Modal Logic, Transition Systems and Processes

Johan van Benthem, *ILLC, Plantage Muidergracht 24, 1018 TV Amsterdam, The Netherlands*

Jan van Eijck, *CWI, Kruislaan 413, 1098 SJ Amsterdam, The Netherlands*

Vera Stebletsova, *Department of Philosophy, University of Utrecht, Heidelberglaan 8, 3584 CS Utrecht, The Netherlands*

Abstract

Transition systems can be viewed either as process diagrams or as Kripke structures. The first perspective is that of process theory, the second that of modal logic. This paper shows how various formalisms of modal logic can be brought to bear on processes. Notions of bisimulation can not only be motivated by operations on transition systems, but they can also be suggested by investigations of modal formalisms. To show that the equational view of processes from process algebra is closely related to modal logic, we consider various ways of looking at the relation between the calculus of basic process algebra and propositional dynamic logic.

More concretely, the paper contains preservation results for various bisimulation notions, a result on the expressive power of propositional dynamic logic, and a definition of bisimulation which is the proper notion of invariance for concurrent propositional dynamic logic.

Keywords: modal logic, transition systems, bisimulation, process algebra

1 Introduction

The purpose of this paper is to compare two traditions of thinking about transition systems (roughly speaking, the computer science tradition and the tradition from modal logic), to bring out analogies, and to further investigate transition systems from a (modal) logical point of view. We will try to demonstrate that existing logical formalisms can go a long way in the study of transition systems, that there is less need for inventing new formalisms in this area than is often realised, and that the perspective of modal logic suggests some interesting further research questions.

The connection between process theory and modal logic is already made in [29, 32], where modal languages for internal description of transition systems are reinvented, so to speak. Modal analogies are suggested in many other places in the process literature as well (see for instance [23]).

The modal perspective on the study of transition systems that is advocated in this paper can also be found in [45], where expressibility issues stemming from modal and temporal logic are brought to bear on the study of processes. Our aim in this paper is to look at the connection between modal logic and process theory in a more systematic way and to point out further questions that are suggested by the modal perspective, answering some of them as we go along to illustrate our purposes.

An important research direction in modal logic is analysis of the expressive power of modal formalisms. This issue can be pursued for basic modal formalisms or for extended formalisms. In classical correspondence theory [3, 4] one studies the way in which modal formulas can be used to formulate first order and higher order relational constraints. The key research question here is: which modal formulas define first-order relational conditions, and how do they do it? Completeness questions involve proposals for axiomatisation and their investigation. See e.g. [26] for an overview.

Both of these directions can also be discerned in the study of newer richer formalisms. Extended formalisms of modal logic consider alternative relations other than the binary accessibility relations considered so far. It turns out that adding a modal operator D for the binary relation of inequality (so $D\varphi$ holds in s if there is an s' different from s where φ holds) increases the expressive power of the modal language [39]. Modal logics with ternary accessibility relations (many-dimensional modal logics) are studied in [46]. In Section 5 we will look at an example of a two-dimensional tense logic. In [25] still another extension, with binary accessibility between states and sets of states, is considered. This logic will be briefly discussed in Section 10.

There are two directions of thought concerning processes: from a notion of bisimulation to a language and from a language to a notion of bisimulation. Also there are two versions of such characterization results: directly on models, or via a preservation theorem in first-order logic (or an even broader formalism). Preservation results linking a modal language L to a bisimulation notion B take the form: a formula of first-order logic is invariant for a bisimulation of kind B iff that formula is equivalent to a translation of a formula from the modal language L.

So here is a whole range of new logical questions that modal logic suggests in connection with the study of transition systems: find preservation results in first-order logic, or higher order logics for the bisimulation notions used to study these transition systems. Also, modal languages suitable for talking about transition systems, or fragments of such languages, may suggest their own bisimulation notions, for which similar questions may be asked.

After a short survey of some standard results on the connection between equivalence relations on transition systems and modal languages, we will explore these further questions, first in connection with some fragments of temporal logic which have proved useful in theoretical computer science, then in connection with programming constructs that are studied in another branch of modal logic, propositional dynamic logic, and finally in connection with process theory. From the standpoint of general modal logic, these are just different, progressively richer formalisms for bringing out relevant structure of transition systems — and we urge our readers to abstract resolutely from the usual competitive discussions concerning their ideological merits.

2 Transition Systems and Process Equivalences

2.1 Transition Systems

Transition systems are the basic stuff that processes are built of. We start with some definitions.

DEFINITION 2.1
By a *transition system* or TS we shall mean a triple $\langle S, A, \rightarrow \rangle$, where S is a set of states, A is a set of labels, and $\rightarrow \subseteq S \times A \times S$.

The relation $\rightarrow$ is the labelled transition relation. If $\langle s, a, s' \rangle \in \rightarrow$ we write $s \xrightarrow{a} s'$ and we call a the label of the transition from s to s'.

DEFINITION 2.2
A *rooted* or *pointed* TS is a TS with one state singled out as the root.

We will use $\mathcal{M}, \mathcal{N}$ to refer to TSs, and $S(\mathcal{M}), S(\mathcal{N})$ to refer to their state sets. For the description of operations on TSs (to be introduced in Section 7) it is necessary to be able to mark states in a TS with a $\surd$ for 'success'. But because we may need other marks later on, we introduce the concept of a valuation for a TS.

DEFINITION 2.3
Let P be a set of proposition letters and $\mathcal{M}$ a TS with state set S. Then V is a *valuation* for $\mathcal{M}$ if V is a function from P to POW S.

Intuitively, $s \in V(p)$ means that proposition p is true in state s. We will sometimes call a TS $\mathcal{M}$ together with a valuation for it an *interpreted transition system.* A TS with an interpretation for $\surd$ is a TS where states may be marked with $\surd$. We will use $\mathcal{M}, \mathcal{N}$ without further ado for TSs with an interpretation for $\surd$. Instead of $s \in V(\surd)$ we will write $s \in \surd$.

2.2 Varieties of Process Equivalence

We look at TSs as process diagrams, so TSs represent processes. However, the question which process is pictured by a given TS has no general answer. We have to bear in mind that processes are identified on the basis of a similarity notion defined on TSs, and there are many such similarity notions. In this paper, we will present a hierarchy of stronger and stronger equivalences. In this section, we start with a brief discussion of three well-known ones.

We will define process equivalences as relations between TSs. Note, however, that it does not matter whether we compare TSs $\mathcal{M}, N$ or look within one TS. We can always combine two TSs $\mathcal{M}, \mathcal{N}$ into one TS by taking their disjoint union.

DEFINITION 2.4
If $\mathcal{M}$ is a transition system and s is a state in $\mathcal{M}$, then T_s, the set of *(finite) traces* from s, is the set of all sequences $a_1 \cdots a_n \in A^*$ such that for some s' with $s' \in \surd$, $s \xrightarrow{a_1} \cdots \xrightarrow{a_n} s'$.

We write $s \xrightarrow{a_1 \cdots a_n} s'$ for $s \xrightarrow{a_1} \cdots \xrightarrow{a_n} s'$.

DEFINITION 2.5
A state s in $\mathcal{M}$ is *finite trace equivalent* with a state r in $\mathcal{N}$ if $T_s = T_r$.

Possible variations here are to also consider infinite traces, and/or to drop the requirement that the finite traces end in a $\surd$ state. Finite state machines are examples of rooted TSs (the root is the start state, and the $\surd$ states are the accepting states). Finite state machines accepting the same language are finite trace equivalent.

DEFINITION 2.6
A relation $C \subseteq S(\mathcal{M}) \times S(\mathcal{N})$ is a simple *left-right simulation* if whenever sCr then s and r have the same valuation, and for every $s' \in S(\mathcal{M})$ with $s \xrightarrow{a} s'$ there is an $r' \in S(\mathcal{N})$ with $r \xrightarrow{a} r'$ and $s'Cr'$. A simple *right-left simulation* is defined similarly. A pair of relations $C_1, C_2 \subseteq S(\mathcal{M}) \times S(\mathcal{N})$ is a *simple simulation equivalence* if C_1 is a simple left-right simulation and C_2 a simple right-left simulation.

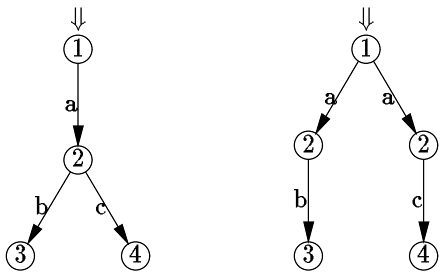

FIG. 1. Finite trace equivalent TSs which are not simply simulation equivalent.

OBSERVATION 2.7
Simple simulation equivalence is a stronger notion than finite trace equivalence.

PROOF. Obviously, simply simulation equivalent TSs have the same finite traces. But simple simulation equivalence is stronger, for consider Figure 1, which gives an example of finite trace equivalent TSs which are not simply simulation equivalent. Corresponding numbers on the nodes indicate the finite trace equivalence. To see that there is no simple simulation equivalence, observe that the 2 node on the left has no choice-preserving counterpart on the right. ∎

DEFINITION 2.8
A relation C between $S(\mathcal{M})$ and $S(\mathcal{N})$ is a *simple bisimulation* if the following three clauses hold:

1. If sCr, then $V(s) = V(r)$,
2. if sCr and $s \xrightarrow{a} s'$, then there is a state r' in $\mathcal{N}$ with $r \xrightarrow{a} r'$ and $s'Cr'$,
3. clause (2) vice versa.

If C is a bisimulation of kind K which relates s and r then we say that s and r are K bisimilar.

The key distinction between a simple (right-left or left-right) simulation and a simple bisimulation lies in the presence of a reverse clause in the latter. Thus, a bisimulation of kind K always brings a simulation equivalence of kind K in its wake: leave out the reverse clause from the bisimulation definition to get a 'directed' simulation definition.

OBSERVATION 2.9
Simple bisimulation is a stronger notion than simple simulation equivalence.

FIG. 2. Simply simulation equivalent, but not simply bisimilar, TSs.

PROOF. Obviously, simply bisimilar TSs are simply simulation equivalent. But the converse does not hold: Figure 2 gives an example of simply simulation equivalent TSs which are not simply bisimilar. A simple left-right simulation: map numbered states to their counterparts on the right. A simple right-left simulation: map numbered states to their counterparts on the left, and the states without numbers to the states indicated by the dotted arrows. Note that the second mapping is not the converse of the first (the dotted arrow pointing to 2 is not reversible). This shows that the two TSs are simply simulation equivalent. Still, the TSs are not simply bisimilar, for there is no way to relate the state in the right TS with $\xrightarrow{c}$ as its only outgoing arrow to a simply bisimilar state on the left. ∎

3 Modal Logic

3.1 *Key Notions*

Modal logic studies the relation between operator languages and structures of possible worlds with alternative relations between them. The semantics of modal logic uses Kripke frames, which are in fact TSs (where the states are the possible worlds), and Kripke models, which are in fact interpreted TSs. Logical description languages for TSs were developed in [29] and [17].

If A is a set of labels and P a set of proposition letters, then the modal language L over A, P is given by the following BNF definition (assume $a \in A, p \in P$):

$L\ \varphi ::= p \mid \neg\varphi \mid (\varphi \wedge \varphi) \mid \langle a \rangle \varphi.$

We add the usual abbreviations: $\perp$ is short for $p \wedge \neg p$, $\top$ is short for $\neg\perp$, $\varphi \vee \psi$ is short for $\neg(\neg\varphi \wedge \neg\psi)$, $\varphi \rightarrow \psi$ is short for $\neg(\varphi \wedge \neg\psi)$, and $[a]\varphi$ is short for $\neg\langle a \rangle \neg\varphi$.

If we wish to forget about the valuations again, we can consider the modal language with just one proposition constant $\top$ which holds in every state. To apply modal logic to processes, we employ a modal language with one propositional constant $\surd$ (and we consider $\top$ as an abbreviation of $\surd \vee \neg\surd$ and $\perp$ as an abbreviation of $\surd \wedge \neg\surd$).

The key semantic notion is the relation of truth of a formula φ in a state s of an ITS $\mathcal{M}, V$, with notation $\mathcal{M}, V, s \models \varphi$. This relation is defined inductively as follows.

DEFINITION 3.1 (Truth)

1. $\mathcal{M}, V, s \models \bot$ never.
2. $\mathcal{M}, V, s \models \top$ always.
3. $\mathcal{M}, V, s \models p$ iff $s \in V(p)$.
4. $\mathcal{M}, V, s \models \neg\varphi$ iff not $\mathcal{M}, V, s \models \varphi$.
5. $\mathcal{M}, V, s \models \varphi \wedge \psi$ iff $\mathcal{M}, V, s \models \varphi$ and $\mathcal{M}, V, s \models \psi$.
6. $\mathcal{M}, V, s \models \langle a \rangle \varphi$ iff there is some s' among the states of $\mathcal{M}$ with $s \xrightarrow{a} s'$ and $\mathcal{M}, V, s' \models \varphi$.

If $\mathcal{M}, V, s \models \varphi$ for every s in the state set of $\mathcal{M}$ we say that φ is true in $\mathcal{M}, V$, notation $\mathcal{M}, V \models \varphi$. If $\mathcal{M}, V \models \varphi$ for every valuation V, we say that φ is true on the TS $\mathcal{M}$, notation $\mathcal{M} \models \varphi$.

3.2 Semantic Invariances

We will now look at suitable modal languages for the equivalences mentioned earlier. The general strategy for linking a similarity notion to a modal language is as follows. See what is needed for the induction step and impose the appropriate requirement on the similarity notion, or the other way around, see what the similarity notion gives us as material to base an induction argument on, and find the language fragment for which this is precisely what is needed. This strategy is illustrated in the arguments in the proofs of Theorems 3.2 and 3.4 below. We note here that the results in this section are well-known from the literature: see in particular Hennessey and Milner [29].

Note that for the application to processes with just $\surd$ markings on their states the condition that bisimilar states have the same valuation boils down to the condition that bisimilar states agree in their $\surd$ markings. Consider the language L_m defined as follows.

$$L_m \ \varphi ::= \surd \mid \neg\varphi \mid (\varphi \wedge \varphi) \mid \langle a \rangle \varphi.$$

L_m is built from $\surd$ with the Booleans and the modal operators. The language L_m gives rise to the notion of simple bisimulation from Section 2.

Theorem 3.2
If s in $\mathcal{M}$ is simply bisimilar to r in $\mathcal{N}$, then all L_m-formulas have the same truth value in $\mathcal{M}, s$ and $\mathcal{N}, r$.

Proof. Induction on the structure of an L_m-formula φ. If φ is $\surd$, φ will hold in s iff φ holds in r by the bisimulation requirement on valuations. If φ is $\neg\psi$ or $\psi \wedge \chi$ the result follows from the induction hypothesis. The crucial case is that of the modal operator. If φ has the form $\langle a \rangle \psi$ and φ holds in s, then this means that there is an s' with $s \xrightarrow{a} s'$ and ψ holds in s'. But because sCr there is an r' with $r \xrightarrow{a} r'$ and $s'Cr'$, so by the induction hypothesis, ψ holds in r'. This shows that $\langle a \rangle \psi$ holds in r. For the other direction, use the other bisimulation clause. ∎

Observation 3.3
Invariance for L_m-formulas does not in general imply simple bisimulation.

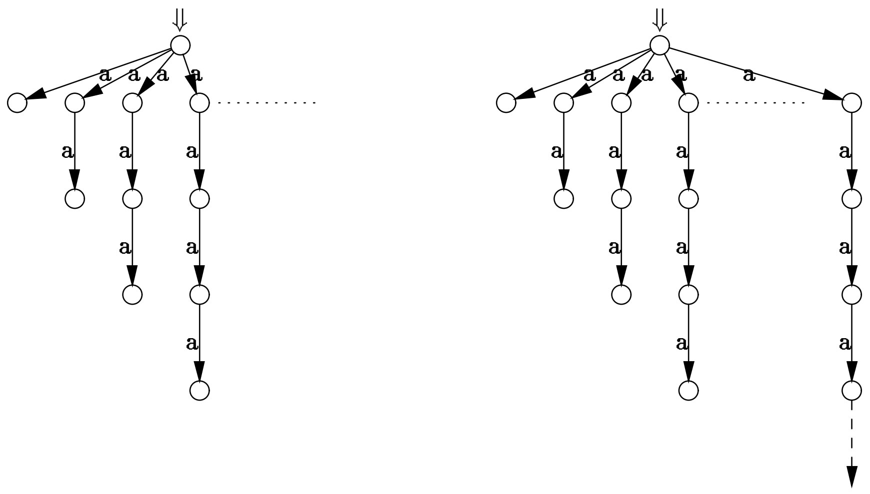

FIG. 3. TSs that are invariant for L_m-formulas but not bisimilar.

PROOF. Figure 3 gives a counterexample. The difference between the two TSs is that the righthand one has an infinite branch which the other TS is lacking. The idea of the equivalence is roughly this. L_m-formulas have finite modal operator depth, so every L_m-formula whose truth (falsity) involves the infinite branch will also be verified (falsified) using a long enough finite branch as a substitute on the left. ∎

The counterexample from Observation 3.3 uses infinitely branching TSs (in some label a). For TSs which are finitely branching in every label a (i.e., for every state s and every label a, the set $\{t \mid s \xrightarrow{a} t\}$ is finite) we have the following

THEOREM 3.4
On TSs which are finitely branching in every label, invariance for L_m-formulas implies simple bisimulation.

PROOF. Assume $\mathcal{M}, s$ and $\mathcal{N}, r$ are finitely branching in every label, and no L_m-formula sees a difference between $\mathcal{M}, s$ and $\mathcal{N}, r$. Call $u \equiv t$ if u and t satisfy the same L_m-formulas. We show that $\equiv$ itself is a simple bisimulation which relates s and r. First, it is clear that $s \equiv r$. To see that $\equiv$ is a simple bisimulation, we have to verify the bisimulation requirements. First it is obvious that $s \models \surd$ iff $r \models \surd$. Next, assume $s \xrightarrow{a} s'$. Then because $\mathcal{M}, s \models \langle a \rangle \top$ and $s \equiv r$, the set $R = \{r' \mid r \xrightarrow{a} r'\}$ is non-empty. Because $\mathcal{N}$ is finitely branching in a, $R = \{r_1, \ldots, r_n\}$, for some $n > 0$. Suppose no r_i has $s' \equiv r_i$. Then there are $\varphi_1, \ldots, \varphi_n$ with $s' \models \varphi_i$ and $r_i \models \neg\varphi_i$. But then $s \models \langle a \rangle (\varphi_1 \wedge \cdots \wedge \varphi_n)$, and $r \models \neg\langle a \rangle (\varphi_1 \wedge \cdots \wedge \varphi_n)$, and we have a contradiction with $s \equiv r$. So there is an r' with $r \xrightarrow{a} r'$ and $s' \equiv r'$. ∎

Proceeding according to the general strategy for linking equivalence notions to modal languages, it is not difficult to find the appropriate language for finite trace equivalence.

L_{tra} $\varphi ::= \psi \mid \neg\varphi \mid \varphi \wedge \psi \quad \psi ::= \surd \mid \langle a \rangle \psi.$

The language L_{tra} is the language of Boolean combinations of path modalities, where a path modality is a formula of the form $\langle a_1\rangle \cdots \langle a_n\rangle \surd$.

PROPOSITION 3.5
$\mathcal{M}, s$ and $\mathcal{N}, r$ are finite trace equivalent iff they are invariant for L_{tra} formulas.

Similarly, it is easy to see that the appropriate language for simple simulation equivalence is the language L_{sim}.

$L_{sim}\ \varphi ::= \surd \mid \neg\surd \mid (\varphi \wedge \varphi) \mid \langle a\rangle\varphi.$

THEOREM 3.6
If $\mathcal{M}, s$ and $\mathcal{N}, r$ are simply simulation equivalent then they are invariant for L_{sim} formulas.

PROOF. Induction on the structure of an L_{sim} formula φ. ∎

THEOREM 3.7
If $\mathcal{M}, s$ and $\mathcal{N}, r$ are finitely branching in every label and invariant for L_{sim} formulas, then they are simply simulation equivalent.

PROOF. Assume $\mathcal{M}, s$ and $\mathcal{N}, r$ are finitely branching in every label, and no L_{sim} formula sees a difference between $\mathcal{M}, s$ and $\mathcal{N}, r$. Call $u \Rightarrow t$ if t satisfies any L_{sim} formula that u satisfies. We show that $\Rightarrow$ is a simple left-right simulation which relates s and r. First, it is clear that $s \Rightarrow r$. To see that $\Rightarrow$ is a left-right simulation, we have to verify the requirements on the relation. It follows from the fact that $\surd$ and $\neg\surd$ are in L_{sim} that $s \models \surd$ iff $r \models \surd$. Next, assume $s \xrightarrow{a} s'$. Assume for convenience that $s' \models \surd$ (otherwise, substitute $\neg\surd$ for $\surd$ in what follows). Then because $\mathcal{M}, s \models \langle a\rangle\surd$ and $s \Rightarrow r$, the set $R = \{r' \mid r \xrightarrow{a} r'\}$ is non-empty. Because $\mathcal{N}$ is finitely branching in a, $R = \{r_1, \ldots, r_n\}$, for some $n > 0$. Suppose no r_i has $s' \Rightarrow r_i$. Then there are $\varphi_1, \ldots, \varphi_n$ in L_{sim} with $s' \models \varphi_i$ and $r_i \models \neg\varphi_i$. But then $s \models \langle a\rangle(\varphi_1 \wedge \cdots \wedge \varphi_n)$, and $r \models \neg\langle a\rangle(\varphi_1 \wedge \cdots \wedge \varphi_n)$, and contradiction with $s \Rightarrow r$. So there is an r' with $r \xrightarrow{a} r'$ and $s' \Rightarrow r'$. The proof that there is a simple simulation in the other direction is completely analogous. ∎

3.3 Embeddings in Standard Logic

It is well known that modal languages are fragments of the full standard predicate logic over TSs. A modal formula φ can be given a first order translation through the following clauses:

$$
\begin{array}{lcll}
p^\bullet & = & Px & \\
\top^\bullet & = & x = x & \\
(\neg\psi)^\bullet & = & \neg\psi^\bullet & \\
(\psi \wedge \chi)^\bullet & = & \psi^\bullet \wedge \chi^\bullet & \\
(\langle a\rangle\psi)^\bullet & = & \exists y(R_a xy \wedge \psi^\bullet[y/x]) & \quad \text{where } y \text{ is some fresh variable.}
\end{array}
$$

Here, variables range over states while the $\xrightarrow{a}$ transitions return as two place predicate symbols R_a. The translation uses a one-place predicate letter P for every atomic proposition p, and a two-place predicate letter R_a for every modality $\langle a\rangle$.

An obvious question now is: which fragment of full predicate logic does this define? Again, bisimulation provides the answer here, which takes the form of a model-theoretic 'preservation theorem'. Theorem 3.9 below tells us which part of standard logic is appropriate for the description of the TSs, if one insists on simple bisimulation as the basic process equivalence. The backbone of the argument for Theorem 3.9 is formed by the following claim (see also [4, 41, 39]).

CLAIM 3.8
If $\mathcal{M}, s$ and $\mathcal{N}, r$ agree on all L_m formulas, then there are elementary extensions $\mathcal{M}^*, \mathcal{N}^*$ with $\mathcal{M}^*, s \equiv \mathcal{N}^*, r$ (where $\equiv$ denotes a simple bisimulation).

PROOF. By a standard result in the model theory of first-order logic, any first-order model $\mathcal{M}$ has an ω-saturated elementary extension $\mathcal{M}^*$ with the property that if $\Gamma(x)$ is a countable set of formulas with only x free and involving at most finitely many state parameters, and $\Gamma(x)$ is consistent with the theory of $\mathcal{M}$, then $\Gamma(x)$ is realized in $\mathcal{M}^*$, i.e., there is a state s in the domain of $\mathcal{M}^*$ of which every $\gamma \in \Gamma(x)$ is true in $\mathcal{M}^*$ (see [13]).

Take two ω-saturated elementary extensions $\mathcal{M}^*, \mathcal{N}^*$ of $\mathcal{M}$ and $\mathcal{N}$. We show that the relation $\equiv$ of agreeing on all (translations of) L_m-formulas is a simple bisimulation between $\mathcal{M}^*$ and $\mathcal{N}^*$ relating s to r.

First observe that $\mathcal{M}^*, s$ and $\mathcal{N}^*, r$ verify the same translations of L_m-formulas, by definition. (It follows, e.g., that s has $\surd$ iff r has.)

If there is an $s' \in S(\mathcal{M}^*)$ with $s \xrightarrow{a} s'$, then each finite subset Δ of the modal theory of s' is satisfiable in some u with $r \xrightarrow{a} u$, because of the fact that $\langle a \rangle \bigwedge \Delta$ holds at s and therefore at r. But then, by ω-saturation, the full modal theory of s' must then be satisfiable at some state r' with $r \xrightarrow{a} r'$, which means that s' and r' bisimulate. The same argument in the other direction takes care of the reverse clause. This proves that $\equiv$ is a simple bisimulation. ∎

THEOREM 3.9
For any formula φ of predicate logic with at most one variable x free, φ is equivalent to some $\psi^\bullet$ with $\psi \in L_m$, iff φ is invariant for simple bisimulations.

PROOF. (Sketch) First note that the direction from left to right is already taken care of in the argument for Theorem 3.2.

For the other direction, assume that $\varphi(x)$ has only x free and is invariant for bisimulations. Let $L(\varphi)$ be the set of semantic consequences of φ which are translations of modal formulas. We show $L(\varphi) \models \varphi$. Compactness then gives us $\psi_1, \ldots, \psi_n \in L(\varphi)$ with $\psi_1, \ldots, \psi_n \models \varphi$, whence the modal formula $\bigwedge_i \psi_i$ defines φ.

Assume $\mathcal{M}, s \models L(\varphi)$. We have to show: $\mathcal{M}, s \models \varphi$.

From $\mathcal{M}, s \models L(\varphi)$ plus the fact that every member of $L(\varphi)$ is a logical consequence of φ, it follows that the following set of formulas must be finitely satisfiable: $\{\varphi\} \cup \{\psi^\bullet \mid \psi$ any L_m-formula with $\mathcal{M}, s \models \psi\}$. Therefore, by compactness, this set has a model $\mathcal{N}$ with a state r where φ holds, and which agrees completely with s on all L_m-formulas.

By Claim 3.8, there are elementary extensions $\mathcal{M}^*, \mathcal{N}^*$ of $\mathcal{M}$ and $\mathcal{N}$ for which the relation of agreeing on all L_m formulas is a simple bisimulation.

Thus, we had $\mathcal{N} \models \varphi\ [r]$, by construction. Therefore $\mathcal{N}^* \models \varphi[r]$ by elementary extension. Now $\mathcal{M}^*, s \models \varphi$ because s and r are simply bisimilar and φ is invariant for bisimulations. Finally, $\mathcal{M}, s \models \varphi$ by elementary descent (the converse of elementary extension). ∎

3.4 From Description Languages to Process Equivalences

For appropriate description languages we have found that finite trace equivalence, simulation equivalence and simple bisimulation imply that the same formulas of an appropriate modal description language are satisfied. The languages for finite trace equivalence, simple simulation equivalence, and simple bisimulation are all (fragments of) multimodal languages. We will extend the picture as we go along and consider more powerful modal languages. This much is well known from the literature (see Hennessey and Milner [29]).

The standard way of reversing the direction of these results has been the method of Theorem 3.4 (also in [29]). The relationship between our approach and that of [29] is somewhat delicate. Finite branching does not imply ω-saturation, and the converse does not hold either. Thus the scope of Claim 3.8 and Theorem 3.4 is different. Nevertheless, the connection can be made tighter. The argument for Claim 3.8 really requires only "2-saturation" [22] which applies to satisfiability at R-successors of some point only. Now, finitely branching TSs are all 2-saturated, and thus the Hennessey and Milner result becomes an instance of the modal analysis.

4 Simulation, Invariance and Logical Definability

4.1 Temporal Notions of Bisimulation: Future and Past

The first use of modal logic in computer science has been via 'temporal logic,' a version of modal logic where the basic modal operator $\langle F\rangle$ (at least once in the future) has a counterpart $\langle P\rangle$ (at least once in the past), which is interpreted via the converse of the relevant transition relation. Thus, we can look in the forward direction $s \xrightarrow{a}$ and in the backward direction $s \xleftarrow{a}$ from any state s. In the following we assume a transition system with a single label a, and we interpret F as the $\xrightarrow{a}$ relation, and P as the $\xleftarrow{a}$ relation (the converse of $\xrightarrow{a}$). Thus, $\langle F\rangle$ equals $\langle a\rangle$. However, as long as we are dealing with TSs with a fixed label a we will suppress the label and use $<$ for the 'later than' relation, and $>$ for its converse.

Here is the modal language with unary temporal operators.

L_{FP} $\varphi ::= p \mid \neg\varphi \mid (\varphi \wedge \varphi) \mid \langle F\rangle\varphi \mid \langle P\rangle\varphi.$

To apply the earlier first-order translation function $^\bullet$ to this language, we add the following clauses:

$$\begin{array}{lcll}
(\langle F\rangle\psi)^\bullet & = & \exists y(x < y \wedge \psi^\bullet[y/x]) & \text{where } y \text{ is some fresh variable,}\\
(\langle P\rangle\psi)^\bullet & = & \exists y(x > y \wedge \psi^\bullet[y/x]) & \text{where } y \text{ is some fresh variable.}
\end{array}$$

The presence of the two modal operators $\langle F\rangle$, $\langle P\rangle$ calls for a modification of the relation of bisimulation, because the definition must reflect that the 'later than' and 'earlier than' relations are each other's converse. In other words, the notion of an F bisimulation (or that of a P bisimulation) misses the connection between the two operators. Here is the appropriate notion for L_{FP}:

FIG. 4. TSs which are F bisimilar but not FP bisimilar.

DEFINITION 4.1
A relation C between $S(\mathcal{M})$ and $S(\mathcal{N})$ is an *FP bisimulation* if the following three clauses hold:

1. If sCr, then $V(s) = V(r)$,
2. • if sCr and $s < s'$, then there is a state r' in $\mathcal{N}$ with $r < r'$ and $s'Cr'$,
   • if sCr and $s' < s$, then there is a state r' in $\mathcal{N}$ with $r' < r$ and $s'Cr'$,
3. clause (2) vice versa.

In Figure 4, corresponding numbers indicate an F bisimulation. There is no FP bisimulation between the TSs, however, for any FP bisimulation has to link the state marked with 1 in the left TS to the 'earliest' state on the right (this is the only state with no incoming arrows), and the earliest states in the two TSs are not F bisimilar.

Again we can prove that no formula of L_{FP} will see a difference between FP bisimilar states, that for TSs which are both 'finitely branching' and 'finitely converging' in every label (every state has a finite number of outgoing and incoming $\xrightarrow{a}$ arrows), states that cannot be distinguished by any tense logical formula are FP bisimilar, and that being equivalent to a first-order translation of a tense logical formula coincides with being invariant for FP bisimulations.

4.2 Temporal Notions of Bisimulation: Until and Since

For further applications of temporal logic in computer science it is useful to be able to talk about 'intermediate stages' of a computation. For this purpose an additional binary modal operator $\mathcal{U}$ (until) may be introduced (see [31, 36]) with the following semantic clause:

Until $\mathcal{M}, V, s \models \varphi\mathcal{U}\psi$ if there is some s' with $s < s'$ and $\mathcal{M}, V, s' \models \psi$, and for all s'' with $s < s'' < s'$ it holds that $\mathcal{M}, V, s'' \models \varphi$.

Note that $\langle F\rangle\varphi$ is definable in terms of $\mathcal{U}$, as $\top\mathcal{U}\varphi$.

The counterpart for $\mathcal{U}$ in the other direction of time is the $\mathcal{S}$ operator. Intuitively, $\varphi\mathcal{S}\psi$ expresses that φ holds since ψ. The semantic clause is as follows.

Since $\mathcal{M}, V, s \models \varphi\mathcal{S}\psi$ if there is some s' with $s' < s$ and $\mathcal{M}, V, s' \models \psi$, and for all s'' with $s' < s'' < s$ it holds that $\mathcal{M}, V, s'' \models \varphi$.

Note that $\langle P\rangle\varphi$ is definable in terms of $\mathcal{U}$, as $\top\mathcal{S}\varphi$. Note also that the semantic clause for $\mathcal{S}$ can be obtained from that for $\mathcal{U}$ by replacing all occurrences of $<$ by $>$.

Here is a first attempt at formulating an appropriate notion of bisimulation for tense logic with $\mathcal{U}$.

DEFINITION 4.2
A relation C between $S(\mathcal{M})$ and $S(\mathcal{N})$ is an $\mathcal{U}$ *bisimulation* if the following three clauses hold:

1. If sCr, then $V(s) = V(r)$,
2. • if sCr and $s < s'$, then there is a state r' in $S(\mathcal{N})$ with $r < r'$ and $s'Cr'$,
   • if $sCr, s < s', r < r', s'Cr'$ and $s < s'' < s'$, then there is a state r'' in $S(\mathcal{N})$ with $r < r'' < r'$ and $s''Cr''$,
3. clause (2) vice versa.

THEOREM 4.3
$\mathcal{U}$ formulas are invariant under $\mathcal{U}$ bisimulations.

PROOF. The proof uses an induction argument. We just treat the crucial clause. Assume $\mathcal{M}, s \models \varphi\mathcal{U}\psi$ and suppose sCr, with C an $\mathcal{U}$ bisimulation. Then by the semantic clause for $\mathcal{U}$, there is an s' with $\mathcal{M}, s' \models \psi$ such that for all s'' with $s < s'' < s'$, $\mathcal{M}, s'' \models \varphi$. Therefore, by the first part of the first clause of the bisimulation definition, there is an r' with $r < r'$ and $s'Cr'$, and we get from the induction hypothesis that $\mathcal{M}, r' \models \psi$. Now take an arbitrary r'' with $r < r'' < r'$. By the second part of the second clause of the bisimulation definition, there is an s'' with $s < s'' < s'$ and $s''Cr''$. By the fact that $s < s'' < s'$, $\mathcal{M}, s'' \models \varphi$, and by the induction hypothesis for φ, $\mathcal{M}, r'' \models \varphi$. This establishes that $\mathcal{M}, r \models \varphi\mathcal{U}\psi$. To derive from $\mathcal{M}, r \models \varphi\mathcal{U}\psi$ that $\mathcal{M}, s \models \varphi\mathcal{U}\psi$, use the bisimulation clauses in the other direction. ■

FIG. 5. TSs which are FP bisimilar but not $\mathcal{U}$ bisimilar.

Figure 5 gives an example of a pair of TSs that are FP bisimilar but not $\mathcal{U}$ bisimilar. To see that the roots do not $\mathcal{U}$ bisimulate, note that in the righthand TS, $\bot\mathcal{U}\top$ is true at the root, while in the left TS this formula is false.

Now we would like to prove a converse result. We extend the earlier translation function $^\bullet$ with a translation clause for $\mathcal{U}$ formulas.

$$(\psi\mathcal{U}\chi)^\bullet \quad = \quad \exists y(x < y \wedge \chi^\bullet[y/x] \wedge \forall z((x < z \wedge z < y) \rightarrow \psi^\bullet[z/x])).$$

The $^\bullet$ translation clause for $\mathcal{S}$ formulas is similar. What we would like to prove is something like the following: a predicate logical formula $\varphi(x)$ with one free variable is equivalent to a first-order translation of a formula in the $\mathcal{U}$ language iff $\varphi(x)$ is invariant for $\mathcal{U}$ bisimulations.

The above argument involving saturation will not work, however. Saturation guarantees us only that every type in x which is consistent with a given set of formulas will be realized, but that is not quite enough to get at an $\mathcal{U}$ bisimulation. The problem is that the $\mathcal{U}$ bisimulation condition relates a state to two other states (by betweenness), and the satisfiability of formulas with one free variable does not provide us with such a link to a pair of individuals.

OPEN PROBLEM 4.4
Find a notion of $\mathcal{U}$ bisimulation that admits a preservation result.

4.3 Decomposition of 'Until': Two-Dimensional Temporal Logic of Statements and Procedures

The trouble with the $\mathcal{U}$ operator is that it involves a combination of an existential and a universal quantification, as its translation clause shows, while our model-theoretic analysis only works smoothly for one existential quantification. Taking our cue from [5] we can remedy this by decomposing the $\mathcal{U}$ operator. We shall pursue this issue here, to demonstrate our earlier point about 'designing bisimulations for languages', but also to show how a new conception of bisimulation will emerge in the process as a relation between tuples of states, rather than single states. We introduce the following two sorted language B of temporal logic with composite 'between' procedures.

B formulas $\varphi ::= \surd \mid \neg\varphi \mid (\varphi \wedge \varphi) \mid Do(\alpha)$.
B procedures $\alpha ::= R\varphi \mid \overline{\alpha} \mid \alpha \cap \alpha \mid \alpha;\alpha$.

The format for a truth definition here is that of a more-dimensional modal logic. The semantic clauses for the Boolean combinations of formulas are as before. The clause for the 'domain' modality Do runs as follows:

- $\mathcal{M}, s \models Do(\alpha)$ iff there is some $s' \in S(\mathcal{M})$ with $s < s'$ and $\mathcal{M}, s, s' \models \alpha$.

The semantic clause for the domain modality is stated in terms of a satisfaction relation $\mathcal{M}, s, s' \models \alpha$ for the B procedure α. Intuitively, $Do(\alpha)$ is true for the domain of the relation given by $\mathcal{M}, s, s' \models \alpha$. The satisfaction relation for B procedures is given by the following clauses:

- $\mathcal{M}, s, s' \models R\varphi$ *iff* $\mathcal{M}, s' \models \varphi$.
- $\mathcal{M}, s, s' \models \overline{\alpha}$ *iff it is not the case that* $\mathcal{M}, s, s' \models \alpha$.
- $\mathcal{M}, s, s' \models \alpha_1 \cap \alpha_2$ *iff* $\mathcal{M}, s, s' \models \alpha_1$ *and* $\mathcal{M}, s, s' \models \alpha_2$.
- $\mathcal{M}, s, s' \models \alpha_1;\alpha_2$ *iff there is some* s'' *with* $s < s'' < s'$ *and* $\mathcal{M}, s, s'' \models \alpha_1$ *and* $\mathcal{M}, s'', s' \models \alpha_2$.

The crucial clause is the one for the composition $\alpha_1;\alpha_2$. It refers to a state in between the beginning and end state of the $\alpha_1;\alpha_2$ relation. Intuitively, the modality $\alpha_1;\alpha_2$ gives the pairs of states s, s' having a state s'' in between, i.e., a state such that $s < s'' < s'$. The other modalities are for bookkeeping. R (Righthand) is for making statements about righthand members of state pairs, and $^-$ and $\cap$ are for making Boolean combinations. Note that the mention of the temporal accessibility relation $<$ in the interpretation of $\alpha_1;\alpha_2$ ensures that $\mathcal{M}, s, s' \models \overline{\alpha_1;\alpha_2}$ iff for all s'' with $s < s'' < s'$ it holds that $\mathcal{M}, s, s'' \not\models \alpha_1$ or $\mathcal{M}, s'', s' \not\models \alpha_2$, and this is precisely what we need to express the meaning of 'Until'. ('Until' is also definable in the language of [40], a dynamic modal language along the same lines as our B language but considerably stronger.)

We give some examples of B formulas and B procedures with their meanings. Let ∇ be short for $R\top$, which denotes the universal relation. Then we have, e.g., the following:

$\mathcal{M}, s, s' \models \nabla$		always,
$\mathcal{M}, s, s' \models R\varphi; \nabla$	iff	$\exists s''$ with $s < s'' < s'$ and $\mathcal{M}, s'' \models \varphi$,
$\mathcal{M}, s, s' \models R\varphi; R\psi$	iff	$\mathcal{M}, s' \models \psi$ and $\exists s''$ with $s < s'' < s'$ and $\mathcal{M}, s'' \models \varphi$,
$\mathcal{M}, s, s' \models \overline{R\neg\varphi; \nabla}$	iff	$\neg\exists s''$ with $s < s'' < s'$ and $\mathcal{M}, s'' \models \neg\varphi$,
$\mathcal{M}, s, s' \models (\overline{R\neg\varphi; \nabla}) \cap R\psi$	iff	$\mathcal{M}, s' \models \psi$ and $\forall s''$ with $s < s'' < s'$ it holds that $\mathcal{M}, s'' \models \varphi$,
$\mathcal{M}, s \models Do(\nabla)$	iff	$\exists s' : s < s'$,
$\mathcal{M}, s \models Do(R\varphi; \nabla)$	iff	$\exists s', \exists s'' : s < s', s < s'' < s'$ and $\mathcal{M}, s'' \models \varphi$,
$\mathcal{M}, s \models Do(R\varphi; R\psi)$	iff	$\exists s' : s < s'$ and $\mathcal{M}, s' \models \psi$ and $\exists s''$ with $s < s'' < s'$ and $\mathcal{M}, s'' \models \varphi$,
$\mathcal{M}, s \models Do(\overline{R\neg\varphi; \nabla})$	iff	$\exists s' : s < s' : \neg\exists s''$ with $s < s'' < s'$ and $\mathcal{M}, s'' \models \neg\varphi$,
$\mathcal{M}, s \models Do((\overline{R\neg\varphi; \nabla}) \cap R\psi)$	iff	$\exists s' : s < s'$ and $\mathcal{M}, s' \models \psi$ and $\forall s''$ with $s < s'' < s'$ it holds that $\mathcal{M}, s'' \models \varphi$.

The examples make clear that we can define the modality $\langle F \rangle \varphi$ as $Do(R\varphi)$, while $\varphi \mathcal{U} \psi$ is rendered in the language B as $Do((\overline{R\neg\varphi; \nabla}) \cap R\psi)$.

REMARK 4.5
The evaluation of a B procedure α may be rephrased in terms of an accessibility relation R_α, which is the propositional dynamic logic format. In fact, the 'between' language bears quite a bit of similarity to the language of propositional dynamic logic (Section 5 below), for the composite temporal procedures are like program modalities. In the program view, $R\varphi; R\psi$ is interpreted as a procedure to go to a $<$ successor, check if φ is true, then go to a next $<$ successor and check if ψ is true.

The B bisimulation notion that works for this language does not only relate states to states, but also pairs of states to pairs of states.

DEFINITION 4.6
A relation C between states in $\mathcal{M}$ and $\mathcal{N}$, and also between pairs of states (s, s') in $\mathcal{M}$ and (r, r') in $\mathcal{N}$ is a *B bisimulation* if the following clauses hold:

1. If sCr, then $V(s) = V(r)$,
2. • if sCr and $s < s'$, then there is an r' with $r < r'$ and $(s, s')C(r, r')$,
   • if $(s, s')C(r, r')$, then sCr and $s'Cr'$
   • if $(s, s')C(r, r')$, then for all s'' with $s < s'' < s'$ there is an r'' with $r < r'' < r'$, $(s, s'')C(r, r'')$ and $(s'', s')C(r'', r)$.
3. clause (2) vice versa.

Figure 6 gives an example of a pair of TSs that are both $\mathcal{U}$ bisimilar and B bisimilar.

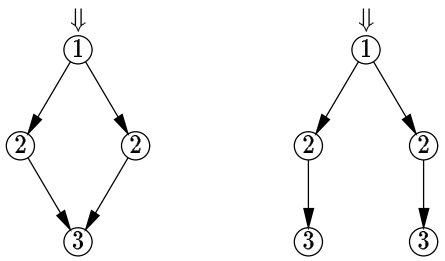

FIG. 6. TSs which are both $\mathcal{U}$ bisimilar and B bisimilar.

OPEN PROBLEM 4.7
From the fact that $\mathcal{M}, s$ and $\mathcal{N}, r$ are B bisimilar, does it follow that they are also $\mathcal{U}$ bisimilar?

PROPOSITION 4.8
Every $\mathcal{U}$ bisimulation can be strengthened to a B bisimulation.

PROOF. Assume C is a $\mathcal{U}$ bisimulation between $\mathcal{M}$ and $\mathcal{N}$. Define C' by: $(s, s')C'(r, r')$ if $s < s'$, $r < r'$, sCr and $s'Cr'$. We claim that $T = C \cup C'$ is a B bisimulation.

If sTr and $s < s'$ then sCr, so by the fact that C is an $\mathcal{U}$ bisimulation, there is an r' with $r < r'$ and $s'Cr'$. It follows that $(s, s')T(r, r')$ by the definition of T.

Assume $(s, s')T(r, r')$. Then by the definition of T, $sCr, s < s', r < r', s'Cr'$, and by the fact that C is a $\mathcal{U}$ bisimulation, for all s'' with $s < s'' < s'$ there is an r'' with $r < r'' < r'$ and $s''Cr''$. Hence $(s, s'')T(r, r'')$ and $(s'', s')T(r'', r')$. ∎

THEOREM 4.9
Let C be a B bisimulation between $\mathcal{M}$ and $\mathcal{N}$ with sCr and $(s, s')C(r, r')$. Then for all B formulas φ it holds that $\mathcal{M}, s \models \varphi$ iff $\mathcal{N}, r \models \varphi$, and for all B procedures α it holds that $\mathcal{M}, s, s' \models \alpha$ iff $\mathcal{N}, r, r' \models \alpha$.

PROOF. The proof again uses an induction argument, but now we need simultaneous induction on formulas and procedures. The interesting cases are the cases of a modal formula $Do(\alpha)$ and of a modal procedure α.

First the case of a modal formula. Assume $\mathcal{M}, s \models Do(\alpha)$. This means that there is some s' with $s < s'$ and $\mathcal{M}, s, s' \models \alpha$. From sCr, $s < s'$ and the fact that C is a B bisimulation, there is an r' with $r < r'$ and $(s, s')C(r, r')$. Now the induction hypothesis for α yields that $\mathcal{N}, r, r' \models \alpha$, and from this plus $r < r'$ we have that $\mathcal{N}, r \models Do(\alpha)$. The other direction is similar.

Next the case of a B procedure. Assume that α has the form $R\varphi$, and the induction hypothesis holds for φ. Then $\mathcal{M}, s, s' \models \alpha$ implies $\mathcal{M}, s' \models \varphi$, so from the induction hypothesis, plus the fact that $(s, s')C(r, r')$ implies $s'Cr'$, we have $\mathcal{N}, r' \models \varphi$, and thus, $\mathcal{N}, r, r' \models \alpha$.

Boolean cases $\overline{\alpha}$, $\alpha_1 \cap \alpha_2$ follow directly by the induction hypothesis.

Finally, for procedures of the form $\alpha_1;\alpha_2$, the reasoning is as follows. Assume $\mathcal{M}, s, s' \models \alpha_1;\alpha_2$. Then there is an s'' with $s < s'' < s'$ and $\mathcal{M}, s, s'' \models \alpha_1$, $\mathcal{M}, s'', s' \models \alpha_2$. Applying the B bisimulation clause to $(s, s')C(r, r')$ and $s < s'' < s'$, we conclude that there is an r'' with $r < r'' < r'$ and $(s, s'')C(r, r'')$, $(s'', s')C(r'', r')$.

Use the induction hypothesis twice to get $\mathcal{N}, r, r'' \models \alpha_1$ and $\mathcal{N}, r'', r' \models \alpha_2$. It follows that $\mathcal{N}, r, r' \models \alpha$. ∎

4.4 First-Order Analysis

We are now in the position to prove a preservation result for the B language, using the following translation from B to first-order logic (note that the translations of B formulas have one free variable, while the translations of B procedures have two):

Formula translations:

$$
\begin{array}{lcl}
p^\bullet & = & Px \\
(\neg\psi)^\bullet & = & \neg\psi^\bullet \\
(\psi \wedge \chi)^\bullet & = & \psi^\bullet \wedge \chi^\bullet \\
(Do(\alpha))^\bullet & = & \exists y(x < y \wedge \alpha^\bullet[y/x])
\end{array}
$$

Procedure translations:

$$
\begin{array}{lcl}
(R\psi)^\bullet & = & \psi^\bullet[y/x] \\
(\overline{\alpha})^\bullet & = & \neg\alpha^\bullet \\
(\alpha_1 \cap \alpha_2)^\bullet & = & \alpha_1^\bullet \wedge \alpha_2^\bullet \\
(\alpha_1; \alpha_2)^\bullet & = & \exists z(x < z \wedge z < y \wedge \alpha_1^\bullet[z/y] \wedge \alpha_2^\bullet[z/x]).
\end{array}
$$

Now the saturation argument works. The following theorem shows that the B language and the notion of B bisimulation 'fit'.

THEOREM 4.10
A first-order formula φ with at most two free variables x, y is equivalent to a $^\bullet$ translation of a formula or procedure of the B language iff φ is invariant for B bisimulations.

PROOF. The left to right direction is taken care of by Theorem 4.9.

For the other direction, we demonstrate the case of B procedures. Assume that $\varphi(x, y)$ has at most x, y free and is invariant for B bisimulations. Let $L(\varphi)$ be the set of semantic consequences of φ which are translations of B formulas, with at most x, y free. We show that $L(\varphi) \models \varphi$. Compactness then gives us $\psi_1, \ldots, \psi_n \in L(\varphi)$ with $\psi_1, \ldots, \psi_n \models \varphi$, whence the modal procedure $\bigwedge_i \psi_i$ defines φ.

Assume $\mathcal{M}, s, s' \models L(\varphi)$. We have to show: $\mathcal{M}, s, s' \models \varphi$. Now the following set of formulas must be finitely satisfiable: $\{\varphi\} \cup \{\psi^\bullet \mid \psi$ any B formula or procedure with $\mathcal{M}, s, s' \models \psi^\bullet\}$. Therefore, by compactness, this set has a model $\mathcal{N}$ with a state pair r, r' where φ holds, and which agrees completely with s, s' on all translations of B formulas. Take two ω saturated elementary extensions $\mathcal{M}^*, \mathcal{N}^*$ of $\mathcal{M}$ and $\mathcal{N}$ (see again [13]). We claim that the relation

$\langle \mathcal{M}^*, s\rangle$ satisfies the same (translations of) B formulas as $\langle \mathcal{N}^*, r\rangle$, and
$\langle \mathcal{M}^*, s, s'\rangle$ satisfies the same (translations of) B procedures as $\langle \mathcal{N}^*, r, r'\rangle$

is a B bisimulation between $\mathcal{M}^*$ and $\mathcal{N}^*$ relating s to r and s, s' to r, r'. Call this relation $\equiv$.

- If $s \equiv r$ then $V(s) = V(r)$, because of the agreement for translations of atomic formulas.

- If $s \equiv r$ and $s < s'$ then for each finite subset Δ of the modal theory of s, s' there is an $r' > r$ such that r, r' satisfy Δ. The reason for this is that $Do(\bigcap \Delta)$ holds in s and hence in r. By saturation, there must be some $r' > r$ such that r, r' satisfy the full modal theory of s, s'. But then $(s, s') \equiv (r, r')$, as required.
- Assume $(s, s') \equiv (r, r')$. Then for all modal formulas φ, $\mathcal{M}^*, s, s' \models R\varphi$ iff $\mathcal{N}^*, r, r' \models R\varphi$. Hence for all modal formulas φ, $\mathcal{M}^*, s' \models \varphi$ iff $\mathcal{N}^*, r' \models \varphi$. Hence $s' \equiv r'$.
- Assume $(s, s') \equiv (r, r')$, and there is an s'' with $s < s'' < s'$. Then for each pair of finite subsets Δ_1, Δ_2 of the modal theories of s, s'' and s'', s', respectively, there is an r'' with $r < r'' < r'$ for which r, r'' satisfies Δ_1 and r'', r' satisfies Δ_2. The reason is that $\mathcal{M}^*, s, s' \models \bigcap \Delta_1; \bigcap \Delta_2$, so $\mathcal{N}^*, r, r' \models \bigcap \Delta_1; \bigcap \Delta_2$, by the assumption. But then by saturation, there must be some r'' with $r < r'' < r'$ such that r, r'' satisfy the full modal theory of s, s'', and r'', r' the full modal theory of s'', s'. This gives $(s, s'') \equiv (r, r'')$ and $(s'', s') \equiv (r'', r')$, as required.

Thus we have $\mathcal{N}, r, r' \models \varphi$, by construction. Therefore $\mathcal{N}^*, r, r' \models \varphi$ by elementary extension, Therefore $\mathcal{M}^*, s, s' \models \varphi$ because s, s' and r, r' are B bisimilar and φ is invariant for B bisimulations. Finally, $\mathcal{M}, s, s' \models \varphi$ by elementary descent. ∎

Many details of the above first-order analysis could be changed easily. As a final illustration, we consider a two-sided 'temporal' variant. A temporal language in which the operators $\mathcal{S}, \mathcal{U}$ are both definable is obtained by extending the B language as follows.

$\mathbf{B}^{\vee}$ **formulas** $\varphi ::= \surd \mid \neg\varphi \mid (\varphi \wedge \varphi) \mid Do(\alpha) \mid Rn(\alpha)$.
$\mathbf{B}^{\vee}$ **procedures** $\alpha ::= R\varphi \mid L\varphi \mid \overline{\alpha} \mid \alpha \cap \alpha \mid \alpha; \alpha$.

The semantic clause for the new $\mathrm{B}^{\vee}$ formula construct will run as follows:

- $\mathcal{M}, s \models Rn(\alpha)$ iff there is some $s' \in S(\mathcal{M})$ with $s' < s$ and $\mathcal{M}, s', s \models \alpha$.

The interpretation of the new procedure is given by:

- $\mathcal{M}, s, s' \models L\varphi$ iff $\mathcal{M}, s \models \varphi$.

Now we can define the modality $\langle P\rangle\varphi$ as $Rn(L\varphi)$ and $\varphi\mathcal{S}\psi$ as $Rn((\overline{R\neg\varphi; \nabla}) \cap L\psi)$.

Again, this language gives rise to a notion of $B^{\vee}$ bisimulation which can be shown to be the proper notion of semantic invariance for the language, while it also allows for a preservation result.

REMARK 4.11
It can be shown that on transitive structures, $\mathrm{B}^{\vee}$ bisimulation is equivalent to the 'trisimulation' from [5]. In fact, using the techniques explained there, one may show that our $\mathrm{B}^{\vee}$ language is *expressively complete* over suitable linear orders.

5 Programming Constructs and Bisimulation

5.1 Programs and Modal Operators

The second variety of modal logic that has been applied to problems of theoretical computer science with considerable success is dynamic logic [26, 38]. Here, as in the case of the temporal languages B and $B^{\vee}$, the modal operators have internal structure: they are in fact based on programs whose structure becomes a second main concern of the formalism. Propositional dynamic logic (PDL) has the following two sorted language of formulas and programs.

PDL formulas $\varphi ::= p \mid \neg\varphi \mid (\varphi \wedge \varphi) \mid \langle\pi\rangle\varphi$.
PDL programs $\pi ::= a \mid \pi^* \mid \pi;\pi \mid \pi \cup \pi \mid \varphi?$

We use the same abbreviations as before.

PDL formulas are evaluated in TSs as follows. The clauses for propositional atoms, Booleans and simple modalities are as before. Using R_a for the set of $\xrightarrow{a}$ arrows in a TS $\mathcal{M}$, we define the relations $[\![\pi]\!]^{\mathcal{M}}$ (i,e., the sets of $\xrightarrow{\pi}$ arrows) of $\mathcal{M}$.

- $[\![a]\!]^{\mathcal{M}} = R_a$.
- $[\![\pi_1;\pi_2]\!]^{\mathcal{M}} = [\![\pi_1]\!]^{\mathcal{M}} \circ [\![\pi_2]\!]^{\mathcal{M}}$.
- $[\![\pi_1 \cup \pi_2]\!]^{\mathcal{M}} = [\![\pi_1]\!]^{\mathcal{M}} \cup [\![\pi_2]\!]^{\mathcal{M}}$.
- $[\![\pi^*]\!]^{\mathcal{M}} = ([\![\pi]\!]^{\mathcal{M}})^*$ (the reflexive transitive closure of $[\![\pi]\!]^{\mathcal{M}}$).
- $[\![\varphi?]\!]^{\mathcal{M}} = \{(s,s) \mid \mathcal{M}, s \models \varphi\}$.

The $[\![\pi]\!]^{\mathcal{M}}$ relations give the sets of state pairs s, s' that are related by the program π. Note that we might just have well have phrased the the truth definition in terms of a two-dimensional modal scheme $\mathcal{M}, s, s' \models \pi$, as we did above for many dimensional temporal logic. In the present set-up with accessibility relations $[\![\pi]\!]^{\mathcal{M}}$, the clause for program modalities becomes:

- $\mathcal{M}, s \models \langle\pi\rangle\varphi$ if there is some s' among the states of $\mathcal{M}$ with $s[\![\pi]\!]^{\mathcal{M}}s'$ and $\mathcal{M}, s' \models \varphi$.

We will use the operator A as an abbreviation of $a_1 \cup \cdots \cup a_n$, where $a_1, \ldots, a_n$ enumerates the set A of action labels, in case A is finite. If A is infinite, allowing A as an operator amounts to an extension of the language, with the stipulation: $\mathcal{M}, s \models \langle A\rangle\varphi$ iff there is an s' with $(s, s') \in \bigcup_{a \in A} R_a$ and $\mathcal{M}, s' \models \varphi$.

There exists a so-called 'filtration technique' by which an infinite TS $\mathcal{M}$ for which $\mathcal{M}, s \models \varphi$, with φ a PDL formula, can always be compressed to a finite TS $\mathcal{M}'$ with a state s' in which φ holds, whose size is bounded by a function involving the length of φ only. It follows that universal PDL validity is decidable. There are also explicit sound and complete axiomatisations of PDL (see e.g. [26]).

5.2 Bisimulation for PDL

Perhaps surprisingly, the same relation of simple bisimulation of Section 2 is the appropriate notion of similarity for PDL as well. We have the following observation.

LEMMA 5.1
If C is a simple bisimulation between $\mathcal{M}, \mathcal{N}$ with sCr, then:

1. For all PDL formulas φ: $\mathcal{M}, s \models \varphi$ iff $\mathcal{N}, r \models \varphi$,
2. For all PDL programs π, if there is an s' with $s[\![\pi]\!]^{\mathcal{M}}s'$ then there is an r' with $r[\![\pi]\!]^{\mathcal{N}}r'$.

PROOF. Simultaneous induction on the structure of φ and π. ∎

In this induction, simple bisimulation turns out to be preserved under the relational operations of $\cup$, $\circ$, * (compare the concern of process algebraists about bisimulation being a congruence for process operations; see [27], and [20] for further analysis). It is easy to see that simple bisimulation is not preserved under e.g. intersection of relations, in the sense that not every simple bisimulation for R_a and R_b is a simple bisimulation for $R_a \cap R_b$. Also, bisimulation is not preserved under taking the reversal $R^\vee$ of a relation R, as is shown by the fact that not every F bisimulation is an FP bisimulation. These observations in fact suggest a new way in which bisimulation invariance affects one's choice of a computational programming repertoire:

DEFINITION 5.2
An operation $\mathcal{O}(R_1, \ldots, R_n)$ on two-place relations is *safe for bisimulation of kind K* if every bisimulation of kind K for $R_1, \ldots, R_n$ is also a bisimulation of kind K for $\mathcal{O}(R_1, \ldots, R_n)$.

In order to prove negative results about safety for bisimulation, i.e., in order to show that certain operations are unsafe for bisimulations of certain kinds, one needs to agree on a language first. For the language of first-order logic the following result was proved in [7] (see also [8]).

THEOREM 5.3
A first-order relational operation $\mathcal{O}(R_1, \ldots, R_n)$ is safe for simple bisimulation iff $\mathcal{O}$ can be defined using the following ingredients:

- atomic relations R_ixy,
- atomic tests $p?$ (i.e., $x = y \wedge Px$),
- relation composition $R_1; R_2$ (i.e., $\exists z(R_1xz \wedge R_2zy)$),
- relation union $R_1 \cup R_2$ (i.e., $R_1xy \vee R_2xy$),
- counterdomain $\sim (R)$ (i.e., $x = y \wedge \neg\exists z Rxz$).

Theorem 5.3 can be viewed as a property of the relation of simple bisimulation, but it also is a reflection of the expressive power of first order logic. This means we can pose the same question for other definition languages (e.g., higher-order logics) and get different answers. Also, the same question can be asked for different bisimulation notions, both for first-order and higher order formalisms. Here is a sample

OPEN PROBLEM 5.4
Which first-order definable operations on relations are safe for B bisimulation?

5.3 Standard Logical Analysis

In order to get a preservation result in standard logic, we need a translation function to the language $L_{\omega_1\omega}$, the language of predicate logic with infinite disjunctions. Here are the translation clauses:

Formula translations:		
Atoms and Booleans:		as before
$(\langle\pi\rangle\varphi)^\bullet$	$=$	$\exists y(\pi^\bullet \wedge \varphi^\bullet[y/x])$
Operator translations:		
$a^\bullet$	$=$	$R_a(x, y)$
$(\pi_1; \pi_2)^\bullet$	$=$	$\exists z(\pi_1^\bullet[z/y] \wedge \pi_2^\bullet[z/x])$
$(\pi_1 \cup \pi_2)^\bullet$	$=$	$\pi_1^\bullet \vee \pi_2^\bullet$
$(\varphi?)^\bullet$	$=$	$(\varphi^\bullet \wedge x = y)$
$(\pi^0)^\bullet$	$=$	$x = y$
$(\pi^{n+1})^\bullet$	$=$	$\exists z(\pi^\bullet[z/y] \wedge (\pi^n)^\bullet[z/x])$
$(\pi^*)^\bullet$	$=$	$\bigvee_{n=0,1,2,\ldots}(\pi^n)^\bullet$.

In the translation instructions, π^n is used as an auxiliary operator, to denote the sequential composition of n copies of π. We will not pursue the matter of formulating a preservation result for PDL here.

Open problem 5.5
Give a preservation result characterizing precisely the PDL formulas in $L_{\omega_1\omega}$.

Mere bisimulation invariance cannot suffice here: Van Benthem and Bergstra [8] characterize this notion structurally as the obvious $L_{\omega_1\omega}$ version of basic modal logic, which properly includes PDL. In Section 7 we will come back to this matter and address the question of a first-order translation for PDL fragments with just one τ^* program, for which we do have a preservation result (in first order logic).

5.4 Appendix: Traces versus Choices

We have seen that all PDL program operations share the property of being safe for bisimulation. This shows that PDL is closely tied up with the branching structure of possible choices in TSs. Nevertheless, some people think of PDL programs in terms of typical 'trace languages' that would be closer to finite trace equivalence. The key issue is the role of *tests*: with only non-modal tests we have pure trace languages, with full modal tests, choices become important. In [28] there is a proof that PDL without test is essentially weaker than full PDL. But we can say more. The ability to perform tests on the branching structure of TSs (by means of 'non-Boolean' tests such as $(\langle a\rangle p)?$) is an essential feature of PDL. We will now show that testing for just trace structure reduces expressive power. This illustrates that PDL is essentially a 'branching language'. Our proof is an adaptation of Harel's proof that PDL without test is weaker than full PDL [28].

Theorem 5.6
PDL with non-Boolean test has greater expressive power than PDL with only Boolean test.

Proof. Consider the TS from Figure 7. Take a PDL language with labels a, b and with p as its only proposition letter. Assume that in the TS of the picture p is true

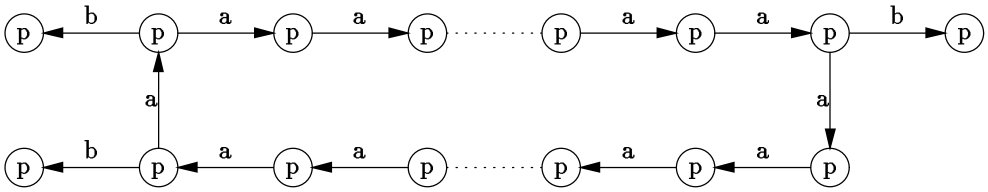

FIG. 7. TS needing non-Boolean test to distinguish top half from bottom half.

everywhere. Then all Boolean tests trivialize, by the following equivalences:

$$\begin{array}{lcl} (p)? & \leftrightarrow & Id \\ (\neg p)? & \leftrightarrow & \bot \\ (\varphi \wedge \psi)? & \leftrightarrow & \varphi?;\psi? \\ (\varphi \vee \psi)? & \leftrightarrow & \varphi? \cup \psi? \end{array}$$

With respect to the TS from Figure 7, the language with Boolean test reduces to a language without Boolean test.

Still we can distinguish the top part from the bottom part of the TS by means of the following modal test formula:

$$\varphi = \langle((\neg\langle b\rangle\top)?;a)^*\rangle(\langle b\rangle\top \wedge \langle a\rangle\langle b\rangle\top).$$

Next one proves, along the lines of the proof in [28] for Boolean test, that φ cannot be equivalent with a test free formula ψ, as any such ψ would be true in both halves of some suitably large TS as in Figure 7, with the size depending on ψ. ∎

6 Intermezzo: Tense Logic and Propositional Dynamic Logic

6.1 A Choice of Perspectives

We will now briefly return to the use of 'temporal' operators in talking about TSs. There are several different ways to make the connection between tense logic and propositional dynamic logic.

In the first place, the temporal operators can be used to describe the 'internal and backward structure' of transitions for some action a. The method here is: fix a transition label a and read the relation $<$ as $\xrightarrow{a}$ and $>$ as $\xleftarrow{a}$. This is a first-order perspective, with the relation symbol R_a substituted for $<$ in the $^\bullet$ translation for the temporal operators.

As we did not assume the temporal precedence relation to be transitive, $\langle F\rangle\varphi$, when interpreted in terms of $\xrightarrow{a}$ transitions, does not express that φ is true at some state that can be reached by following an $\xrightarrow{a}$ path, but rather that φ is true in some immediate $\xrightarrow{a}$ successor. Of course, if the $\xrightarrow{a}$ relation is transitive this makes no difference, but in general it does.

This reflection leads to a second perspective. We can single out special relations $<$ and $>$ with reasonable properties as a 'flow of time' and let F and P refer to those. This gives rise to possible interactions between R_a and $<$, e.g., it is reasonable to ask that $<$ be transitive, and that the transitive closure of every $\stackrel{a}{\rightarrow}$ transition relation be included in $<$. The following axioms accomplish this much.

$$\begin{array}{ll}
\varphi \rightarrow [P]\langle F\rangle\varphi & R_P \subseteq R_F^{\vee} \\
\varphi \rightarrow [F]\langle P\rangle\varphi & R_F \subseteq R_P^{\vee} \\
\langle F\rangle\langle F\rangle\varphi \rightarrow \langle F\rangle\varphi & \text{transitivity of } < (= R_F) \\
\langle a; a^*\rangle\varphi \rightarrow \langle F\rangle\varphi & R_a^+ \subseteq < \\
\langle a^{\vee}; a^{\vee *}\rangle\varphi \rightarrow \langle P\rangle\varphi & R_{a^{\vee}}^+ \subseteq >
\end{array}$$

TSs satisfying these axioms have one distinguished transition relation $\stackrel{later}{\longrightarrow}$ to monitor the passing of time. Time is viewed as having an intrinsic quality of its own. The 'action' that gets performed by making a move along a $\stackrel{later}{\longrightarrow}$ transition is the action of 'moving into the future', i.e., of doing what we all do by doing nothing at all.

The F operator is very useful to describe properties of TSs having to do with the availability of $\surd$ states, for it enables us to express things like 'some $\surd$ state is reachable', by means of $\langle F\rangle\surd$, and 'all reachable states without outgoing arrows are $\surd$ states', by means of $[F]([A]\bot \rightarrow \surd)$.

In the perpective just sketched temporal transitions stand apart from the other transitions of a TS. It is also possible to think of time as generated by action. This leads to a third perspective on the connection between temporal logic and dynamic logic. In this view, the temporal operator F is interpreted using the relation of 'being reachable via a finite number (one or more) of transitions from the current state, no matter what their labels are'. Now the temporal languages L_{FP} and $\mathcal{U}$ that we have considered are fragments of PDL. $\langle A; A^*\rangle\varphi$ expresses precisely what $\langle F\rangle\varphi$ expresses. For translating the past operators, use propositional dynamic logic with reversal. The appropriate fragments of PDL for the 'temporal' similarity notions under this perspective are given in Figure 8.

similarity notion	language for this notion
F bisimulation	$\varphi ::= \surd \mid \neg\varphi \mid (\varphi \wedge \varphi) \mid \langle A; A^*\rangle\varphi$.
FP bisimulation	$\varphi ::= \surd \mid \neg\varphi \mid (\varphi \wedge \varphi) \mid \langle A; A^*\rangle\varphi \mid \langle A^{\vee}; A^{\vee *}\rangle\varphi$.

FIG. 8. Temporal Fragments of PDL.

Reading $<$ as $(\bigcup R_a)^+$, which is what this third perspective amounts to, makes for a big increase in the computational complexity of the resulting version of temporal logic. This is not surprising, as the logic becomes infinitary, in the sense that the tense fragments of PDL are fragments of $L_{\omega_1\omega}$. See [43] for some comparisons. Also, some temporal modalities still elude us. To see why this is so, we must look briefly at the distinction between linear time and branching time temporal logic.

6.2 Linear Time, Branching Time and PDL

In linear time temporal logic one considers just one execution sequence of a process, in branching time temporal logic one looks at the several possible futures of a process that might go different ways. A logic for the study of branching time was introduced by Clarke and Emerson [15] (see also [26]).

The temporal languages that we have considered are certainly appropriate for linear time temporal logic. For linear time, the unary future operator $\langle F\rangle$ (at least once in the future) comes with a counterpart $[F]$ (always in the future). Using $(\bigcup R_a)^+$ to interpret the future operator, this is reflected in PDL as the distinction between $\langle A; A^*\rangle$ and $[A; A^*]$.

For branching time, quantification over time points in the future has two dimensions: sometimes versus always along one time branch, and sometimes versus always along every time branch. If a statement φ is true somewhere along some time branch, this indicates that φ may happen. If φ is true somewhere along every time branch, this means that φ is bound to happen. If φ is true everywhere along some time branch, this means that it is possible that φ will hold forever. Finally, if φ is true everywhere along every time branch, then this means that it is inevitable that φ holds forever.

Thus, for our unary future modality, we get the following interpretation clauses:

- $\exists\langle F\rangle\varphi$: somewhere along some time path, φ.
- $\forall\langle F\rangle\varphi$: somewhere along every time path, φ.
- $\exists[F]\varphi$: everywhere along some time path, φ.
- $\forall[F]\varphi$: everywhere along every time path, φ.

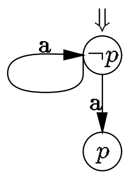

FIG. 9. TS verifying $[a][((\neg p)?; a)^*]\langle a^*\rangle p$.

It is well known from the theoretical computer science folklore that not all of the unary branching operators are expressible in PDL; see e.g., [18]. At first sight, $[A][((\neg\varphi)?; A)^*]\langle A^*\rangle\varphi$ looks like a reasonable translation of $\forall\langle F\rangle\varphi$. To see why this does not work, consider the TS of Figure 9.

In the root note of this TS, $[a][((\neg p)?; a)^*]\langle a^*\rangle p$ is true, but still, not for all paths through this TS does it hold that eventually p will become true on that path, for the process could loop from the root state to the root state indefinitely. The problem is that PDL does not allow us to express the presence or absence of such loops.

There are extensions of PDL with constructions for loop detection in which the branching time operators are expressible. Such extensions call for generalizations over previous concerns, e.g., the study of a notion of 'bisimulation with loop detection'.

In Section 8 we return to the topic of branching, in connection with the treatment of silent actions in TSs.

7 Modal Logic and Process Algebra

7.1 Varieties of Process Semantics

The process semantics of finite trace equivalence, simulation, simple bisimulation, τ bisimulation, simple branching bisimulation, and branching bisimulation can all be described as fragments of PDL. It will not come as a surprise that it is possible to isolate appropriate description languages as fragments of PDL for various other proposals of process semantics. Here are some examples of this.

The appropriate language for 'completed trace equivalence' (a variant of finite trace equivalence where the finite traces have to end in states without outgoing transitions):

L_{ct} $\varphi ::= \psi \mid \varphi \wedge \psi \quad \psi ::= ([A]\bot \wedge \surd) \mid ([A]\bot \wedge \neg\surd) \mid \langle a \rangle \psi.$

The appropriate language for 'failure semantics' [30]:

L_{fs} $\varphi ::= \psi \mid \varphi \wedge \psi \mid (\varphi \wedge [X]\bot) \quad (X \subseteq A) \quad \psi ::= \surd \mid \neg\surd \mid \langle a \rangle \psi.$

To get at the appropriate language for 'ready-trace semantics' [1] (called 'barbed wire semantics' in [37]), we take $X!$ to be an abbreviation of

$$(\bigwedge_{x \in X} \langle x \rangle \top \wedge \bigwedge_{y \in A - X} [y]\bot)?.$$

L_{rt} $\varphi ::= \psi \mid \varphi \wedge \psi \quad \psi ::= \surd \mid \neg\surd \mid \langle a \rangle \psi \mid \langle X! \rangle \psi \quad (X \subseteq A).$

7.2 Operations on Transition Systems

An algebraic approach to TSs is provided in process algebra, where TSs are introduced as solutions for algebraic equations formulated in terms of construction operations on TSs [2, 10], and processes are defined as equivalence classes of TSs modulo bisimulation.

The construction operations of process algebra in the spirit of [10] work on equivalence classes of rooted TSs with an interpretation for $\surd$. We prefer to introduce the operations at the level of the TSs themselves, so we are looking in fact at graph operations in the spirit of [16].

Definition 7.1
A *basic* TS is a TS of the form $s \xrightarrow{a} \surd$, where s is the root state, a is a label from a set of action labels A, and $\surd$ denotes the state that the arrow is pointing to and indicates that it is a success state.

Apart from the basic TSs, there are two special TSs.

Definition 7.2
ϵ is the TS consisting of one state marked by $\surd$ and without arrows; δ is the TS consisting of one state not marked by $\surd$ and without arrows.

Next, we need operations on TSs to construct further TSs. The construction operation $+$ (sum) takes two rooted TSs (with disjoint state sets) and identifies the roots to form a new rooted TS. A slight complication here is that the root of one or both TSs may have incoming arrows, spoiling the idea of a 'choice' between two processes. Therefore the sum operation has to be defined on rooted TSs with roots without incoming arrows. If a rooted TS has a root with an incoming arrow, we use the following root unwinding operation ρ to remedy this. $\rho(\mathcal{M}, s) = (\mathcal{M}', s')$ where s' is a new state which has $\surd$ iff s has $\surd$ and which has an arrow $s' \xrightarrow{a} r$ in $\mathcal{M}'$ for all arrows $s \xrightarrow{a} r$ in $\mathcal{M}$, and where $\mathcal{M}'$ is such that all the arrows and states of $\mathcal{M}$ which cannot be reached from s' are left out.

Definition 7.3
The construction operation $+$ (sum) takes two rooted TSs (with disjoint state sets), first unwinds the roots and then identifies them to form a new rooted TS. A $\surd$ mark on one of the roots is inherited by the new root.

Because of the way in which $\surd$ is inherited, δ is a neutral element for this operation.

Definition 7.4
The construction operation $\cdot$ (product) takes two rooted TSs (with disjoint state sets) and identifies the root of (a copy of) the second one with all nodes of the first which have the $\surd$ mark, while erasing this $\surd$ mark.

It is not difficult to see that ϵ is a neutral element for this operation.

There are also versions of process algebra where the $\cdot$ operation is replaced by operations of 'action prefixing' $a\cdot$ (see e.g. [33]).

Definition 7.5
The construction operation $||$ (free merge) takes two rooted TSs and makes a new one by taking the cartesian product of the state sets as the new state set, taking the ordered pair of the two roots as the new root, marking $\langle s, r\rangle$ with $\surd$ if both s and r have $\surd$ in the original TSs, and taking $\langle s, r\rangle \xrightarrow{a} \langle s', r'\rangle$ as a new transition arrow if

- $s = s'$ and $r \xrightarrow{a} r'$, or
- $r = r'$ and $s \xrightarrow{a} s'$, or
- $s \xrightarrow{a} s'$ and $r \xrightarrow{a} r'$.

More complex constructions on TSs exist (abstraction, pruning, recursion), but these will not be discussed in this paper.

7.3 Modal Languages and Process Term Languages

What we would like to do here is to put a new research direction for modal logic on the research agenda: the investigation of the connections between the process operations given above on one hand and syntactic and semantic operations in modal logic on the other, and of the connections between equational process languages and modal languages. There is an intriguing relation between the 'process' operations on TSs and various syntactic operations on modal formulas satisfying these TSs.

The key fact in the earlier application of modal logic to process diagrams (rooted TSs) is that modal languages give an 'internal' description of a TS, describing possible 'runs' of the associated process, while bisimulation invariance gives an 'external' description of the TS, and for suitable choices of language and bisimulation relation, these two descriptions match. Let us now consider the connections with process algebra.

For a start, it may seem that there is a difference between our earlier bisimulations and the bisimulation notions as employed in Process Algebra (e.g., in [2]), where bisimulation is understood to be a relation on one big domain of TSs. But this is not a real difference with the present approach, for it is always possible to collect TSs in one super TS, by taking disjoint unions. In Process Algebra, characterization results take the form: $s = r$ in some Process Algebra calculus iff s and r bisimulate in some TS containing both of these states according to some definition of bisimulation. We propose to look at s, r in their own TSs, for which we can take the generated submodels.

DEFINITION 7.6
If $\mathcal{M}$ is a TS with $s \in S(\mathcal{M})$, then $\mathcal{M}^s$, the subTS or *submodel generated by* s, is defined as follows. Its domain is the smallest subset of $S(\mathcal{M})$ containing s which is closed under $\xrightarrow{a}$-successors for all $a \in A$. The interpretation of $\surd$ in $\mathcal{M}^s$ is the interpretation of $\surd$ in $\mathcal{M}$ restricted to $S(\mathcal{M}^s)$. The transitions of $\mathcal{M}^s$ are the transitions of $\mathcal{M}$ restricted to $S(\mathcal{M}^s)$.

Evidently, the inclusion from $\mathcal{M}^s$ to $\mathcal{M}$ is a simple bisimulation.

These two links, one between Process Algebra calculi and bisimulation notions, and the other between bisimulation notions and formalisms of modal logic (that can also be axiomatized), suggest the possibility of direct translations between process algebra equations and modal formulas. We will now explore this idea in several directions.

7.4 First Approach: Process Term Equalities as Program Equivalences in PDL

The most straightforward link between process algebra and PDL is a mapping from term equalities to PDL Equivalences. Here is an example of what we have in mind.

Term translations:

$$
\begin{array}{lcl}
a^\circ & = & a \\
(x+y)^\circ & = & x^\circ \cup y^\circ \\
(xy)^\circ & = & x^\circ ; y^\circ \\
(\epsilon)^\circ & = & \top ? \\
(\delta)^\circ & = & \bot ?
\end{array}
$$

Equality translation:

$$
(x = y)^\circ \quad = \quad \langle x^\circ \rangle \surd \leftrightarrow \langle y^\circ \rangle \surd .
$$

The following table gives the axioms of Basic Process Algebra, with their PDL counterparts under the $^\circ$ translation.

$$\begin{array}{lll}
x+y=y+x & \rightsquigarrow & \langle \pi_1 \cup \pi_2\rangle\surd \leftrightarrow \langle \pi_2 \cup \pi_1\rangle\surd \\
(x+y)+z=x+(y+z) & \rightsquigarrow & \langle (\pi_1 \cup \pi_2) \cup \pi_3\rangle\surd \leftrightarrow \langle \pi_1 \cup (\pi_2 \cup \pi_3)\rangle\surd \\
x+x=x & \rightsquigarrow & \langle \pi \cup \pi\rangle\surd \leftrightarrow \langle \pi\rangle\surd \\
(x+y)z=xz+yz & \rightsquigarrow & \langle (\pi_1 \cup \pi_2);\pi_3\rangle\surd \leftrightarrow \langle \pi_1;\pi_3 \cup \pi_2;\pi_3\rangle\surd \\
(xy)z=x(yz) & \rightsquigarrow & \langle (\pi_1;\pi_2);\pi_3\rangle\surd \leftrightarrow \langle \pi_1;(\pi_2;\pi_3)\rangle\surd \\
x+\delta=x & \rightsquigarrow & \langle \pi \cup \bot?\rangle\surd \leftrightarrow \langle \pi\rangle\surd \\
\delta x=\delta & \rightsquigarrow & \langle \bot?;\pi\rangle\surd \leftrightarrow \langle \bot?\rangle\surd \\
x\epsilon=x & \rightsquigarrow & \langle \pi;\top?\rangle\surd \leftrightarrow \langle \pi\rangle\surd \\
\epsilon x=x & \rightsquigarrow & \langle \top?;\pi\rangle\surd \leftrightarrow \langle \pi\rangle\surd
\end{array}$$

PROPOSITION 7.7
1. Translation $^\circ$ is 'sound'. 2. Translation $^\circ$ is not 'faithful'.

PROOF. (1) follows from the fact that the translation of every Basic Process Algebra axiom is a valid principle of PDL.

(2) The translation is too coarse, for it maps process terms which are not equivalent to PDL equivalent formulas. For example, $x(y+z) \neq (xy+xz)$, but $(x(y+z))^\circ \rightsquigarrow x^\circ;(y^\circ \cup z^\circ)$, and this PDL program is equivalent to the program $(x^\circ;y^\circ) \cup (x^\circ;z^\circ)$, which is the $^\circ$ translation of $xy+xz$. ∎

OPEN PROBLEM 7.8
Is there a faithful embedding of the Basic Process Algebra axioms in PDL?

We should look at the Process Algebra equations as telling the story of the TS operations from Section 7.2. The equation $x+y=y+x$ tells us that the rooted TS $\mathcal{M}, s+\mathcal{N}, r$ cannot be distinguished from the rooted TS $\mathcal{N}, r+\mathcal{M}, s$. But because of the fact that the notion of similarity at the basis of this is the same as the similarity notion of (a suitable fragment of) PDL, we can use modal logic to formulate syntactic operations corresponding to the process operations on TSs, and then see which modal principles correspond with the process algebra equations.

7.5 Second Approach: Canonical Approximations at Finite Depth

In Process Algebra, process terms denote rooted TSs, and process equations stipulate which TSs are viewed as descriptions of the same process. In modal logic, formulas denote classes of rooted TSs, and fragments of the modal languages induce similarity notions telling us which rooted TSs cannot be told apart by a formula from the fragment. In general, it is not possible to associate 'normal form' modal formulas with single TSs, for formulas are inherently finite, and TSs are not. However, if we agree not to probe into the TSs beyond a given finite depth n, the situation changes.

The finite depth perspective immediately leads to a parallel between modal logic and process algebra.

DEFINITION 7.9
The *tree unwinding operation* T of rooted TSs $\mathcal{M}, s$ is given by: $T(\mathcal{M}, s)$ has a root corresponding to the empty path in $\mathcal{M}, s$, a state θ for every finite path of $\mathcal{M}, s$, and an arrow $\theta \xrightarrow{a} \theta'$ iff the path corresponding to θ' in $\mathcal{M}$ is an extension with $\xrightarrow{a}$ of the path corresponding to θ in $\mathcal{M}$.

The depth of a state in a tree is the length of the (unique) path from the root of the tree to that state. Tree unwinding is well known in process theory [9], and pre-existed in modal logic under the name of 'unravelling' ([42]; see also [12], Section 13).

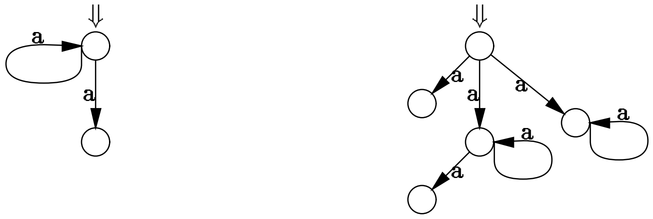

FIG. 10. TSs that can be unwound to the trees of the next figure.

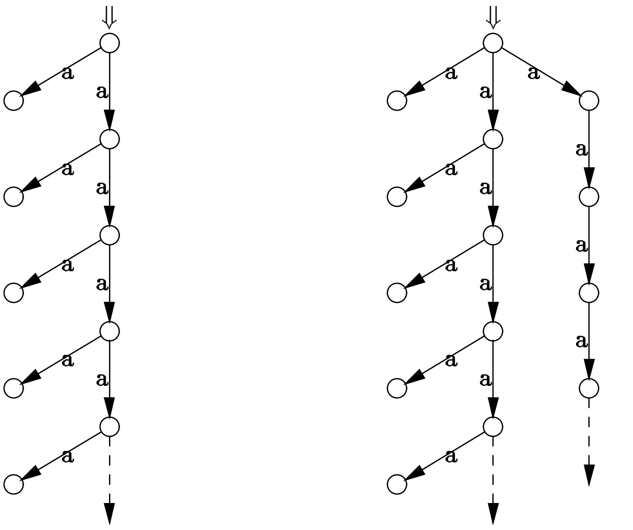

FIG. 11. Unwindings of the TSs of the previous figure.

The TSs in Figure 11 give the tree unwindings of the TSs of Figure 10. Note also that for any rooted TS $\mathcal{M}, s$ it holds that $\mathcal{M}, s$ and $T(\mathcal{M}, s)$ are simply bisimilar.

DEFINITION 7.10
The *projection up to depth n* of TS $\mathcal{M}, s$, $\pi_n(\mathcal{M}, s)$, is obtained by first unwinding $\mathcal{M}, s$ to a tree $\mathcal{M}', s'$, then removing all arrows leaving from a state at depth n, and finally removing all inaccessible states and arrows.

DEFINITION 7.11 (Operator depth)
The operator depth of a multimodal formula φ is given by:

- $d(p) = 0$,

- $d(\neg\varphi) = d(\varphi)$,
- $d(\varphi \wedge \psi) = \max(d(\varphi), d(\psi))$,
- $d(\langle a\rangle\varphi) = d(\varphi) + 1$.

PROPOSITION 7.12
For all φ with $d(\varphi) \leq n$: $\mathcal{M}, s \models \varphi$ iff $\pi_n(\mathcal{M}, s) \models \varphi$.

The next definition is from [21].

DEFINITION 7.13 (Bisimilarity up to depth n)

$$\begin{array}{lcl}
\mathcal{M}, s \sim_0 \mathcal{N}, r & \stackrel{\text{def}}{\Longleftrightarrow} & s \in \surd \iff r \in \surd. \\[1em]
\mathcal{M}, s \sim_{n+1} \mathcal{N}, r & \stackrel{\text{def}}{\Longleftrightarrow} & s \in \surd \iff r \in \surd, \\
& & \forall a \in A \forall s' \in S(\mathcal{M}) \\
& & (s \xrightarrow{a} s' \Rightarrow \exists r' \in S(\mathcal{N})(r \xrightarrow{a} r' \ \&\ s' \sim_n r')), \\
& & \forall a \in A \forall r' \in S(\mathcal{N}) \\
& & (r \xrightarrow{a} r' \Rightarrow \exists s' \in S(\mathcal{M})(s \xrightarrow{a} s' \ \&\ s' \sim_n r')).
\end{array}$$

The following results are standard:

LEMMA 7.14
For every n, the number of equivalence classes induced by $\sim_n$ on the class of rooted TSs is finite.

PROPOSITION 7.15
$\mathcal{M}, s \sim_n \mathcal{N}, r$ iff for all multimodal formulas φ with $d(\varphi) = n$: $\mathcal{M}, s \models \varphi$ iff $\mathcal{N}, r \models \varphi$.

PROOF. From left to right: see e.g. [21] for the idea.

From right to left: For any $n \geq 0$, associate a normal form formula with any state $\mathcal{M}, s$ as follows:

$n = 0$. $\varphi = \begin{cases} \surd & \text{if } s \in \surd \\ \neg\surd & \text{otherwise.} \end{cases}$

$n > 0$. By Lemma 7.14, there is a finite list $\psi_1, \ldots, \psi_m$ of normal form formulas for $n-1$. Let $a_1, \ldots, a_k$ be a finite enumeration of A (the set of labels of the language). Then $\varphi = \varphi_0 \wedge \bigwedge_{i=1}^{k} \bigwedge_{j=1}^{m} \chi_{ij}$, where φ_0 is the normal form formula for $\mathcal{M}, s$ for the 0 case, and $\chi_{ij} = \langle a_i\rangle\psi_j$ in case there is some t with $s \xrightarrow{a_i} t$ and ψ_j is associated with $\mathcal{M}, t$ for $n-1$, and $\chi_{ij} = \neg\langle a_i\rangle\psi_j$ otherwise.

We can now prove by induction on n that, for any $\mathcal{M}, s$ and any TS $\mathcal{N}$, the formula φ associated with $\mathcal{M}, s$ for n defines the set of states $\{u \in S(\mathcal{N}) \mid \mathcal{N}, u \sim_n \mathcal{M}, s\}$, in the sense that for all and only these $u \in S(\mathcal{N})$ we have $\mathcal{N}, u \models \varphi$. By the construction of φ, we have that $\mathcal{M}, s \models \varphi$. Also, $d(\varphi) = n$, so by the assumption of the proposition, $\mathcal{N}, u \models \varphi$. Finally, by the fact that φ is a normal form formula for n, it follows that $\mathcal{M}, s \sim_n \mathcal{N}, r$. ∎

Note that the proof of Proposition 7.15 uses general logic: the notion of operator depth is closely related to that of quantifier depth, which makes for obvious connections with the topics of Ehrenfeucht games and partial isomorphisms in standard first-order logic.

If we only look at finite processes, then we can derive 'normal form' modal formulas from process terms, as follows. In the following translation, we assume $\top$ to be a proposition symbol (not an abbreviation for $\surd \vee \neg\surd$).

$$\begin{array}{lcl} \varphi_\epsilon & = & \surd \\ \varphi_\delta & = & \top \\ \varphi_a & = & \langle a \rangle \surd \\ \varphi_{x+y} & = & \varphi_x \wedge \varphi_y \\ \varphi_{x\cdot y} & = & \varphi_x[\varphi_y/\surd]. \end{array}$$

PROPOSITION 7.16
BPA $\vdash x = y$ iff $\models \varphi_x \leftrightarrow \varphi_y$.

PROOF. From left to right: just check that all BPA axioms yield modal equivalences under this translation.

From right to left: it follows from the definition of φ_x, φ_y that $\models \varphi_x \leftrightarrow \varphi_y$ iff there exists a simple bisimulation between x and y in the term model P of BPA. Thus $\models \varphi_x \leftrightarrow \varphi_y$ iff $P \models x = y$ iff BPA $\vdash x = y$. ∎

OPEN PROBLEM 7.17
Can this translation be extended to other process operators?

7.6 Behaviour of Modal Formulas Under Process Operations

Still another way to study the process operations is by determining which modal formulas they preserve. This is in fact a new systematic question for Modal Logic (for which only a few haphazard precedents exist in the literature): given TSs $\mathcal{M}_i$ and modal formulas φ_i with $\mathcal{M}_i \models \varphi_i$, what happens to the truth of $\{\varphi_i\}$ on $\oplus_i(\mathcal{M}_i)$, for various operations $\oplus$ on TSs?

In modal logic, various operations on Kripke models have been studied, e.g., the 'rooting' of $\mathcal{M}, s$ and $\mathcal{N}, r$, notation $\mathcal{M} \oplus \mathcal{N}, x$: combine $\mathcal{M}, s$ and $\mathcal{N}, r$ by adding a state x and two transitions $x \to s$, $x \to r$. This operation is defined for models with a single accessibility relation R, i.e., TSs with one unlabelled $\to$ transition. But it can also easily be applied to TSs with different transition labels, via the following

DEFINITION 7.18
Let $\mathcal{M}$ be a TS with an interpretation for P and set of action labels A. Then $U\mathcal{M}$, the unlabeled TS based on $\mathcal{M}$, is the TS which has as its set of states

$$S(\mathcal{M}) \cup \{a_{s,t} \mid s \xrightarrow{a} t \text{ in } \mathcal{M}\},$$

as its accessibility relation

$$\{s \to a_{s,t} \mid t \in S(\mathcal{M}), s \xrightarrow{a} t \text{ in } \mathcal{M}\} \cup \{a_{s,t} \to t \mid s \in S(\mathcal{M}), s \xrightarrow{a} t \text{ in } \mathcal{M}\},$$

and as its valuation the function $V : P \cup A \to \text{POW}\,(S(U\mathcal{M}))$ with

$$V(p) = V_{\mathcal{M}}(p) \ \ (p \in P) \qquad V(a) = \{a_{s,t} \mid s \xrightarrow{a} t \text{ in } \mathcal{M}\} \ \ (a \in A).$$

Consider the following translation from the multimodal logic over proposition letters P and action labels A to modal logic with proposition letters $P \cup A$ and modality $\Diamond$.

$$\begin{array}{lcl} p^\diamond & = & p \\ (\neg\varphi)^\diamond & = & \neg\varphi^\diamond \\ (\varphi \wedge \psi)^\diamond & = & (\varphi^\diamond \wedge \psi^\diamond) \\ (\langle a\rangle\varphi)^\diamond & = & \Diamond(a \wedge \Diamond\varphi^\diamond). \end{array}$$

PROPOSITION 7.19
For all labelled TSs $\mathcal{M}$ and all multimodal formulas φ: $\mathcal{M}, s \models \varphi$ iff $U\mathcal{M}, s \models \varphi^\diamond$.

PROOF. Induction on the structure of φ. ∎

We can use this translation to apply the 'rooting' operation $\oplus$, which is defined for TSs without labels, to labelled TSs. This leads to the following

PROPOSITION 7.20
If $\mathcal{M}, s \models \varphi$ then $U\mathcal{M} \oplus U\mathcal{N}, x \models \Diamond\varphi^\diamond$.

It should be noted, though, that $U\mathcal{M} \oplus U\mathcal{N}$ still does not coincide with the Process Algebra operation $U\mathcal{M} + U\mathcal{N}$.

Another well known construction from modal logic is the 'direct product' of $\mathcal{M}$ and $\mathcal{N}$, notation $\mathcal{M} \otimes \mathcal{N}$, which has $S(\mathcal{M}) \times S(\mathcal{N})$ as its state set and $(s, r) \xrightarrow{a} (s', r')$ iff $s \xrightarrow{a} s'$ and $r \xrightarrow{a} r'$. Here we have the following preservation result.

PROPOSITION 7.21
If φ has the 'Horn format' $\langle a\rangle p_1 \wedge \cdots \wedge \langle a\rangle p_n \rightarrow [a]q$, then $\mathcal{M} \models \varphi$ implies $\mathcal{M} \otimes \mathcal{N} \models \varphi$.

But again, the operation $\otimes$ is subtly different from the nearest process algebra operation $\|$.

Finally we have the operation of 'glueing at specified positions', $\mathcal{M} \cdot_p \mathcal{N}$, of which the Process Algebra operation $\mathcal{M} \cdot \mathcal{N}$ represents the special case where $p = \surd$.

OPEN PROBLEM 7.22
What is the precise relation between 'rooting' and 'direct product' and the Process Algebra operations?

The modal study of the Process Algebra operations should start with an investigation of their preservation properties. Here are some preliminary observations for existential multimodal formulas.

L_{ex} $\varphi ::= p \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \langle a\rangle\varphi$.

PROPOSITION 7.23
If $\varphi \in L_{ex}$ then $\mathcal{M}, s \models \varphi$ implies $\mathcal{M} + \mathcal{N}, s \models \varphi$.

PROPOSITION 7.24
For all $\varphi, \psi \in L_{ex}$: $\mathcal{M}, s \cdot_p \mathcal{N}, r \models \varphi[\psi/p]$ iff $\mathcal{M}, s \models \varphi$ and $\mathcal{N}, r \models \psi$.

While the above observations focus on preservation, it should be borne in mind that at some point reduction questions will have to be tackled as well: how can modal statements about some Process Algebra construct be reduced to statements about its components? This systematic question would also be of independent interest in Modal Logic.

8 Transition Systems with Silent Steps

8.1 Silence and Branching

An important theme in the Process Algebra literature is the treatment of actions which are invisible from outside the system, the so-called *silent moves*. If one considers transition systems with a special *silent action* τ, one can study equivalence between TSs 'if τ steps do not matter'. We review two proposals from the literature and add one of our own. Our main claim in this section will be that this extension fits naturally into our framework so far.

Let $s \stackrel{\tau *}{\to} s'$ if s' can be reached from s through a finite number of τ transitions (possibly 0), and let $s \stackrel{\tau * a \tau *}{\longrightarrow} s'$ if s' can be reached from s through a finite number of τ transitions (possibly 0) followed by one a transition (we assume $a \neq \tau$), followed again by a finite number of τ transitions (possibly 0).

Definition 8.1
A relation $C \subseteq S(\mathcal{M}) \times S(\mathcal{N})$ is a τ *bisimulation* if the following three clauses hold:

1. If sCr then $V(s) = V(r)$,
2. • if sCr and $s \stackrel{\tau *}{\to} s'$ then there is a state r' with $r \stackrel{\tau *}{\to} r'$ and $s'Cr'$,
   • if $s \stackrel{\tau * a \tau *}{\longrightarrow} s'$ then there is a state r' with $r \stackrel{\tau * a \tau *}{\longrightarrow} r'$ and $s'Cr'$,
3. clause (2) vice versa.

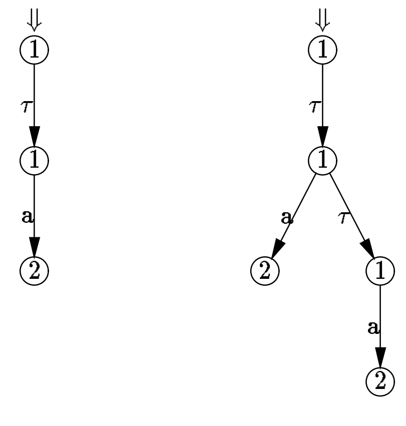

Fig. 12. τ bisimilar, but not simply bisimilar, TSs.

Figure 12 gives an example of τ bisimilar TSs (corresponding numbers indicate τ bisimilar states) which are clearly not simply bisimilar.

The definition of process equivalences that take the 'invisibility' of τ steps into account is an important topic in the process algebra literature. The definition of τ bisimulation is from Hennessey and Milner [29]. Process algebraists from Amsterdam have been looking for alternatives to ensure that τ steps do not discard choice options. Their intuition was as follows. To match a τ step in the other structure there are two possibilities: (1) in case the τ step didn't change any choice options, don't move, (2) in case the τ step did make a change, then make τ transitions through states that are all indistinguishable, until you can match the τ move on the other side (see [24]). This gives the following definition for *branching bisimulation* (called 'branching' because it preserves choice structure in both directions).

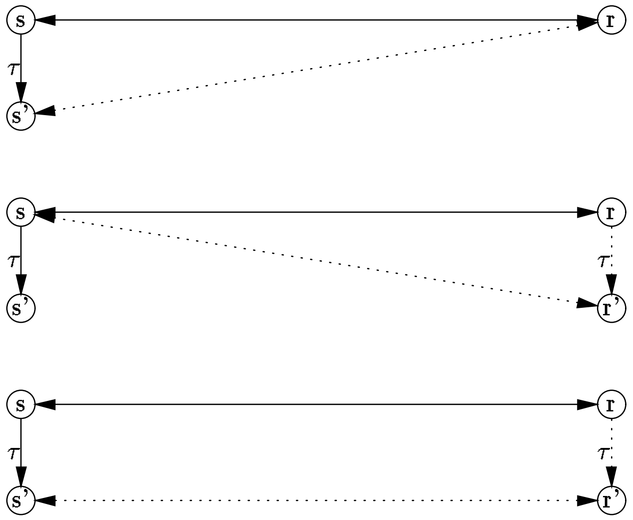

FIG. 13. Relations between τ steps for branching bisimulation.

DEFINITION 8.2
A relation C between $S(\mathcal{M})$ and $S(\mathcal{N})$ is a *branching bisimulation* if the following three clauses hold:

1. If sCr then $V(s) = V(r)$,
2. • if sCr and $s \xrightarrow{\tau} s'$, then either $s'Cr$ or there is a state $r' \in S(\mathcal{N})$ with $r \xrightarrow{\tau} r'$ and either $s'Cr'$ or ($r' \neq r$ and sCr'),
   • if $s \xrightarrow{\tau *} s' \xrightarrow{a} s''$ then there are states r', r'' in $\mathcal{N}$ with $r \xrightarrow{\tau *} r' \xrightarrow{a} r''$, $s'Cr'$ and $s''Cr''$,
3. clause (2) vice versa.

Figure 13 gives the relation between τ steps that can be made from branching bisimilar states, according to this definition.

This 'symmetric' formulation of the definition of branching bisimulation is slightly more general than the definition from the process algebra literature [24, 2]. To establish the precise connection, we need the concept of being finitely branching in a non-atomic label. We say that a TS is finitely branching in τ^* if for all of its states s the set $\{t \mid s \xrightarrow{\tau *} t\}$ is finite.

We need this concept because from the fact that a TS is finitely branching in the label τ it does not follow that for every state s the set $\{t \mid s \xrightarrow{\tau *} t\}$ is finite. See Figure 14 for a counterexample.

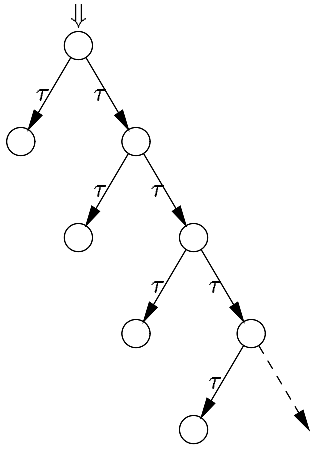

FIG. 14. TS which is finitely branching in τ but not in τ^*.

LEMMA 8.3
If C is a branching bisimulation between $S(\mathcal{M})$ and $S(\mathcal{N})$, and sCr, $s \xrightarrow{\tau} s'$, and $S(\mathcal{N})$ is finitely branching in τ^*, then either

- $s'Cr$, or
- there are $r_0, \ldots, r_n$, with $n \geq 1$, $r_0 \xrightarrow{\tau} \cdots \xrightarrow{\tau} r_n$, $r_0 = r$, $r_i = r_j$ implies $i = j$ (i.e., the states are all different), sCr_i for all $i < n$, and $s'Cr_n$.

PROOF. Assume $s \in S(\mathcal{M})$, $r \in S(\mathcal{N})$ and sCr, with C a branching bisimulation. Suppose $s \xrightarrow{\tau} s'$. Then either $s'Cr$ and we are done, or there is a state $r' \in S(\mathcal{N})$ with $r \xrightarrow{\tau} r'$ and either (i) $s'Cr'$ or (ii) $r' \neq r$ and sCr'. In case (i) we have found our r_n and we are done. In case (ii) we apply the definition again for the situation where sCr', $s \xrightarrow{\tau} s'$ and not $s'Cr'$. This gives an r'' with $r \xrightarrow{\tau} r' \xrightarrow{\tau} r''$. Repeating the argument and using the assumption that $S(\mathcal{N})$ is finitely branching in τ^* we find $r_0 \xrightarrow{\tau} \cdots \xrightarrow{\tau} r_n$ with the desired property. ∎

We conclude with an equivalence of our own, which lies in between τ bisimulation and branching bisimulation, and which preserves as much of the choice structure as we will ever need. Our intuition is that all τ paths should have a match, but that individual τ steps need not. It does not matter if some individual τ steps do not have counterparts, as long as the choice structure before and after any visible action is matched.

DEFINITION 8.4
A relation $C \subseteq S(\mathcal{M}) \times S(\mathcal{N})$ is a *simple branching bisimulation* if the following three clauses hold:

1. If sCr then $V(s) = V(r)$,
2. • if sCr and $s \xrightarrow{\tau*} s'$ then there is a state r' with $r \xrightarrow{\tau*} r'$ and $s'Cr'$,
   • if $s \xrightarrow{\tau*} s' \xrightarrow{a} s''$ then there are states r', r'' with $r \xrightarrow{\tau*} r' \xrightarrow{a} r''$, $s'Cr'$ and $s''Cr''$,

3. clause (2) vice versa.

When commenting on a draft version of this paper, Rob van Glabbeek mentioned to us that this notion has already made a brief appearance in the literature (see [14], where this equivalence is called 'quasi branching').

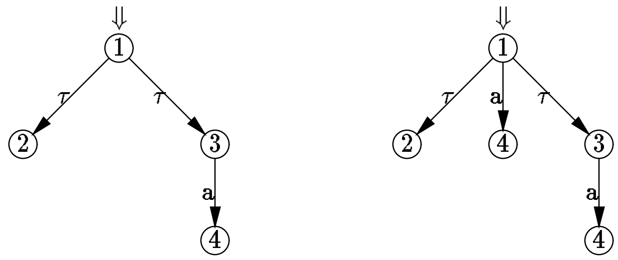

FIG. 15. τ bisimilar, but not simply branching bisimilar, TSs.

Note that the τ bisimulation from Figure 12 is also a simple branching bisimulation.

OBSERVATION 8.5
Simple branching bisimulation is a stronger notion than τ bisimulation.

PROOF. Obviously, every simple branching bisimulation is also a τ bisimulation. Figure 15 shows, however, that the two notions do not coincide. The corresponding numbers in this figure indicate a τ bisimulation. To see that there is no simple branching bisimulation, observe that in the TS on the right, the $\xrightarrow{a}$ step between the nodes marked 1 and 4 cannot be linked to the only $\xrightarrow{a}$ step in the left TS, for the source states of the $\xrightarrow{a}$ steps are not simply branching bisimilar. ∎

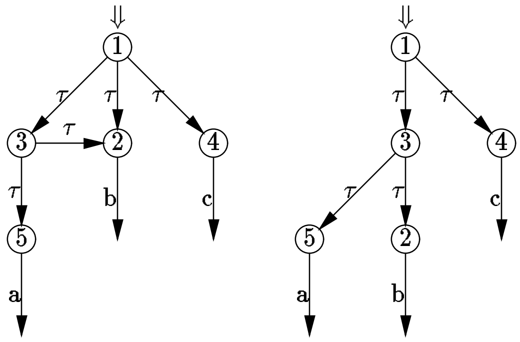

FIG. 16. Simply branching bisimilar, but not branching bisimilar, TSs.

OBSERVATION 8.6
Branching bisimulation is a stronger notion than simple branching bisimulation.

PROOF. Again, it is obvious that every branching bisimulation is a simple branching bisimulation. To see that simple branching bisimulation does not coincide with branching bisimulation, consider Figure 16, which gives an example of simply branching bisimilar TSs (with the simple branching bisimulation indicated by the matching numbers on the nodes) which are not branching bisimilar. To see that there is no branching bisimulation, observe that the $\xrightarrow{\tau}$ step from the state marked 1 to the state marked 2 in the left TS cannot be matched with a $\xrightarrow{\tau}$ step in the right TS between a pair of states of which the first is branching bisimilar to 1 and the second to 2. ∎

REMARK 8.7
Silence is a natural notion from a general logical point of view: 'Flatten' some previous action α to some τ without internal structure. This operation could be viewed as a kind of 'projection' at the level of action labels, which suggests that it is amenable to ordinary model-theoretic analysis.

8.2 Semantic Invariances Involving Silent Steps

The considerations about semantic invariance from Section 3 are easily extended to TSs involving silent steps.

CLAIM 8.8
Semantic invariances involving silent steps can be described by modal languages which are fragments of PDL.

Inspection of the notion of a τ bisimulation readily yields the appropriate modal language for it:

$L_\tau \ \varphi ::= \surd \mid \neg\varphi \mid (\varphi \wedge \varphi) \mid \langle \tau^* \rangle \varphi \mid \langle \tau^* a \tau^* \rangle \varphi.$

The semantic clauses for the new modalities become:

1. $\mathcal{M}, V, s \models \langle \tau^* \rangle \varphi$ if there is some s' among the states of $\mathcal{M}$ with $s \xrightarrow{\tau *} s'$ and $\mathcal{M}, V, s' \models \varphi$.
2. $\mathcal{M}, V, s \models \langle \tau^* a \tau^* \rangle \varphi$ if there is some s' among the states of $\mathcal{M}$ with $s \xrightarrow{\tau * a \tau *} s'$ and $\mathcal{M}, V, s' \models \varphi$.

THEOREM 8.9
If $s \in S(\mathcal{M})$ is τ-bisimilar to $r \in S(\mathcal{N})$, then for all L_τ formulas φ: $\mathcal{M}, s \models \varphi$ iff $\mathcal{N}, r \models \varphi$.

PROOF. Induction on the structure of an L_τ formula φ. ∎

THEOREM 8.10
On TSs that are finitely branching in every atomic label and also in τ^*, invariance for L_τ formulas implies τ-bisimulation.

PROOF. Same method as in Theorems 3.4 and 3.7. ∎

Let L_{sb} be the following language:

$L_{sb} \ \varphi ::= \surd \mid \neg\varphi \mid (\varphi \wedge \varphi) \mid \langle \tau^* \rangle \varphi \mid \langle \tau^*; \varphi?; a \rangle \varphi.$

Note that L_{sb} properly extends L_τ: the L_{sb} formula $\langle\tau^*;\top?;a\rangle\langle\tau^*\rangle\varphi$ has the same meaning as the L_τ formula $\langle\tau^* a\tau^*\rangle\varphi$, so we can assume the latter formula to belong to L_{sb} as well.

Consider Figure 15 again, of a pair of τ bisimilar TSs that are not simply branching bisimilar. The following L_{sb} formula distinguishes between them:

$$\varphi = \langle\tau^*;(\langle\tau^*\rangle[\tau^* a]\bot)?;a\rangle\top.$$

Formula φ asserts that some τ^* path leads to a state from which a $\xrightarrow{a}$ transition is possible, and from which also is a τ^* path to a state from which $\xrightarrow{a}$ is not reachable anymore via any τ^* path. This holds in the root state of the right TS, but is false in the root state of the left TS.

THEOREM 8.11
If s and r are simply branching bisimilar then for all $\varphi \in L_{sb}$: $(s \models \varphi$ iff $r \models \varphi)$.

PROOF. Induction on the complexity of φ. ∎

THEOREM 8.12
If $\mathcal{M}$, $\mathcal{N}$ are finitely branching in every atomic label and also in τ^*, then invariance for L_{sb} formulas implies simple branching bisimulation.

PROOF. Call $s \equiv r$ if s and r satisfy the same L_{sb} formulas. We show that $\equiv$ is a simple branching bisimulation. Assume $s \in S(\mathcal{M})$, $r \in S(\mathcal{N})$, $s \equiv r$.

Assume $s \xrightarrow{\tau*} s'$. Then because $s \models \langle\tau^*\rangle\top$ and $s \equiv r$, the set $R = \{r' \mid r \xrightarrow{\tau*} r'\}$ is non-empty. Because $\mathcal{N}$ is finitely branching in τ^*, $R = \{r_1, \ldots, r_n\}$, for some $n > 0$. Suppose no r_i has $s' \equiv r_i$. Then there are $\varphi_1, \ldots, \varphi_n$ with $s' \models \varphi_i$ and $r_i \models \neg\varphi_i$. But then $s \models \langle\tau^*\rangle(\varphi_1 \wedge \cdots \wedge \varphi_n)$, and $r \models \neg\langle\tau^*\rangle(\varphi_1 \wedge \cdots \wedge \varphi_n)$, and contradiction with $s \equiv r$. So there is an r' with $r \xrightarrow{\tau*} r'$ and $s' \equiv r'$.

Assume $s \xrightarrow{\tau*} s' \xrightarrow{a} s''$. Then $R = \{(r', r'') \mid r \xrightarrow{\tau*} r' \xrightarrow{a} r''\}$ is non-empty. Because $\mathcal{N}$ is finitely branching in τ^* and in a, $R = \{(r_1, u_1), \ldots, (r_n, u_n)\}$, for some $n > 0$. Suppose no pair (r_i, u_i) has both $s' \equiv r_i$ and $s'' \equiv u_i$. Then there are $\varphi_1, \ldots, \varphi_n, \psi_1, \ldots, \psi_n$ with $s' \models \varphi_i$ and $r_i \models \neg\varphi_i$ or $s'' \models \psi_i$ and $u_i \models \neg\psi_i$. Let φ be the conjunction of the φ_i, and ψ the conjunction of the ψ_i. Then $s \models \langle\tau^*;\varphi?;a\rangle\psi$ and $r \models \neg\langle\tau^*;\varphi?;a\rangle\psi$, and we have a contradiction with $s \equiv r$. So there is a pair r', r'' with $r \xrightarrow{\tau*} r' \xrightarrow{a} r''$ and $s' \equiv r'$, $s'' \equiv r''$. ∎

Let L_b be the following language:

$L_b\ \varphi ::= \surd \mid \neg\varphi \mid (\varphi \wedge \varphi) \mid \langle(\varphi?;\tau)^*\rangle\varphi \mid \langle\tau^*;\varphi?;a\rangle\varphi.$

Note that L_b properly extends L_{sb}: the L_b formula $\langle(\top?;\tau)^*\rangle\varphi$ has the same meaning as the L_{sb} formula $\langle\tau^*\rangle\varphi$, so we can assume the latter formula to belong to L_b as well.

Consider the TSs from Figure 16. An L_b formula that distinguishes between them is:

$$\varphi = \langle(\langle\tau^* c\rangle\top?;\tau)^*\rangle([\tau^* a]\bot \wedge \langle\tau^* b\rangle\top \wedge [\tau^* c]\bot).$$

Formula φ asserts that it is possible to do τ steps while a $\xrightarrow{c}$ transition is reachable by a τ^* path and arrive at a state with the property that a $\xrightarrow{b}$ transition is reachable by τ^* path, but $\xrightarrow{a}$ or $\xrightarrow{c}$ transitions are not. This is true in the root state of the left TS (move to the state marked 2), false in the root state of the right TS in Figure 16.

THEOREM 8.13
If $\mathcal{M}$, $\mathcal{N}$ are finitely branching in every atomic label and also in τ^* and $s \in S(\mathcal{M})$ is branching bisimilar to $r \in S(\mathcal{N})$, then for all $\varphi \in L_b$: ($s \models \varphi$ iff $r \models \varphi$).

PROOF. Induction on the complexity of φ. We just give the case of formulas of the form $\langle(\varphi?;\tau)^*\rangle\psi$. Assume C is a branching bisimulation relating s and r, and the induction hypothesis holds. Suppose $s \models \langle(\varphi?;\tau)^*\rangle\psi$. Then either $s \models \psi$ or there are $s_1, \ldots, s_n$ with $n \geq 1$, $s_0 \xrightarrow{\tau} s_1 \xrightarrow{\tau} \cdots \xrightarrow{\tau} s_n$, $s_0 = s$, $s_i \models \varphi$ for all $i < n$, and $s_n \models \psi$. We may assume without loss of generality that the s_i are all different. Applying Lemma 8.3 n times, (here we need the assumption that the TSs are finitely branching in τ^*), we find for each pair s_i, s_{i+1} a finite sequence $r_0 \xrightarrow{\tau} r_1 \xrightarrow{\tau} \cdots \xrightarrow{\tau} r_m$, with the r_j all different, and with for all r_j with $j < m$, $s_i C r_j$, and $s_{i+1} C r_m$. Applying the induction hypothesis m times, we find $r_j \models \varphi$ for all $j < m$. If $i \neq n$ we also have $r_m \models \varphi$, otherwise we have $r_m \models \psi$. It follows from this that there is a finite τ path from r with everywhere on the path except in the last state φ, and in the last state ψ. Therefore, $r \models \langle(\varphi?;\tau)^*\rangle\psi$. ∎

THEOREM 8.14
If $\mathcal{M}$, $\mathcal{N}$ are finitely branching in every atomic label and also in τ^*, then invariance for L_b formulas implies branching bisimulation.

PROOF. Call $s \equiv r$ if s and r satisfy the same L_b formulas. We show that $\equiv$ is a simple branching bisimulation. Assume $s \in S(\mathcal{M})$, $r \in S(\mathcal{N})$, $s \equiv r$.

Let $s \xrightarrow{\tau} s'$. In case $s' \equiv r$, there is nothing to prove, so assume that there is a $\chi \in L_b$ with $s' \models \chi$ and $r \models \neg\chi$. By $s \equiv r$, also $s \models \neg\chi$. Because $s \models \langle(\neg\chi;\tau)^*\rangle\chi$, and $s \equiv r$, the set

$$R = \{(r_0, \ldots r_n) \mid r_0 = r, r_0 \xrightarrow{\tau} r_1 \cdots \xrightarrow{\tau} r_n, r_i = r_j \Rightarrow i = j, i < n \Rightarrow r_i \models \neg\chi, r_n \models \chi\}$$

is non-empty. Because $\mathcal{N}$ is finitely branching in τ^*, $R = \{\sigma_1, \ldots, \sigma_m\}$, for some $m > 0$. Suppose no sequence σ_i has $\sigma_i(j) \equiv s$ ($j < last(\sigma_i)$) and $\sigma_i(last(\sigma_i)) \equiv s'$. Then there are $\varphi_1, \ldots, \varphi_m, \psi_1, \ldots \psi_m$ with $s \models \varphi_i$ and either $\sigma_i(j) \models \neg\varphi_i$ for some j with $0 \leq j < last(\sigma_i)$, or $\sigma_i(last(\sigma_i)) \models \neg\psi_i$. Let φ be the conjunction of the φ_i and ψ the conjunction of the ψ_i. Then $s \models \langle(\varphi;\tau)^*\rangle\psi$. and $r \models \neg\langle(\varphi;\tau)^*\rangle\psi$. and contradiction with $s \equiv r$. So there is a state r' with $r' \neq r$, $r \xrightarrow{\tau} r'$ and either $s \equiv r'$ or $s' \equiv r'$.

The reasoning for $s \xrightarrow{\tau*} s' \xrightarrow{a} s''$ is the same as for the case of simple branching bisimulation. ∎

De Nicola and Vaandrager [17] prove a similar result for a slightly different language. Our use of a fragment of a well known modal language (PDL) leads to further questions.

8.3 First-Orderization

The observation that the languages L_τ, L_{sb} and L_b are all fragments of PDL gives rise to a broadening of perspective. One would like to know if one can characterize these PDL fragments by means of invariance for the corresponding bisimulation notions? A problem here is the infinitary character of the operations τ^* and $(\varphi?;\tau)^*$.

The first-order translation instruction for multimodal languages from Section 3 breaks down for the language L_τ, and for all other fragments of PDL containing the * operator, in fact (cf. Section 5.3). The modalities $\langle\tau^*\rangle$ and $\langle\tau^* a\tau^*\rangle$ would give rise to infinite disjunctions of the form:

$$\begin{array}{rcl}(\langle\tau^* a\tau^*\rangle\psi)^\bullet & = & \exists y(R_a xy \wedge \psi^\bullet[y/x]) \\ & \vee & \exists y\exists z(R_\tau xy \wedge R_a yz \wedge \psi^\bullet[z/x]) \\ & \vee & \exists y\exists z(R_a xy \wedge R_\tau yz \wedge \psi^\bullet[z/x]) \\ & \vee & \exists y\exists z\exists w(R_\tau xy \wedge R_\tau yz \wedge R_a zw \wedge \psi^\bullet[w/x]) \\ & \vee & \ldots\end{array}$$

This shows that the language of first-order logic is not quite appropriate for a translation of modalities involving τ^*, and it is natural to move to the infinitary version $L_{\omega_1\omega}$.

Open problem 8.15
Give preservation results characterizing precisely the PDL fragments L_τ, L_{sb} and L_b in $L_{\omega_1\omega}$.

Here we will pursue a different question, by attempting to give a first order analysis after all. The problem with a first order analysis of PDL is that there is no first-order way to describe the reflexive transitive closure of a relation. Now, there are various standard options for re-analyzing this situation, so as to make it first-order again.

One strategy, which works as long as the number of different reflexive transitive closure relations remains manageable, is to represent each starred transition by a new two place relation symbol. In the case of L_τ and L_{sb} we can get by by just translating τ^* as R (for 'silent reachability') and making sure that R gets interpreted as a relation that *contains* the reflexive transitive closure of the $\xrightarrow{\tau}$ relation. There is a connection between this strategy and the notion of Δ saturation from [11] (see also [8]).

T $\forall x\forall y((x = y \rightarrow Rxy) \wedge (R_\tau xy \rightarrow Rxy) \wedge (\exists z(Rxz \wedge Rzy) \rightarrow Rxy))$.

In any first-order model where T holds and where R_τ is interpreted as the $\xrightarrow{\tau}$ relation, R will be interpreted as a relation containing the $\xrightarrow{\tau*}$ relation. Thus, for the translation of L_τ and L_{sb} it suffices to restrict attention to models where T holds and translate τ^* by means of R:

Formula translations:

$$\begin{array}{lcl}p^\bullet & = & Px \\ \top^\bullet & = & x = x \\ (\neg\psi)^\bullet & = & \neg\psi^\bullet \\ (\psi\wedge\chi)^\bullet & = & \psi^\bullet \wedge \chi^\bullet \\ (\langle\pi\rangle\psi)^\bullet & = & \exists y(\pi^\bullet xy \wedge \psi^\bullet[y/x])\end{array}$$

Program translations:

$$\begin{array}{lcl}(\tau^*)^\bullet & = & Rxy \\ (\tau^* a\tau^*)^\bullet & = & \exists z\exists w(Rxz \wedge R_a zw \wedge Rwy) \\ (\tau^*;\psi?;a)^\bullet & = & \exists z(Rxz \wedge \psi^\bullet[z/x] \wedge R_a zy).\end{array}$$

Now we can prove preservation results for τ bisimilarity and simple branching bisimilarity. Call φ and ψ T-equivalent if for every model $\mathcal{M}$ for the language and every assignment s: $\mathcal{M}, s \models T \wedge \varphi$ iff $\mathcal{M}, s \models T \wedge \psi$

THEOREM 8.16
A first-order formula φ with at most one free variables x is T-equivalent to a $^\bullet$-translation of a formula of the language L_τ iff φ is invariant for τ bisimulations.

THEOREM 8.17
A first-order formula φ with at most one free variable x is T-equivalent to a $^\bullet$ translation of a formula of the language L_{sb} iff φ is invariant for simple branching bisimulations.

The proofs of these theorems follow a by now familiar routine.

In the case of L_b we need a translation for $(\varphi?;\tau)^*$, so here a more general account of first-orderizing transitive closures is needed. One possible way to proceed is to add a new relation symbol R^*_φ for every first order formula $\varphi(x,y)$ with x,y as its only free variables. This gets complicated by the fact that the new relation symbols should also be allowed to occur in φ.

There is also another option, which is more specific to the analysis of silence and branching: an analysis in terms of two sorted first order logic, with states and branches (see also [44]). To work this out, one needs appropriate predicates:

$$
\begin{array}{lcl}
Tx & : & \textit{x is a } \tau \textit{ branch} \\
Fxy & : & \textit{x is the first state on branch y} \\
Lxy & : & \textit{x is the last state on branch y} \\
Oxy & : & \textit{state x occurs on branch y.}
\end{array}
$$

Use these predicates to translate as follows (note that the program translations have a free variable for a τ branch and a free variable for an end state (which in the case of $\tau^*;\psi?;a$ is reached by doing an $\xrightarrow{a}$ step *after* walking through the τ branch):

Formula translations:

$$
\begin{array}{lcl}
p^\natural & = & Px \\
\top^\natural & = & x = x \\
(\neg\psi)^\natural & = & \neg\psi^\natural \\
(\psi\wedge\chi)^\natural & = & \psi^\natural \wedge \chi^\natural \\
(\langle\pi\rangle\psi)^\natural & = & \exists b \exists y(\pi^\natural \wedge \psi^\natural[y/x])
\end{array}
$$

Program translations:

$$
\begin{array}{lcl}
(\tau^*;\psi?;a)^\natural & = & Tb \wedge \exists z(Lzb \wedge \psi^\natural[z/x] \wedge R_a zy) \\
(\langle(\psi?;\tau)^*\rangle)^\natural & = & Tb \wedge Lyb \wedge \forall z((Ozb \wedge z \neq y) \rightarrow \psi^\natural[z/x]).
\end{array}
$$

To ensure that Tb gets interpreted as 'b is a τ branch' we may take Tb as shorthand for the following formula:

Tb $\forall x(Oxb \leftrightarrow (\neg Lxb \rightarrow \exists y(Oyb \wedge R_\tau xy)) \wedge (\neg Fxb \rightarrow \exists y(Oyb \wedge R_\tau yx)))$.

Now a notion of two sorted branching bisimulation (for states and for τ branches) can be defined. and a preservation result can be proved for this:

DEFINITION 8.18
A relation C between $S(\mathcal{M})$ and $S(\mathcal{N})$ and also between the set of τ branches of $\mathcal{M}$ and the set of τ branches of $\mathcal{N}$ is a *two sorted branching bisimulation* if the following two clauses hold:

- The restriction of C to states is a branching bisimulation,
- if $\sigma C \theta$ (where σ, θ are τ branches) then
  1. $\sigma(1) C \theta(1)$,
  2. if $\sigma(i) C \theta(j)$ and $i \neq \mathit{last}(\sigma)$, then either
     - $\sigma(i+1) C \theta(j)$, or
     - $j \neq \mathit{last}(\theta)$ and either $\sigma(i+1) C \theta(j+1)$ or $\sigma(i) C \theta(j+1)$,
  3. clause (2) vice versa.

THEOREM 8.19
A first-order formula φ with at most two free variables x, y is equivalent to a $\sharp$-translation of a formula or program of the language L_b iff φ is invariant for two sorted branching bisimulations.

8.4 Comparison of Similarity Notions Involving Silent Steps

We have proved preservation results for three notions of similarity involving silent steps, two from the literature and a third one of our own. One might reasonably ask if there is anything to choose between the three notions.

A technical comparison would be useful, for instance: Can we say something illuminating about modal translations of the τ laws from process algebra? Or the other way around: does the notion of simple branching bisimulation give rise to process algebra τ laws of its own, and if so, how do these compare to the τ laws engendered by the competing similarity notions? Frits Vaandrager suggested the following τ law for simple branching bisimulation:

$$\tau(\tau x + y) = \tau(\tau x + y) + \tau x.$$

To investigate the question of the comparison further one would also have to look a bit closer at the philosophical intuitions behind τ abstraction, and see what we can derive from the intrinsic motivation for τ abstraction.

9 Concurrency and Bisimulation

9.1 Concurrent PDL

Although we have postponed the modal analysis of $||$, the concurrency notion from process algebra, for future work, we do have something to say about concurrency. We will briefly look here at an extension of PDL with an operator for concurrency of programs. Concurrent PDL [35] has the following syntax.

CPDL formulas $\varphi ::= p \mid \neg\varphi \mid (\varphi \wedge \varphi) \mid \langle\pi\rangle\varphi \mid [\pi]\varphi$.
CPDL programs $\pi ::= a \mid \pi^* \mid \pi;\pi \mid \pi \cup \pi \mid \pi \cap \pi \mid \varphi?$

The idea of the new operation $\pi_1 \cap \pi_2$ is: do π_1 and π_2 concurrently.

Because CPDL allows programs π to compute in parallel, several states may be reached in parallel by computing processes within π that act independently at the same time. This means that the reachability relation R_π of a program π connects a state s to the set of all states that are reached at the same time by the processes within π. We also allow for indeterminism, so π may give rise to several R_π successor sets in one given state.

For the modal clauses of CPDL, we follow Nerode and Wijesekera [34] and Goldblatt [25]:

- $\mathcal{M}, V, s \models \langle\pi\rangle\varphi$ if there is some $T \subseteq S(\mathcal{M})$ with $sR_\pi T$ and for all $t \in T$, $\mathcal{M}, V, t \models \varphi$.
- $\mathcal{M}, V, s \models [\pi]\varphi$ if for all $T \subseteq S(\mathcal{M})$, $sR_\pi T$ implies that for all $t \in T$, $\mathcal{M}, V, t \models \varphi$.

Note that under these stipulations, $\langle\pi\rangle$ and $[\pi]$ are no longer duals.

The definitions of the reachability relations R_π are slightly complicated by the type shift in the second arguments of the relations. The crucial definitions are the following two set operations.

$$R \otimes S = \lambda x \lambda Y \cdot \exists Y_1, Y_2 : xRY_1 \wedge xSY_2 \wedge Y = Y_1 \cup Y_2.$$

$$R \odot S = \lambda x \lambda Y \cdot \exists Z : xRZ \wedge \exists f : Z \to \text{POW}\,(S(\mathcal{M}))$$
$$\textit{with}\ \forall z : zSf(z) \ \wedge\ Y = \bigcup\{f(z) \mid z \in Z\}.$$

The first of these is used for $\pi_1 \cap \pi_2$, the second for $\pi_1;\pi_2$. The operation for iteration is in need of a slight modification. It is defined in terms of $\odot$ and $\cup$ as follows:

$$\begin{array}{lcl} Id & = & \{(s, \{s\}) \mid s \in S(\mathcal{M})\} \\ R^{(0)} & = & Id \\ R^{(n+1)} & = & Id \cup (R \odot R^{(n)}) \\ R^{(*)} & = & \bigcup\{R^{(n)} \mid n \in \omega\}. \end{array}$$

The full interpretation of program relations in a model $\mathcal{M}$ now runs as follows:

- The accessibility relations for the atoms are given.
- $R_{\pi_1;\pi_2} = R_{\pi_1} \odot R_{\pi_2}$.
- $R_{\pi_1 \cup \pi_2} = R_{\pi_1} \cup R_{\pi_2}$.
- $R_{\pi_1 \cap \pi_2} = R_{\pi_1} \otimes R_{\pi_2}$.
- $R_{\pi^*} = R_\pi^{(*)}$.
- $R_{\varphi?} = \{\langle s, \{s\}\rangle \mid \mathcal{M}, s \models \varphi\}$.

Peleg [35] gives computational motivation, Goldblatt [25] has a completeness result.

9.2 CPDL Bisimulation

Now, we introduce the appropriate generalization of our central notion of process equivalence.

Definition 9.1
A relation $C \subseteq S(\mathcal{M}) \times S(\mathcal{N})$ is a *CPDL bisimulation* (see Figure 17) if the following three clauses hold:

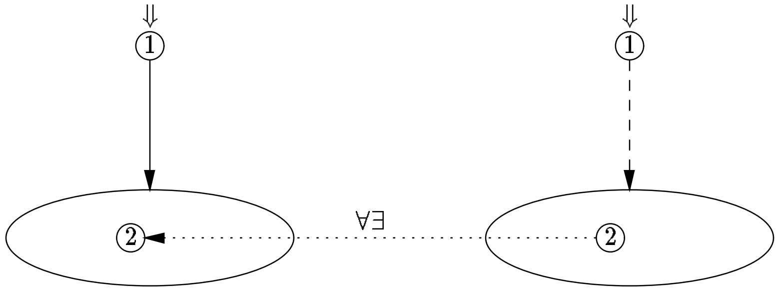

FIG. 17. CPDL bisimilarity.

1. If sCr then $V(s) = V(r)$,
2. if sCr and sRZ then there is a $U \subseteq S(\mathcal{N})$ with rRU and for all $u \in U$ there is a $z \in Z$ with zCu,
3. clause (2) vice versa.

This definition was in fact motivated by the following result:

LEMMA 9.2

1. CPDL formulas are invariant for CPDL bisimulations.
2. The operations $\varphi?$ (test), $\cup$, $\otimes$, $\odot$ and $^{(*)}$ are safe for CPDL bisimulation.

PROOF. 1. Induction on the complexity of φ. The case of atomic propositions and Booleans is trivial. For φ of the form $\langle\pi\rangle\psi$ and $[\pi]\psi$, use part (2) of this lemma.

2. We treat the relational operations one by one.

Test: It follows from part (1) of this lemma that test is safe for CPDL bisimulation.

Choice: Assume $s(R \cup S)T$ and sCr. We have to show that there is a state set $U \subseteq S(\mathcal{N})$ with $r(R \cup S)U$ and for all $u \in U$ there is a $t \in T$ with tCu. From $s(R \cup S)T$, either sRT or sST. By the fact that C is a CPDL bisimulation for R and S, there is a state U with either rRU or rSU, and for all $u \in U$ there is a $t \in T$ with tCu. It follows that $r(R \cup S)U$ and for all $u \in U$ there is a $t \in T$ with tCu.

Concurrency: Assume $s(R \otimes S)T$. We have to show that there is a state set $U \subseteq S(\mathcal{N})$ with $r(R \otimes S)U$ and for all $u \in U$ there is a $t \in T$ with tCu. By the definition of $\otimes$, there exist T_1, T_2 with $T = T_1 \cup T_2$ and sRT_1 and sST_2. By the fact that C is a bisimulation for R, there is a $U_1 \subseteq S(\mathcal{N})$ with rRU_1 and for all $u_1 \in U_1$ there is a $t_1 \in T_1$ with t_1Cu_1. By the fact that C is a bisimulation for S, there is a $U_2 \subseteq S(\mathcal{N})$ with rSU_2 and for all $u_2 \in U_2$ there is a $t_2 \in T_2$ with t_2Cu_2. So setting $U = U_1 \cup U_2$ yields the required property.

Composition: Assume $s(R \odot S)T$. We have to show that there is a state set $U \subseteq S(\mathcal{N})$ with $r(R \odot S)U$ and for all $u \in U$ there is a $t \in T$ with tCu.

By the definition of $\odot$, there exists Z, f with sRZ, $\forall z \in Z, zSf(z)$ and $T = \bigcup\{f(z) \mid z \in Z\}$.

By the fact that C is a bisimulation for R, there is a $V \subseteq S(\mathcal{N})$ with rRV and there is a $g : V \to Z$ with $g(v)Cv$. Now, from the fact that C is also a CPDL bisimulation for S, combined with $g(v)Cv$, we have that $g(v)Sf(g(v))$ implies that there is an X_v with vSX_v and $\forall x \in X_v \exists y \in f(g(v)) : yCx$. In other words, there is a function $h : V \to \text{POW}\,(S(\mathcal{N}))$ with for all $v \in V$: $vSh(v)$ and $\forall x \in h(v) \exists y \in f(g(v)) : yCx$. It follows that $U = \bigcup\{h(v) \mid v \in V\}$ has the required properties.

Iteration: From the result for $\odot$ we get that CPDL bisimilarity for R implies CPDL bisimilarity for $R^{(n)}$ (any $n \in \omega$). Because CPDL bisimilarity is preserved under taking unions, CPDL bisimilarity for $R^{(*)}$ follows. ∎

Open problem 9.3
Which first-order definable operations on relations are safe for CPDL bisimulation?

9.3 Embeddings in Standard Logic

There are two sources of non-first-orderness in CPDL:

- the infinary construction *, which calls for an analysis in $L_{\omega_1\omega}$,
- the quantification over sets in xRY, which calls for a second order analysis.

This leads to the following 'obvious' translation in infinitary second order logic:

Formula translations:

$$
\begin{array}{lcl}
p^\circ & = & Px \\
(\neg\varphi)^\circ & = & \neg\varphi^\circ \\
(\varphi \wedge \psi)^\circ & = & \varphi^\circ \wedge \psi^\circ \\
(\langle\pi\rangle\varphi)^\circ & = & \exists Y(\pi^\circ \wedge \forall y(Yy \to \varphi^\circ[y/x])) \\
([\pi]\varphi)^\circ & = & \forall Y(\pi^\circ \to \forall y(Yy \to \varphi^\circ[y/x]))
\end{array}
$$

Program translations:

$$
\begin{array}{lcl}
a^\circ & = & R_a xY \\
(\pi_1;\pi_2)^\circ & = & \exists Z \exists F(\pi_1^\circ[Z/Y] \wedge \forall x \forall U \forall V((FxU \wedge FxV) \to U = V) \\
 & & \wedge \forall y(Yy \leftrightarrow \exists z \exists U(Zz \wedge FzU \wedge Uy \wedge (\pi_2^\circ[z/x, U/Y])))) \\
(\pi_1 \cap \pi_2)^\circ & = & \exists Z_1 \exists Z_2(\pi_1^\circ[Z_1/Y] \wedge \pi_2^\circ[Z_2/Y] \wedge \\
 & & \forall y(Yy \leftrightarrow (Z_1 y \vee Z_2 y))) \\
(\pi_1 \cup \pi_2)^\circ & = & \pi_1^\circ \vee \pi_2^\circ \\
(\varphi?)^\circ & = & \varphi^\circ \wedge \forall y(Yy \leftrightarrow y = x) \\
(\pi^0)^\circ & = & \forall y(Yy \leftrightarrow y = x) \\
(\pi^{n+1})^\circ & = & \forall y(Yy \leftrightarrow y = x) \vee (\pi;\pi^n)^\circ \\
(\pi^*)^\circ & = & \bigvee_{n=0,1,2,\ldots}(\pi^n)^\circ.
\end{array}
$$

Open problem 9.4
Give a preservation result characterizing precisely the CPDL formulas in second order infinitary logic.

There are several ways to analyse the situation in first-order terms again. For $*$, we can either introduce predicates for reflexive transitive closures of relations, or introduce a separate sort of branches, plus vocabulary to talk about those and their relations to states (see Section 8.3). For talking about sets of states, we can introduce a sort for those, too, and a predicate Mxy, for 'state x is a member of state set y'. We also need Fx to express that x is a function, and $Afxy$ to express that function f applied to argument x yields value y.

Formula translations:

$$
\begin{array}{lcl}
p^\bullet & = & Px \\
(\neg\varphi)^\bullet & = & \neg\varphi^\bullet \\
(\varphi \wedge \psi)^\bullet & = & \varphi^\bullet \wedge \psi^\bullet \\
(\langle\pi\rangle\varphi)^\bullet & = & \exists y(\pi^\bullet \wedge \forall z(Mzy \rightarrow \varphi^\bullet[z/x])) \\
([\pi]\varphi)^\bullet & = & \forall y(\pi^\bullet \rightarrow \forall z(Mzy \rightarrow \varphi^\bullet[z/x]))
\end{array}
$$

Program translations:

$$
\begin{array}{lcl}
a^\bullet & = & R_a xy \\
(\pi_1;\pi_2)^\bullet & = & \exists z \exists f(\pi_1^\bullet[z/y] \wedge Ff \wedge \forall u \forall v(Afuv \leftrightarrow Muz) \wedge \\
 & & \forall w(Mwy \leftrightarrow \\
 & & \exists u \exists v(Muz \wedge Mwv \wedge Afuv \wedge \pi_2^\bullet[u/x, v/y]))) \\
(\pi_1 \cap \pi_2)^\bullet & = & \exists z_1 z_2(\pi_1^\bullet[z_1/y] \wedge \pi_2^\bullet[z_2/y] \wedge \\
 & & \forall w(Mwy \leftrightarrow (Mwz_1 \vee Mwz_2))) \\
(\pi_1 \cup \pi_2)^\bullet & = & \pi_1^\bullet \vee \pi_2^\bullet \\
(\varphi?)^\bullet & = & \varphi^\bullet \wedge \forall z(Mzy \leftrightarrow z = x) \\
(\pi^*)^\bullet & = & \textit{several options (see Section 8.3).}
\end{array}
$$

We assume that the one-place predicate letters P are interpreted as properties of individual states and the two-place predicate letters R_a as relations between individual states and state sets.

OPEN PROBLEM 9.5
Give a preservation result characterizing precisely the CPDL formulas in first order logic.

10 Conclusion and Further Topics of Investigation

10.1 What We Hope to Have Established

To summarize our claims, we hope to have established parallels between modal logic, in a suitably broad sense (including temporal logic and propositional dynamic logic) and process theories over TSs. The basics of these connections are well known from the literature [29, 33], but we have given more detailed analogies in techniques, leading to a next range of questions, some of them answered in the text, other listed in passing as open problems.

We will now wind up our story by giving a list of further topics of investigation, most of them hinted at in the paper and put aside with a $\surd$ mark for 'merits further attention'. These topics of further study are in three main areas: general logic, process algebra and modal logic.

10.2 General Logical Issues

Higher-Order and Infinitary Logic Versus First-Order Logic In the above we have confined attention to first-order formalisms over transition systems. For a glimpse of what happens when one considers higher order formalisms over transition systems see [19, 20]. Nothing seems to be known at present concerning preservation properties of modal fragments of higher-order infinitary logics.

10.3 Process Algebra

The Modal Analysis of Sophisticated Process Operations At the end of Section 7 we have made a first attempt at a modal analysis of the simple process operations such as sum and product. As a second step, one would like to extend this analysis to the operations of free merge, I abstraction and I pruning. Finally, a modal account is needed of recursive definitions of TSs from a class of given TSs.

Special Classes of TSs One might also wish to consider axiomatisations of special classes of TSs (acyclic, finite, finitely branching) in modal logic. But modal logic is blind for these distinctions: the minimal modal logic is complete for all of these. This observation leads directly to the next topic.

Infinitary Processes We have seen in Section 5 that the expressive power of PDL is not sufficient to fully describe the infinitary trace behaviour of processes. We mentioned extending PDL with an operator for loop detection as a possible remedy. This strenghtening of the language brings a stronger similarity notion in its wake which merits further attention. There are also other means of strengthening the expressive power of PDL: see [28] for some comparisons. How do the similarity notions they engender compare? Which preservation properties hold?

10.4 Modal Logic

Correspondence and Completeness Theory Although we have said something about preservation results for modal fragments, modal logic has quite a bit more by way of technique. Correspondence theory in the sense of [3] was not really used. Also, nothing was used from modal proof theory and completeness theory. Finally, the issue of a complexity analysis of processes in terms of complexity of the modal languages that describe them is still wide open.

Extended Modal and Dynamic Logic In Section 4.3 we have used an extended modal formalism with two-dimensional temporal procedures. A much more powerful system in the same spirit is presented and analysed in [40]. It seems worth one's while to investigate the use of such systems in the analysis of processes.

Maarten de Rijke (personal communication) suggests the use of an extended modal formalism that matches the Process Algebra term formation operators a bit more closely. Assume a three-place relation $Cxyz$ for 'x has a choice between y and z', and introduce a two-place modal operator $\triangle$ with the following semantic clause.

- $\mathcal{M}, s \models \varphi \triangle \psi$ iff $\exists tu(Cstu\ \&\ \mathcal{M}, t \models \varphi\ \&\ \mathcal{M}, u \models \psi)$.

Suitable restrictions have to be imposed on the interpretation of C.

Newer Conceptual Developments One of the newer developments in modal logic and propositional dynamic logic is the advance of arrow logic, where transition arrows themselves become first class citizens, and states may be (but need not be) demoted to a more marginal status [6].

Arrow logic is a kind of dynamic logic based on a weaker set of underlying assumptions than is usual. In propositional dynamic logic or relational algebra, every program relation brings a converse relation in its wake, and every pair of relations a composition. The idea is to get rid of these facts by dropping the analysis of transitions as composed of pairs of states. Taking the transitions themselves as primitive and calling them arrows has the advantage that we can stipulate what relations hold between them: composition and reversal themselves become two place relations on the set of arrows. Because of the primacy of transitions, arrow logic looks very promising as a tool for the study of TSs.

Acknowledgements

The research of Vera Stebletsova was sponsored by project NF 102/62-356 ('Structural and Semantic Parallels in Natural Languages and Programming Languages'), funded by the Netherlands Organization for the Advancement of Research (N.W.O.).

Thanks are due to Jan Bergstra, Patrick Blackburn, Claudia Brovedani, Tim Fernando, Rob van Glabbeek, Wilfried Meyer Viol, Maarten de Rijke, Frits Vaandrager, Albert Visser and an anonymous referee of this Journal for criticisms, hints and helpful discussions.

References

[1] J.C.M. Baeten, J.A. Bergstra, and J.W. Klop. Ready-trace semantics for concrete process algebra with the priority operator. *The Computer Journal*, 30(6):498–506, 1987.

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

[3] J. van Benthem. Correspondence theory. In D. Gabbay and F. Guenthner, editors, *Handbook of Philosophical Logic, Volume II*, pages 167–247. Reidel, Dordrecht, 1984.

[4] J. van Benthem. *Modal Logic and Classical Logic.* Bibliopolis, Naples, 1985.

[5] J. van Benthem. *Language in Action: categories, lambdas and dynamic logic.* Studies in Logic 130. Elsevier, Amsterdam, 1991.

[6] J. van Benthem. A note on dynamic arrow logic. In: Van Eijck and Visser (eds.), Logic and Information Flow, MIT Press, Cambridge Mass., 1994.

[7] J. van Benthem. Which program constructions are safe for bisimulation? Manuscript, University of Amsterdam, 1993.

[8] J. van Benthem and J. Bergstra. Logic of transition systems. Technical Report CT-93-03, ILLC, 1993.

[9] J. Bergstra and J.W. Klop. Algebra of communicating processes. In J.W. de Bakker, M. Hazewinkel, and J.K. Lenstra, editors, *Mathematics and Computer Science.* North-Holland, Amsterdam, 1986. Proceedings CWI Symposium November 1983.

[10] J. Bergstra and J.W. Klop. Process algebra: specification and verification in bisimulation semantics. In J.K. Lenstra M. Hazewinkel and L.G.L.T. Meertens, editors, *Mathematics and Computer Science II*. North-Holland, Amsterdam, 1986.

[11] J. Bergstra and J.W. Klop. A complete inference system for regular processes with silent moves. In F. Drake and J. Tuss, editors, *Logic Colloquium '86*, pages 21–81. North-Holland, Amsterdam, 1988.

[12] R. Bull and K. Segerberg. Basic modal logic. In D. Gabbay and F. Guenthner, editors, *Handbook of Philosophical Logic, volume II*, pages 1–88. Reidel, Dordrecht, 1984.

[13] C.C. Chang and H.J. Keisler. *Model Theory*. North-Holland, Amsterdam, 1973.

[14] F. Cherief. *Contributions à la sémantique du Parallélisme: Bisimulations pour le Raffinement et le Vrai Parallélisme*. PhD thesis, Institut National Polytechnique de Grenoble, 1992.

[15] E.M. Clarke and E.A. Emerson. Design and synthesis of synchronisation skeletons using branching time temporal logic. In D. Kozen, editor, *Logics of Programs*, Lecture Notes in Computer Science 131, pages 52–71. Springer, Berlin, 1981.

[16] B. Courcelle. Graph rewriting: An algebraic and logic approach. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science, Volume B*, pages 193–242. Elsevier, Amsterdam, 1990.

[17] R. De Nicola and F. Vaandrager. Three logics of branching bisimulation. In *Proceedings 5th LICS Conference*, pages 118–129. Computer Society Press, 1990. Full version available as Report SI-92/07 of the Department of Computer Science, University of Rome.

[18] E.A. Emerson. Temporal and modal logic. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science, Volume B*, pages 995–1072. Elsevier, Amsterdam, 1990.

[19] T. Fernando. Bisimulations and predicate logic. Technical Report CS-R9333, CWI, Amsterdam, March 1993. Journal of Symbolic Logic, in press.

[20] T. Fernando. Comparative transition system semantics. In *Computer Science Logic: Selected Papers from CSL '92*. Springer-Verlag, Berlin, in press.

[21] K. Fine. Logics containing K4. *Journal of Symbolic Logic*, 39:31–42, 1974.

[22] K. Fine. Some connections between elementary and modal logic. In S. Kanger, editor, *Proceedings of the third Scandinavian Logic Symposium, Uppsala 1973*, pages 15–31. North-Holland, Amsterdam, 1975.

[23] R. van Glabbeek. *Comparative Concurrency Semantics and Refinement of Actions*. PhD thesis, Free University, Amsterdam, 1990.

[24] R.J. van Glabbeek and W.P. Weijland. Branching time and abstraction in bisimulation semantics (extended abstract). In G.X. Ritter, editor, *Information Processing 89*, pages 613–618. North-Holland, Amsterdam, 1989. Full version available as Report CS-R9120, CWI, Amsterdam, 1991.

[25] R. Goldblatt. Concurrent dynamic logic with independent modalities. *Studia Logica*, 51:551–578, 1992.

[26] R. Goldblatt. *Logics of Time and Computation, Second Edition, Revised and Expanded*, volume 7 of *CSLI Lecture Notes*. CSLI, Stanford, 1992 (first edition 1987). Distributed by University of Chicago Press.

[27] J.F. Groote and F. Vaandrager. Structured operational semantics and bisimulation as a congruence. *Information and Computation*, 100(2):202–260, 1992.

[28] D. Harel. Dynamic logic. In D. Gabbay and F. Guenthner, editors, *Handbook of Philosophical Logic*, pages 497–604. Reidel, Dordrecht, 1984. Volume II.

[29] M. Hennessey and R. Milner. Algebraic laws for nondeterminism and concurrency. *Journal of the ACM*, 32, pages 137–161, 1985.

[30] C.A.R. Hoare. *Communicating Sequential Processes*. Prentice Hall, New York, 1985.

[31] H. Kamp. *Tense Logic and the Theory of Order*. PhD thesis, UCLA, 1968.

[32] R. Milner. A modal characterisation of observable machine-behaviour. In G. Astesiano and C. Böhm, editors, *Proceedings CAAP '81*, pages 25–34. Springer, Berlin, 1981.

[33] R. Milner. *Communication and Concurrency*. Prentice Hall, New York, 1989.

[34] A. Nerode and D. Wijesekera. Constructive concurrent dynamic logic i. Technical Report 90-43, Mathematical Science Institute, Cornell University, 1990.

[35] D. Peleg. Concurrent dynamic logic. *Journal of the ACM*, 34:450–479, 1987.

[36] A. Pnueli. The temporal logic of programs. In *Proceedings 18th Annual IEEE Symposium on Foundations of Computer Science*, pages 46–57, 1977.

[37] A. Pnueli. Linear and branching structures in the semantics and logics of reactive systems. In W. Brauer, editor, *Proceedings ICALP 85*, Lecture Notes in Computer Science 194, pages 15–32. Springer, Berlin, 1985.

[38] V. Pratt. Semantical considerations on Floyd–Hoare logic. *Proceedings 17th IEEE Symposium on Foundations of Computer Science*, pages 109–121, 1976.

[39] M. de Rijke. The modal logic of inequality. *Journal of Symbolic Logic*, 57(2):566–584, 1992.

[40] M. de Rijke. A system of dynamic modal logic. Technical Report LP-92-08, ILLC, University of Amsterdam, 1992.

[41] P.H. Rodenburg. *Intuitionistic Correspondence Theory*. PhD thesis, University of Amsterdam, 1986.

[42] H. Sahlqvist. Completeness and correspondence in first and second order semantics for modal logic. In S. Kanger, editor, *Proceedings of the Third Scandinavian Logic Symposium*, pages 110–143. North-Holland, Amsterdam, 1975.

[43] E. Spaan. *Complexity of Modal Logics*. PhD thesis, University of Amsterdam, 1993.

[44] V. Stebletsova. First-orderizing the logic of branching bisimulation. Manuscript, CWI, Amsterdam, 1993.

[45] C. Stirling. Modal and temporal logics. In S. Abramsky, D.M. Gabbay, and T.S.E. Maibaum, editors, *Handbook of Logic in Computer Science, Volume 2*. Clarendon Press, Oxford, 1992.

[46] Y. Venema. *Many-dimensional modal logic*. PhD thesis, University of Amsterdam, 1992.

Received 7 June 1993