Journal of Logic, Language, and Information **3:** 247–283, 1995.

Logic of Transition Systems

JOHAN VAN BENTHEM and JAN BERGSTRA
Institute for Logic, Language and Computation, University of Amsterdam, Amsterdam, The Netherlands

(Received 19 August 1994)

Abstract. Labeled transition systems are key structures for modeling computation. In this paper, we show how they lend themselves to ordinary logical analysis (without any special new formalisms), by introducing their standard first-order theory. This perspective enables us to raise several basic model-theoretic questions of definability, axiomatization and preservation for various notions of process equivalence found in the computational literature, and answer them using well-known logical techniques (including the Compactness theorem, Saturation and Ehrenfeucht games). Moreover, we consider what happens to this general theory when one restricts attention to special classes of transition systems (in particular, finite ones), as well as extended logical languages (in particular, infinitary first-order logic). We hope that this puts standard logical formalisms on the map as a serious option for a theory of computational processes. As a side benefit, our approach increases comparability with several other existing formalisms over labeled transition systems (such as Process Algebra or Modal Logic). We provide some pointers to this effect, too.

Key words: Labeled transition system, bisimulation, modal logic, process algebra, invariance, definability, first-order model theory.

1. Transition Systems

1.1. Basic Structures

DEFINITION. A *labeled transition system* (LTS) is a triple $(S, A, \rightarrow)$, where S is a set of 'states', A of atomic 'actions', and $\rightarrow$ a ternary relation over $S \times A \times A$ providing 'labeled transitions' $\rightarrow^a$ between states for each atomic action a.

Remark. A mathematically equivalent definition would use structures of the form $(S, \{R_a \mid a \in A\})$, where the R_a are binary transition relations between states. These are, of course, the well-known Kripke models for polymodal logic.

In the literature, one finds several further elements to LTSs. For instance, there may be special arrows $\rightarrow^a \surd$ indicating a succesfully completed a-transition, or special markers for 'termination' or 'dead-lock'. This structure may be added as follows:

$\surd$ is a unary predicate holding at 'success' or 'termination' states,
∂ is a unary predicate holding at 'failure' or 'dead-lock' states.

If desired, these may be provided with suitable meaning postulates, such as

$$\forall x \ (\surd x \vee \delta x \rightarrow \neg \exists y R_a x y) \text{ (for all } a \in A)$$

$$\forall x \ (\bigwedge_{a \in A} \neg \exists y R_a xy \to \surd x \vee \delta x)$$
$$\forall x \ (\surd x \to \neg \delta x).$$

LTSs decorated with such unary predicates will be called 'labeled transition systems' too, whenever appropriate (in practice, this will not cause any confusion).

1.2. Notions of Process Equivalence

Two different LTSs may be instances of the same 'process'. The appropriate notion of 'process equivalence' here may depend on one's computational aims, and therefore, we consider a hierarchy from coarser to finer grain levels of comparison. There are two notable choice points in setting up these definitions. We shall mainly compare different LTSs, but the literature often compares states inside one 'super LTS'. These are equivalent views, as one can always merge two different LTSs into their disjoint union (for some purposes below, the latter is even the more convenient perspective). But a genuine choice is the following. One can view process equivalence relations either as 'respecting' certain observational properties in two LTSs under comparison (this is the "if" direction in the definitions to follow) or as completely 'determined' by the latter properties (this is the "iff" version). Both alternatives occur in the literature.

DEFINITION. Let $M = (S, A, \to)$ and $M' = (S', A', \to')$ be two LTSs. A binary relation $\equiv$ over $S \times S'$ is a *finite trace equivalence* if, whenever $s \equiv s'$, then s, s' start the same successful finite traces, i.e., finite sequences of labeled transitions ending in a success state. (Thus, we presuppose the enriched notion of LTS here.) If the relation $\equiv$ consists of precisely the latter pairs, then it is a *complete finite trace equivalence.*

This is about the coarsest notion of 'behavioural equivalence' for processes. We now consider some progressively more demanding comparisons.

DEFINITION. A relation $\equiv$ as above is a *full trace equivalence* if any two states with $s \equiv s'$ start the same countable traces. The latter include all terminating finite traces (successful or not) and all countably infinite ones $s = s_1 \to^{a1} s_2 \to^{a2} s_3 \to^{a3} \ldots$ *Complete full trace equivalence* is the case where the latter property even defines the relation $\equiv$.

The next notion is basic in both the computational and the modal literature. Essentially, it adds respect for the 'choice structure' of processes:

DEFINITION. A binary relation $\equiv$ between two LTSs is a *bisimulation* if the following back-and-forth conditions hold:
(i) if $s \equiv s'$ and $s \to^a t$, then there exists t' with $s' \to^a t'$, $s' \equiv t'$

(ii) and vice versa.

Moreover, $\equiv$ should only connect states agreeing in their atomic markings (usually just $\surd$ and ∂). The *maximal bisimulation* between two LTSs is the union of all bisimulations between them (which is necessarily a bisimulation itself).

Finally, we consider one even stronger notion of process equivalence.

DEFINITION. A relation $\equiv$ is a *generated graph isomorphism* if, whenever $s \equiv s'$, then the 'generated subLTSs' M_s, $M'_{s'}$ are isomorphic. (The latter are the submodels containing s, s' and all states that can be reached from these via some finite sequence of available atomic transitions.) As a defining characteristic again, the latter relation gives *complete generated graph isomorphism.*

Here are some simple connections between these various notions, stated with some obvious abbreviations (the relevant proofs and counter-examples are straightforward):

PROPOSITION.
(i) *GGI implies BISIM implies FullTE implies FinTE*
(ii) *None of these implications can be reversed.*

Evidently, one can vary the above notions of process equivalence in several ways. One major purpose of our logical analysis below is to provide a general model-theoretic perspective upon all of them.

2. Logical Description Languages

2.1. Standard Formalisms

Labeled transition systems are standard graph-like mathematical structures, and as such, they may be described using standard logical formalisms. Here, we shall work mainly with the following two languages:

First-Order Predicate Logic over LTSs ($L_{\omega\omega}$) (cf. Chang and Keisler 1973)
The basic vocabulary consists of binary predicates $\{R_a | a \in A\}$ and unary predicates P (including $\surd$, ∂), while the logical operators are the usual Boolean connectives and first-order quantifiers with variables ranging over states in LTSs. Henceforth, we shall use both notations $R_a xy$ and $x \rightarrow^a y$ for relational atoms, where the choice depends on what fits best with the relevant (logical or computational) literature.

Example. Some Basic Computational Notions.
'Action a is deterministic' $\forall xy((R_a xy \wedge R_a xz \rightarrow y = z))$
'Action a is confluent' $\forall xyz((R_a xy \wedge R_a xz) \rightarrow \exists u(R_a yu \wedge R_a zu))$

'Action a enables b' $\forall xy(R_a xy \rightarrow \exists z R_b yz)$
'Tree-like LTS' $\forall xyzu((R_a xy \wedge R_b zu) \rightarrow \neg y = u)$ for all distinct actions a, b.

For many computational purposes, however, the natural formalization involves some form of 'finiteness' beyond the expressive power of the first-order language, and hence, the following formalism is sometimes relevant too:

Infinitary Predicate Logic over LTSs ($L_{\omega 1 \omega}$) (cf. Keisler 1971)
Its vocabulary is the same as above, while the logical operations now also include *countably infinite* conjunctions and disjunctions of formulas in at most some fixed finite number of free variables.

Example. Expressing Some Higher-Order Computational Notions.
(One has to assume here that there are only countably many atomic actions.)
'Acyclic LTS' $\bigwedge_{n \geq 1, a_1, \ldots, a_{n-1} \in A} \neg(\exists x_1 \ldots x_n(\bigwedge_{1 \leq i \leq n} x_i \rightarrow^{a_i} x_{i+1} \wedge x_1 = x_n))$
'Action a terminates' $\forall x(\surd(x) \vee \bigvee_{n \geq 1} \exists x_1 \ldots x_n(\bigwedge_{1 \leq i < n} x_i \rightarrow^a x_{i+1} \wedge \surd(x_n)))$.

This formalism has been advocated in Salwicki 1970, Goldblatt 1982, Harel 1984. Some natural computational notions are still beyond it, witness Section 2.3 below.

Remark. First-Orderization.
In Logic, things are not always what they seem. Higher-order formulations may often be 'first-orderized' after all by bringing in suitable additional vocabulary. For instance, we can also design a *two-sorted* first-order version of Termination by taking 'paths' or 'branches' to be a new sort of object (in addition to states), with some plausible predicates about them, and then stating that every state lies on a succesful a-path. Such a technical move may even have intrinsic interest, leading to an explicit theory of computation paths. Other cases of this first-order strategy, on LTSs with silent steps (cf. Section 9 below) occur in Van Benthem *et al.* 1993.

2.2. Other Formalisms over Transition Systems

2.2.1. *Modal and Dynamic Logic*

At least two other well-developed traditions are around in the study of LTSs. First, there are Modal and Dynamic Logics over such models (Harel 1984, Stirling 1989), with the usual box and diamond notations. For our present purposes, these may be viewed as *fragments* of the above first-order and higher-order languages via the so-called 'standard translation' (cf. van Benthem 1985), which works as illustrated here:

Example. Sample Modal Translations.
(i) $[a]\langle b \cup c\rangle p$ translates into $\forall y(R_a xy \rightarrow \exists z((R_b yz \vee R_c yz) \wedge Pz))$
(ii) $\langle a^* \rangle [b]q$ translates into $\exists y(\bigvee_{n\in N} R_a^n xy \wedge \forall z(R_b yz \rightarrow Qz)))$.

From now on, we will work with the following notion.

DEFINITION. The *modal fragment* of the above first-order logic consists of all those formulas that can be formed using unary atoms, Boolean operations, and restricted existential quantifiers of the form $\exists z(R_a yz \wedge$ (with y, z distinct variables).

Usually, we shall be concerned with unary formulas $\phi(x)$ describing properties of states – although Dynamic Logic adds binary formulas $\pi(x, y)$ for program expressions denoting binary transition relations between states (cf. Section 6 below).

2.2.2. *Process Algebra*

Next, there is the tradition of Process Algebra, which considers LTSs consisting of processes with additional structure (Milner 1980, Bergstra and Klop 1984). For instance, one typical axiom system to be used in Section 10 below looks as follows (this is a fragment of the system CCS in Milner 1980). There is one binary operation of addition or 'choice' (+) plus a family of unary operations a; ('prefixing action a'), satisfying the following algebraic identities:

Example. Elementary Process Algebra (EPA).

(1) $x + x = x$	idempotence
(2) $x + y = y + x$	commutativity
(3) $x + (y + z) = (x + y) + z$	associativity
(4) $0 + x = x$	zero element

There are actually several ways of relating the two approaches. In Section 10 below, we shall bring in ideas from Process Algebra by endowing LTSs with additional operations on states. But one can also think of Process Algebra as a calculus for combining whole LTSs into new ones, via operations of 'choice', 'product' or 'merge' mirroring the algebraic operations. On the latter view, Modal Logic is an 'internal' description language for LTSs, which is in harmony with the latter 'external' calculus. (Van Benthem *et al.* 1993 presents a more detailed comparison).

2.2.3. *Standard Logic of Both Actions and States*

Finally, from the first-order point of view on LTSs, one could describe operational structure also by enriching the vocabulary type as follows. Introduce a second sort of objects called *actions* in addition to states, with corresponding quantifiers

and variables. (Also, specific atomic actions might have constant names in the language.) Labeled transition systems are still models for this language, with the domains S and A now treated on a par. The earlier arrow $\rightarrow$ then becomes a ternary predicate letter relating actions and states, and one can directly express various new computational properties, both first-order and infinitary, such as:

- 'Endpoints exist' $\exists x \neg \exists a x \rightarrow^a y$
- 'Finite branching' $\forall x \bigvee_{n \geq 1} \exists a_1 \ldots a_n \forall a y (x \rightarrow^a y \rightarrow \bigvee_{1 \leq i \leq n} a = a_i)$
- 'Every action has a converse' $\forall a \exists b \forall x y (x \rightarrow^a y \leftrightarrow y \rightarrow^b x)$
- 'Actions are extensional' $\forall a b (\forall x y (x \rightarrow^a y \leftrightarrow x \rightarrow^b y) \rightarrow a = b)$.

This richer perspective (not pursued here) is a natural next stage of formalization, and it suggests several logical issues concerning the interplay of state and action domains for LTSs that seem to have escaped the literature so far. Here are two illustrations.

(1) Each assertion in the earlier state languages refers to at most some countable set of specific atomic actions, and hence it is invariant for arbitrary changes in the action domain of LTSs leaving all transitions for these 'relevant actions' undisturbed. This invariance is easily turned into a semantic characterization of what may be called the 'pure state fragment' of the full two-sorted state-action formalism, which only allows quantification over states, not over actions.

(2) A natural related question is which statements are preserved when actions are *added* to an LTS (while keeping the state domain fixed). The answer is a two-sorted Los-type Preservation Theorem, as found in van Benthem (1985) (Lemma 17.19). Modulo logical equivalence, these are all formulas constructed using atoms and their negations, the connectives $\wedge$ and $\vee$, quantifiers $\forall$, $\exists$ over states and only existential quantifiers $\exists$ over actions. Many further logical questions concerning state languages found in this paper can be extended to this broader formalism.

2.3. From Languages to Simulations, and Conversely

One striking difference between the more 'logical' and the more 'computational' literature in this area is one of emphasis. On the former approach, one tends to start from a given formalism, and then studies its expressive power by inventing some matching notion of 'semantic equivalence' between models for which the language is invariant. A typical example of this route are Ehrenfeucht games in first-order logic (cf. Immerman and Kozen 1987, Doets 1987, 1993). In the computational approach, however, one rather tends to start from intuitions concerning processes and their equivalences, and then designs some appropriate language respecting the latter. Sub specie aeternitatis, however, both approaches are two sides of the same coin, and often produce the same results. Thus, the computational notion of bisimulation (Park 1981) had already been proposed as the preferred invariance

for modal logic in Van Benthem (1976) (generalizing the notion of a 'p-morphism' going back to De Jongh and Troelstra 1966) – and in fact, it may also be seen as a clipped version of Ehrenfeucht Games, appropriate to the modal fragment of our first-order logic of LTSs.

Here are some specific examples illustrating this general point. As we shall see in Section 5), the first-order formulas $\phi(x)$ expressing unary properties of states that are invariant under bisimulation are essentially those earlier 'modal' ones having one free variable x. The latter reflect precisely the back-and-forth steps in the above definition of bisimulation. From a more general point of view, all these formulas belong to the 'two-variable fragment' of predicate logic, whose formulas can be written using only the two variables x, y (free or bound). This fragment also allows us to express further properties of LTSs, referring also, e.g., to predecessors rather than successors in the ordering of labeled transitions. Its proper notion of invariance involves an interesting extension of the notion of bisimulation to matchings between pairs of states, where the next state can now be chosen freely:

DEFINITION. A *2-simulation* is a relation C between states in S and S' as well as between pairs of states in S and pairs of states in S' satisfying:

(i) the matches in C are partial isomorphisms with respect to all predicates of our first-order language,

(iia) if $s\ C\ s'$, then for arbitrary states t in S, there exists some t' in S' with $(s,t)\ C\ (s',t')$,

(iib) and vice versa,

(iii) C is closed under taking restrictions, that is, if $(s,t)\ C\ (s',t')$, then both $s\ C\ s'$ and $t\ C\ t'$.

This notion is stronger than bisimulation, in that it allows even arbitrary choices of states in the back-and-forth clauses (not just successors of the previous one). An easy induction shows that C-matching states or pairs of states in two LTSs satisfy the same formulas from the two-variable fragment of first-order logic. This notion may be generalized to arbitrary *n-variable fragments*, with an obvious corresponding notion of *n-simulation* involving matching sequences of states up to length n. Then we have the following general result (for a proof, cf. Van Benthem 1991, Chapter 17):

THEOREM. A predicate-logical formula $\phi(x_1,\ldots,x_n)$ is definable inside the n-variable fragment if and only if it is invariant for n-simulations.

The next level beyond ordinary bisimulation in this hierarchy has $n = 3$. Here lie the usual languages of Temporal Logic, which allow one to also express properties of intermediate stages of labeled transitions, such as the well-known operator

$$\text{Until}_a\ pq \qquad \exists y(Py \wedge \forall z((x \rightarrow^a z \wedge z \rightarrow^a y) \rightarrow Qz)).$$

Such fragments suggest intermediate notions of 3-simulation, working only with various restricted choices for the next state continuing the current matching.

3. Definability of Process Equivalences

What is the complexity of definition for the various notions of process equivalence introduced in the above? For the purposes of this section, we think of single models and let our relations $\equiv$ connect pairs of states inside them. Let $\mathbf{K}$ be the class of models where this relation satisfies one of the above definitions. We call the relevant notion of process equivalence *elementary* ('EC' for short) if $\mathbf{K}$ is definable by means of some first-order sentence τ ($\equiv$), and EC_Δ if it is definable by means of some infinite set of first-order sentences T ($\equiv$). An obvious example is 'bisimulation', whose very definition has a simple first-order form, whence it is EC – at least for the case where only *finitely many atomic actions* are considered. We shall make the latter assumption henceforth, to exclude trivial sources of complexity.

Remark. Explicit Definability.
One might also consider a stronger notion of definability for the relevant classes $\mathbf{K}$, namely in the form $\forall xy(x \equiv y \leftrightarrow \phi(x,y))$, where the first-order formula ϕ does not contain the predicate $\equiv$. Obviously, this will imply ECness, but the converse does not hold. 'Bisimulation' is a counter-example, as may be seen by an application of Padoa's Method: the same LTS can support different bisimulations. In fact, by Beth's Theorem, explicit definability of this kind is equivalent to EC_Δ-ness plus implicit definability of the relation $\equiv$ in terms of the other vocabulary.

Finite Trace Equivalence

From the earlier definition of FinTE, the following is easy to read off:

PROPOSITION. *Finite Trace Equivalence is EC_Δ.*

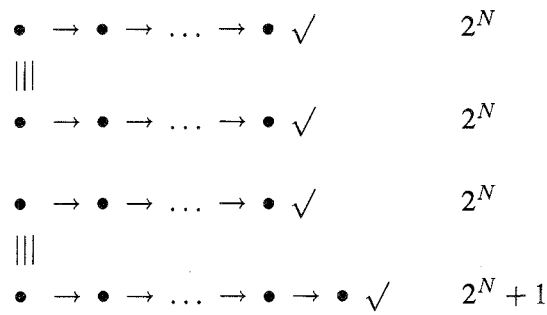
Proof. The above definition had the form $\forall xy(x \equiv y \rightarrow (\phi(x) \leftrightarrow \phi(y)))$, where the formulas ϕ are arbitrary 'succesful path formulas' of the form

$$\exists x_1(x \rightarrow^{a1} x_1 \wedge \ldots \wedge \exists x_n(x_{n-1} \rightarrow^{an} x_n \wedge \surd(x_n))\ldots). \qquad \blacksquare$$

PROPOSITION. *Finite Trace Equivalence is not EC.*

Proof. We refute this definability, even in the case with just one atomic label a. Let ϕ ($\equiv$) be any purported first-order definition, of quantifier depth N. Consider the following two models. One has two disjoint chains of length 2^N ending

with success, starting at states which are $\equiv$-related (and this match is the only pair in $\equiv$), the other has two such chains of lengths 2^N, $2^N + 1$:



Using an Ehrenfeucht game just as in the usual completeness proofs for linear order (cf. Doets 1987), one easily shows that these two models are indistinguishable by means of first-order sentences up to quantifer depth N. The point is to match the obvious initial and terminal points in the above two LTSs, while maintaining a judicious 'invariant' throughout with i more moves to go: viz. on both sides, keeping relative distances $< 2^i - 1$ equal between all points chosen so far. Our initial situation will then allow the 'analogy' player N safe moves against any opponent. But this yields a contradiction, since the relation $\equiv$ is a finite trace equivalence in the first model, but not in the second, whereas the purported definition ϕ ($\equiv$) cannot see any difference between these two LTSs. ■

PROPOSITION. *Complete Finite Trace Equivalence is not EC_Δ.*

Proof. Suppose that T ($\equiv$) defined this notion. Then we have, in particular, that

$$T,\ \{\phi(x) \leftrightarrow \phi(y) \mid \text{ all succesful path formulas } \phi\} \models x \equiv y.$$

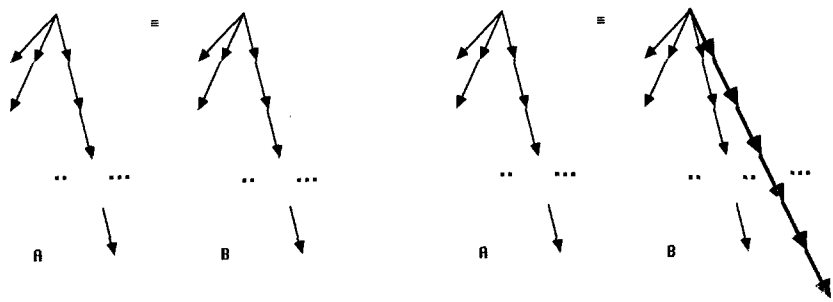
By the Compactness Theorem, some finite set of such path equivalences already implies $x \equiv y$ on the basis of T. But then we have complete finite trace equivalence whenever we have it up to some fixed finite depth: which is plainly false. ■

Full Trace Equivalence

PROPOSITION. *Full Trace Equivalence is not EC_Δ.*

Proof. Consider one atomic transition only. Take an LTS N consisting of one root from which finite branches fan out of arbitrary length, all ending in success. By a standard Compactness argument, N has an elementary extension N' containing at least one infinite branch from the root. Our first model M consists of two disjoint copies of N (denoted in our language by unary predicates A, B) and the second model M' of the same A-copy but with the elementary extension N' in its B-part. Moreover, in both cases, the relation $\equiv$ consists of just the pair connecting

the roots in the two components. In a picture, we have:



CLAIM. *M' is an elementary extension of M.*

Proof. In these models, any first-order formula is equivalent to a Boolean compound of completely A-relativized or B-relativized formulas whose free variables all lie within the relevant component. In particular, the atom '$x \equiv y$' amounts to stating that one of these states is in A and the other in B and that both are roots within their component, while $x \rightarrow^a y$ only holds with x, y lying in the same component. This decomposition then allows us to use the two separate facts that both components of M' are elementary extensions of their counterparts in M. ■

To complete the main argument, it is clear that $\equiv$ is a full trace equivalence in M, but not in M': whence it cannot be first-order definable. ■

PROPOSITION. *Complete Full Trace Equivalence is not EC_Δ.*

Proof. A simple counter-example runs as follows. On the LTS $(N, >)$ with transitions (i, j) with $i > j$ and success only at the initial state 0, identity of full trace sets coincides with real identity, whence $\equiv$ must be equal to $=$ (i.e., the first-order formula $\forall xy(x \equiv y \leftrightarrow x = y)$ holds). Now take any elementary extension of this LTS: it will contain at least one copy of the integers in its tail, while $\equiv$ still remains real identity. But all states in the integer tails have the same trace sets, and hence share complete full trace equivalence. This shows that the class to be defined is not closed under elementary equivalence, and so it cannot be first-order definable. ■

DIGRESSION. *Compatible Process Equivalences.*
There is a more general reason for the previous outcome. Except for bisimulation, the earlier notions of process equivalence fell into the following pattern. Some 'complete' invariance relation $\equiv$ is defined by a structural condition on labeling patterns, while there is a weaker 'compatible' version that makes the relation merely *imply* the former. (Alternatively, the former is the 'completion' of the latter.) Now, we have this

OBSERVATION. *If some complete process equivalence is EC_Δ, then so is its 'compatible version', and in fact, both will be EC.*

Proof. Suppose that a first-order set T ($\equiv$) defines our complete process equivalence. The uniqueness of a completed equivalence implies that $\equiv$ is implicitly defined in all models of T (in terms of the remaining vocabulary). Hence, by Beth's Definability Theorem, there must be some explicit first-order definition ϕ (not involving $\equiv$) such that $T \models \forall xy(x \equiv y \leftrightarrow \phi(x, y))$. By a standard argument then, the theory T ($\equiv$) may be equivalently formulated as (1) the set of all its $\equiv$-free consequences, plus (2) the preceding definition for $\equiv$. Moreover, since any LTS can be expanded with the completed equivalence, the former part must be universally valid (no LTS can be excluded), so that T is essentially just the latter definition. But then, it is immediate that the simple version of our process equivalence will be defined by the implication $\forall xy(x \equiv y \rightarrow \phi(x, y))$. ■

'Compatibility' is a general operation generating new process equivalences from old, and one can study various questions of definability and axiomatizability from this paper systematically under this view-point. We do not pursue this matter here.

Bisimulation

As was already observed before, we have (at least in the case with finitely many atomic actions; otherwise, the definition involves infinitely many back-and-forth assertions and it becomes EC_Δ):

PROPOSITION. *Bisimulation is EC.*

Proof. By a straightforward transcription of the above definition. ■

PROPOSITION. *Maximal Bisimulation is not EC_Δ.*

Proof. Consider the natural numbers N with predecessor as the single atomic action:

$$0 \leftarrow 1 \leftarrow 2 \leftarrow \cdots \leftarrow n \leftarrow \cdots$$

The identity relation is the maximal bisimulation in this case, as is easy to see (any non-trivial identifications would eventually clash at zero). But now, consider some elementary extension of this structure. This will contain states with infinitely many predecessors (and successors) lying in copies of the integers, inside which all elements can be identified via a bisimulation. Therefore, the maximal bisimulation will not be real identity in the latter case. But, if the latter notion were definable by some set of first-order sentences T ($\equiv$), then this theory would hold in both LTSs

described: which yields a contradiction as before. ■

Generated Graph Isomorphism

PROPOSITION. *Generated Graph Isomorphism is not* EC_Δ.

Proof. The argument is similar to that for Full Trace Equivalence. The relation $\equiv$ in the model M depicted there satisfies the condition of Generated Graph Isomorphism. But the pair of the two roots no longer qualifies in the elementary extension constructed, as the two components have now become non-isomorphic. ■

PROPOSITION. *Complete Generated Graph Isomorphism is not* EC_Δ.

Proof. Again, one can use an earlier counter-example, namely that of the reverse natural numbers $(N, >)$ with one of their elementary extensions. ■

The above negative results leave the question unanswered just where the non-EC_Δ cases are located in higher-order logical formalisms. For instance, 'complete trace equivalence' is naturally definable in $L_{\omega 1 \omega}$, as it involves a conjunction of path equivalences of each finite length. 'Maximal Bisimulation' is naturally Δ^1_2, however, having the form $\forall xy(x \equiv y \leftrightarrow \exists R$ ("R is a bisimulation" and $Rxy))$, which becomes a conjunction of a Σ^1_1 and a Π^1_1 sentence. We shall not pursue these matters here.

4. Axiomatizability of Process Equivalences

Another approach to the description of process equivalences is via axiomatization of their first-order consequences. In particular, when does the set **K** of models for a process equivalence $\equiv$ as described above have a recursively enumerable set of first-order consequences? We consider some of the above notions here, demonstrating different possible reasons for a positive outcome.

PROPOSITION. *The following process equivalences have a recursive axiomatization: Finite Trace Equivalence, Bisimulation, Generated Graph Isomorphism.*

Proof. Finite Trace Equivalence had a first-order definition which is recursively enumerable, and hence the latter also serves as an RE axiomatization for its first-order consequences, which has an equivalent recursive axiomatization by a well-known result of Craig's. The same reasoning applies to Bisimulation, which even had a finite first-order definition.

To axiomatize the first-order consequences of Generated Graph Isomorphism, we need a more elaborate argument, relying on the proof of a preservation result

in Section 5 below. That proof shows that (i) GGI preserves all 'restricted first-order formulas' $\phi(x)$ constructed from arbitrary atoms by Boolean operations and restricted existential quantifiers $\exists y(R_a xy \wedge$ (with x, y distinct). (ii) In any countable 'recursively saturated' model, the relation of agreeing on all restricted formulas is a generated graph isomorphism between states. Now, the relevant first-order theory is axiomatized by all formulas of the form

$$\forall xy(x \equiv y \rightarrow (\phi(x) \leftrightarrow \phi(y)),$$

where ϕ is a restricted first-order formula. Obviously, these formulas all follow from GGI, by observation (i). Conversely, suppose that some first-order ψ does not follow from these axioms. Then there must be some countable model for the latter where ψ fails (by the Completeness and Löwenheim-Skolem theorems), which also has a countable recursively saturated elementary extension (cf. Keisler 1977). But then, $\equiv$ will be a generated graph isomorphism in the latter model, whence Ψ fails in an LTS of the original class. ■

Next, we turn to some negative results.

PROPOSITION. *The first-order consequences of Maximal Bisimulation and Complete Trace Equivalence are not RE, and not even arithmetically definable.*

Proof. We treat the case of Maximal Bisimulation only, that of Complete Trace Equivalence being similar. First, recall the following observations from earlier proofs. On the natural numbers with predecessor as an atomic action, the maximal bisimulation coincides with the relation of identity. Now, consider a first-order language whose vocabulary type includes enough predicates to express the basic Peano Axioms for zero, successor, predecessor, addition and multiplication (but without the Induction Schema). Call the latter theory PA*. Now, on LTSs satisfying PA* where the binary relation $\equiv$ coincides with the maximal bisimulation with respect to 'predecessor' actions, the following formula defines the initial segment of the natural numbers (since all numbers in the non-standard 'integer tails' admit of non-trivial bisimulations):

$$\phi(x) = \forall y(x \equiv y \rightarrow x = y).$$

But this implies the following effective reduction of standard truth for arithmetical sentences α in terms of their syntactically relativized versions $(\alpha)^{\phi}$:

IN $\models \alpha$ iff

PA* $\rightarrow (\alpha)^{\phi}$ holds in all LTSs where $\equiv$ is the maximal bisimulation.

Thus, the first-order consequences of Maximal Bisimulation are at least as complex as True Arithmetic, which forms a non-arithmetical theory by Tarski's Theorem. (Using a suitable encoding, this argument can be improved to work for non-arithmetical vocabularies too.) ■

Remark. Concrete Axioms.
The preceding result does not mean that nothing concrete can be said about the theory of completed or maximal process equivalences. In particular, these will contain the whole theory of their 'compatible notion', while adding certain 'induction principles'. For instance, both Complete Finite Trace Equivalence and Maximal Bisimulation satisfy the following principle (with proposition letters, a small addition is needed):

$$\forall xy((\forall z(R_a xz \to \exists u(R_a yu \wedge z \equiv u)) \\ \wedge \forall z(R_a yz \to \exists u(R_a xu \wedge z \equiv u))) \to x \equiv y).$$

Finally, results on axiomatizability may also have consequences for definability:

COROLLARY. *Maximal Bisimulation is not* Σ^1_1*-definable.*

Proof. The reason is that the first-order consequences of any Σ^1_1-sentence are RE, since, for all first-order ϕ, ψ, $\exists P_1 \ldots P_k \phi(P_1, \ldots, P_k) \models \psi$ iff $\phi(P_1, \ldots, P_k) \models \psi$, where the right-hand notion is RE by the Completeness Theorem. ■

5. Simulation Invariance and Logical Definability

Now, we want to study the first-order effects of notions of proces equivalence from another angle. Given any of the above relations $\equiv$ between models M, M', what can we say about first-order properties of processes that are preserved by them? More technically, we shall call a first-order formula $\phi(x)$ over states *invariant for* $\equiv$ if,

whenever $s \equiv s'$, $M \models \phi[s]$ if and only if $M' \models \phi[s']$.

In general, this will take us to fragments of the full first-order language. Here are some relevant results, stating which fragments match various notions of bisimulation.

Remark. Meaning Postulates.
Here and elsewhere, we work with arbitrary LTSs. All our results will go through, however, if one imposes arbitrary first-order conditions on our model class, such as the earlier meaning postulates on success and dead-lock.

THEOREM. *A first-order formula is invariant for Finite Trace Equivalence iff it is definable as a Boolean combination of successful path formulas.*

Proof. From right to left, this was already observed above. Conversely, suppose that $\phi(x)$ is invariant for FinTE. Consider the set BPF (ϕ) of all Boolean combinations

of succesful path formulas that follow logically from $\phi(x)$. The desired definability assertion is an obvious consequence of the following

CLAIM. *BPF* $(\phi) \models \phi(x)$.

Proof. Let M, $s \models$ BPF (ϕ). Then, by a simple argument, the set consisting of $\phi(x)$ together with all Boolean combinations of succesful path formulas true at M, s is finitely satisfiable. Therefore, by Compactness, it must be simultaneously satisfiable, say in some LTS N, s'. This means that the relation of complete finite trace equivalence between these two models relates s to s'. By the given invariance of $\phi(x)$, this implies that M, $s \models \phi(x)$. ■

Note that the preceding analysis also establishes that the same syntactic class captures invariance with respect to Complete Finite Trace Equivalence. In general, of course, invariance for the completed equivalence notion is implied by, but need not imply, that under the original one. Next, we move from successful path formulas to arbitrary ones, being existential descriptions of finite chains of transitions with arbitrary success or dead-lock behaviour in their states.

THEOREM. *A first-order formula is invariant for Full Trace Equivalence iff it is definable as a Boolean combination of arbitrary path formulas.*

Proof. The argument is similar to the one above. This time, however, one passes from M, s and N, s' to two *ω-saturated* elementary extensions, being models in which each finitely satisfiable set of first-order formulas involving only finitely many parameters from the domain is simultaneously satisfiable. (Each model has such an elementary extension: cf. Chang and Keisler 1973. Usually, one considers only sets with one free variable x, but the result also holds if we allow a countable sequence of free variables.) Now, in the latter LTSs, coincidence of arbitrary path formulas at two states gives in fact full trace equivalence. For any countably infinite chain on one side can be described using some countable set of formulas which will be finitely satisfiable on the other side, so that saturation gives us the whole chain there, too. ■

Next, we come to a basic relevant result from Modal Logic in the seventies. Recall the 'modal fragment' of our first-order language introduced in Section 2.2.1:

THEOREM. *A first-order formula $\phi(x)$ is invariant for Bisimulation iff it is definable as a modal formula with one free variable x.*

Proof. (Cf. van Benthem 1976, 1991). Modal formulas are invariant for bisimulation, by an obvious induction. (The back-and-forth clauses are tailored exactly to take care of the existential quantifier case.) To prove the converse direction, let

mod (φ) be the set of all modal consequences of φ. We show that

CLAIM. mod $(\varphi) \models \varphi$,

from which the desired modal definability follows by the Compactness Theorem. For, then ϕ will be implied by, and hence equivalent to, some finite conjunction of its modal consequences. So, let $M, w \models$ mod (φ). By a standard model-theoretic argument, there must be a model N, v satisfying

(i) $N, v \models \varphi$,

(ii) (M, w) and (N, v) verify the same modal formulas.

Now, take any two ω-saturated elementary extensions of M, N: say, M^* and N^*. In such saturated models, the following stipulation defines a bisimulation:

$$M^*, x \text{ satisfies the same modal formulas as } N^*, y. \qquad \#$$

For instance, if x' is any R_a-successor of x in M^*, then each *finite* subset Δ of its modal theory is satisfiable in some R_a-successor of y in N^*: since the modal formula $\Diamond \bigwedge \Delta$ holds at x and therefore also at y. But then, its *full* modal theory must be satisfiable in some R_a-successor y' of y, by countable saturation: and such a state y' is the required match for x' in the zigzag clause of bisimulation. Thus, we can conclude our argument by the following observations, starting from N, $v \models \varphi$:

$N^*, v \models \varphi$ (by elementary extension),
$M^*, w \models \varphi$ (by bisimulation invariance),
$M, w \models \varphi$ (by elementary descent). ■

As before, the same argument also characterizes another invariance. For, note that the bisimulation constructed above is in fact the maximal one between M^* and N^*. Therefore, we have also shown that the same syntactic class captures invariance with respect to Maximal Bisimulation.

A slightly refined version of this argument shows the following result.

THEOREM. *A first-order formula is invariant for Generated Graph Isomorphism iff it is definable as a restricted formula.*

Proof. Consider the *restricted fragment* of first-order logic mentioned in Section 4 above, which is constructed like the modal fragment while also allowing binary atoms (in addition to unary ones). We show that, in an obvious notation: rest $(\phi) \models \phi$, by making a few changes in the above argument, without loss of generality. First, the models M, N can be chosen to be *countable* by the Löwenheim-Skolem Theorem. Then, we pass on to a countable *recursively saturated* elementary extension MN^* of the model pair M, N (cf. Keisler 1977 for this technique), which has saturation with respect only to RE sets of first-order formulas. (The latter restriction allows us to make do with countable models.) Then, equality of restricted

theories as in the above stipulation # yields the additional information that there is an isomorphism between the generated submodels of x and y in MN^*. This may be obtained as follows. One starts from complete enumerations $x = a_1, a_2, \ldots$ and $y = b_1, b_2, \ldots$ of the generated submodels. Then one uses a Cantor-style zigzag argument to extend the initial match to one between progressive sequences, maintaining the following invariant:

$$M, a_1, \ldots, a_n \text{ satisfies the same restricted formulas as } N, b_1, \ldots, b_n.$$

With each extension with a new object in the enumeration, use the RE set of formulas

$$\phi(a_1, \ldots, a_n, x_{n+1}) \leftrightarrow \phi(b_1, \ldots, b_n, x_{n+1})$$

where ϕ is restricted, to locate a matching object on the other side. Here, one may use restricted quantifiers only to describe the behaviour of some object x_{n+1} sharing finitely many restricted properties with, say, a_{n+1}. This is possible, as each object in a generated submodel is reachable from its origins via some finite sequence of labeled transitions. ■

Again, by the same argument, the above syntactic class also captures invariance with respect to Complete Generated Graph Isomorphism.

Remark. Complexity of Fragments.
Membership in these first-order fragments may be *undecidable.* Modal formulas are a case in point, because of the following effective reduction. Let $\alpha(x)$ be any first-order formula, and P (unary), R (binary) new predicate letters outside of α.

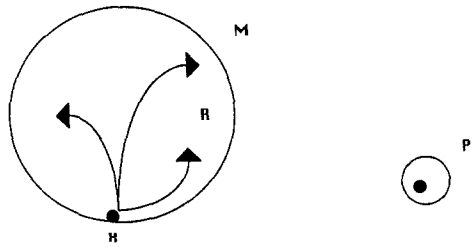
CLAIM. *The following two assertions are equivalent:*

(i) α *is universally valid*

(ii) $\exists y Py \wedge (\neg\alpha)^\rho$ *is bisimulation invariant, where* $(\neg\alpha)^\rho$ *is the formula* $\neg\alpha$ *syntactically relativized to the subdomain* $\lambda z \bullet Rxz \vee x = z$.

Proof. If α is universally valid, then $\exists y Py \wedge (\neg\alpha)^\rho$ is equivalent to a contradiction, which is trivially invariant. If α is not universally valid, then $\neg\alpha(x)$ is true in some LTS M at x. Now let R be the universal relation in M – and then, add one unrelated

point where P holds to obtain a model M^+:



In the new model M^+, the point x validates $\exists y Py \wedge (\neg\alpha)^\rho$. But this formula fails in the R-closed submodel M generated by x, thereby violating the assumed invariance for bisimulation. ■

6. Analyzing Programming Repertoire

Labeled transition systems only deal with atomic actions. One way of introducing complex program structure is by moving from basic Modal Logic to Propositional Dynamic Logic (Harel 1984), which has *propositional formulas* describing states, while also describing complex transitions over LTSs by means of regular *program expressions* π (including tests on propositional formulas) and their corresponding modalities $\langle\pi\rangle$:

$$F : = AT\text{-}F \mid \neg F \mid (F \wedge F) \mid \langle P\rangle F$$

$$P : = AT\text{-}P \mid (P;P) \mid (P \cup P) \mid P^* \mid (F)?$$

Here again, there is invariance of dynamic formulas ϕ for bisimulations $\equiv$ between two models – but there is also a new aspect. Intertwined with the old proof of invariance, one also has to show that the usual back-and-forth clauses in bisimulation are inherited by the regular program constructions: and indeed, each binary transition relation $[[\pi]]$ shows this behaviour, upward from the atomic ones. More precisely:

PROPOSITION. *If $\equiv$ is a bisimulation between two models M, M', and $s \equiv s'$, then*

(i) *s, s' verify the same formulas of propositional dynamic logic*
(ii) *if $s[[\pi]]^M t$, then there exists t' with $s'[[\pi]]^{M'} t'$ and $s' \equiv t'$.*

This observation motivates the following notion of invariance for program operations (where we indulge in a slight abuse of notation):

DEFINITION. An operation $O(P_1, \ldots, P_n)$ on binary relations is *safe for bisimulation* if, whenever $\equiv$ is a relation of bisimulation between two models M, M' with respect to relations $R_1, \ldots, R_n$ and $R'_1, \ldots, R'_n$, then it is also a bisimulation

for the complex relations $O(R_1, \ldots, R_n)$, $O(R'_1, \ldots, R'_n)$.

It is easy to show that, e.g., the regular operations of relational composition ; and choice $\cup$ (i.e., Boolean union) have this property. Another example are the standard test relations $(\phi)?$ for modal formulas ϕ. (These examples reflect the key observations in the straightforward inductive proof of the previous Proposition.) For further reference, we also mention the safety of one less familiar relational operation, being a strong negation ('counter-domain'):

$$\sim(R) = \{(x, y) \mid x = y \text{ and for no } z : xRz\}.$$

Note, e.g., that all complex modal tests $(\phi)?$ can be reduced to atomic ones using only the regular operations ; and $\sim$, by the following three valid identities:

$$\begin{aligned} (\phi \wedge \psi)? &= (\phi)?;(\psi)? \\ (\neg\phi)? &= \sim(\phi)? \\ (\langle a\rangle\phi)? &= \sim\sim(a;(\phi)?) \end{aligned}$$

Now, the natural question arises whether there exists some equivalent of our earlier preservation theorems, characterizing those programming operations that guarantee safety for bisimulation. Again, to make the question precise, we go to the standard first-order description language over labeled transition systems – this time, adding arbitrary binary relation symbols for transition relations. Let us call a programming operation *first-order* if it can be defined using a formula $\theta(x, y)$ of this language with two free variables. All earlier operations are first-order in this sense:

$$\begin{aligned} &(R_1;R_2) && \exists z(R_1xz \wedge R_2zy) \\ &(R_1 \cup R_2) && R_1xy \vee R_2xy \\ &\sim(R) && x = y \wedge \neg\exists z Rxz \\ &(P)? && x = y \wedge Px \end{aligned}$$

Then, we have the following main result:

THEOREM. *A first-order relational operation $O(R_1, \ldots, R_n)$ is safe for bisimulation if and only if it can be defined using atomic relations R_ixy, atomic tests (q)? for propositional atoms in our models, using just the three operations ;, $\cup$ and $\sim$.*

To prove this result, one needs an excursion into the model theory of Modal Logic, whence we do not give details of the argument here (cf. van Benthem 1993). Safety of programming operations is a property which makes sense for other notions of process equivalence, too. We leave the matter of their complete safe repertoires open here.

7. Infinitary First-Order Theory of Transition Systems

In Section 2.1, several computational notions concerning LTSs received a natural formalization in the infinitary first-order logic $L_{\omega 1\omega}$. The purpose of this section is to show that the earlier theory extends to this formalism too. For a survey of relevant model-theoretic techniques replacing Compactness, cf. Keisler 1971.

Definability

Here are some observations concerning the earlier definability questions for process equivalences in this formalism. For a start, Complete Finite Trace Equivalence becomes definable by an infinite conjunction of succesful path equivalences. Next, there are several negative results:

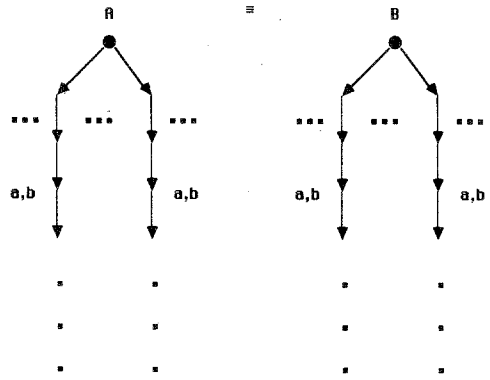
PROPOSITION. *Full Trace Equivalence, Complete Full Trace Equivalence, Maximal Bisimulation, Generated Graph Isomorphism and Complete Generated Graph Isomorphism are not $L_{\omega 1\omega}$-definable.*

Proof. The case of Maximal Bisimulation goes as follows. Note that on well-orderings with labeled transitions corresponding to downward arrows, the maximal bisimulation must be the identity. Moreover, on non-well-orders, any downward infinite chain will allow some non-trivial bisimulation. Now, suppose that Maximal Bisimulation were $L_{\omega 1\omega}$-definable. Then the formula stating that some relation

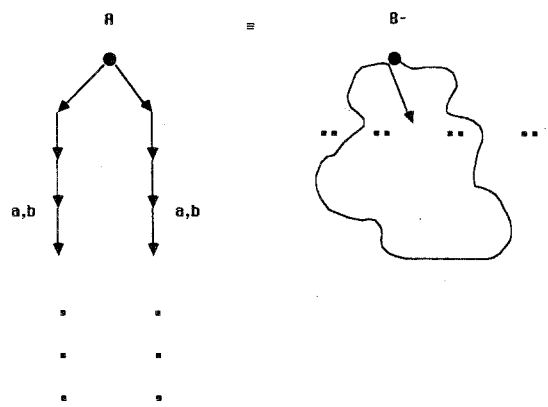
$$< \text{ is a linear order and } \equiv \text{ is the maximal bisimulation,}$$
$$\text{while } \forall xy(x \equiv y \leftrightarrow x = y),$$

will define the well-orderings with their maximal bisimulation in $L_{\omega 1\omega}$. But this contradicts the non-$L_{\omega 1\omega}$-definability of well-order.

The other negative assertions in this sequence all depend on the Löwenheim-Skolem Theorem for our infinitary language. The example of Full Trace Equivalence will suffice. Take a model with two uncountable trees of branches showing every possible alternation of a- and b-labels, which are $\equiv$-connected in their roots, separated by unary predicates A, B as in a previous argument:



Next, take a countable elementary submodel of one of the components. The result will be an elementary submodel where $\equiv$ is no longer a full trace equivalence – and thus we have refuted $L_{\omega 1\omega}$-definability: ■



Scott Isomorphism Theorem

A corollary of the earlier proof of the preservation theorem for bisimulation is the fact that two *finite* LTSs satisfy the same modal formulas in states x, y if and only if there exists a bisimulation $\equiv$ between them with $x \equiv y$. This result may be extended to $L_{\omega 1\omega}$ via the following modification of 'Scott's Theorem'. Here, the 'modal' fragment of this formalism is defined as for first-order logic, but this time allowing infinite conjunctions and disjunctions too.

THEOREM. *For each countable LTS M with state x, there exists a modal $L_{\omega 1\omega}$-formula $\phi(u)$ such that, for each countable LTS N and state y, $N, y \models \phi$ iff there exists a bisimulation $\equiv$ between M, N with $x \equiv y$.*

COROLLARY. *In countable LTSs M, N, states x, y verify the same modal $L_{\omega 1\omega}$-theory iff there exists a bisimulation connecting them.*

Proof. Define the following modal formulas by induction, for each y in M:

$$\begin{aligned}
\phi_y^0(u) &:= \text{a complete description of unary atoms true at } y \text{ in } M \\
\phi_y^\alpha(u) &:= \textstyle\bigwedge_{\beta<\alpha} \phi_y^\beta(u) \quad \text{if } \alpha \text{ is a limit ordinal} \\
\phi_y^{\alpha+1}(u) &:= \phi_y^\alpha(u) \wedge \textstyle\bigwedge_{R_c yz} : \exists u(R_c yu \wedge \phi_z^\alpha(u)) \wedge \\
&\qquad \forall u(R_c yu \to \textstyle\bigvee_{R_c yz} \phi_z^\alpha(u)).
\end{aligned}$$

Since M is countable, we have:

$$\exists \alpha < \omega_1 \forall \beta \geq \alpha : \forall u : \phi_y^\alpha(u) \leftrightarrow \phi_y^\beta(u)$$

(Choose α_z with $\neg\phi_y^{\alpha z}(z)$, if existent, for all y: $\sup_{z \in M}(\alpha_z) < \omega_1$).

And for the same reason,

$$\exists \alpha < \omega_1 \forall yu : \phi_y^\alpha(u) \leftrightarrow \forall \beta \geq \alpha \, \phi_y^\beta(u).$$

Now, for the α found here, define the above formula $\phi(u)$ as

$$\Phi_x^\alpha \qquad \phi_x^\alpha(u) \wedge \wedge_{z \in M} \forall w(R^* uw \to (\phi_z^\alpha(w) \to \phi_z^{\alpha+1}(w)),$$

where R^* describes the transitive closure of the union of all atomic transition relations. Note that Φ_x^α can be taken to be modal by rewriting the part '$\forall w(R^* uw \to$' in the following form:

$$\forall w \left(\vee_n (\cup_c R_c)^n \, uw \to \right.$$

i.e.,

$$\wedge_n \forall w \left((\cup_c R_c)^n \, uw \to \right.,$$

which can be unpacked to single-restriction quantifiers. Now, set (with some harmless abuse of notation):

$$u \equiv v \qquad \text{iff} \qquad N, v \models \Phi_u^\alpha.$$

CLAIM. *This relation* $\equiv$ *is a bisimulation.*

Proof.
(i) 'Atomic equivalence' is obvious.
(ii) Suppose that $u \equiv v$ and $uR_c u'$ in M.

Then $\quad N, v \models \Phi_u^\alpha(v)$,
so $\quad N, v \models \phi_u^\alpha(v)$ (use the first conjunct of Φ_u^α),
and $\quad N, v \models \phi_u^{\alpha+1}(v)$ (use the second conjunct of Φ_u^α),

whence $N, v \models \exists w(R_c uw \wedge \phi^{\alpha}_{u'}(w))$,
i.e., there exists some v' with $R_c vv'$ in N such that
$N, v' \models \Phi^{\alpha}_{u'}(v')$.
Moreover, $N, v' \models \bigwedge_{z \in M} \forall w(R^* uw \rightarrow (\phi^{\alpha}_z(w) \rightarrow \phi^{\alpha+1}_z(w))$,
because every R^*-successor of v' is also one of v (since $vR_c v'$).
Hence, $N, v' \models \Phi^{\alpha}_{u'}$: and $u' \equiv v'$.

Next, suppose that $u \equiv v$ and $vR_c v'$ in N.

Then $N, v \models \Phi^{\alpha}_u(v)$,
so $N, v \models \phi^{\alpha}_u(v)$ and $N, v \models \phi^{\alpha+1}_u(v)$ (as before),
whence $N, v \models \forall w(R_c uw \rightarrow \bigvee_{R_c uz} \phi^{\alpha}_z(w))$,
i.e., $N, v' \models \Phi^{\alpha}_z(u)$ for some z with $R_c uz$ in M: set $u' = z$.
Moreover, the second conjunct of $\Phi^{\alpha}_{u'}$ holds at v', for the same reason as above,
so that $N, v' \models \Phi^{\alpha}_{u'}$: and again $u' \equiv v'$.

This completes the proof of the Claim, and hence of the Theorem. ■

Question. Would this argument also go through for the infinitary modal fragment corresponding exactly to the standard translation of propositional dynamic logic?

Invariance under Bisimulation

Our earlier preservation results may also be extended to this setting. Here is a key example of how we can modify the earlier proof method. (The following proof would also have worked for the first-order case!)

THEOREM. *An $L_{\omega 1 \omega}$-formula $\phi(x)$ is invariant for bisimulation iff it is equivalent to a modal $L_{\omega 1 \omega}$-formula.*

Proof. The non-trivial direction is from left to right. Consider the family of all finite sets of the form (assuming a language with one transition relation, for convenience):

"atoms $x \equiv y$ expressing intended bisimulation links" $\cup \Sigma(\mathbf{x}) \cup \Delta(\mathbf{y})$,

which have the property of 'Modal Consistency': for no modal formula $\Psi(x_1, \ldots, x_n)$ there are atoms $x_1 \equiv y_1, \ldots, x_n \equiv y_n$ available such that $\Sigma(\mathbf{x}) \models \Psi$ and $\Delta(\mathbf{y}) \models \neg\Psi$.

Remark. These extended modal formulas may have more than one free variable.

(It is easy to show that these must be equivalent to, possibly infinitary, Boolean combinations of unary modal formulas.)

CLAIM. *This family is a Consistency Property, and moreover, we can construct a model for each of its members while forcing* $\equiv$ *to be a bisimulation.*

If this is so, then we get the desired result as follows. Suppose that $\phi(x)$ is invariant for bisimulation. Then the set $\{x \equiv y, \phi(x), \neg\phi(y)\}$ has no model of the above kind, whence it must be modally inconsistent. That is, there exists some modal formula Ψ such that $\phi(x) \models \Psi(x)$ and $\neg\phi(x) \models \neg\Psi(x)$: i.e., $\models \phi(x) \leftrightarrow \Psi(x)$. ■

Let us now check the requirements for a consistency property (cf. Keisler 1971):

(i) If $\neg\phi \in \Sigma$ or $\neg\phi \in \Delta$, then its 'inversion' $\phi\neg$' can be added. This is immediate given the defining condition for our family.

(ii) The same holds for the cases 'adding a conjunct' and 'adding a witness for an existential quantifier' (note that these cases add no bisimulation atoms).

(iii) Disjunction. Suppose that $\bigvee_i \phi_i \in \Sigma$ (the case of Δ is similar), while no set of the form "old atoms AT, $\Sigma \cup \{\phi_i\}$, Δ" qualifies. Then, there exist modal formulas α_i with AT, $\Sigma \cup \{\phi_i\} \models \alpha_i(\mathbf{x})$ and AT, $\Delta \models \neg\alpha_i(\mathbf{y})$. But then AT, $\Sigma \models \bigvee_i \alpha_i(\mathbf{x})$ (recall that $\bigvee_i \phi_i \in \Sigma$!), and also AT, $\Delta \models \bigwedge_i \neg\alpha_i(\mathbf{y}) \leftrightarrow \neg\bigvee_i \alpha_i(\mathbf{y})$: which is a contradiction.

(iv) No contradiction occurs in Σ versus Δ: as $\bot$ is a modal formula.

Finally, here is how Bisimulation can be achieved during the model construction. One can intertwine the above extension process with steps of the following form (using some suitable enumeration of the *countably* many relevant formulas):

$x \equiv y \in$ AT and $Rxz \in S$:
add a *new* constant u with $z \equiv u \in$ AT, $Ryu \in \Delta$; and vice versa.

This move cannot disturb Modal Consistency:

If $\Sigma \models \alpha(\mathbf{x}, z)$ and $\Delta \cup \{Ryu\} \models \neg\alpha(\mathbf{y}, u)$,
then $\Sigma \models \exists z(Rxz \wedge \alpha(\mathbf{x}, z))(Rxz \in \Sigma)$
and $\Delta \models \neg\exists u(Ryu \wedge \alpha(\mathbf{y}, u))$

which is a contradiction. ■

Safety for Bisimulation

As for safety of programming operations under bisimulation, we merely conjecture that the only addition to the previous characterization are *arbitrary infinite unions*. Note that these include Kleene iterations as well as arbitrary unwindings for fixed-point recursions $\mu p \bullet \phi(p)$ with respect to p-positive formulas ϕ.

Remark. Full Infinity.
Finally, several of the above results could be extended to the infinitary language $L_{\infty\omega}$ with set conjunctions and disjunctions of arbitrary cardinality. For instance, maximal bisimulation is in fact the same as elementary equivalence in the latter language.

8. Special Classes of Transition Systems

There are various special classes of LTS where it makes sense to study the above phenomena of definability and invariance separately. We provide a first exploration.

Finite Models

PROPOSITION. *Every earlier notion of process equivalence is* EC_Δ.

Proof. All these definitions describe classes of models that are closed under isomorphic images. And every class of finite models, in a finite vocabulary, that has the latter closure property is definable in EC_Δ-form, viz. by the set of negations of all complete first-order descriptions of its finite non-members. ■
PROPOSITION. *No notion of process equivalence becomes EC which was not already so on arbitrary models.*

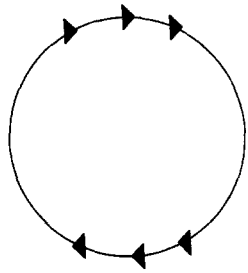
Proof. The method of proof here uses Ehrenfeucht Games. E.g., the earlier argument for Finite Trace Equivalence still applies. Another typical case is Complete Finite Trace Equivalence: its definability by any first-order sentence of quantifier depth N may be refuted via an Ehrenfeucht Game comparing the following two models:

(1) $\bullet \rightarrow\rightarrow\rightarrow\rightarrow\rightarrow\rightarrow\rightarrow\rightarrow\rightarrow\rightarrow\rightarrow\rightarrow\rightarrow\rightarrow\rightarrow\rightarrow \surd$

a chain of immediate successors, of length 2^N, with $\equiv$ as identity (being the full trace equivalence here),

(2) $\bullet \rightarrow\rightarrow\rightarrow\rightarrow\rightarrow\rightarrow\rightarrow\rightarrow\rightarrow\rightarrow\rightarrow\rightarrow\rightarrow\rightarrow\rightarrow\rightarrow \surd$

the same chain together with a suitably large 'disjoint cycle' (again, with $\equiv$ as the identity relation):



Up to N steps, the 'analogy player' has a winning strategy here, even though $\equiv$ is not full trace equivalence in the second model. (Doets 1993 suggests an alternative style of argument, with a detour via the infinite model $\mathbf{N} + \mathbf{Z} + \mathbf{N}^*$.) ■

As for axiomatizability, rather than definability, some things simplify here:

PROPOSITION. *A notion of process equivalence has a finite axiomatization iff it is EC.*

Proof. From right to left, this is obvious, as the defining first-order formula provides the axiomatization. Conversely, suppose that δ axiomatizes our notion $\equiv$. Then, it holds in any model for this process equivalence (being one of its own consequences). But also conversely, suppose that δ holds in some finite model M, which is fully described by the first-order sentence α_M. Then δ does not imply $\neg\alpha_M$ on the finite models, and hence there is a finite model in our intended class verifying α_M, which must therefore be isomorphic to M itself. So, M is in our intended class. ■

Finally, a generalization of our earlier preservation results for simulation invariances to the finite case might be difficult, witness some recent failures of classical model-theoretic results in current Finite Model Theory (Gurevich 1985).

Further Classes of Transition Systems

Besides finite models, other natural restricted classes of LTSs occur in the literature. Here are some illustrations of what may happen to our previous concerns. On *countable models*, no difference in outcomes arise at all, as all first-order arguments so far used countable LTSs only (mainly thanks to the Löwenheim-Skolem theorem). (There were differences in outcomes, of course, with the infinitary language $L_{\omega 1 \omega}$.) To take another kind of restriction, on *finitely branching trees*, here is a typical difference in outcome (taking 'finite trace equivalence' now with respect to arbitrary finite traces, whatever the status of their end-points):

PROPOSITION. *Finite Trace Equivalence and Full Trace Equivalence are equivalent.*

Proof. FullTE always implied FinTE. Conversely, suppose that two nodes x, y are finite trace equivalent (in our extended sense). Let some infinite trace start at x, and consider the subgraph at y matching all its initial segments. By König's Lemma for finitely branching trees, there must be some infinite branch at y, which matches the infinite trace at x. ■

Finitely, to demonstrate another kind of change, we consider LTSs whose transition relations are all *well-founded.*

PROPOSITION. *Maximal Bisimulation is EC on well-founded models.*

Proof. For convenience, we restrict attention to the case with one atomic action. The definition that works here consists of the first-order definition of bisimulation plus the 'Induction Principle' stated toward the end of Section 4. Let C be any relation on some well-founded LTS (S, R_a) satisfying these two requirements which does not equal the maximal bisimulation $\equiv$. As C is a bisimulation, it is contained in $\equiv$. Now, by well-foundedness, take some minimal point x where a situation of type $x \equiv y$, $\neg xCy$ occurs, with some y exemplifying this mismatch. Then, by minimality, it is easy to see that the condition for the induction principle will be satisfied for all R_a-successors of x and y. (For instance, if $x \rightarrow^a z$, then z must have a $\equiv$-matching $\rightarrow^a$-successor u of y, because $\equiv$ is a bisimulation. By the minimality of x with respect to $\equiv / C$ mismatches, this implies that zCu. The proof for the converse direction is analogous.) Thus, the induction principle implies that xCy after all: which yields a contradiction. ■

9. Adding Process Structure 1: Silence and Branching

One typical feature in more realistic process theories are so-called 'silent steps' $\rightarrow^\tau$ that reflect unobservable transitions. These may arise from hiding certain transitions in an LTS under this distinguished relation. Structures will then have the form

$$(S, A, \rightarrow, \rightarrow^\tau).$$

An interesting class are the Δ-*saturated* LTSs (introduced in Bergstra and Klop 1988) satisfying the following condition for all atomic actions a, and all states x, y, z:

$$((x \rightarrow^a y \wedge y \rightarrow^\tau z) \rightarrow x \rightarrow^a z)$$

$$\wedge\ ((x \to^{\tau} y \wedge y \to^{a} z) \to x \to^{a} z)$$
$$\wedge\ ((x \to^{\tau} y \wedge y \to^{\tau} z) \to x \to^{\tau} z)).$$

The 'Δ-saturation' of an LTS is obtained by adding the smallest set of silent transitions that makes these Horn clauses true.

We show how various earlier questions extend to this setting. For a start, the most basic notion of process equivalence for these structures is as follows.

DEFINITION. A binary relation $\equiv$ is a *weak bisimulation* between two transition systems with silent steps if the following conditions hold, whenever $x \equiv y$:

(i) atomic markings agree on x and y,

(ii) if $x \to^{a} x'$, then there exists some finite sequence of transitions on the other side which is of the form $y = y_1 \to^{\tau} \ldots \to^{\tau} y_n \to^{a} y_{n+1} \to^{\tau} \ldots \to^{\tau} y_{n+k} = y'$ such that $x' \equiv y'$; and vice versa,

(iii) if $x \to^{\tau} x'$, then there exists some finite sequence of transitions on the other side of the form $y = y_1 \to^{\tau} \ldots \to^{\tau} y_n = y'$ such that $x' \equiv y'$; and vice versa.

More sophisticated alternatives may be found in De Nicola and Vaandrager 1990 (cf. van Benthem et al. 1993 for logical discussion, including possible first-orderization via a new sort of 'silent branches').

PROPOSITION. *Weak bisimulation is EC on Δ-saturated LTSs. Weak bisimulation is not EC_{Δ} on arbitrary LTSs.*

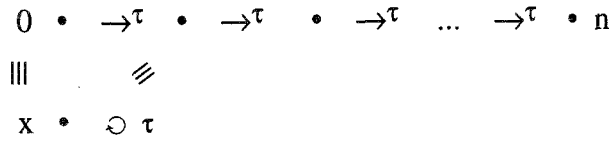
Proof. On Δ-saturated models, weak bisimulation reduces to ordinary bisimulation (with respect to the modified relation $\to^{a} \vee =$), which had a first-order transcription. In general, however, the above definition is irreducibly infinitary in $L_{\omega 1 \omega}$. Suppose that the set Σ ($\equiv$) of first-order sentences defined weak bisimulation. Then the union of the following three sets of first-order sentences is finitely satisfiable:

(i) Σ ($\equiv$)

(ii) "there is one atomic transition $\to^{\tau}$ which is a one-to-one function t",

(iii) $x \equiv 0, \neg x \equiv t(0), \neg x \equiv tt(0), \neg x \equiv ttt(0), \ldots$

To see this, take suitably large finite chains starting at 0 with a disjoint point for x where the function t loops, putting $x \equiv 0$, $x \equiv n$:



But then, by the Compactness Theorem, there must be one model for the whole set, whose $\equiv$ cannot be a weak bisimulation, even though the purported definition $\Sigma(\equiv)$ holds. For we have $x \equiv 0$, whereas x lacks any $\equiv$-connection with points lying at some finite number of t-steps from 0. This is the required contradiction. ■

We can also extend earlier invariance results to this setting. On Δ-saturated models, everything remains basically as before. But on arbitrary LTSs, results are not quite so straightforward, because of a certain 'asymmetry' in the above definition. A restricted existential quantifier for a $\rightarrow^a$ successor step on one side need not match with a similar quantifier step on the other, because of the possible finite τ-prefixes and suffixes. Therefore, the proper format of restricted quantification here is over successors in compound relations τ^*; a; τ^*. But then, we conjecture that we will have a preservation theorem for invariance under weak bisimulation in $L_{\omega 1 \omega}$ just like the one in Section 7. Further questions in the logic of silence must be left unstated here.

10. Adding Process Structure 2: Algebraic Operations

10.1. First-Order Process Theories

Axiomatic process theory is studied mostly in an algebraic setting. We consider some versions that continue our earlier formalisms, simplifying our task by restricting attention to a very small process algebra signature (*elementary process algebra* with finitely many atomic actions). This system is a subset of the calculus CCS in Milner 1980, with a slightly modified notation:

0	P	zero process
$+$	$P \times P \rightarrow P$	alternative composition ('choice')
$a;$	$P \rightarrow P$	action prefixing, for each atomic $a \in A$.

Our discussion will focus on bisimulation semantics, omitting recursion and infinite branching. The relevant axioms are those already mentioned in Section 2.2.2:

EPA (A)
$$x + y = y + x$$
$$(x + y) + z = x + (y + z)$$
$$x + x = x$$
$$x + 0 = x$$

The initial algebra for these axioms is called A^e_ω. (This system is just one choice among various directions, including the system BPA (A) of Bergstra and Klop

1984, 1986, whose relation to transition systems is less immediate. We refer to Baeten and Bergstra 1993 for a systematic comparison, investigating both EPA-based CCS and BPA-based ACP.) Indeed, several first-order theories are relevant here. For instance, it turns out useful to introduce a further axiomatic system EPT, employing a relation of inclusion defined in the usual way:

$$x \subseteq y \Leftrightarrow x + y = y.$$

The theory EPT(A) adds the following principles to EPA(A) (these include inequalities, so that we move up in first-order complexity):

$$0 \subseteq x$$

$$(x + y) \subseteq z \quad \Leftrightarrow \quad x \subseteq z \wedge y \subseteq z$$

$$a; x \subseteq (y + z) \quad \Leftrightarrow \quad a; x \subseteq y \vee a; x \subseteq z$$

$$a; x \subseteq a; y \quad \Leftrightarrow \quad x \subseteq y \wedge y \subseteq x$$

$$\neg a; x \subseteq b; y \qquad \text{if } a, b \text{ are distinct atomic actions}$$

$$\neg a; x \subseteq 0$$

PROPOSITION. *EPT(A) is sound and complete for validity of both ground equations and inequalities in* A^e_ω.

Proof. Soundness is clear, since EPT(A) holds in A^e_ω. As for completeness, true equations are derivable in EPA(A), and hence in EPT(A), by the usual completeness theorem for equational logic. Finally, if some inequality holds in A^e_ω, then one of its inclusions must fail. Now, the EPT axioms will decompose closed terms, so as to get the following property

CLAIM. $A^e_\omega \models \neg t_1 \subseteq t_2$ *implies that EPT(A)* $\vdash \neg t_1 \subseteq t_2$.

The proof is a straightforward induction on length (t_1) + length (t_2). Finally, the latter assertion immediately implies that EPT(A) $\vdash \neg t_1 = t_2$. ■

Remark. These results can also be extended to cover process logics with silent steps. Additional axioms will include such principles as

$$a; x \subseteq \tau; y \quad \Leftrightarrow \quad a; x \subseteq y$$

$$\tau; x \subseteq \tau; y \quad \Leftrightarrow \quad (x \subseteq y \wedge y \subseteq x) \vee \tau; x \subseteq y.$$

Finally, there is nothing to prevent us from considering still more complex first-order systems, such as the full first-order theory of the initial algebra A^e_ω. Equivalently, consider the structure **G** of *finite graphs* identified *under bisimulation*, with the following operations (well-defined over representatives):

- 0 single success node
- $+$ fusion of graphs at the root
- $a;$ prefixing a single incoming a-arrow to the root.

Its first-order theory $Th(\mathbf{G})$ may actually be formulated in various ways:

PROPOSITION. *The following theories are effectively equivalent:*
(i) $Th((\mathbf{G}, 0, +, \{a; \mid a \in A\}))$
(ii) $Th((\mathbf{G}, \subseteq, \{a; \mid a \in A\}))$

Proof. Theory (ii) can be embedded into (i), as the relation $\subseteq$ has already been defined above in terms of +. Conversely, theory (i) can be embedded into (ii), by the following valid first-order equivalences in the Finite Graph Model:

$$x = 0 \leftrightarrow \forall y\, x + y = y$$

$$x = y + z \leftrightarrow \text{“}x \text{ is the supremum of } y \text{ and } z \text{ in the ordering } \subseteq\text{”}.$$ ■

This theory contains a lot of information concerning these graphs, such as a full lattice structure for the ordering $\subseteq$. Moreover, it contains some special features not found in the general case, such as

$$E \qquad \forall xy(\bigwedge_{a \in A} \forall z(x \rightarrow^a z \leftrightarrow y \rightarrow^a z) \rightarrow x = y) \qquad \text{Extensionality}$$

Indeed, $Th(\mathbf{G})$ amounts to the theory of a well-known mathematical structure:

THEOREM. *The first-order theory of the Finite Graph Model under Bisimulation is effectively equivalent to True Arithmetic.*

Proof. For simplicity, let us think of $\mathbf{G}$ as a labeled transition system consisting of equivalence classes $\ulcorner G \urcorner$ of all finite rooted graphs G under bisimulations connecting their roots. Its relational structure has identity $=$ as well as transition relations $\rightarrow^a$ (for all a in some finite set A), defined via 'root prefixing' (which is well-defined on the equivalence classes, as this relation is invariant for bisimulation). As will be shown shortly, the first-order theory $Th(\mathbf{G})$ gives us the whole process algebra structure with $\subseteq$, + and all prefix operations $a;$. As usual, True Arithmetic is the complete first-order theory of the model $(\mathrm{IN}, +, \bullet)$. Now, there are two directions to the above assertion:

From Graphs to Natural Numbers

Using any obvious effective coding scheme, the following arithmetical predicates are recursive, and hence arithmetically first-order definable:

GRAPH (n)	“n codes some finite graph”
BISIM (m, n)	“m, n code bisimilar finite graphs”
a-TRANS (m, n)	“m encodes a graph having an a-transition from its root to the root of of a generated subgraph encoded by n”

Now define the following translation * from first-order formulas about the above graphs to arithmetical ones:

$x = y$	BISIM (x, y)
$x \rightarrow^a y$	$\exists zu$: BISIM $(x, z) \wedge$ BISIM $(y, u) \wedge a-$TRANS (z, u)
$\neg, \wedge$	Boolean connectives are translated homomorphically
$\exists x$	existential quantifiers become restricted numerical quantifiers over codes of graphs: $\exists x$(GRAPH $(x) \wedge \ldots$).

The relevant reduction can be proved by a straightforward induction on formulas:

CLAIM. *For all finite graphs $G_1, \ldots, G_k$ with numerical codes $g_1, \ldots, g_k$, and all first-order formulas ϕ, the following equivalence holds*:

$$\mathbf{G} \models \phi[\ulcorner G_1 \urcorner, \ldots, \ulcorner G_k \urcorner] \text{ iff IN} \models (\phi)^*[g_1, \ldots, g_k].$$

From Natural Numbers to Graphs

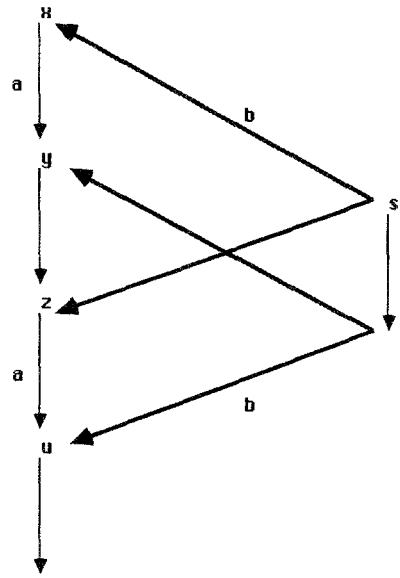
Going in the opposite direction, one needs some auxiliary first-order predicates on (equivalence classes of) finite graphs, that will help us to encode the basic numerical operations. Here is an informal explanation of the main ideas:

(1) Each natural number n may be identified with the (equivalence class of) a finite linear order $\rightarrow^a$ over $\{0, \ldots, n\}$. More precisely, one can introduce 'numerical graph objects' by defining a first-order predicate

$a-$NUM(x)	"x is a graph with just a-transitions, on which the relation $\rightarrow^a$ is transitive, irreflexive and linear".

(2) Isomorphism of two disjoint intervals on such linear graphs can be defined via the existence of some larger graph containing these intervals as subgraphs, with a second relation $\rightarrow^b$ playing the role of a bisimulation between them. More precisely, this involves the following first-order predicates:

BISIM (x, y, s)	"s has only $\rightarrow^a$- and $\rightarrow^b$-successors and $s \rightarrow^b x$, the $\rightarrow^a$-successors of s form a transitive, irreflexive linear ordering, and $\rightarrow^b$ is a bisimulation between $\rightarrow^a$-successors of s and points in the closed $\rightarrow^a$-interval $[x, y]$ with respect to the relation $\rightarrow^a$"
EQUI (x, y, z, u)	$\exists s$(BISIM $(x, y, s) \wedge$ BISIM (z, u, s)) :



One can verify in the Finite Graph Model that the latter predicate holds of two intervals on a numerical graph if and only if they have the same cardinality.

Remark. Graphs versus Equivalence Classes.
Some care is needed here, as the objects in the model **G** are not graphs, but their equivalence classes under bisimulation. Nevertheless, there is a useful strong analogy. For, the map sending finite graphs G to their equivalence classes $\ulcorner G \urcorner$ is a modal 'p-morphism' with respect to the prefix relations $\rightarrow^a$ (it is a homomorphism, and it also satisfies an existential backward clause, for both successors and predecessors). Therefore, at least for suitable 'restricted' first-order formulas (cf. Section 5 above, or van Benthem 1985), there is no difference between the two kinds of object. (The precise extent of this analogy remains to be determined for this specific graph model.)

(3) Now, one may describe addition by a first-order predicate SUM (x, y, z) which first distinguishes some marginal cases (e.g., "$x = 0$" and "$y = 0$" are trivial), and then states that x, y lie in some sequence on the linear order z, say $x \geq y$, where the top interval $[z, x]$ is equinumerous to the bottom interval on z starting from y.

(4) Multiplication TIMES (x, y, z) is encoded via a division of the linear order z into y intervals, all of equal length (this may be done by introducing a larger graph with suitable new $\rightarrow^c$ transitions marking the boundary points), and stating that each interval is equinumerous with x while the subgraph of boundary points (these are identifiable via the new relation $\rightarrow^c$) is equinumerous with y. (Again,

some trivial cases are to be treated separately.)

Finally, the relevant first-order translation $^{\#}$ can be defined as follows. Without loss of generality, the arithmetical first-order language may be restricted to the use of only atoms $x = y + z$ and $x = y \bullet z$. These will be translated via the above two predicates SUM (x, y, z) and TIMES (x, y, z). Boolean connectives are again treated homomorphically, while numerical existential quantifiers become restricted to the above predicate a-NUM defining the numerical graph equivalence classes (cf. (1)). The relevant reduction is then straightforward from the above motivation for our translating predicates:

CLAIM. *For all natural numbers $n_1, \ldots, n_k$, with corresponding graph (classe)s $N_1, \ldots, N_k$, and for all first-order formulas ϕ, the following equivalence holds*:

$$\mathrm{IN} \models \phi[n_1, \ldots, n_k] \quad \text{iff} \quad \mathbf{G} \models (\phi)^{\#}[N_1, \ldots, N_k]. \qquad \blacksquare$$

Question. What is the complexity of the first-order theory of the Finite Graph Model with only one single atomic action? In particular, does it become equivalent to Additive Presburger Arithmetic?

Another type of logical question concerns, not axiomatizability, but *definability* of process operations in these models. In the BPA/ACP tradition (as distinct from EPA/CCS), one would have an additional product operation on graphs, corresponding to full product of processes. This operation can be defined inside the above first-order theory, but there remains this

Question. Can full product be explicitly defined in terms of just the above signature of elementary process algebra?

We conjecture that the answer is negative, which would highlight the difference between the two mentioned traditions. Interestingly, the above theory also contains other graph operations, not found so far in process algebra. For instance, given the above lattice structure for $\subseteq$ on graphs, there is not just a *supremum* +, but also a natural *infimum* * dual to it, whose algebraic behaviour might be worth investigating.

10.2. Process Algebras and Labeled Transition Systems

Here is how one can return to the original perspective of this paper. Any model M for the earlier process theories generates a labeled transition system **TS** (M) by the following stipulation:

$$x \rightarrow^a y \;\Leftrightarrow\; \exists z\, x = a; y + z.$$

At this point, several open questions of definability and axiomatizability arise, similar to those encountered in earlier sections. We mention a few:

- Is the class **TS** (MOD (EPA (A)) EC or EC_Δ?
- Is the class **TS** (MOD (EPT (A)) EC or EC_Δ?

Our conjecture is that all answers are negative. Nevertheless, both these classes are Σ^1_1-definable in second-order logic, as their defining condition states the existence of some predicates or functions in an LTS satisfying the finite lists of EPA or EPT axioms. By an earlier logical observation in Section 4, this implies at least recursive axiomatizability for their first-order theories. Hence, we have this further question:

- To axiomatize the first-order theories of the above classes explicitly.
- Are they even finitely axiomatizable?

Here are some partial results. Analyzing the signature $(0, +, \{a; \mid a \in A\})$ through the above stipulation, we get the following first-order principles:

A_1 $\exists x \bigwedge_{a \in A} \neg \exists y : x \rightarrow^a y$ End-points

A_2 $\forall xy \exists z \bigwedge_{a \in A} \forall u (z \rightarrow^a u \leftrightarrow (x \rightarrow^a u \vee y \rightarrow^a u))$ Sums

A_3 $\forall x \bigwedge_{a \in A} \exists y (\forall z (y \rightarrow^a z \leftrightarrow z = x) \wedge$
$\bigwedge_{b \in A, b \text{ distinct from } a} \neg \exists z\, y \rightarrow^b z)$ Predecessors

PROPOSITION. **TS** (A^e_ω) *verifies* A_1, A_2, A_3, *as well as Extensionality* (E).

More generally, we have the following property:

PROPOSITION. *MOD* $(\{A_1, A_2, A_3, E\}) \subseteq$ **TS** *(EPA).*

Proof. By a simple deduction, each of these four principles follows from the axioms of EPT under the above stipulation. Conversely, given the above principles, one can introduce obvious operations of 'sum' and 'action prefixing', which turn out to have the right properties by direct verification. For instance, Associativity follows because the two ways of computing a 'sum' of three objects lead to points having the same successors, which must therefore be identical by Extensionality. ■

Remark. The theory $\{A_1, A_2, A_3, E\}$ encodes much of elementary process algebra. Without Extensionality, however, we do not have obvious uniqueness for the process operations induced by our LTS – and only some partial results have been obtained so far, whose formulation we omit here.

Essentially, the above questions ask to which extent the richer structure of a process algebra is already available within its derived transition system. From the earlier full first-order perspective, this will often be the case, witness our final observation:

PROPOSITION. *The following first-order theories are effectively equivalent for the model* $M = (G, 0, +, \{a; \mid a \in A\})$:
(i) $Th(M)$
(ii) $Th(\mathbf{TS}(M))$.

Proof. this is implicit in the argument for the previous Proposition. ■

11. Further Directions

The purpose of this paper has been to put standard logics of transition systems on the map as a computational formalism, to prove a number of technical results showing their theoretical interest, and finally, to suggest some main directions for their further investigation. Of the many general questions that arise in this way (in addition to the many specific pointers in our text), we mention the following:

(1) To develop the practical application of first-order and infinitary formalisms.
(2) To explore computational consequences of other aspects of their known theory, such as cut-free proof methods or interpolation.
(3) To generalize our results in the first-order case systematically to special kinds of LTSs, and to different formalisms (infinitary, or state-action languages).
(4) To carry our analysis to the case of process algebra with further operations such as various merges, recursion, or encapsulation.

References

Baeten, J. and Bergstra, J.A., 1993, 'On Sequential Composition, Action Prefixes and Process Prefix', Departments of Computer Science, University of Amsterdam and University of Eindhoven.

Bergstra, J.A. and Klop, J.-W., 1984, 'Process Algebra for Synchronous Communication', *Information and Control* **60**, 109–137.

Bergstra, J.A. and Klop, J.-W., 1986, 'Process Algebra: Specification and Verification in Bisimulation Semantics', in *Mathematics and Computer Science II*, M. Hazewinkel *et al.*, eds., North-Holland, Amsterdam, pp. 61–94.

Bergstra, J.A. and Klop, J.-W., 1988, 'A Complete Inference System for Regular Processes with Silent Moves', in *Logic Colloquium '86*, F. Drake and J. Truss, eds., North-Holland, Amsterdam, pp. 21–81.

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

De Jongh, D. and Troelstra, A., 1966, 'On the Connection of Partially Ordered Sets with Some Pseudo-Boolean Algebras', *Indagationes Mathematicae* **28**, 317–329.

De Nicola, R. and Vaandrager, F., 1990, 'Three Logics of Branching Bisimulation', in *Proceedings 5th LICS Conference*, Computer Society Press, pp. 118–129.

Doets, K., 1987, *Completeness and Definability. Applications of the Ehrenfeucht Game in Second-Order and Intensional Logic*, Dissertation, Mathematical Institute, University of Amsterdam.

Doets, K., 1993, *Model Theory*, Lecture Notes for the Fifth European Summer School in Logic, Language and Information, University of Lisbon.

Goldblatt, R., 1982, *Axiomatizing the Logic of Computer Programming*, Springer, Berlin.
Gurevich, Y., 1985, 'Logic and the Challenge of Computer Science', Computing Research Laboratory, The University of Michigan, Ann Arbor.
Harel, D., 1984, 'Dynamic Logic', in *Handbook of Philosophical Logic*, Vol. II, D. Gabbay and F. Guenthner, eds., Reidel, Dordrecht, pp. 497–604.
Hennessy, M. and Milner, R., 1985, 'Algebraic Laws for Nondeterminism and Concurrency', *Journal of the ACM* **32**, 137–161.
Immerman, N. and Kozen, D., 1987, 'Definability with Bounded Number of Bound Variables', in *Proceedings IEEE 1987*, pp. 236–244.
Keisler, H.J., 1971, *Model Theory for Infinitary Logic*, North-Holland, Amsterdam.
Keisler, H.J., 1977, 'Fundamentals of Model Theory', in *Handbook of Mathematical Logic*, J. Barwise, ed., North-Holland, Amsterdam, pp. 47–103.
Milner, R., 1980, *A Calculus of Communicating Systems*, Springer, Berlin.
Park, D., 1981, 'Concurrency and Automata on Infinite Sequences', in *Proceedings 5th GI Conference*, Springer, Berlin, pp. 167–183.
Salwicki, A., 1970, 'Formalised Algorithmic Languages', *Bulletin Polish Academy of Sciences*, Series Sci. Math. Astr. Phy., Vol. 18, pp. 227–232.
Stirling, C., 1989, 'Modal and Temporal Logics', to appear in *Handboook of Logic in Computer Science*, S. Abramsky *et al.*, eds., Oxford University Press, Oxford.
Van Benthem, J., 1976, *Modal Correspondence Theory*, Dissertation, Mathematical Institute, University of Amsterdam.
Van Benthem, J., 1985, *Modal Logic and Classical Logic*, Bibliopolis, Napoli.
Van Benthem, J., 1991, *Language in Action. Categories, Lambdas and Dynamic Logic*, North-Holland, Amsterdam.
Van Benthem, J., 1993, 'Which Program Constructions Are Safe for Bisimulation?', Institute for Logic, Language and Computation, University of Amsterdam. (To appear in D. Richard et al., eds., Logic Colloquium. Clermont-Ferrand 1994, Elsevier Science Publishers, Amsterdam).
Van Benthem, J., van Eyck, J. and Stebletsova, V., 1993, 'Modal Logic, Transition Systems and Processes', Centre for Mathematics and Computer Science (CWI), Amsterdam. (To appear in the Journal of Logic and Computation).