fragment of UL and of DPL with respect to (1) and (2). This will be the issue at stake in the next two paragraphs.

4.3.2 Inverse logic for UL negation

We have seen how the negation clause in UL does not validate the above anti-revisionist principles. Still, it makes sense to isolate special subclasses of programs that make these principles valid. In this way, we are taking as variables of our 'inverse logic' policy dynamic programs: given a constraint, we are asking ourselves which programs do meet it.

4.3.1. PROPOSITION. *UL negation validates (1) for just those programs π that are* idempotent, *i.e. such that, for any information state W:*

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

Proof: $[\![\neg\pi]\!]([\![\pi]\!]W) = ([\![\pi]\!]W) - ([\![\pi]\!]([\![\pi]\!]W)) = ([\![\pi]\!]W) - ([\![\pi]\!]W) = \emptyset$ ∎

On the other hand:

4.3.2. PROPOSITION. *UL negation validates (2) for just those programs π that are* progressive, *i.e. such that, for any information state W:*

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

Proof: $[\![\pi]\!]([\![\neg\pi]\!]W) = [\![\pi]\!](W - [\![\pi]\!]W) = \emptyset$ ∎

Note that, as it is easily seen, a program π is progressive if and only if its negation is idempotent.

The art is now finding out the idempotent and the progressive fragments of UL. This will possibly bring to the fore interesting peculiarities of the update set-up. In order to do this, we need to prove the following simple facts (cf. definition 1.3.4 for the notion of test):

4.3.3. PROPOSITION. *For all UL programs π_1, π_2, if π_1 is idempotent and π_2 is a test, then the program $\pi_1;\pi_2$ is idempotent.*

Proof: Given an information state W, we want to show that:
$[\![\pi_1;\pi_2]\!]([\![\pi_1;\pi_2]\!]W) = [\![\pi_1;\pi_2]\!]W$
Now, since π_2 is a test, there are only two possible cases:
Case 1: $[\![\pi_2]\!]([\![\pi_1]\!]W) = [\![\pi_1]\!]W$.
Then: $[\![\pi_1;\pi_2]\!]([\![\pi_1;\pi_2]\!]W) = [\![\pi_1;\pi_2]\!]([\![\pi_1]\!]W) =$
$[\![\pi_2]\!]([\![\pi_1]\!]([\![\pi_1]\!]W)) = [\![\pi_1;\pi_2]\!]W$, since π_1 is idempotent.
Case 2: $[\![\pi_2]\!]([\![\pi_1]\!]W) = \emptyset$.
Then: $[\![\pi_1;\pi_2]\!]([\![\pi_1;\pi_2]\!]W) = [\![\pi_1;\pi_2]\!]\emptyset = \emptyset = [\![\pi_1;\pi_2]\!]W$. ∎

4.3.4. PROPOSITION. *For all UL programs π_1, π_2, if π_1 is progressive and π_2 is a test, then the program $\pi_1;\pi_2$ is progressive.*

Proof: Given an information state W, we want to show that:
$[\![\neg(\pi_1;\pi_2);(\pi_1;\pi_2)]\!]W = [\![\pi_2]\!]([\![\pi_1]\!](W - [\![\pi_2]\!]([\![\pi_1]\!]W))) = \emptyset$.
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. ∎

Moreover, it's easy to prove that tests and continuous programs (cf. definition 1.3.6) are both idempotent and progressive:

4.3.5. PROPOSITION. *For all UL programs π, if π is a test then it is idempotent.*

Proof: straightforward. ∎

4.3.6. PROPOSITION. *For all UL programs π, if π is a test then it is progressive.*

Proof: Given a state W, there are only two possibilities, given the fact that π is a test. Suppose $[\![\pi]\!]W = \emptyset$. Then $[\![\pi]\!](W - [\![\pi]\!]W) = [\![\pi]\!]W = \emptyset$. Next, suppose $[\![\pi]\!]W = W$. Then $[\![\pi]\!](W - [\![\pi]\!]W) = [\![\pi]\!]\emptyset = \emptyset$. ∎

4.3.7. PROPOSITION. *For all UL programs π, if π is continuous then it is idempotent.*

Proof: If π is state-continuous, then, given a state W, $[\![\pi]\!]W = W \cap P$, where P is the 'characteristic' set for π. Thus, $[\![\pi]\!]([\![\pi]\!]W) = (W \cap P) \cap P)$ ∎

4.3.8. PROPOSITION. *For all UL programs π, if π is continuous then it is progressive.*

Proof: If π is continuous, then $[\![\pi]\!](W - [\![\pi]\!]W) = (W - (W \cap P)) \cap P = \emptyset$. ∎

Using facts 1.3.8 and 1.3.11 we are now able to demonstrate the following corollaries of propositions 4.3.3 and 4.3.4:

4.3.9. COROLLARY. *The ;-free fragment of UL is idempotent and progressive.*

Proof: Obvious, from facts 1.3.8, 1.3.11, and facts 4.3.5-4.3.8. ∎

4.3.10. COROLLARY. *If π has the form: $\pi_1; \ldots; \pi_n; \pi_{n+1}; \ldots; \pi_m$, where π_i is continuous for $1 \leq i \leq n$ and π_i is a test for $n < i \leq m$, then:*

1. *π is idempotent*
2. *π is progressive*

Proof: Immediate from propositions 4.3.3 and 4.3.4 and facts 1.3.8 and 1.3.11. ∎

We have thus isolated a 'normal' fragment of UL with respect to the 'you can't change your mind' principles above. Next, we will consider the case of Dynamic Predicate Logic.

4.3.3 Inverse logic for DPL negation I

In this paragraph we will pursue the same task of isolating a 'normal' fragment for the case of DPL. We have already hinted to the fact that the principle (2) above is universally valid in DPL, as one can see just by looking at the definition of DPL negation. On the other hand, it is easy to prove that:

4.3.11. PROPOSITION. *DPL negation makes the principle (1) valid for just those programs π that meet the 'weak update property', namely such that:*

$$\forall w, w'(w' \in [\![\pi]\!]w \rightarrow [\![\pi]\!]\{w'\} \neq \emptyset)$$

We need one more definition:

4.3.12. DEFINITION. [Reflexive programs] A DPL program is said to be *reflexive* if for all states w the following holds:

$$w \in [\![\pi]\!]\{w\}$$

Note that only random assignments are in general reflexive.
Armed with this definition, we can now prove this fact:

4.3.13. PROPOSITION. *If π_1 is reflexive and π_2 has the weak update property, then $\pi_1;\pi_2$ has the weak update property.*

Proof: We want to prove that, given two states w and w', if $w \in [\![\pi_1;\pi_2]\!]w'$ then $[\![\pi_1;\pi_2]\!]\{w\} \neq \emptyset$. Thus, suppose $w \in [\![\pi_2]\!]([\![\pi_1]\!]w')$. From this it follows that $[\![\pi_2]\!]\{w\} \neq \emptyset$, because of the fact that π_2 has the weak update property. But then we can immediately conclude that $[\![\pi_2]\!]([\![\pi_1]\!]w) \neq \emptyset$, simply because π_1 is reflexive and therefore $w \in [\![\pi_1]\!]w$. ∎

We now need a bunch of facts along the line of the preceding paragraph:

4.3.14. FACT. Random assignments are reflexive.

4.3.15. FACT. Eliminative programs have the weak update property.

4.3.16. FACT. Reflexive programs are closed under ;.

Using the fact 1.3.10 we are now able to prove the following proposition, that gives us the aimed DPL normal fragment:

4.3.17. PROPOSITION. *If a DPL program π has the form: $\pi_1;\ldots;\pi_n;\ldots;\pi_m$, where π_i is an assignment for $1 \leq i \leq n$ and an eliminative program for $n < i \leq m$, then π has the weak update property.*

Proof: Straightforward from 4.3.13 and the facts above. ∎

4.3.4 Inverse logic for DPL Negation II

In this section, we will pursue another 'inverse logic' strategy applied to the DPL negation. This time, the 'variables' will be the semantic clauses: starting from a set of desiderata, we ask ourselves which actual clause for negation meet them. We use the relational algebra set-up, and prove the following (cf. [14]):

4.3.18. THEOREM. *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.

4.3.19. LEMMA. *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:

- $z = 1 - \{x\}$ is our intended choice.
- $z = \{x\}$. This is in conflict with 4.
- $z = 0$. This can be disposed of by the same argument as above.
- $z = \{y\}$. 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 $\neg(\{\langle x,y\rangle\}) =$ Id $-(\{\langle x,x\rangle,\langle y,y\rangle\})$. Then, $\neg(\{\langle x,y\rangle\}) = \cup_{z\neq x,y}(\{\langle z,z\rangle\})$. Hence, 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.

4.3.20. FACT. 2 and 3 imply that $\neg\pi \cap \pi = 0$.

Proof: By lemma 4.3.19 $\neg\pi \leq Id$. From this we have, using valid principles from Relational Algebra: $\neg\pi \cap \pi = \neg\pi \cap \pi \cap Id = (\neg\pi \cap Id) \cap (\pi \cap Id) = (\neg\pi \cap Id); (\pi \cap Id) \leq \neg\pi; \pi = 0$. ∎

In fact, one may observe that all non-validities for negation that we find in relational algebra seem to be refutable even within the special domain of DPL. Interestingly, there are other places where the parallel with relational algebra is intriguing; for instance, as it is remarked in [13], one of the key features of DPL, namely the rightward extension of the scope of η across sequential composition (as in $\eta x; \varphi; \psi$) is just an instantiation of the law for composition, namely Associativity:

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

These and other similar facts, motivate the following more general conjecture:

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

4.3.5 Discussion of the previous results

This paragraph will be devoted to showing how our 'foundational' analysis on dynamic connectives can raise interesting questions and disclose new points of view both on the linguistico-philosophical interpretation of Dynamic Logic and on its technical structure. In particular, we will try here to discuss and evaluate the eventual impact of 'inverse logic for dynamic negation' on the dynamic ideology.

Let us first briefly recall the nutshell of one of the inverse logic strategies we have proposed. In order to get a better insight on dynamic negation, we have asked ourselves if and when this connective behaves according to the two 'you can't change your mind' principles. It turned out that both the two principles are not universally valid in UL and that only one (...) is valid in DPL. The next move then consisted in finding out under which conditions the requested principles hold in UL and in DPL. What we are after now is clarifying the intuitive relation between the two principles and their validating conditions. In order to do this, we need to get back to a general distinction.

As we have said at various points, there are two main linguistic philosophical interpretations of dynamic programs (taking for granted the analogy states of a dynamic model = mental or informational states): they can be seen either as

proper information bits, in a knowledge-updating perspective, or as cognitive instructions, carrying 'information about the language'.

Now, it seems very plausible that the two principles on negation should hold for those programs that are to convey 'information about the world', unless a revision mechanism is available; take for instance the case of UL. UL typically represents a non-revisionist set-up where programs are read as pieces of information or as tests on the open possibilities. Thus, a UL update informing the processor that p is supposed to eliminate from the current epistemic horizon the worlds where p does not hold. And there is no way of getting these worlds back: this is the UL anti-revisionism. But DPL also is anti-revisionist, even if the point is here more subtle. Random assignment allows us in principle to retrieve worlds that have been previously eliminated, at least if the re-use of a variable is permitted. Still, it should be clear that this does not account for any kind of revision: the retrieving of lost worlds can only happen in a formula of the form '$\eta x; \ldots; \eta x; \ldots$', where the only revision that is going on concerns the knowledge about the language[1]. Given that, our question becomes now more precise: if UL and DPL represent knowledge-updating (where by knowledge we mean proper knowledge or knowledge about the world) in a non-revisionist style, then how comes that their negations do not meet the obvious 'you can't change your mind' requirements? This could be seen as a symptom of something wrong in the semantic clauses for DPL and for UL negation, something that do not agree with the intended interpretation of these set-ups. Fortunately, the analysis of the previous section allows us to make an encouraging diagnosis. Let us start with the case of Update Logic.

First of all, recall that both '$\pi; \neg\pi$' and '$\neg\pi; \pi$' are UL valid if π is sequential composition-free. Therefore, the current interpretation of UL programs as pieces of informations that progressively restrict the epistemic space of the processor, without any possibility of revision, looks unproblematic as far as ;-free programs are concerned. But we know more: we know that also idempotent/progressive programs (i.e. idempotent program whose negation is also idempotent) respect the anti-revisionist principles of paragraph 4.3.1. And this gives us the key for our optimistic diagnosis. If the principles in point do not always hold for UL programs, the reason is that non all UL programs can be seen as pieces of information. In other words, non all ;-sequences of UL programs are 'packages' of information. 'Maybe it's raining ; no, it is not raining' cannot be considered *as a whole* a piece of information. Rather, it represents a process, and it makes sense that negating a process does not respect the obvious laws for negating a piece of knowledge about the world in a non-revisionist set-up. Moreover, it makes sense that the negation is 'non-revisionist' for idempotent programs,

[1] processing the discourse: 'A man walks in ; he is tall ; a man walks in ; he is not tall' does not amount to revise our knowledge of the world. The only thing that gets revised is our knowledge about the language, in as much as we first use the description 'a man' to speak of some tall guy and afterwards to pick up a short guy

since idempotence is in a sense the hallmark of information: if a program π is a package of information, then processing it twice will produce the same effect as processing it once (namely the answer of the processor at the second time will be 'I know'). Therefore: if a program is an information chip then it is idempotent, and if it is idempotent then it respects the 'you can't change your mind' principles.

Or, in other words: UL negation is designed on the idea that programs are knowledge-updates, i.e. knowledge bits; and it works in the intended way when applied to knowledge bits.

Concerning DPL, we think the same intuition applies here. We have seen in fact how DPL fails to respect principle (1) of paragraph 4.3.1 for programs that do not meet the weak update property. We now briefly show that this principle amount in a sense to idempotence, and therefore we have once more that negation behaves in the intended manner when applied to idempotent programs = information bits. We have in fact that:

4.3.22. PROPOSITION. *In the regular fragment of DPL, a program π has the weak update property only if it is idempotent (where by 'regular' fragment we mean the fragment where: if $\pi = \pi_1; \ldots; \pi_n; \eta x; \ldots; \pi_m$ then x does not occur in $\pi_1; \ldots; \pi_n$).*

Proof: It is enough to prove that: if $v \in [\![\pi]\!]\{w\}$ then either $[\![\pi]\!]\{v\} = \emptyset$ or $[\![\pi]\!]\{v\} = [\![\pi]\!]\{w\}$. This we prove by induction on π.

π = atom trivial from the eliminativity of atoms

$\pi = \neg\pi_1$ trivial from the eliminativity of negated programs

$\pi = \eta x$ trivial from the semantic clause for η

$\pi = \pi_1; \pi_2$ in two steps:

1) if π_1 has the requested property and π_2 is a test, then $\pi_1; \pi_2$ has the requested property. Suppose that $v \in [\![\pi_2]\!]([\![\pi_1]\!]w)$. Therefore, since π_2 is a test, $v \in [\![\pi_1]\!]w$. But since π_1 has the requested property, there are two possibilities:

*) $[\![\pi_1]\!]v = [\![\pi_1]\!]w$, which implies that $[\![\pi_2]\!]([\![\pi_1]\!]v) = [\![\pi_2]\!]([\![\pi_1]\!]w)$;

**) $[\![\pi_1]\!]v = \emptyset$, from which it follows that $[\![\pi_2]\!]([\![\pi_1]\!]v) = \emptyset$.

2) if π_1 has the requested property and π_2 is an assignment, then $\pi_1; \pi_2$ has the requested property. Suppose that $v \in [\![\eta x]\!]([\![\pi_1]\!]w)$. Then there must be a state $u \in [\![\pi_1]\!]w$ such that $v =_x u$. But, since we are in the regular fragment of DPL and π_1 has the requested property, there are only two possibilities:

*) $[\![\pi_1]\!]v = ([\![\pi_1]\!]w)/x$, where $([\![\pi_1]\!]w)/x =_x [\![\pi_1]\!]w$; from which it follows that: $[\![\eta x]\!]([\![\pi_1]\!]v) = [\![\eta x]\!]([\![\pi_1]\!]w)$.

**) $[\![\pi_1]\!]v = \emptyset$, and therefore $[\![\eta x]\!]([\![\pi_1]\!]v) = \emptyset$ ∎

4.4 Common patterns across different systems

We hope to have shown that the notion of dynamic negation is quite controversial. In particular, we have seen how difficult it is to give a technical instantiation to an intuitive idea of 'negating a program', mainly because different programs within the same syntax do have different meanings and different behaviours (cf. the previous distinction between programs=information chips and programs=processes). Therefore, we now want to give a few general directions for defining semantic clauses for dynamic negations, that emerge from the actual clauses we have taken into account up to now. This will bring to the fore an intriguing common pattern that shows up at several points across different dynamic systems.

4.4.1 Local truth

All the dynamic systems we have mentioned so far (namely DPL, UL, TV, GSV, DMPL, TKV) follow in a sense the same strategy when interpreting $\neg\pi$: they basically curtail the input state, and eliminate a part of it that we shall call 'local truth of π'. The common intuition is in fact that a program $\neg\pi$ should transform the initial state into a state from which π would fail if run. Thus, we define the operation '$\top$', as the dynamic connective that, given a program π, picks from an input state the local truth of π. The negation clauses of the systems above can be defined according to the following common form:

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

Accordingly, it is possible to retrieve the semantic clauses for $\top$ in the systems examined so far, just by looking at their clauses for negation:

TV	$[\![\top\pi]\!]W = \{w \in W \mid [\![\pi]\!]\{w\} \neq \emptyset\}$
UL	$[\![\top\pi]\!]W = [\![\pi]\!]W$
DMPL	$[\![\top\pi]\!]^s_u W = \{w \in W \mid s = u \text{ and } \exists r \text{ with } w \in [\![\pi]\!]^s_r W\}$
GSV	$[\![\top\pi]\!]W = \{w \in W \mid \exists w' : w \leq w' \text{ and } w' \in [\![\pi]\!]W\}$
TKV	$[\![\top\pi]\!]W = \{w \in W \mid \exists w' : w \succ_\pi w' \text{ and } w' \in [\![\pi]\!]W\}$

Interestingly, there are other points in the above where the same operator is implicitly 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. [43], [70]), in particular:

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

[2] which gets parametrized in TVK

which amounts in practice to the following:

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

And obviously, $\top$ plays a role in defining dynamic implication. Thus, it becomes of interest to pursue some characteristic semantic properties of the local truth operator $\top$ by itself.

4.4.2 Technical discussion

Let us now sketch a brief technical analysis of this new dynamic operator.

4.4.1. DEFINITION. [Normality] The operator $\top$ is *normal* iff for all programs π, for all information states W, the following conditions hold:

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

More concretely, if $\top$ is 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).

4.4.2. DEFINITION. [$\emptyset$-preservation] $\top$ is *$\emptyset$-preserving* if for all states W and programs π:

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

4.4.3. DEFINITION. [$\emptyset$-continuity] π is *$\emptyset$-continuous* if for all W, W':

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

4.4.4. PROPOSITION. *If $\top$ is $\emptyset$-preserving and 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:*

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

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

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

4.4.5. PROPOSITION. *The ;-free fragment of TV is normal.*

Proof: Easy induction on ;-free formulas. ∎

4.4.6. PROPOSITION. *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:

4.4.7. PROPOSITION. *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. 4.1.5).

4.4.8. PROPOSITION. *If $\top$ is normal then it is permutation invariant.*

Proof: By a simple calculation, using the fact that $\top$ has a set-theoretic definition. ∎

Then, from a logical 'local truth' operator we can obtain a logical negation and logical modalities:

4.4.9. PROPOSITION. *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:
$[\![\neg(\alpha;\pi;\alpha^{-1})]\!]W =$
$[\![\alpha^{-1}]\!]([\![\neg\pi]\!]([\![\alpha]\!]W)) =$
$[\![\alpha^{-1}]\!]([\![\alpha]\!]W - [\![\top\pi]\!]([\![\alpha]\!]W)) =$
$W - [\![\alpha;\top\pi;\alpha^{-1}]\!]W =$
$W - [\![\top(\alpha;\pi;\alpha^{-1})]\!]W =$
$[\![\neg(\alpha;\pi;\alpha^{-1})]\!]W$ ∎

4.4.10. PROPOSITION. *If $\top$ is permutation invariant then the matching modality is permutation invariant.*

Proof: Analogous. ∎

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

Chapter 5

Philosophical Repercussions

This chapter is meant as a 'philosophical agenda'. We do not develop a full dynamic investigation of a specific logico-philosophical problem. Rather, connecting up with the philosophical themes that have emerged throughout the dissertation, we suggest a few possible dynamic strategies to be developed - hopefully - in future work. Besides this programmatic character, the following philosophical remarks are also meant as a conclusive moral of the whole story. The story started with a list of extra-logical questions that eventually support our approach to Dynamic Logic, in that, as we claimed, they can be faced within an enlarged dynamic set-up. Now, we want to end the story by getting back to a few (philosophical) items in that list: we will briefly show (or in some cases recall) how the dynamic framework that emerges from the present dissertation can say something about them.

5.1 Dynamic Logic for the Dynamic shift

Reading back the opening section of Chapter 1 ('From truth to action') the first philosophical theme we encounter is the metaphor mind/computer. As we said, this intriguing metaphor was not originally born in a purely philosophical environment. Rather, this parallel was inspired by the basic principles of Artificial Intelligence and consequently by the so-called informational approach in Psychology. Still, since what we are interested here is a general cultural trend (including these disciplines), we will consider this metaphor at a more general level, namely in its philosophical import.

A little further within the same section, the wittgensteinian *language games* come into play, witnessing once more the feasibility of a dynamic approach; then the Lewis 'score board' metaphor, and finally the so-called 'dynamic conception

of meaning', in turn the newest and most actual ground of dynamics.

As the reader will remind, Dynamic Logic is then advocated (cf. section 1.2) as a suitable formal tool for modelling and representing this cultural dynamic shift, as it shows up in the different (philosophical) contexts above. In the course of the whole dissertation, we have tried never to forget these - deep, at least according to our point of view - motives for enlarging the boundaries of traditional informatic-oriented Dynamic Logic. Still, it is now time to make the point of the situation. Now that the reader has seen how the 'generalized' Dynamic Logic we wanted to build (or to sketch) looks like, we can fairly test to which extent this formal apparatus can model the phenomena above.

Let us start then with the parallel mind/computer. The idea of comparing human minds with microchips is certainly a challenge for philosophers, in that it brings to the fore new questions and possibly new insights on the way we think, understand, learn etc. We do not dare giving an historical reconstruction of the whole debate; rather, we will focus on one specific issue that we find particularly relevant to our purposes.

When a computer receives an input of some sort, it will start processing it according to the instructions of the programmer, and finally provide the user with the requested output. The process between the input and the output will be often ignored by the user, that will keep track instead of the flux inputs-outputs. This set-up, we think, suggests one of the most deep guiding allegories of contemporary studies on human intelligence: that we are like users of our minds, that process information according to unknown programs, without being conscious of what happens between the sensorial input and the emotional or behavioural or whatever output. Note that, as we said, this is in a sense the proper subject of Cognitive Psychology: what happens between the stimulus and the answer, no matter the misleading image our conscience suggests us. The influence of this allegory in the recent debate in the philosophy of mind community is testified for instance by the influential book by Hofstadter and Dennet [50], in which the question is tackled if this perspective on human minds allows us to maintain the conviction (so important for our western culture) that we do have a soul and a conscience. Now, what are the repercussions of the dynamic-logical framework on this point of view, suggesting that mysterious processes goes on in our brains while we talk, listen, understand etc.? As a formal tool, (extended) Dynamic Logic can precisely represent the mental processes that run between the stages of the elaboration of knowledge, by means of its many-aimed programs, that do not necessarily mimic the way actual computers function. In the variational perspective we have been suggesting, for instance, the process of selecting and labelling objects is formalized by means of the program η, being this (presumably unconscious) process the basis of the competent use of pronouns. Besides, thanks to its two typed syntax and semantics, it can account for the interface between static inputs/outputs and dynamic processes. An im-

portant proviso has to be made here: Dynamic Logic *per se* does not impose any commitment on how these processes are actually structured and work. It only provides with a suitable *formal language* to describe them, or, to be precise, to describe the various theories about them.

Concerning Wittgenstein's language games, we do not dare saying that Dynamic Logic would constitute a *formal paradigm* of them. We do not even think that the pure wittgensteinian word would admit the legitimacy of formalizing linguistic phenomena! What we want to stress here is rather a sort of deep general intuition about how language works that, according to our view, emerges from both the Philosophische Untersuchungen ([76]) and the new dynamic approach in logic and in philosophical logic. According to this intuition, language is not something traced after reality, so to speak; rather, it emerges in its manifold variety out of a number of cognitive processes, out of use, out of a communitary practice. Language games are a beautiful philosophical tool, and they emphasize the virtual infinite complexity of this dense intertwine of life, mental activities and language. On the other hand, Dynamic Logic is a formal set-up, and it confines itself to the more modest analysis of small fragments of that complex intertwine. Still, it carries this analysis by putting the accent on the cognitive activities that are 'behind' the external linguistic expression. For instance, the use of modalities is accounted for in dynamic terms in such a way as to bring to the fore the process of testing/updating that underlies them.

Let us now take into account the 'score board' metaphor by David Lewis, as described in [58]. As it is shown in [29], this metaphor well encapsulates the spirit of Dynamic Logic in its application to the study of Natural Language. According to this metaphor, the process of understanding Natural Language is somewhat similar to the process of updating a score board within the course of a game (with answers to questions like 'who has thrown the dice?', 'who is winning?' and so on). They are both aimed at accounting for the eventual changes that occur in the context. Accordingly, Dynamic Logic represents the successive process of updating one's picture of the world (the context of the game), on the basis, again, of the changes that occur. Remember for instance the semantic variation of the parameter Interpretation, that can well model the acceptance of an answer to a question like 'who has the property P?'. A sequence of programs of Dynamic Logic can be seen as a sequence of instructions for updating a score board, where the game can consist in understanding a situation, or in forecasting how the situation will develop, or *simply* in competently using linguistic resources.

Concerning the dynamic conception of meaning, we hope to have shown what it amounts to along all the previous chapters. Now, we will focus on a special theme, very up-to-date in the 'dynamic community', and consider it as a case study for expounding once more this new approach to meaning.

5.2 Individuals and Modality

In this section we want to focus on an issue that we have already encountered at various points within the previous chapters. Namely, we will now discuss at some length if and how the dynamic apparatus (in the format we have investigated) can be fruitfully used for solving some thorny problems that arise when making modal statements about individuals. More precisely, we will check if and to which extent a dynamic interpretation can clear up some deep doubts concerning Quantified Modal Logic (QML for short). The present section will be devoted to drawing a rough map of the problems in QML, while also sketching the main 'static' solutions that have been advocated in the literature. Section 5.3 will consider the possible dynamic 'impact' on this controversial area.

5.2.1 The main problems

The question of making modal statements about individuals raised an extensive and 'hot' debate within the course of the current century. Given that, we do not aim at an all-embracing historical reconstruction. Rather, we will try to spell out the most fundamental problems and objections that have often threatened the legitimacy of QML.

The first problem we want to face is very dramatic, and directly brings us to the work of W.O. Quine, that certainly has been the most inexorable censor of QML. His main technical contribute to the (attempted) liquidation of QML consisted in bringing to the fore the so called question of 'quantifying in'. Before going into details, we resume Quine's fundamental claim: quantifying into modal (more generally, referentially opaque) contexts is not technically correct, and leads us to a puzzling dilemma. Let us see the argument step by step.

First, a matter of fact: there are contexts where the apparently obvious law of substitutivity of identical fails. These context are called, by definition, referentially opaque. Among them, there are modal contexts (together with belief contexts, quotation contexts, and so on. We assume the readers has a certain familiarity with this very well-known typization). An easy example. The following statement seems out of discussion:

9 is necessarily greater than 7

Moreover:

9 = number of planets

But - here the failure of the 'Leibniz law' shows up - it is false that:

The number of planets is necessarily greater than 7

Therefore, the context 'it is necessary that ...' (or, in symbols, '$\Box$ ...) is referentially opaque, since within it the Leibniz laws does not hold. In other words,

the singular terms that occur within such a context are not *purely referential*, since not only *what* they denote matters, but also *how* they denote it.

Second step: since the ultimate mean of reference in a theory are the variables of quantification, the failure of the substitutivity of identicals must have an effect of some sort at the very level of quantification. Quine takes into account the rule of *existential generalization*, and checks what happens when it is applied to referentially opaque contexts. Following up with the previous example, take the sentence:

9 is necessarily greater than 7

By applying the rule in question, we get:

$\exists x(x$ is necessarily greater than 7)

But, is Quine's qualm, what is the number that has the property of being necessarily greater than 7? 9? or the number of planets? A clear quotation: 'In a word, to be necessarily greater than 7 is not a trait of a number, but depends on the manner of referring to the number'.

Third step, i.e. the moral: the application of a quantifier to a variable within a referentially opaque context produces 'unintended sense or nonsense'. Namely, quantifying into referentially opaque contexts is not formally sound.

The second technical difficulty with QML is of a totally different nature. While the question of 'quantifying in' concerns the very rightfulness of QML, this second point we are going to discuss takes it for granted, so to speak, and involves an 'inner' feature of QML systems. This point we have encountered at many stages in the course of the present work: how is it possible to express both de re and de dicto modalities within one single logical system? As we shall see in what follows, this becomes a relevant issue also in connection with the problem of 'quantifying in', that can be possibly clarified if not solved, by a fine-grained formalization that accounts for both interpretations of modalities (cf. paragraph 5.2.2). Another, maybe more important, motive for facing this question concerns the application of QML to the analysis of Natural Language. It is quite uncontroversial that both the de re and the de dicto reading of intensional operators are 'at work', so to speak, in common linguistic practice. By the way, this is already a good reason for justifying the philosophically dubious[1] de re modalities. As it is properly remarked in [39], it is not methodologically correct to let a philosophical objection of any kind to influence our analysis of Natural Language: 'we want descriptions of how we speak, not of how we would have to speak in order to earn the approval of philosophers' (cf. [39], p. 47).

[1]Herewith we have only taken into account the technical criticism by Quine to QML. Still, the quinean problem is more complex, since he refutes - in most cases - de re modalities because they commit one to essentialism. We will not pursue this issue here, for we consider it somehow less actual than the 'concrete' technical questions about QML.

Finally, we want to stress a last question concerning the theme 'Individuals and Modality'. This question is in a sense marginal in the literature, but we think it is quite relevant from the point of view of Philosophy of Language and, again, of Linguistics. The question, that we have already implicitly mentioned in Chapter 3, looks as follows: which part is to be played by the choice of models and which by the syntactical distribution of modal operators in order to determine their intended reading?

5.2.2 A few solutions to the question of 'quantifying in'

In this paragraph we will briefly recall (some of) the solutions that have been advocated to the first problem above. Although the question of 'quantifying in' won't be directly involved in our dynamic analysis, we think it can be useful to outline the main strategies, in that they connect up with the key problem we will face 'dynamically', namely the twofold reading of modal operators.

The solutions to this dilemma, if by solutions we mean the attempts to license QML, are all aimed to show that Quine's argument against quantification within opaque contexts has a gap of some sort.

We now outline a list of what we consider the more significant strategies in finding out this advocated gap:

Kaplan in [55] provides a quasi-formal reconstruction of Quine's argument against the quantifying in; in this reconstruction, he finds a wrong step, caused by a hidden confusion between non-referential occurrences and non-referential contexts; this is an interesting point, since, as we have already hinted in Chapter 3, it is not clear how the intended reading of a modal sentence should depend on its structure (i.e. on the context) or on the intension towards terms (i.e. on the interpretation of occurrences). We will get back to this issue in paragraph 5.3.1.

Fine in [37] claims that Quine's criterion for non-referentiality (i.e. the failure of Leibniz law) is too narrow, in as much as it rules out referential terms too. Example: take 'The man behind Fred saw him leave'. Suppose 'the man behind Fred = the man before Bill'. Although 'the man behind Fred' in the first sentence looks certainly referential, substitution with 'the man before Bill' does not preserve the truth: 'The man before Bill saw him leave' is false.

Plantinga in [63] demonstrates that the various cases where the Leibniz law fails all display a certain confusion between the de re reading and the de dicto reading. For instance, in the example of the number of planets, the conclusion ('the number of planets is necessarily odd') is surely false if read de dicto, but it becomes unquestionably true if read de re.

This last strategy rises a relevant point, in as much as it connects up with the second problem of paragraph 5.2.1. If it is true that the distinction de re/de dicto can contribute clearing Quine's quandaries, then the question of finding a way for representing both the two readings of modalities becomes more urgent!

5.2.3 Possible worlds semantics

As we have already said at various stages (cf. in particular Chapter 3), there are manifold 'degrees of freedom' when designing a possible worlds model of Quantified Modal Logic, and different choices at the semantic level correspond to different interpretations of modal operators, to different 'ontological' assumptions, and to different technical traits. We do not want to go through the whole range of possibilities here. Rather, we will focus on the choices that are more relevant with respect to our favourite philosophical point, concerning the distinction de re/de dicto. There are two main ways:

Rigid Terms As the reader will know, a possible worlds model with Rigid Terms will assign to any term of the language[2] the same object through all possible worlds. Thus, terms are treated as proper names, with no intensions. From another point of view: the peculiarity of possible worlds models with Rigid Terms lies in that they interpret terms as *individuals*.

World-relative Terms All terms, including variables, possibly become 'non purely referential' (cf. above). In other words, it may be said that terms are interpreted here as *individual concepts*. This definition comes from Carnap [19], but the common intuition behind individual concepts is not faithful to the carnapian word: an individual concept is often pictured as a sort of story (temporal, if possible worlds are interpreted as points in time, or 'metaphysical', in case they are mere possibilities) of an individual. Interestingly, an individual concept can also be pictured as a definite description in its 'attributive' reading (for the distinction between 'attributive' and 'referential' use of definite descriptions see [25]): ' the killer', namely the person who has committed the murder, no matter who he actually is (he may have different instantiations in different possible worlds).

Let us see now what these alternatives can tell us concerning the two readings of modalities.

If one interpretes a QML system over a possible worlds model with Rigid Terms, the true advantage is that the Leibniz law above holds without restrictions (given the fact that identities are all necessary!), and de re modalities are

[2]Obviously, semantics are available where only variables or only constants are fixed. We will not pursue this distinction here, since we think that the reading of modal operators within a logical system, linked for many aspects to the interpretation of terms, is basically transverse ..

properly expressed within this context. On the other hand, problems arise concerning de dicto modalities. In the case of properties of singular terms, like in the sentence 'the killer is bad', one can make it de dicto necessary by performing a syntactical manoeuvre:

$$\Box\ (Kx \rightarrow Bx)$$

opposite to the de re counterpart:

$$(Kx \rightarrow \Box\ Bx)$$

This is standard. But we want to stress that a pretty strong assumption is implicit here: namely that the intended reading of modal operators is a function of their syntactic distribution. This is at least controversial if we take the instance of representing Natural Language seriously: the phenomenological evidence supporting this claim is not strong enough.

The situation becomes even worst when trying to represent modal identities: as we said, rigid terms make all identities necessary. Which means that two relevant modes in Natural Language are ruled out: first, totally de dicto identities, that may be contingent (it is not necessary that the US Postmaster being the same individual as the inventor of bifocals). Second, mixed modal identities, that say about a certain individual, labelled as x, that he could have the property of being the same as y; more clearly, suppose we want to say about the individual who happened to be the postman (about *him* - de re intention), that he could be the killer. In this case, we want to keep track of our postman through all possible worlds, and check if somewhere this individual happen to have the property of being equal to y, the killer (de dicto intention).

We can now consider the case of models with World-relative Terms. In fact, we have already seen what happens with dropping the assumption that terms are rigid: the system that we have called static GSV was nothing but a system of Predicate Modal Logic interpreted over possible worlds models with World-relative Assignments. We recall that the advantage of that system is that both a de re and a de dicto reading of modal operators is accounted for. But we already know that the price for achieving this is quite high. First, in connection with the last question of paragraph 5.2.1, we have already stressed that the common interpretation of quantification within a World-relative Terms semantics commits to a strong assumption: that the reading of modalities is a function of their syntactical distribution. Or, in other words, the fact that terms are not rigid implies that they are interpreted as *individuals* when they are within the scope of a quantifier, and as *individual concepts* when they are outside the scope of quantifiers. This is certainly a way for coping with both individuals/de re intensions from one hand and individual concepts/de dicto intentions from the other; but, once more, we doubt that this really catches what goes on in Natural Language. Finally, we want to mention a typical technical problem of models with World-relative Terms: classical principles of

instantiation become here invalid. An example:

$$\forall x \varphi x \not\rightarrow \varphi t$$

As remarked in [40], this clearly shows that in World-relative Terms models, quantification acts on rigid terms, while terms in general are non rigid.

Summing up, we don't know of any uncontroversial representational tool here. Multimodal logic can be of some help, in as much as it can blend the different possible models for QML within the same semantics. Still, the solution is not uniform. We have seen as systems with Worlds-relative Terms, like static GSV, can model both readings at the same time. But even there, we hope to have shown that a questionable assumption has to be made assuming a link between the interpretation of intensional operators and the binding structure of the discourse.

5.3 Dynamic Individuals and Modality

5.3.1 What is on the market?

We have seen basically three kinds of dynamic systems that seems to be good candidates for saying something about the old querelle on individuals and modality. Let us briefly sum up their main contributions.

The system DMPL, opting for Fixed Assignments, is not fine-grained enough as to give a satisfactory account of the double reading of modal operators. In particular, DMPL validates the following principle:

$$\Diamond\, s = t \rightarrow \Box\, s = t$$

From this point of view, DMPL does not differ from any static system of Modal Predicate Logic interpreted over Fixed Assignments models. Still, DMPL, inspired by [43] and by [44], adds an important question, if not a solution, to the debate on individuals and modality: how to cope with the standard puzzles of the literature when pronouns are involved? In DMPL it becomes possible to make de re modal statements about pronouns without violating the principle of compositionality; but further system (cf. what follows) give further contributes to this point.

The system GSV represents the distinction de re/de dicto by adopting a semantics with World-relative Assignments. This leads to a different reading of the existential quantifier, that searches for its witnesses by setting the whole logical universe (namely all the possible worlds) on a value at a time. To sum up our general claim, this option presupposes a controversial assumption on the structure of Natural Language: according to the GVS interpretation of existential quantification, a modality is to be read de re when inside the scope of an existential quantifier, de dicto when outside. Since existential quantifiers

are more or less explicitly at work in many places in Natural Language, this is also a commitment on the syntactic linguistic occurrence of de re and de dicto modalities. More generally, GSV assumes that terms are interpreted as individual concepts, except in case they are inside the scope of a quantifier. Again, we don't find any uncontrovertible linguistic evidence fo this. Moreover, we want to stress that this particular solution to the problem of quantified modality is not, once again, dynamic at all, since, as we have shown in Chapter 3, the same reading of existential quantifiers can be adopted for a static system.

Still, the system GSV adds an important contribution to the question of modalities applied to pronouns. As we hinted, this question is by now the most original contribute of the dynamic approach to the querelle on individuals and modality. Thanks to the World-relative Assignments, GSV manages to give a very fine-grained treatment of identities involving pronouns; we recall here (but see Chapter 3 for a more detailed explanation) the following two discourses:

1. Someone has done it. There is someone hiding in the closet who might be the one who has done it.

2. Someone has done it. There is someone hiding in the closet. He might be the one who has done it.

that get the following formalization in GSV:

1. $\eta x Dx; \eta y(Qy; \Diamond y = x)$

2. $\eta x Dx; \eta y Qy; \Diamond y = x$

Interestingly, the $\Diamond$ in the first formula is 'de re' only with respect to y, since it is inside the scope of ηy. (i.e. y has the property of being possibly equal to x. Thus $\Diamond y = x$ does not imply $\Box y = x$). Rather, the $\Box$ in 2 is totally 'de dicto'. Unfortunately, as we have shown in Chapter 3, GSV does not allow a convincing formalization of a discourse like:

There is someone hiding in the closet who might be Alessandro

where Alessandro is given a fixed instantiation in all possible worlds.

Finally, the system TKV manages to account for both the de re and the de dicto reading of modal operators, via a refinement of both the syntax and the semantics: the first one explicitly distinguishes among different modalities, while at the level of models this corresponds to different kinds of accessibility relations. In one word, this multimodal dynamic system uses the standard idea of coping with the different readings of modalities by means of different constraints within the same model, with corresponding modal operators. Again, its peculiarity lies in its being multimodal, and not quite in its being dynamic. Still, the treatment of modal puzzles involving pronouns is quite satisfactory, since it combines the advantages of the η (allowing a compositional treatment of intersentential bindings) with the advantages of an extremely fine-grained kit of modalities.

5.3.2 Waiting for a dynamic cut

In this section, we want to suggest a possible purely dynamic approach to the problems we have been considering. This is not a structured proposal, rather we aim to hint at a feasible *direct* application of the dynamic apparatus to the problem of individuals and modality. Recall that the dynamic contributions to this issue we have see so far are rather of an heuristic kind, in that they suggest new questions and points of view instead of positively proposing new solutions.

We take our cue from the last 'static' strategy that we have mentioned in the previous paragraph. More specifically, we assume that the puzzles concerning the Leibniz law and consequently the quantification within opaque contexts, depends on a subtle back and forth shift between de re reading and de dicto reading.

9 is necessarily odd

seems to Quine above discussion. We are tempted to say that he gives a de re reading of it. He would probably reply that no, it is a de dicto reading. In fact, under the assumption that the name '9' is a rigid designator, the distinction vanishes. But this is controversial. In other worlds, '9' could name something else. On the other hand:

the number of planets is necessarily odd

is false according to Quine. We have already said that it is de dicto false. De re it would be true. Moreover, the identity:

9 = number of planets

is modally unspecified, since it is read de re (in the sense that both '9' and 'the number of planets' are purely referential). But it would become contingent if read de dicto (requiring then an explicit modal specification).

The moral we want to infer is that Quine seems to give to any modal statement a sort of *preferred* reading, independent from the syntactical structure. This makes a typical question for the dynamist arise: how does the mechanism for picking out the intended reading of a modal operator work? We can make some examples in order to show how this question makes sense.

Suppose someone has been murdered. Suppose you have seen the killer. In some sense, you *know* him. You talk about *him*, i.e. about the individual you have seen. You could say something like 'he is surely very scared', because you saw he was trembling. This epistemic modality 'surely' is surely de re: you talk about an individual, not about a description.

On the other hand, take the same situation. Suppose you don't know the killer, in any sense. You only know there is a killer. And you are investigating. Since the homicide was very cruel, you utter the following: 'he is surely very violent'. But in this case, you are talking about a description, about 'the killer', and the 'surely' is surely de dicto.

This example shows an interesting feature of the distinction de re/de dicto. It shows that the syntactic distribution of modal operators is not the only relevant fact in order to determine their reading. Sometimes, it is a matter of *cognitive focus*!

Other interesting examples can be done about identities. We can take the case of the guy hiding in the closet who might have done it:

> Someone has done it. Someone is hiding in the closet. He might be the one who has done it

In [45], the authors make an interesting claim. In case you know a little something about that someone (in particular: it is not your elder son, since you heard a subtle voice from the closet, and you elder has already a rough voice), then you are inclined to make de re modal assertions about that someone. In other words, knowing something makes your focus pass from a mere description to a (partial) individual. Thus, when saying 'he might be the one who has done it' you are meaning to read the might de re with respect to 'he' (namely with respect to the hiding guy), while presumably de dicto with respect to 'the one who has done it' (in other words: you are using the pronoun 'he' to pick a partial individual, while 'the one who has done' is just a description). On the other hand, if you don't know absolutely anything about the guy who is hiding, then you are lead to stay at the de dicto level. 'He might have done it' becomes then totally de dicto.

If all these examples make some sense, than this is a good dynamic question: how does the mechanism of 'cognitive focus', guiding the interpretation of modalities, work?

It should be clear that we are already in the middle of the second technical problem of paragraph 5.2.1. In fact, bringing to the fore the process of selecting the intensions of modalities would answer - in dynamic terms - the representational question also. What we want to shortly remark here is the following: as we have already claimed in Chapter 3, the approach to the problem of representing de re and de dicto modalities as offered in [45] is not dynamic. We have shown that a static version of GSV can be designed that provides us with the same approach to this representational question. That solution is still bound to the view that the reading of modalities depends on their syntactical distribution. The dynamic cut we advocate suggests a different perspective: the reading of modalities depends on a cognitive process of some sort, that is just to be unveiled. We hope that the system TKV, presented in Chapter 3 can represent a first step in this (complex) research program.

Appendix A

More on Dynamic Modal Predicate Logic

In this Appendix we will give a detailed description of the system DMPL (cf. Chapter 3), while also providing a complete axiomatization and a large set of examples of its application to Natural Language analysis.

A.1 Semantics for DMPL

The syntax of DMPL is like the syntax of DPL, but with a construction for epistemic 'might' added:

DMPL $\pi := Rt \cdots t \mid t = t \mid v :=? \mid \pi;\pi \mid \neg\pi \mid \Diamond\pi.$

We evaluate with respect to sets of first order models over the same universe M, i,e., we consider a DMPL model $\mathcal{M}$ as a pair $\langle M, W\rangle$ where W is a set of first order interpretations over M. Variable assignments for $\mathcal{M}$ are elements of M^V. Index sets for $\mathcal{M}$ are subsets of W.

We give a functional semantics that maps triples consisting of an index set, an input assignment and an output assignment to a new index set. Intuitively, ${}^{\mathcal{M}}_{s}[\![\pi]\!]^{I}_{u} = J$ means that given input assignment s and output assignment u for $\mathcal{M}$, π maps index set I to index set J. Suppressing the parameter $\mathcal{M}$ for ease of reading, we can express the same as $[\![\pi]\!]^{s}_{u}(I) = J$. The advantage of this functional formulation is that it clearly shows the two dimensions of parametric variation, the dimension of assignments and the dimension of index sets, with their interplay.

The index set W pictures the case of complete ignorance. Of course, even in this case one has to make up his mind about what the pronouns (free variables) are supposed to refer to, both at the start and at the end of the processing;

namely, one has to fix an input and an output assignment function to be able to compute an output index set. If the discourse produces new information, the index set that results from the semantic processing will be a proper subset of the initial index set. On the other hand, a state of maximal information is given by an index set $\{i\}$, indicating that i is the only state of affairs compatible with one's information. Finally, the index set $\emptyset$ pictures the case of inconsistent information: no state of affairs at all is compatible with what one knows or believes.

A.1.1. DEFINITION. [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. $[\![t_1 = t_n]\!]^s_u(I) = \{i \in I \mid s = u \text{ and } M, i \models_s t_1 = t_2\}$.
3. $[\![\pi_1;\, \pi_2]\!]^s_u(I) = \{i \in I \mid \text{there is an } r \text{ with } i \in [\![\pi_2]\!]^r_u([\![\pi_1]\!]^s_r(I))\}$.
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. $[\![v :=?]\!]^s_u(I)$

   $= \{i \in I \mid u = s(v|d) \text{ for some } d \in M\}$

   $= \begin{cases} I & \text{if } u = s(v|d) \text{ for some } d \in M, \\ \emptyset & \text{otherwise.} \end{cases}$
6. $[\![\Diamond\pi]\!]^s_u(I)$

   $= \{i \in I \mid s = u \text{ and there is an } r \text{ with } [\![\pi]\!]^s_r(I) \neq \emptyset\}$

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

The clauses for basic relations $Rt_1 \cdots t_n$ and identities $t_1 = t_2$ say that atomic predicates serve to weed out the set of indices, given what the assignment function tells us about the referents of the variables. For every input index set, the output index set simply consists of those items from the input index set that satisfy the predicate, given the assignment function, which itself remains unchanged. Note that identity is a logical relation which behaves the same in every world.

The clause for program concatenation says that the members of the output index set of the concatenated program are the output index set of the second program, when applied on the output index set of the first program for the original input index set, provided that some intermediate assignment function establishes the link between the input and the output assignment. Thus, along the index set dimension we have composition of update functions, as in UL, while along the assignment dimension we have relational composition, as in DPL.

Similarly, negation is boolean complement along the index set dimension, but computed in terms of possible assignment continuations along the assignment dimension.

As in DPL, dynamic implication $\pi_1 \Rightarrow \pi_2$, is defined as $\neg(\pi_1; \neg\pi_2)$. The DMPL semantics gives rise to the following clause for $\Rightarrow$:

$$[\![\pi_1 \Rightarrow \pi_2]\!]^s_u(I) = \{i \in I \mid s = u \text{ and}$$

$$\text{for all } r \text{ with } i \in [\![\pi_1]\!]^s_r(I) \text{ there is a } p \text{ with } i \in [\![\pi_2]\!]^s_p([\![\pi_1]\!]^s_r(I))\}.$$

Thus, dynamic implication weeds out those indices that for some intermediate assignment function r satisfy the antecedent program but do not have an output assignment for which they also satisfy the consequent program.

The clause for random assignment to v checks whether the output assignment function is a v variant of the input assignment and returns the input index set if it does, the empty set if not.

Finally, the clause for $\diamond$ checks whether the output assignment equals the input assignment and whether it is consistent with π.

As in UL, we have an elimination lemma:

A.1.2. LEMMA. *For all programs π of DMPL, all models $\mathcal{M}$, all index sets I and assignments s, u:*
$[\![\pi]\!]^s_u(I) \subseteq I$.

Also, it is easy to semantically characterize the test programs. Test programs are the programs for which $[\![\pi]\!]^s_u(I) \neq \emptyset$ implies that $s = u$.

Acceptability and acceptance are defined in the spirit of UL.

A.1.3. DEFINITION. A program π is *acceptable* (in model $\mathcal{M}$) for input index set I and input assignment s if there is an u for which $[\![\pi]\!]^s_u(I) \neq \emptyset$.

Intuitively, if the information conveyed by program π is accepted by index set I then it does not weed out any of the current epistemic alternatives:

A.1.4. DEFINITION. A program π is *accepted* (in model $\mathcal{M}$) by index set I, given input assignment s, if there is an u for which $[\![\pi]\!]^s_u(I) = I$.

Validity is defined in terms of acceptance, as follows.

A.1.5. DEFINITION. A program π is *valid* if for all models $\mathcal{M}$, for all I, s for $\mathcal{M}$, π is accepted by I, given s.

Notation: write $\downarrow [\![\pi]\!]^s(I)$ for $\{i \in I \mid \text{there is some } u \text{ with } i \in [\![\pi]\!]^s_u(I)\}$.

A.1.6. DEFINITION. π_1 *dynamically entails* π_2, notation $\pi_1 \models \pi_2$, if for all models $\mathcal{M}$ all index sets I and assignments s, u for $\mathcal{M}$,

$$\downarrow [\![\pi_2]\!]^u([\![\pi_1]\!]^s_u(I)) = \quad [\![\pi_1]\!]^s_u(I).$$

We leave it to the reader to check that this definition combines the features of DPL entailment and UL entailment, both in the appropriate dimensions.

To see that DMPL is a conservative extension of Update Logic, consider the restriction of DMPL to the update fragment, i.e., the fragment which only has 0-ary predicates and no quantification. It is easily seen that the assignment parameters in the semantics become redundant, and the definition of DMPL entailment reduces to the Update Logic definition.

If, on the other hand, we consider the DPL fragment of DMPL (all formulas without occurrences of the $\Diamond$ operator), then we may replace evaluation at an index set by evaluation at a single world, and we are back at the semantics of DPL. In this case the index set parameter becomes redundant and the definition of DMPL entailment reduces to the DPL definition.

A.2 Some Simple Examples

In order to gain insight in the semantics of DMPL it is useful to look at some very simple examples. Let us assume a model $\mathcal{M}$ with a set of worlds W consisting of three worlds over a universe $\{d, d'\}$, where a one place predicate P is defined as follows.

$$F_{w_1}(P) = \{d, d'\}.$$

$$F_{w_2}(P) = \{d\}.$$

$$F_{w_3}(P) = \emptyset.$$

Now suppose also we only have one variable x in the language. Then M^V consists of just two assignment functions, $x \mapsto d$ and $x \mapsto d'$, which we will refer to as s and u respectively. Here are the results of evaluating some very

simple programs.

$$[Px]^s_s(W) = \{w_1, w_2\}$$

$$[Px]^u_u(W) = \{w_1\}.$$

$$[Px]^s_u(W) = [Px]^u_s(W) = \emptyset$$

$$[\neg Px]^s_s(W) = \{w_3\}.$$

$$[\neg Px]^u_u(W) = \{w_2, w_3\}.$$

$$[\neg Px]^s_u(W) = [\neg Px]^u_s(W) = \emptyset$$

$$[\eta x : Px]^s_s(W) = \{w_1, w_2\}.$$

$$[\eta x : Px]^u_u(W) = \{w_1\}.$$

$$[\eta x : Px]^s_u(W) = \{w_1\}.$$

$$[\eta x : Px]^u_s(W) = \{w_1, w_2\}.$$

$$[\Diamond Px]^s_s(W) = W.$$

$$[\Diamond Px]^u_u(W) = W.$$

$$[\Diamond Px]^s_u(W) = [\Diamond Px]^u_s(W) = \emptyset.$$

The first litmus test of the system is whether it can deal with the contrast between '*He may be present ... He isn't present.*' and '*He isn't present ... *He may be present.*', no matter what the referent of 'he' happens to be. We illustrate with the example interpretation that the semantics of DMPL make the translation of the first acceptable, but that of the second unacceptable:

$$\begin{aligned}[\Diamond Px;\ \neg Px]^s_s(W) &= [\neg Px]^s_s([\Diamond Px]^s_s(W)) \\ &= [\neg Px]^s_s(W) \\ &= \{w_3\}.\end{aligned}$$

$$\begin{aligned}
[\![\Diamond Px;\ \neg Px]\!]^u_u(W) &= [\![\neg Px]\!]^u_u([\![\Diamond Px]\!]^u_u(W)) \\
&= [\![\neg Px]\!]^u_u(W) \\
&= \{w_2, w_3\}.
\end{aligned}$$

$$\begin{aligned}
[\![\neg Px;\ \Diamond Px]\!]^s_s(W) &= [\![\Diamond Px]\!]^s_s([\![\neg Px]\!]^s_s(W)) \\
&= [\![\Diamond Px]\!]^s_s(\{w_3\}) \\
&= \emptyset.
\end{aligned}$$

$$\begin{aligned}
[\![\neg Px;\ \Diamond Px]\!]^u_u(W) &= [\![\Diamond Px]\!]^u_u([\![\neg Px]\!]^u_u(W)) \\
&= [\![\Diamond Px]\!]^u_u(\{w_2, w_3\}) \\
&= \emptyset.
\end{aligned}$$

Secondly, the *maybe* operator should not eliminate any of the current epistemically possible worlds. A litmus test for this is that a sentence like *He may be present, but he may just as well not be present* should come out true in the situation of complete ignorance, and leave us in a state of complete ignorance. And this is precisely what we find, regardless of how the reference for 'he' gets fixed:

$$\begin{aligned}
[\![\Diamond Px;\ \Diamond\neg Px]\!]^s_s(W) &= [\![\Diamond\neg Px]\!]^s_s([\![\Diamond Px]\!]^s_s(W)) \\
&= [\![\Diamond\neg Px]\!]^s_s(W) \\
&= W.
\end{aligned}$$

$$\begin{aligned}
[\![\Diamond Px;\ \Diamond\neg Px]\!]^u_u(W) &= [\![\Diamond\neg Px]\!]^u_u([\![\Diamond Px]\!]^u_u(W)) \\
&= [\![\Diamond\neg Px]\!]^u_u(W) \\
&= W.
\end{aligned}$$

We want to argue that if *may* in the following examples is taken in its epistemic sense, then (A.1) should turn out to be acceptable, but (A.2) should not.

> Everyone may have escaped from the fire ... (A.1)
> Oh dear! Someone hasn't made it.
> Someone hasn't escaped from the fire. (A.2)
> *Everyone may have escaped.

One can easily imagine a fire fighter uttering discourse (A.1) as a comment to a colleague during an inspection round. But (A.2) would sound very weird in those same circumstances.

It is clear, however, that *may* or *might* has more senses than the purely epistemic sense that we are focusing on here. Note, for instance, that the following discourse is intuitively acceptable.

Someone hasn't escaped from the fire. (A.3)
Everyone might have escaped.

The acceptability of (A.3) means that *might* in this example does not express a purely epistemic modality. The second sentence of the discourse does not express that a state of affairs where everyone has escaped is in accordance with one's epistemic state (it is not, witness the information contained in the first sentence). The second sentence expresses something quite different: things might have turned out different from how they in fact turned out. To account for this kind of example, we would have to introduce a new operator for alethic modality, which looks at all indices, irrespective of whether they are still 'in the game' as epistemic alternatives. This would boil down to evaluation in yet another dimension. We will not pursue this issue here.

A.3 Quantified Dynamic Modal Logic

To gain further insight in DMPL, our next goal is to provide an axiomatization. For this we take our cue from two existing axiomatizations for DPL and Update Logic, respectively. Van Eijck and De Vries [32] use Hoare logic to axiomatize DPL, and in Van Eijck and De Vries [33] they apply the same methods to Update Logic. The extension of DPL with the epistemic operator suggests the use of modal predicate logic as assertion language in a Hoare style calculus for DMPL programs.

Rather than confining ourselves to Hoare style *implications* we want to be able to use the full range of logical connections between static assertions from modal predicate logic and programs from DMPL. We will therefore define a version of quantified dynamic modal logic that gives us the expressive power we need. This logic is inspired by Pratt [64]. See also Goldblatt [42] for a general survey of dynamic logics, Van Eijck [26] for a reformulation of the Hoare style calculus of Van Eijck and De Vries [32] in quantified dynamic logic, and Van Eijck and De Vries [34] for a reformulation of the Hoare style calculus for Update Logic in S5 propositional dynamic logic.

QDML programs $\pi := Rt\cdots t \mid t = t \mid v :=? \mid \pi;\pi \mid \neg\pi \mid \Diamond\pi.$

QDML formulas $\varphi ::= Rt\cdots t \mid t = t \mid \neg\varphi \mid \varphi \wedge \varphi \mid \exists v\varphi \mid \Diamond\varphi \mid \langle\pi\rangle\varphi.$

The programs of QDL are the DMPL programs, the formulas are the formulas of modal predicate logic, with an extra modality for DMPL programs added.

We use $\top$ as an abbreviation of $\forall x(x = x)$ and $\bot$ as an abbreviation of $\neg\top$. As is customary, we abbreviate $\neg(\neg\varphi\wedge\neg\psi)$ as $(\varphi\vee\psi)$. Also, we write $\neg(\varphi\wedge\neg\psi)$ as $(\varphi \to \psi)$, $\neg\Diamond\neg\varphi$ as $\Box\varphi$, $\langle\neg(\pi_1;\ \neg\pi_2)\rangle\varphi$ as $\langle\pi_1 \Rightarrow \pi_2\rangle\varphi$, $\neg\langle\pi\rangle\neg\varphi$ as $[\pi]\varphi$ and $\neg\exists x\neg\varphi$ as $\forall x\varphi$. Also, we omit outermost parentheses for readability.

We consider index sets I as pointers to universal Kripke models consisting of those worlds which are the current epistemic alternatives, with accessibility relation $I \times I$. Recall from the literature that the modal logic determined by the class of finite universal frames is S5. Moreover, for any universal model $\mathcal{M}$ (a universal frame with first order valuations in some domain M assigned to all of its worlds) there is a submodel $\mathcal{M}'$ where all worlds have different valuations, and such that $\mathcal{M}'$ validates the same formulas. $\mathcal{M}'$ can be got by throwing away the extra copies of the worlds with identical valuations: because of the universal accessibility this makes no difference to validity.

We define the notion of truth in a model $\mathcal{M}$, with respect to an index set I, an index $i \in I$, and an assignment s for $\mathcal{M}$.

A.3.1. DEFINITION. [Truth in $\mathcal{M}$ for index set I, index i, ass s]

1. $\mathcal{M}, I, i, s \models Rt_1 \cdots t_n$ iff $\mathcal{M}, i \models_s Rt_1 \cdots t_n$.
2. $\mathcal{M}, I, i, s \models t_1 = t_2$ iff $\mathcal{M}, i \models_s t_1 = t_2$.
3. $\mathcal{M}, I, i, s \models \neg\varphi$ iff it is not the case that $\mathcal{M}, I, i, s \models \varphi$.
4. $\mathcal{M}, I, i, s \models \varphi \wedge \psi$ iff $\mathcal{M}, I, i, s \models \varphi$ and $\mathcal{M}, I, i, s \models \psi$.
5. $\mathcal{M}, I, i, s \models \exists v\varphi$ iff for some $d \in M$, $\mathcal{M}, I, i, s(v|d) \models \varphi$.
6. $\mathcal{M}, I, i, s \models \Diamond\varphi$ iff there is some $j \in I$ with $\mathcal{M}, I, j, s \models \varphi$.
7. $\mathcal{M}, I, i, s \models \langle\pi\rangle\varphi$ iff there is an assignment u for which $i \in [\![\pi]\!]^s_u(I)$ and $\mathcal{M}, [\![\pi]\!]^s_u(I), i, u \models \varphi$.

More global notions of truth are now defined by universally quantifying over the various parameters, as usual:

A.3.2. DEFINITION. [Truth for I and i, Truth for I, Truth, Validity]

- $\mathcal{M}, I, i \models \varphi$ if for all assignments s for $\mathcal{M}$: $\mathcal{M}, I, i, s \models \varphi$.
- $\mathcal{M}, I \models \varphi$ if for all $i \in I$: $\mathcal{M}, I, i \models \varphi$.
- $\mathcal{M} \models \varphi$ if for all $I \subseteq W$: $\mathcal{M}, I \models \varphi$.
- $\models \varphi$ if for all $\mathcal{M}$: $\mathcal{M} \models \varphi$.

The consequence relation for QDML is defined as follows:

A.3.3. DEFINITION. [Consequence for QDML] $\Gamma \models \varphi$ if for all triples $\mathcal{M}, I, i$ the following holds: if there is an assignment s for $\mathcal{M}$ with $\mathcal{M}, I, i, s \models \gamma$ for all $\gamma \in \Gamma$, then $\mathcal{M}, I, i, s \models \varphi$.

It is convenient to define the following operation on index sets.

A.3.4. DEFINITION. [Pruning of index set I by φ, given s]
$I^s_\varphi \stackrel{\text{def}}{=} \{i \in I \mid \mathcal{M}, I, i, s \models \varphi\}$.
Note that $\downarrow [\![\pi]\!]^s(I) = I^s_{\langle \pi \rangle \top}$.

If φ, ψ are formulas of QDML, then $\varphi{\downarrow}\psi$, the localisation of φ to ψ, is defined as follows.

A.3.5. DEFINITION. [Localised QDML formulas $\varphi{\downarrow}\psi$]

$$
\begin{array}{lcll}
Rt_1 \cdots t_n{\downarrow}\psi & = & Rt_1 \cdots t_n \wedge \psi & \\
t_1 = t_2{\downarrow}\psi & = & t_1 = t_2 \wedge \psi & \\
(\varphi_1 \wedge \varphi_2){\downarrow}\psi & = & (\varphi_1{\downarrow}\psi) \wedge (\varphi_2{\downarrow}\psi) & \\
(\neg\varphi){\downarrow}\psi & = & \psi \wedge \neg(\varphi{\downarrow}\psi) & \\
(\exists v\varphi){\downarrow}\psi & = & \exists w(\varphi[w/v]{\downarrow}\psi) & (w \text{ a new variable})
\end{array}
$$

$$
\begin{array}{lcl}
(\Diamond\varphi){\downarrow}\psi & = & \psi \wedge \Diamond(\varphi{\downarrow}\psi) \\
(\langle \boldsymbol{Rt_1 \cdots t_n} \rangle\varphi){\downarrow}\psi & = & (Rt_1 \cdots t_n \wedge \varphi){\downarrow}\psi \\
(\langle \boldsymbol{t_1 = t_2} \rangle\varphi){\downarrow}\psi & = & (t_1 = t_2 \wedge \varphi){\downarrow}\psi \\
(\langle \boldsymbol{\pi_1; \pi_2} \rangle\varphi){\downarrow}\psi & = & (\langle \boldsymbol{\pi_1} \rangle\langle \boldsymbol{\pi_2} \rangle\varphi){\downarrow}\psi \\
(\langle \boldsymbol{\neg\pi} \rangle\varphi){\downarrow}\psi & = & (\varphi{\downarrow}[\pi]\bot){\downarrow}\psi \\
(\langle \boldsymbol{v :=?} \rangle\varphi){\downarrow}\psi & = & (\exists v\varphi){\downarrow}\psi \\
(\langle \boldsymbol{\Diamond\pi} \rangle\varphi){\downarrow}\psi & = & (\varphi \wedge \Diamond\langle \boldsymbol{\pi} \rangle\top){\downarrow}\psi.
\end{array}
$$

The localisation operator will play an important role in the axioms to be presented in Section A.4. The following lemma makes clear what localisation accomplishes.

A.3.6. LEMMA. *For all $\mathcal{M} = \langle M, W \rangle$, all $I \subseteq W$, and all assignments s for $\mathcal{M}$: $I^s_{\varphi{\downarrow}\psi} = (I^s_\psi)^s_\varphi$.*

Proof: What we have to prove is that $\mathcal{M}, I, i, s \models \varphi{\downarrow}\psi$ iff $\mathcal{M}, I, i, s \vdash \psi$ and $\mathcal{M}, I^s_\psi, i, s \vdash \varphi$. The proof uses induction on the structure of φ; we merely give some example clauses.

For atomic formulas $R\bar{t}$ we have: $\mathcal{M}, I, i, s \models R\bar{t}{\downarrow}\psi$ iff $\mathcal{M}, I, i, s \models R\bar{t} \wedge \psi$ iff both $\mathcal{M}, I, i, s \models \psi$ and $\mathcal{M}, I^s_\psi, i, s \models R\bar{t}$.

The case of existential quantification:

	$\mathcal{M}, I, i, s \models (\exists v\varphi){\downarrow}\psi$
iff	$\mathcal{M}, I, i, s \models \exists w(\varphi[w/v]{\downarrow}\psi)$, with w new
iff	there is some $d \in M$ with $\mathcal{M}, I, i, s(w\|d) \models \varphi[w/v]{\downarrow}\psi$
iff (ind hyp)	$\mathcal{M}, I, i, s(w\|d) \models \psi$ and $\mathcal{M}, I^{s(w\|d)}_\psi, i, s(w\|d) \models \varphi[w/v]$
iff (w fresh)	$\mathcal{M}, I, i, s \models \psi$ and $\mathcal{M}, I^s_\psi, i, s(w\|d) \models \varphi[w/v]$
iff	$\mathcal{M}, I, i, s \models \psi$ and $\mathcal{M}, I^s_\psi, i, s \models \exists v\varphi$.

The modal operator case:

	$\mathcal{M}, I, i, s \models (\Diamond\varphi)\downarrow\psi$
iff	$\mathcal{M}, I, i, s \models \psi \wedge \Diamond(\varphi\downarrow\psi)$
iff	$\mathcal{M}, I, i, s \models \psi$ and there is a $j \in I$ such that
	$\mathcal{M}, I, j, s \models \varphi\downarrow\psi$
iff (ind hyp)	$\mathcal{M}, I, i, s \models \psi$ and there is a $j \in I$ such that
	$\mathcal{M}, I, j, s \models \psi$ and $\mathcal{M}, I^s_\psi, j, s \models \varphi$
iff	$\mathcal{M}, I, i, s \models \psi$ and $M, I^s_\psi, i, s \models \Diamond\varphi$.

The definition of localisations for formulas starting with a program modality decomposes this operator, so in the program modality case all we have to do is check that the decomposition agrees with the semantic clause for the program construct and apply the induction hypothesis. This completes the induction argument and the proof. ∎

A.4 A Calculus for QDML

The calculus for QDML to be presented in this section provides the explicit link between static meaning and dynamic meaning for DMPL. The calculus has five sets of axiom schemata: (i) propositional and quantificational schemata, (ii) S5 schemata for the epistemic modality $\Box$, (iii) K-schemata for the program modalities, (iv) atomic test schemata and an assignment schema for the atomic program modalities, and (v) program composition schemata.

Propositional and Quantificational Schemata We start by taking the axiom schemata of propositional logic and first order quantification:

1. A. $\varphi \to (\psi \to \varphi)$.

2. A. $(\varphi \to (\psi \to \chi)) \to ((\varphi \to \psi) \to (\varphi \to \chi))$.

3. A. $(\neg\varphi \to \neg\psi) \to (\psi \to \varphi)$.

4. A. $\forall v\varphi \to [t/v]\varphi$, *provided* t *is free for* v *in* φ.

5. A. $\varphi \to \forall v\varphi$, *provided* v *has no free occurrences in* φ.

6. A. $\forall v(\varphi \to \psi) \to (\forall v\varphi \to \forall v\psi)$.

7. A. $v = v$.

8. A. $v = w \to (\varphi \to \varphi')$, *where* φ' *results from replacing some* v*-occurrence(s) in* φ *by* w.

See e.g. Enderton [36] for discussion and motivation.

The S5 Schemata for $\Box$

9. A. $\Box(\varphi \to \psi) \to (\Box\varphi \to \Box\psi)$.

10. A. $\Box\varphi \to \varphi$.

11. A. $\Box\varphi \to \Box\Box\varphi$.

12. A. $\Diamond\Box\varphi \to \varphi$.

These are the propositional S5 modalities. The next axiom gives the Barcan Schema, which takes care of the interaction of quantifiers and the S5 modal operator.

13. A. $\forall v\Diamond\varphi \to \Diamond\forall v\varphi$.

The K-schema for the program modalities

14. A. $[\pi](\varphi \to \psi) \to ([\pi]\varphi \to [\pi]\psi)$.

Atomic Test Schemata

15. A. $\langle \boldsymbol{Rt_1 \cdots t_n} \rangle\varphi \leftrightarrow \varphi{\downarrow}Rt_1 \cdots t_n$.

16. A. $\langle \boldsymbol{t_1 = t_2} \rangle\varphi \leftrightarrow \varphi{\downarrow}t_1 = t_2$.

Assignment Schema

17. A. $\langle \boldsymbol{v :=?} \rangle\varphi \leftrightarrow \exists v\varphi$.

Program Composition Schemata The schemata for complex programs.

18. A. $\langle \boldsymbol{\pi_1; \pi_2} \rangle\varphi \leftrightarrow \langle \boldsymbol{\pi_1} \rangle\langle \boldsymbol{\pi_2} \rangle\varphi$.

19. A. $\langle \neg\pi \rangle\varphi \leftrightarrow \varphi{\downarrow}[\pi]\bot$.

20. A. $\langle \Diamond\pi \rangle\varphi \leftrightarrow \varphi \wedge \Diamond\langle \pi \rangle\top$.

Rules of inference The rules of inference of the calculus are Universal Generalization (conclude from $\vdash \varphi$ to $\vdash \forall v\varphi$), Necessitation for $\Box$ (conclude from $\vdash \varphi$ to $\vdash \Box\varphi$) and Modus Ponens (conclude from $\vdash \varphi \to \psi$ and $\vdash \varphi$ to $\vdash \psi$). It turns out that necessitation for program modality can be derived (Proposition A.4.1). The notions of theoremhood ($\vdash \varphi$) and derivability from a set of premisses ($\Gamma \vdash \varphi$) in the calculus are defined in the standard way.

A.4.1. PROPOSITION.
If $\vdash \varphi$, *then* $\vdash [\pi]\varphi$.

Proof: Induction on the complexity of π. ∎

A.4.2. THEOREM. *If* $\Gamma \vdash \varphi$ *then* $\Gamma \models \varphi$.

Proof: Checking of axioms and rules, in the usual manner. ∎

Completeness of the calculus is also straightforward, as is to be expected from the fact that there are no difficult iterative phenomena (no Kleene star among the program operators).

A.4.3. THEOREM. *If* $\Gamma \models \varphi$ *then* $\Gamma \vdash \varphi$.

Proof: First observe that the following translation function * from QDML to S5 modal predicate logic preserves truth for index set, index and assignment.

$$\begin{array}{lcl}
(Rt_1 \cdots t_n)^* & = & Rt_1 \cdots t_n \\
(t_1 = t_2)^* & = & t_1 = t_2 \\
(\varphi \wedge \psi)^* & = & \varphi^* \wedge \psi^* \\
(\neg\varphi)^* & = & \neg\varphi^* \\
(\exists v\varphi)^* & = & \exists v\varphi^* \\
(\langle \boldsymbol{Rt_1 \cdots t_n} \rangle\varphi)^* & = & \varphi^*{\downarrow}Rt_1 \cdots t_n \\
(\langle \boldsymbol{t_1 = t_2} \rangle\varphi)^* & = & \varphi^*{\downarrow}t_1 = t_2 \\
(\langle \boldsymbol{\pi_1;\ \pi_2} \rangle\varphi)^* & = & (\langle \boldsymbol{\pi_1} \rangle\langle \boldsymbol{\pi_2} \rangle\varphi)^* \\
(\langle \boldsymbol{\neg\pi} \rangle\varphi)^* & = & \varphi^*{\downarrow}([\pi]\bot)^* \\
(\langle \boldsymbol{v :=?} \rangle\varphi)^* & = & (\exists v\varphi)^* \\
(\langle \boldsymbol{\Diamond\pi} \rangle\varphi)^* & = & \varphi^* \wedge \Diamond(\langle \pi \rangle\top)^*
\end{array}$$

Thus, it follows from $\Gamma \models \varphi$ that $\Gamma \models \varphi^*$. Next, use the completeness of S5 modal predicate logic to conclude from $\Gamma \models \varphi^*$ that $\Gamma \vdash \varphi^*$. Finally, note that the translation steps and their inverses in the definition of * are licenced by the atomic test schemata and the program composition schemata of the calculus. This allows us to conclude from $\Gamma \vdash \varphi^*$ that $\Gamma \vdash \varphi$. ∎

A.5 Calculating Meanings

The translation function * from Theorem A.4.3 derives more or less directly from our calculus. We will now demonstrate that it can be used for calculating the meanings of DMPL programs as formulas of S5 modal predicate logic.

Please note that in this paper we are not concerned with giving a translation algorithm from natural language to our representation language, although it is clear that this can be done using standard techniques from Montague grammar; see, e.g., Bouchez, Van Eijck and Istace [17] or Muskens [60]. All that we want to establish here is that DMPL is a reasonable representation language for those aspects of natural language meaning that involve epistemic modalities and anaphoric linking, by showing that the translation function * that is implied by our calculus can be used to derive truth conditions for natural language texts in S5 modal predicate logic.

The demonstration will proceed by analyzing some example natural language texts.

A man walked out. Maybe he was angry. (A.4)

A reasonable DMPL translation is the following (note that we ignore tense).

$$\eta x : \mathit{man}\ x;\ \mathit{walk\text{-}out}\ x;\ \Diamond \mathit{angry}\ x. \tag{A.5}$$

We want to find a specification of the conditions under which this DMPL program succeeds, for these are the truth conditions of the natural language example (in the intended reading, as specified by the DMPL translation). In other words, we want the truth conditions of the following QDML formula.

$$\langle \eta x : \mathit{man}\ x;\ \mathit{walk\text{-}out}\ x;\ \Diamond \mathit{angry}\ x\rangle \top. \tag{A.6}$$

We apply the translation function.

$$(\langle \eta x : \mathit{man}\ x;\ \mathit{walk\text{-}out}\ x;\ \Diamond \mathit{angry}\ x\rangle \top)^*. \tag{A.7}$$

What we get is:

$$\begin{array}{ll}
(\langle \eta x : \mathit{man}\ x;\ \mathit{walk\text{-}out}\ x;\ \Diamond \mathit{angry}\ x\rangle \top)^* & = \\
(\langle \eta x : \mathit{man}\ x\rangle\langle \mathit{walk\text{-}out}\ x\rangle\langle \Diamond \mathit{angry}\ x\rangle \top)^* & = \\
\exists x(\langle \mathit{man}\ x\rangle\langle \mathit{walk\text{-}out}\ x\rangle\langle \Diamond \mathit{angry}\ x\rangle \top)^* & = \\
\exists x((\langle \mathit{walk\text{-}out}\ x\rangle\langle \Diamond \mathit{angry}\ x\rangle \top)^*\downarrow \mathit{man}\ x) & = \\
\exists x(\mathit{man}\ x \wedge (\langle \mathit{walk\text{-}out}\ x\rangle\langle \Diamond \mathit{angry}\ x\rangle \top)^*) & = \\
\exists x(\mathit{man}\ x \wedge (\langle \Diamond \mathit{angry}\ x\rangle \top)^*\downarrow \mathit{walk\text{-}out}\ x) & = \\
\exists x(\mathit{man}\ x \wedge \mathit{walk\text{-}out}\ x \wedge (\langle \Diamond \mathit{angry}\ x\rangle \top)^*) & = \\
\exists x(\mathit{man}\ x \wedge \mathit{walk\text{-}out}\ x \wedge \Diamond(\langle \mathit{angry}\ x\rangle \top)^*) & = \\
\exists x(\mathit{man}\ x \wedge \mathit{walk\text{-}out}\ x \wedge \Diamond(\top\downarrow \mathit{angry}\ x)) & = \\
\exists x(\mathit{man}\ x \wedge \mathit{walk\text{-}out}\ x \wedge \Diamond \mathit{angry}\ x). &
\end{array}$$

Before we proceed to the next example, it is useful to list some derived translation instructions.

$$\begin{array}{lcl}
([\boldsymbol{Rt_1 \cdots t_n}]\varphi)^* & = & Rt_1 \cdots t_n \rightarrow (\varphi^*\downarrow Rt_1 \cdots t_n) \\
([\boldsymbol{t_1 = t_2}]\varphi)^* & = & t_1 = t_2 \rightarrow (\varphi^*\downarrow t_1 = t_2) \\
([\boldsymbol{\pi_1;\ \pi_2}]\varphi)^* & = & ([\pi_1][\pi_2]\varphi)^* \\
([\neg\pi]\varphi)^* & = & [\pi]\bot \rightarrow \varphi^*\downarrow([\pi]\bot)^* \\
(\langle \boldsymbol{\pi_1 \Rightarrow \pi_2}\rangle\varphi)^* & = & \varphi^*\downarrow([\pi_1]\langle \pi_2\rangle \top)^* \\
([\boldsymbol{\pi_1 \Rightarrow \pi_2}]\varphi)^* & = & ([\pi_1]\langle \pi_2\rangle \top)^* \rightarrow \varphi^*\downarrow([\pi_1]\langle \pi_2\rangle \top)^* \\
([\boldsymbol{\eta v : \pi}]\varphi)^* & = & \forall v([\pi]\varphi)^* \\
([\Diamond\pi]\varphi)^* & = & \Diamond(\langle \pi\rangle \top)^* \rightarrow \varphi^*
\end{array}$$

We can now tackle next example.

If a man walks out, then maybe he is angry. (A.8)

A reasonable DMPL translation:

$$(\eta x : man\ x;\ walk\text{-}out\ x) \Rightarrow \Diamond angry\ x. \tag{A.9}$$

We have to calculate the truth conditions of the following QDML sentence.

$$\langle (\boldsymbol{\eta x : man\ x;\ walk\text{-}out\ x}) \Rightarrow \Diamond angry\ \boldsymbol{x}\rangle \top. \tag{A.10}$$

Again the * function allows us to translate this into S5 modal predicate logic.

$$\begin{array}{ll}
(\langle(\boldsymbol{\eta x : man\ x;\ walk\text{-}out\ x}) \Rightarrow \Diamond angry\ \boldsymbol{x}\rangle \top)^* & = \\
\top\!\!\downarrow([\boldsymbol{\eta x : man\ x;\ walk\text{-}out\ x}]\langle\Diamond angry\ \boldsymbol{x}\rangle\top)^* & = \\
([\boldsymbol{\eta x : man\ x;\ walk\text{-}out\ x}]\langle\Diamond angry\ \boldsymbol{x}\rangle\top)^* & = \\
([\boldsymbol{\eta x : man\ x}][walk\text{-}out\ \boldsymbol{x}]\langle\Diamond angry\ \boldsymbol{x}\rangle\top)^* & = \\
\forall x([man\ \boldsymbol{x}][walk\text{-}out\ x]\langle\Diamond angry\ \boldsymbol{x}\rangle\top)^* & = \\
\forall x(man\ x \rightarrow ([walk\text{-}out\ \boldsymbol{x}]\langle\Diamond angry\ \boldsymbol{x}\rangle\top)^*) & = \\
\forall x(man\ x \rightarrow (walk\text{-}out\ x \rightarrow \langle\Diamond angry\ \boldsymbol{x}\rangle\top)^*) & = \\
\forall x(man\ x \rightarrow (walk\text{-}out\ x \rightarrow \Diamond angry\ x)). &
\end{array}$$

Appendix B

Outline of a completeness proof for TKVL

In this appendix we will sketch a completeness proof for (a fragment of) of the TKVL system (for a detailed proof see [21]). The system TKVL is just the system TKV (cf. Chapter 3) enriched with a 'static' part, which will basically be a version of Modal Predicate Logic and with the 'switching' operator $\langle\ \rangle$. See next section for a precise definition.

B.1 The question of dynamic completeness

The question of proving the completeness of TKVL is just an instantiation of the well-researched issue of proving the completeness of dynamic systems. In fact, we will basically use standard techniques from the literature. In particular, we will take our clue from the strategy proposed in [26] (partly based on the classical metatheory of Dynamic Logic - cf. for instance [42]) and also used in [35] (cf. the previous Appendix).

Concretely, we will get our completeness theorem in three steps:

- first, we will give an axiomatization of TKVL;
- second, using the previous axiomatization, we will translate (via a function *) the system TKVL into a multimodal static system TKVL*, which will be in fact a subsystem of TKVL;
- finally, we will prove the completeness of TKVL* (its axioms being the 'static' axioms for TKVL, and its models being just TKVL models), by building its canonical model; this will automatically give us the completeness of TKVL, because of the following two facts:

1. the function * preserves validity of TKVL formulas (i.e., if $\models \varphi$ then $\models \varphi*$);
2. if a formula $\varphi*$ is derivable in TKVL*, then φ will be derivable in TKVL.

We should stress that the peculiarity of the problem we are now going to cope with lies in the fact that the system TKVL is quite complex, both in its syntax and in its semantics. Concretely, TKVL has a great amount of different modalities, that call for a different semantic interpretation. Since, as we have said, we will prove our completeness result at a 'static' level, many of the classical problems in the completeness theory for Modal Predicate Logic will come up here. The art will consist then in merging a number of techniques from the literature in order to solve these problems.

B.2 The system TKVL

Let us start with the first step: defining a 'dynamic logic' in Pratt style for talking about Tarskian Kripkean Variations. For the sake of simplicity, we will not take into account the whole range of TKV programs, but rather confine our attention to the following kit, in which all semantic parameters are represented:

TKVL Programs :: $= Pt_1 \dots t_n \mid \neg\pi \mid \pi_1;\pi_2 \mid \eta \mid \mu \mid \delta \mid \Diamond^{\eta}\pi \mid \Diamond^{\mu}\pi \mid \Diamond^{\delta}\pi$

On the basis of this repertoire of programs, let us now define a Dynamic Logic for Tarskian Kripkean Variations, á la Pratt. Recall that this will be a multimodal system, where programs will only occur as 'labels' for modalities.

TKVL formulas: $Pt_1 \dots t_n \mid t_1 = t_2 \mid \neg\varphi \mid \varphi \wedge \psi \mid \exists x\varphi \mid \Diamond^{\mu}\varphi \mid \Diamond^{\eta}\varphi \mid \Diamond^{\delta}\varphi \mid \langle\pi\rangle\varphi$

As for the semantic, here is an obvious adaptation of TV models:

B.2.1. DEFINITION. [TVKL model] A *TVKL 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: $w, v, u, \dots$. These models carry the following 'shift relations':

1. $w =_A v$: w differs from v at most in its A-value
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

Let us see the semantic clauses for TVKL formulas on these models.

B.2.2. DEFINITION. [TKVL truth in W, w]

1. $W, w \models Pt_1 \ldots t_n$ iff $w \models Pt_1 \ldots t_n$
2. $W, w \models t_1 = t_2$ iff $w \models t_1 = t_2$
3. $W, w \models \neg\varphi$ iff not $W, w \models \varphi$
4. $W, w \models \varphi \wedge \psi$ iff $W, w \models \varphi$ and $W, w \models \psi$
5. $W, w \models \exists x\varphi$ iff there is a $d \in D^w$ such that $W, w(x|d) \models \varphi$ where $w(x|d)$ is like w except for the fact that it assigns to x the element d
6. $W, w \models \Diamond^{\mu}\varphi$ iff there is a v such that $w =_I v$ and $W, v \models \varphi$
7. $W, w \models \Diamond^{\eta}\varphi$ iff there is a v such that $w =_A v$ and $W, v \models \varphi$
8. $W, w \models \Diamond^{\delta}\varphi$ iff there is a v such that $w =_D v$ and $W, v \models \varphi$
9. $W, w \models \langle\pi\rangle\varphi$ iff there is a v such that $w \succ_{\pi} v$ and $v \in [\![\pi]\!]W$ and $W, v \models \varphi$

Truth in a model and validity can be defined as follows:

B.2.3. DEFINITION. [Truth in a model W] φ is *true* in W, notation $W \models \varphi$, if for all $w \in W$: $W, w \models \varphi$.

B.2.4. DEFINITION. [Validity] φ is *TKVL valid*, notation $\models \varphi$, if for all TKVL models W: $W \models \varphi$.

B.3 Localization

We will now define the notion of 'localization' (cf. also Appendix A), which will play an important role in the axiomatization of next section. Intuitively, the localization of a formula φ to a formula ψ (notation: $Loc^{\psi}\varphi$) restricts the range for φ to be evaluated to the extension of ψ. Concretely, if φ contains existential modalities, then its localization to ψ will keep the search for the witnesses within the bounds of the extension of ψ.

This becomes clear in the following lemma:

B.3.1. LEMMA. *For all TKVL models W and all worlds w the following hold:*

$$W, w \models Loc^{\psi}\varphi \quad \textit{iff} \quad W_{\psi}, w \models \varphi$$

where $W_{\psi} = \{w \in W \mid W, w \models \psi\}$.

Proof: by induction on the definition of *Loc* (cf. Definition B.3.2). ∎

Let us then see the definition of this non-commutative operation *Loc*:

B.3.2. DEFINITION. [Localization of φ to ψ]

1. $Loc^{\psi}(Pt_1 \ldots t_n) = Pt_1 \ldots t_n$
2. $Loc^{\psi}(t_1 = t_2) = t_1 = t_2$
3. $Loc^{\psi}(\neg\varphi) = \neg Loc^{\psi}(\varphi)$
4. $Loc^{\psi}(\varphi_1 \wedge \varphi_2) = (Loc^{\psi}(\varphi_1) \wedge (Loc^{\psi}(\varphi_2))$

5. $Loc^{\psi}(\exists x\varphi) = \exists x Loc^{\psi}(\varphi)$
6. $Loc^{\psi}(\Diamond^{\mu}\varphi) = \Diamond^{\mu}(Loc^{\psi}(\varphi) \wedge \psi)$
7. $Loc^{\psi}(\Diamond^{\eta}\varphi) = \Diamond^{\eta}(Loc^{\psi}(\varphi) \wedge \psi)$
8. $Loc^{\psi}(\Diamond^{\delta}\varphi) = \Diamond^{\delta}(Loc^{\psi}(\varphi) \wedge \psi)$
9. $Loc^{\psi}(\langle \mathbf{Pt_1} \ldots \mathbf{t_n}\rangle\varphi) = Loc^{\psi}(Pt_1 \ldots t_n \wedge \varphi)$
10. $Loc^{\psi}(\langle \mathbf{t_1} = \mathbf{t_n}\rangle\varphi) = Loc^{\psi}(t_1 = t_n \wedge \varphi)$
11. $Loc^{\psi}(\langle \pi_1; \pi_2\rangle\varphi) = Loc^{\psi}(\langle \pi_1\rangle\langle \pi_2\rangle\varphi)$
12. $Loc^{\psi}(\langle \neg\pi\rangle\varphi) = Loc^{\psi}(Loc^{[\pi]\bot}(\varphi))$
13. $Loc^{\psi}(\langle \Diamond^{\mu}\pi\rangle\varphi) = Loc^{\psi}(\varphi \wedge \Diamond^{\mu}\langle \pi\rangle\top)$
14. $Loc^{\psi}(\langle \Diamond^{\eta}\pi\rangle\varphi) = Loc^{\psi}(\varphi \wedge \Diamond^{\eta}\langle \pi\rangle\top)$
15. $Loc^{\psi}(\langle \Diamond^{\delta}\pi\rangle\varphi) = Loc^{\psi}(\varphi \wedge \Diamond^{\delta}\langle \pi\rangle\top)$
16. $Loc^{\psi}(\langle \eta^{\mu} : \pi\rangle\varphi) = Loc^{\psi}(cl\Diamond^{\mu}\langle \pi\rangle\varphi)$
17. $Loc^{\psi}(\langle \eta^{\eta} : \pi\rangle\varphi) = Loc^{\psi}(cl\Diamond^{\eta}\langle \pi\rangle\varphi)$
18. $Loc^{\psi}(\langle \eta^{\delta} : \pi\rangle\varphi) = Loc^{\psi}(cl\Diamond^{\delta}\langle \pi\rangle\varphi)$

Note that the notion of Localization is very closed to well-known *relativization* of modal formulas. For a detailed discussion on this connection, see [34].

Using the operation *Loc*, we are now able to define the connective $\downarrow$, with the following semantic clause:

$$W, w \models \varphi \downarrow \psi \text{ iff } W, w \models \psi \text{ and } W, w \models Loc^{\psi}\varphi$$

Armed with this $\downarrow$, let us now give our set of axioms for TKVL.

B.4 Axiomatization

We have seven sets of axiom schemata:

1. Propositional and quantificational axioms:
2. S5 axioms for all the modalities.
3. K schema for program modalities.
4. Tarskian schemata:
   - Barcan formula for: $\Diamond^{\mu}$, $\Diamond^{\eta}$.
   - $\Diamond^{\mu}(x = y) \rightarrow (x = y)$ and $(x = y) \rightarrow \Box^{\mu}(x = y)$ and similarly for $\Diamond^{\delta}$ and its dual.
   - $x = a \rightarrow \Box^{\delta}(x = a)$ where x is a variable and a is a constant

- $\varphi(\bar{a}) \rightarrow \Box^{\eta}\varphi(\bar{a})$
  and similarly for $\Box^{\delta}$.

5. Atomic test schemata
   - $\langle Pt_1 \ldots t_n\rangle\varphi \leftrightarrow \varphi \downarrow Pt_1 \ldots t_n$
   - $\langle t_1 = t_n\rangle\varphi \leftrightarrow \varphi \downarrow t_1 = t_n$
6. Assignments schemata
   - $\langle \eta\rangle\varphi \leftrightarrow \Diamond^{\eta}\varphi$
   - $\langle \mu\rangle\varphi \leftrightarrow \Diamond^{\mu}\varphi$
   - $\langle \delta\rangle\varphi \leftrightarrow \Diamond^{\delta}\varphi$
7. Complex programs schemata
   - $\langle \pi_1; \pi_2\rangle\varphi \leftrightarrow \langle \pi_1\rangle\langle \pi_2\rangle\varphi$
   - $\langle \neg\pi\rangle\varphi \leftrightarrow \varphi \downarrow [\pi]\bot$
   - $\langle \Diamond^{\mu}\pi\rangle\varphi \leftrightarrow \varphi \wedge \Diamond^{\mu}\langle \pi\rangle\top$
   - $\langle \Diamond^{\eta}\pi\rangle\varphi \leftrightarrow \varphi \wedge \Diamond^{\eta}\langle \pi\rangle\top$
   - $\langle \Diamond^{\delta}\pi\rangle\varphi \leftrightarrow \varphi \wedge \Diamond^{\delta}\langle \pi\rangle\top$

Let us now give some brief comments on this axiomatization. The first 3 sets of axioms should be quite perspicuous by themselves. Concerning the 'tarskian schemata', their role is to 'capture' the peculiarities of the different modalities. Therefore, we have the Barcan Formula only for $\Diamond^{\eta}$ and for $\Diamond^{\mu}$, whose correspondent accessibility relation only connects worlds with the same Domain. Similarly, only $\Diamond^{\mu}$ and $\Diamond^{\delta}$ can validate the schema for rigid variables, while $\Diamond^{\delta}$ also validates the schema for rigid terms. Finally, the rigidity of interpretation only holds for $\Diamond^{\eta}$ and $\Diamond^{\delta}$. As for the Atomic test schemata, the Assignments schemata and the Complex programs schemata, we only remark that they inductively cut down the dynamic potential of TKVL, so to speak, in that they make it possible to translate all the formulas with program modalities into formulas without program modalities. This particular feature of this axioms system will play a crucial role within our strategy for proving the completeness of TKVL.

Here are the standard inference rules:

1. Modus Ponens
2. Necessitation for all the modalities

3. $\frac{\varphi \to \psi}{\varphi \to \forall x \psi}$ where x is not free in φ

4. $\frac{\varphi \to \Box^{\delta} \psi}{\varphi \to \Box^{\delta} \forall x \psi}$ where x is not free in φ

5. $\frac{\varphi \to \neg x = t}{\neg \varphi}$

The soundness of this axioms system can be trivially proved by first showing that all axioms are valid and then that the inference rules preserves validity.

B.5 From TKVL to TKVL*

We must now define the translation that embeds TKVL into its static subsystem TKVL*.

B.5.1. DEFINITION. [Translation *]

1. $(\varphi)* = \varphi$ if φ does not contain program modalities;
2. $(\langle P t_1 \dots t_n \rangle \varphi)* = \varphi \downarrow P t_1 \dots t_n$
3. $(\langle t_1 = t_n \rangle \varphi)* = \varphi \downarrow t_1 = t_n$
4. $(\langle \eta \rangle \varphi)* = \Diamond^{\eta} \varphi$
5. $(\langle \mu \rangle \varphi)* = \Diamond^{\mu} \varphi$
6. $(\langle \delta \rangle \varphi)* = \Diamond^{\delta} \varphi$
7. $(\langle \pi_1 ; \pi_2 \rangle \varphi)* = \langle \pi_1 \rangle \langle \pi_2 \rangle \varphi$
8. $(\langle \neg \pi \rangle \varphi)* = \varphi \downarrow [\pi] \bot$
9. $(\langle \Diamond^{\mu} \pi \rangle \varphi)* = \varphi \wedge \Diamond^{\mu} \langle \pi \rangle \top$
10. $(\langle \Diamond^{\eta} \pi \rangle \varphi)* = \varphi \wedge \Diamond^{\eta} \langle \pi \rangle \top$
11. $(\langle \Diamond^{\delta} \pi \rangle \varphi)* = \varphi \wedge \Diamond^{\delta} \langle \pi \rangle \top$

It's important to remark that the translation * truly amounts to a plain application of the groups of axioms 5, 6 and 7. From this we can immediately deduce the two facts that we need in order to transfer the completeness result for TKVL* to TKVL (see paragraph B.1), namely:

1. the function * preserves validity of TKVL formulas (i.e., if $\models \varphi$ then $\models \varphi*$);

2. if a formula $\varphi*$ is derivable in TKVL*, then φ will be derivable in TKVL.

B.6 The proof

Therefore, we now need to prove that the system TKVL* is complete. TKVL* is a multimodal quantified system, and its key feature is that its three modalities blend within the same set-up the two crucial problems in the completeness theory for Modal Predicate Logic. Concretely, we have that:

1. Domains are not in general fixed, and the modality $\Diamond^\delta$ essentially ranges over the non-fixed Domains submodels of TKVL* models; thus, as we have already seen, the Barcan Formula does not hold for $\Diamond^\delta$;

2. on the other hand, terms are not in general rigid, since $\Diamond^\eta$ ranges over alternative 'extensions' of the variables, while $\Diamond^\mu$ allows for a world-relative interpretation of the constants.

Our strategy for coping with these problems will be as follows:

1. we will follow Thomason ([69]) and Garson ([40]) in defining an ad hoc notion of saturation (recall that when BF is not available, it becomes difficult to define a canonical model where the worlds have the $\forall$ property and provide the suitable witnesses for the existential modalities in their predecessors - see for instance [51]);

2. as for non-rigid terms, we will follow Goldblatt ([42]) and take a set of rigid designators as witnesses for the $\forall$ property.

We can now sketch the structure of the completeness proof.

Let us start with defining the notion of saturation.

B.6.1. DEFINITION. [Saturation for TKVL*] A set Γ of TKVL* formulas is *saturated* if the following conditions hold:

1. Γ is TKVL* consistent;
2. Γ is negation closed (namely, for every formula φ, one of the following holds: or $\varphi \in \Gamma$ or $\neg\varphi \in \Gamma$;
3. Γ has the $\forall$-property, in the following, general, form: if $(\chi \&^\delta \ldots \&^\delta \exists x\psi) \in \Gamma$, then $(\chi \&^\delta \ldots \&^\delta (\exists z(z = y) \wedge \psi^y/x)) \in \Gamma$ for a variable y, where $\varphi \&^\delta \psi$ stands for $\Diamond^\delta(\varphi \wedge \psi)$[1];
4. Γ has the witness property in the following, general, form: if $(\chi \&^\eta \ldots \&^\eta \top) \in \Gamma$, then for every term t there is a $x \in V$ such that $(\chi \&^\eta \ldots \&^\eta (x = t)) \in \Gamma$.

We want to remark the following - predictable - facts, that highlights the rationale of the requirements 3 and 4:

[1]Note that the 'bag' of 'antecedents' $\chi \&^\delta \ldots$ could be empty. In this case, this is the standard notion of $\forall$-property.

- if Γ has the standard $\forall$-property (i.e.: if $\exists x\varphi \in \Gamma$ then $\exists z(z = y \wedge \varphi^y/x) \in \Gamma$ for a variable y), then Γ automatically meets the general $\forall$-property as in 3 above for $\&^\eta$ and for $\&^\mu$.

- if Γ has the standard witness property (i.e.: if for every term t there is a variable $x \in V$ such that $x = t \in \Gamma$), then Γ automatically meets the general witness property as in 4 for $\&^\delta$ and $\&^\mu$.

Second, we now tackle the question of non-rigid terms modalities. In concrete, we expand the language of TKVL* with a set $U = \{u_1, u_2, \ldots\}$ of variables foreign to it. Armed with this kit of fresh variables, it is possible to prove that a saturated extension can be built from any consistent set Γ of formulas of TKVL* $\cup U$.

B.6.2. LEMMA. *Given a consistent set Γ of formulas of TKVL* $\cup U$, Γ has a saturated extension.*

Proof: see [21] ∎

The next lemma is crucial:

B.6.3. LEMMA. *If Γ is saturated and W is the closure of $\{\Gamma\}$ under the relation R (defined as follows: $\Delta R\Delta'$ iff $\Box^-\Delta = \{\varphi \mid \Box\varphi \in \Delta\} \subseteq \Delta'$), then for all $\Delta \in W$, if $\Diamond\varphi \in \Delta$ then there exists a $\Delta' \in W$ such that $\Delta R\Delta'$ and $\varphi \in \Delta'$.*

Proof: see [69]. ∎

The canonical model can be now easily built:

B.6.4. DEFINITION. [Canonical model for TKVL* $\cup U$] The *canonical model* CM for TKVL* $\cup U$ is as follows:

$$CM = \langle W, \{R_{PAR} \mid PAR \text{ is a tarskian parameter}\}, D_{CM}, A_{CM}, I_{CM}\rangle$$

where:

- W is the set of all saturated sets of formulas of TKVL* $\cup U$;
- D_{CM} is the set $U = \{u_1, u_2 \ldots\}$
- $wR_{PAR}w'$ iff $\Box^{PAR^-}(w) \subseteq w'$
- $A^w_{CM}(x) = u$ iff $x = u \in w$
- $\langle t_1 \ldots t_n\rangle \in I^w_{CM}(P^n)$ iff $P^n(t_1 \ldots t_n) \in w$
- $I^w_{CM}(a) = u$ iff $a = u \in w$

The completeness proof is standard (it uses the fundamental lemma saying that for all $w \in CM$, it holds that: $W, w \models \varphi$ iff $\varphi \in w$).

Appendix C

Back to Classical Logic

In this appendix, we will show how the 'inverse logic' methods of Section 4.3 can be applied also to more complex cases. Therefore, we will take into account one more type of dynamic operator, being projections taking dynamic propositions to classical ones. One such operator was the DPL local truth $\top$, assigning to each program π its 'domain':

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

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

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

First note the following:

C.0.5. FACT. $\top^*$ is permutation invariant.

(This follows from its set-theoretic definition; cf. Theorem 4.1.5). 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:

C.0.6. FACT. $\top^*$ 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:

C.0.7. THEOREM. *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:

C.0.8. PROPOSITION. *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:

C.0.9. FACT. F^* is logical.

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

C.0.10. PROPOSITION. *There are only two logical maps F^* sending states to atoms in the update algebra.*

Proof: We have $s \stackrel{F^*}{\longmapsto} \alpha_{W,w}$, 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. ∎

Bibliography

[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] D. Beaver (1993), *What comes first in Dynamic Logic*, manuscript, University of Edimburgh.

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

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

[6] 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.

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

[8] 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.

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

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

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

[12] 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.

[13] J. van Benthem (1995), *Logic and the flow of information*, forthcoming.

[14] J. van Benthem and G. Cepparello (1994), *Tarskian Variations. Dynamic Parameters in Classical Semantics*, Report CS-R9419 of the Centre for Mathematics and Computer Science (CWI), Amsterdam.

[15] 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.

[16] 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.

[17] O. Bouchez, J. van Eijck and O. Istace (1993), *A strategy for dynamic interpretation: a fragment and an implementation*, in: S. Krauwer, M. Moortgat and Louis des Tombe (eds.), *Proceedings of the Sixth Conference of the European Chapter of the Association for Computational Linguistics*, ACL.

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

[19] R. Carnap (1947), *Meaning and Necessity*, The University of Chicago Press, Chicago.

[20] G. Cepparello (1991), *What are individuals? New Semantics for Predicate Modal Logic*, Master's Thesis, Dipartimento di Filosofia, Universitá degli Studi di Pisa.

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

[22] G. Cepparello and R. Presilla (1995), *Why can't Moderate Holism be good?*, to appear in: *Proceedings of the 10th International Congress of Logic, Methodology and Philosophy of Science. Florence 1995.*

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

[24] P. Dekker (1994), *Transsentential meditations. Ups and Downs in Dynamic Semantics*, ILLC Dissertation Series, Amsterdam.

[25] K. Donnellan (1966), *Reference and Definite Descriptions*, in: The Philosophical Review, 75.

[26] J. van Eijck (1992), *Axiomatizing Dynamic Predicate Logic with Quantified Dynamic Logic*, to appear in: J. van Eijck and A. Visser (eds.), *Logic and Information Flow*, MIT Press, Cambridge Massachussets.

[27] J. van Eijck (1993), *The Dynamics of Descriptions*, in: The Journal of Semantics, 10.

[28] J. van Eijck (1993), *The Dynamics of Theory Extension*, to appear in: P. Dekker and M. Stokhof (eds.), *Proceedings of the 9th Amsterdam Colloquium*, ILLC, Amsterdam.

[29] J. van Eijck (1994), *Presupposition failure. A comedy of errors*, in: Formal Aspects of Computing, 6A.

[30] J. van Eijck (1994), *Presuppositions and Dynamic Logic*, Report CSLI-94-186 of the CSLI, Stanford.

[31] J. van Eijck (1995), *Presuppositions and Information Updating*, to appear in: M. Kanazawa, C..J. Piñon and H. de Swart (eds.), *Quantifiers, Deduction, and Context*, CSLI, Stanford.

[32] J. van Eijck and F. de Vries (1992), *Dynamic Interpretation and Hoare Deduction*, in: Journal of Logic, Language and Information, 1.

[33] J. van Eijck anf F. de Vries (1992), *A Sound and Complete Calculus for Update Logic*, in: P. Dekker and M. Stokhof (eds.), *Proceedings of the 8th Amsterdam Colloquium*, ILLC, Amsterdam.

[34] J. van Eijck and F. de Vries (1993), *Reasoning about Update Logic*, in: Journal of Philosophical Logic, 24.

[35] J. van Eijck and G. Cepparello (1994), *Dynamic Modal Predicate Logic*, in: M. Kanazawa and C.J. Piñon (eds.), *Dynamics, Polarity and Quantification*, CSLI, Stanford.

[36] H.B. Enderton (1972), *A Mathematical Introduction to Logic*, Academic Press.

[37] K. Fine (1984), *Reference, Essence and Identity*, unpublished manuscript.

[38] G. Frege (1892), *Über Sinn und Bedeutung*, in: Zeitschrift für Philosophie und Philosophische Kritik, 100.

[39] L.T.F. Gamut (1991), *Logic, Language and Meaning*, vol II, The University of Chicago Press, Chicago.

[40] J.W. Garson (1984), *Quantification in Modal Logic*, in: D. Gabbay and F. Guenthner (eds.), *Handbook of Philosophical Logic*, vol II, Reidel Publishing Company, Dordrecht.

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

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

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

[44] 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.

[45] J. Groenendijk, M. Stokhof and F. Veltman (1994), *Coreference and Modality*, handout for the 'Third CSLI Workshop on Language, Logic and Computation', June 1994, Stanford.

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

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

[48] D. Hilbert and P. Bernays (1939), *Grundlagen der Mathematik*, Berlin.

[49] C.A.R. Hoare (1969), *An axiomatic basis for computer programming*, in: Communications of the ACM, 12.

[50] D. Hofstadter and D. Dennet (1981), *The minds I. Fantasies and reflections on self and soul*, Basic Books.

[51] G.E. Hughes and M.J. Cresswell (1984), *A Companion to Modal Logic*, Methuen and Co., London.

[52] J. Jaspars (1994), *Calculi for Constructive Communication*, ILLC Dissertation Series, Amsterdam.

[53] M. Kameyama (1994), *Indefeasible semantics and defeasible pragmatics*, Report CS-R9441 of the Centre for Mathematics and Computer Science (CWI), Amsterdam.

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

[55] D. Kaplan (1986), *Opacity*, in: L.E. Hahn and P.A. Schilpp (eds.), *The Philosophy of W.V. Quine*, Open Court, La Salle.

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

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

[58] D. Lewis (1979), *Score keeping in a language game*, in: Journal of Philosophical Logic, 8.

[59] G. Miller, E. Galanter and K. Pribram (1960), *Plans and the structure of behaviour*, Holt, Rinehard and Winston, New York.

[60] R. Muskens (1991), *Anaphora and the Logic of Change*, in: J. van Eijck (ed.), *Logics in AI*, Proceedings of the European Workshop JELIA 90, Springer-Verlag.

[61] U. Neisser (1967), *Cognitive Psychology*, Appleton-Century-Crofts, New York.

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

[63] A. Plantinga (1974), *The Nature of Necessity*, Oxford University Press, Oxford.

[64] V. Pratt (1976), *Semantical Considerations on Floyd-Hoare Logic*, in: *Proceedings of the 17th IEEE Symposium on Foundations of Computer Science.*

[65] W.O. Quine (1963), *Reference and Modality*, in: W.O. Quine, *From a Logical Point of View*, Harvard University Press, Cambridge.

[66] 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.

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

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

[69] R.H. Thomason (1970), *Some completeness results for Modal Predicate Calculi*, in: K. Lambert (ed.), *Philosophical Problems in Logic*, Reidel, Dordrecht.

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

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

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

[73] K. Vermeulen and M. Hollenberg (1995), *Counting Variables in a Dynamic Setting*, Report IR-373 of the Department of Computer Science, Free University, Amsterdam.

[74] J.B. Watson (1924), *Behaviourism*, Norton, New York.

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

[76] L. Wittgenstein (1953), *Philosophische Untersuchungen*, Basil Blackwell, Oxford.

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

Titles in the ILLC Dissertation Series:

ILLC DS-94-01: **Harold Schellinx**
The Noble Art of Linear Decorating

ILLC DS-94-02: **Jan Willem Cornelis Koorn**
Generating Uniform User-Interfaces for Interactive Programming Environments

ILLC DS-94-03: **Nicoline Johanna Drost**
Process Theory and Equation Solving

ILLC DS-94-04: **Jan Jaspars**
Calculi for Constructive Communication, a Study of the Dynamics of Partial States

ILLC DS-94-05: **Arie van Deursen**
Executable Language Definitions, Case Studies and Origin Tracking Techniques

ILLC DS-94-06: **Domenico Zambella**
Chapters on Bounded Arithmetic & on Provability Logic

ILLC DS-94-07: **V. Yu. Shavrukov**
Adventures in Diagonalizable Algebras

ILLC DS-94-08: **Makoto Kanazawa**
Learnable Classes of Categorial Grammars

ILLC DS-94-09: **Wan Fokkink**
Clocks, Trees and Stars in Process Theory

ILLC DS-94-10: **Zhisheng Huang**
Logics for Agents with Bounded Rationality

ILLC DS-95-01: **Jacob Brunekreef**
On Modular Algebraic Protocol Specification

ILLC DS-95-02: **Andreja Prijatelj**
Investigating Bounded Contraction

ILLC DS-95-03: **Maarten Marx**
Algebraic Relativization and Arrow Logic

ILLC DS-95-04: **Dejuan Wang**
Study on the Formal Semantics of Pictures

ILLC DS-95-05: **Frank Tip**
Generation of Program Analysis Tools

ILLC DS-95-06: **Jos van Wamel**
Verification Techniques for Elementary Data Types and Retransmission Protocols

ILLC DS-95-07: **Sandro Etalle**
Transformation and Analysis of (Constraint) Logic Programs

ILLC DS-95-08: **Natasha Kurtonina**
Frames and Labels. A Modal Analysis of Categorial Inference

ILLC DS-95-09: **G.J. Veltink**
Tools for PSF

ILLC DS-95-10: **Giovanna Cepparello**
Studies in Dynamic Logic

ILLC DS-95-11: **W.P.M. Meyer Viol**
Instantial Logic. An Investigation into Reasoning with Instances

ILLC DS-95-12: **Szabolcs Mikulás**
Taming Logics

ILLC DS-95-13: **Marianne Kalsbeek**
Meta-Logics for Logic Programming

ILLC DS-95-14: **Rens Bod**
Enriching Linguistics with Statistics: Performance Models of Natural Language

ILLC DS-95-15: **Marten Trautwein**
Computational Pitfalls in Tractable Grammatical Formalisms

ILLC DS-95-16: **Sophie Fischer**
The Solution Sets of Local Search Problems

ILLC DS-95-17: **Michiel Leezenberg**
Contexts of Metaphor

ILLC DS-95-18: **Willem Groeneveld**
Logical Investigations into Dynamic Semantics

ILLC DS-95-19: **Erik Aarts**
Investigations in Logic, Language and Computation

ILLC DS-95-20: **Natasha Alechina**
Modal Quantifiers

ILLC DS-96-01: **Lex Hendriks**
Computations in Propositional Logic

ILLC DS-96-02: **Erik de Haas**
Categories for Profit

ILLC DS-96-03: **Martin H. van den Berg**
Some Aspects of the Internal Structure of Discourse: the Dynamics of Nominal Anaphora

Finito di stampare nel giugno 1996
in Pisa dalle
EDIZIONI ETS