Bisimulation Respecting First-Order Operations.

Marco Hollenberg
Utrecht University, Department of Philosophy
Heidelberglaan 8, 3584 CS Utrecht, the Netherlands
hollenb@phil.ruu.nl
http://www.phil.ruu.nl/home/marco/

Abstract

We identify two features of common process algebra operations: their first-order flavour and the fact that they respect bisimulation in a uniform manner. For this purpose two notions are introduced: first, a notion of *first-order definable operations on process graphs* and second, *respect for sequence extension*. In the first part of the paper those first-order definable operations are characterised whose defining formulas respect sequence extension. The second part uses the resulting format to calculate *modal preconditions*: it gives an algorithm that reduces modal truth in the output of certain process graph operations to modal truth in the inputs of this operation.

1 Introduction

This paper fits into the recent tradition ([5, 6, 8]) of attempts to apply modal logical techniques to process algebra ([2]). These attempts are motivated by two observations. First, rooted transition systems (or *process graphs*) play a role in both process algebra and modal logic. In process algebra, process graphs are the objects of the so-called *graph model* ([2]). On the modal side, they appear as models for polymodal languages. The second observation is that *(strong) bisimulation* ([3, 19]) is an important concept in both fields. In process algebra, bisimulation plays the role of process equivalence. Admittedly, process algebra has a whole spectrum of process equivalences, but strong bisimulation is one of the most basic among these. In modal logic, bisimulation plays another role: it does for modal logic what partial isomorphism does for first-order logic ([20]). In particular, modal formulas are *invariant* for bisimulation.

What can be done with these connections between the fields of process algebra and modal logic? The most obvious is to use invariance for bisimulation of modal formulas to provide modal logical *witnesses* ([10]) to nonbisimilarity: in certain classes of process graphs (such as the class of finite ones), whenever two points are not bisimilar there will be a modal formula true at one point, but not at the other. Such witnesses are useful for telling us *why* certain processes are not equivalent.

Van Benthem, in a draft paper ([6]), takes another path from modal logic to process algebra. He discusses the possibility of defining process operations (i.e. operations on process graphs) by means of formulas from the first-order part of dynamic logic (basically just some sort of modal logic). In his paper various operations are dealt with, such as action-prefix and alternative composition, but others, such as direct product and merge, where more structured states are needed, remain elusive to his approach. The present paper is an attempt to extend or alter this approach so that these more elaborate operations can also be handled.

Besides this we will also present an algorithm that, given the definition of an operation O in a certain format (sufficiently broad to cover the well-known ACP operations, cf. [9]) and a modal formula ϕ, reduces the modal truth of ϕ at the root of $O(\mathfrak{S}_1, \ldots, \mathfrak{S}_m)$ to modal truth at the roots of the input graphs, irrespective of the particular $\mathfrak{S}_1, \ldots, \mathfrak{S}_m$ that serve as input to the operation. we say that the algorithm calculates *modal preconditions*. This answers a question posed in [8].

The proof of the main theorem of this paper (theorem 5.3) is quite similar to the proof of the main theorem in [17]. The latter paper expands the notion of *safety* for bisimulation (see [5]) and gives a characterisation theorem for first-order formulas satisfying this notion. As the concerns there are purely modal, we have chosen to write two separate papers.

Notational Conventions:
Throughout this paper we will often introduce a first-order formula as: 'Consider a formula $\phi(x_1, \ldots, x_n)$ ...'. By this we mean: 'Consider a formula ϕ, whose free variables occur among $x_1, \ldots, x_n$. Later in the text we may refer to this formula just by ϕ. If ϕ has been introduced in this fashion, $\phi(y_1, \ldots, y_n)$ will be used to denote the formula we get by first renaming all bound variables in ϕ such that $x_1, \ldots, x_n, y_1, \ldots, y_n$ do not occur as bound formulas in ϕ and then simultaneously substituting x_1 for y_1, x_2 for y_2 etc. Finally, if $\mathcal{M}$ is some appropriate model, then $\mathcal{M} \models \phi[e_1, \ldots, e_n]$ means that ϕ holds in $\mathcal{M}$ under the assignment that sends x_i to e_i. □

2 Process operations

A common way of representing nondeterministic processes is by means of process graphs. Process graphs can be defined in different ways, all of which are interchangeable. We prefer a modal logic perspective on them, and equate the term 'process graph' with 'rooted polymodal Kripke model'. That is, a process graph is a structure of the form:

$$\mathfrak{S} = (S, \stackrel{a}{\rightarrow}, V, s)_{a \in A}$$

where S is any nonempty set, A is a (possibly empty) set of labels, $\stackrel{a}{\rightarrow}$ is a binary relation on S for every $a \in A$, $s \in S$ and V is a *valuation*, a function from a set PROP $= \{p, q, r, p_0, p_1, \ldots\}$ of propositional variables to subsets of S.

Such a Kripke model should be viewed as follows: S is some domain of states, say assignments to internal variables of some machine. $x \stackrel{a}{\rightarrow} y$ if the *atomic action* 'a' can bring us from state x to state y. Note that, as we are modelling nondeterministic processes, $\stackrel{a}{\rightarrow}$ need not be functional. The state s is the *initial state*, or *root*, of the process: it is the state where the computation commences. Finally PROP is a set of predicates about states. $p \in$ PROP is assumed to be true of a state x if $x \in V(p)$. For the purposes of this paper, we need not be specific about PROP, but in examples we will always assume it to be a singleton set $\{\surd\}$. $\surd$ is intended to be true only at those states at which the process may *terminate*.

Process graphs may also be seen as first-order models for a language $\mathcal{L}$ consisting of a constant s, binary relation-symbols R_a for every $a \in A$ and a unary predicate P for every $p \in$ PROP. A rooted Kripke model such as the above $\mathfrak{S}$ can be transformed into a model $\mathcal{M}$ for this language as follows. The universe of $\mathcal{M}$ is S, while R_a, s and P are interpreted by respectively $\stackrel{a}{\rightarrow}$, s and $V(p)$. Conversely, a model $\mathcal{M}$ for $\mathcal{L}$ is easily transformed into a rooted Kripke model: simply define $V(p) := P^{\mathcal{M}}$.

The reason we equate process graphs with Kripke models is that we will have cause to evaluate *modal formulas* at points in process graphs. With 'modal formulas' we mean the set defined as follows:

$$\phi ::= \top \mid p \mid \langle a \rangle \phi \mid \phi \vee \psi \mid \neg \phi$$

where $a \in A$ and $p \in$ PROP. We assume the reader is familiar with the truth definition that comes with these formulas, as well as with standard abbreviations such as $[a]\phi := \neg\langle a \rangle \neg \phi$. We write $\mathfrak{S}, s \Vdash \phi$, or simply $s \Vdash \phi$ for 'ϕ is true at the point s'.

The literature of process algebra (for a comprehensive introductory book, see [2]) abounds with examples of *operations* on process graphs: operations that on input of some sequence of process graphs produce another process graph. Examples include sequential composition, alternative composition, iteration and merge with or without communication.

Most process operations seem to have a certain first-order quality to them. Van Benthem ([5]) contains tentative steps towards exploiting this first-order flavour. In this paper we will expand upon

Van Benthem's work. We will study a notion of *first-order definable operation*, making it applicable to a wider class of operations than is possible in Van Benthem's approach.

What *is* a first-order definable operation? It is an operation that describes the output model in terms of the input models by means of first-order formulas in some language. To be able to talk about both the input models and the output model at the same time with the aid of first-order formulas, we need some larger model (which we will call a *superstructure*) in which all these other models can be interpreted. This is where we need to make choices: we need to pick a suitable language for the superstructure and we need to specify the superstructure itself.

Definition 2.1 *Given some finite set* CONS *of fresh constants and an* $m \in \mathbb{N}$*, let* $\mathcal{L}_m$ *be the language signature containing:*

1. *Unary predicates* S_i*, for* $1 \leq i \leq m$*.*
2. *A unary predicate* P_i *(*$1 \leq i \leq m$*) for each predicate* P *corresponding to some* $p \in$ PROP*.*
3. *Binary relation-symbols* $R_{(a,i)}$ *(*$a \in A$ *and* $1 \leq i \leq m$*).*
4. *Constants* s_i *and* c *(*$1 \leq i \leq m$*,* $c \in$ CONS*).*
5. *A ternary relation-symbol* C *(for 'concatenation').*
6. *Identity* $=$*.* □

Definition 2.2 (Superstructure) *Let* $\mathfrak{S}_i$ *be disjoint rooted Kripke models, for* $1 \leq i \leq m$*. Fix a finite set of constants* CONS *(disjoint from the domains* S_i*). Then* $\mathcal{M} = \mathbb{S}^n(\mathfrak{S}_1, \ldots, \mathfrak{S}_m)$ *is the following* $\mathcal{L}_m$*-model:*

1. *Universe: all sequences* σ *of elements of* $S_1 \cup \ldots \cup S_m \cup$ CONS*, such that the length* $|\sigma|$ *of the sequence is at most* n*. We denote the empty sequence by* λ *and sequence concatenation by* $\cdot$*. However,* $\sigma \cdot \tau$ *will often be simply denoted by* $\sigma\tau$*. Also, a sequence of length one is identified with the unique element in the sequence.*
2. $\mathcal{L}_m$ *is interpreted within* $\mathcal{M}$ *as:*

$$P_i^{\mathcal{M}} := P^{\mathfrak{S}_i}. \qquad S_i^{\mathcal{M}} := S_i.$$
$$R_{(a,i)}^{\mathcal{M}} := R_a^{\mathfrak{S}_i}. \qquad s_i^{\mathcal{M}} := s_i.$$
$$c^{\mathcal{M}} := c, \text{ for } c \in \text{CONS}. \qquad C^{\mathcal{M}}(u,v,w) \text{ iff } u = v \cdot w.$$

To make this definition correct, we need to assume that $n \geq 1$ *if* $m \geq 1$ *or* CONS $\neq \emptyset$*.* □

Clearly each $\mathfrak{S}_i$ $(1 \leq i \leq m)$ 'lives' inside the model $\mathcal{M} = \mathbb{S}^n(\vec{\mathfrak{S}})$: $\mathfrak{S}_i \cong (S_i^{\mathcal{M}}, R_{(a,i)}^{\mathcal{M}}, V, s_i^{\mathcal{M}})$, where $V(p) = P_i^{\mathcal{M}}$. So input graphs can be interpreted in the superstructure.

The class $\mathsf{SUP} = \{\mathcal{M} \mid \text{there are } \mathfrak{S}_1, \ldots, \mathfrak{S}_m \text{ such that } \mathcal{M} \cong \mathbb{S}^n(\mathfrak{S}_1, \ldots, \mathfrak{S}_m)\}$ (given some finite set CONS) is a basic elementary one: there exists a first-order sentence ϕ_{SUP} such that $\mathcal{M} \models \phi_{\mathsf{SUP}}$ iff $\mathcal{M}$ is isomorphic to some $\mathbb{S}^n(\vec{\mathfrak{S}})$. We will say that two formulas ϕ and ψ are SUP-equivalent iff $\mathsf{SUP} \models \phi \leftrightarrow \psi$. Similarly, notions of consequence may be relativised to SUP. Note that SUP-consequence still satisfies compactness, as SUP is first-order definable. This would no longer be the case if we allowed sequences of arbitrary finite length in our superstructures.

From now on, whenever we mention the language $\mathcal{L}_m$, we will assume that either the set CONS is clear from context, or that it is some fixed set. We will also often drop the superscript n from $\mathbb{S}^n$, assuming it fixed, or clear from context.

Definition 2.3 *Given* CONS$, n$ *and* m*, a* **first-order definable operation** O *is given by:*

1. *A first-order* $\mathcal{L}_m$*-formula* $\phi_a(x,y)$ *in two variables, for every* $a \in A$*.*

2. *A first-order $\mathcal{L}_m$-formula $\phi_p(x)$ for every $p \in$ PROP.*

3. *A sequence $\sigma = t_1 \dots t_k$, where each t_j $(1 \leq j \leq k)$ is a constant of $\mathcal{L}_m$ (so either $t_j \in$ CONS, or $t_j = s_i$ for some $i = 1, \dots, m$) and $k \leq n$.* □

Given process graphs $\mathfrak{S}_1, \dots, \mathfrak{S}_m$, the process graph $\mathfrak{T} = O(\mathfrak{S}_1, \dots, \mathfrak{S}_m)$ is defined as follows. Its domain is the same as the superstructure $\mathcal{M} = \mathbb{S}(\vec{\mathfrak{S}})$. Furthermore:

$$\begin{array}{rcl} R_a^{\mathfrak{T}} & := & \{(u,v) \in M^2 \mid \mathcal{M} \models \phi_a[u,v]\} \\ V^{\mathfrak{T}}(p) & := & \{u \in M \mid \mathcal{M} \models \phi_p[u]\} \\ s^{\mathfrak{T}} & := & t_1^{\mathcal{M}} \dots t_k^{\mathcal{M}}. \end{array}$$

where M is the universe of $\mathcal{M}$.

Let us give a few examples to show that most familiar process operations are first-order definable.

Sequential Composition $(\cdot)$ is defined with $m = 2$ (it is a binary operation), CONS $= \emptyset$ and $n = 1$. The defining formulas and constant are:

$$\begin{array}{lcl} \phi_a(x,y) & := & xR_{(a,1)}y \vee xR_{(a,2)}y \vee (\surd_1(x) \wedge s_2 R_{(a,2)} y) \\ \phi_{\surd}(x) & := & \surd_2(x) \vee (\surd_1(x) \wedge \surd_2(s_2)) \\ \sigma & := & s_1 \end{array}$$

Alternative Composition $(+)$ is defined with $m = 2$, $n = 1$ and CONS $= \{c\}$.

$$\begin{array}{lcl} \phi_a(x,y) & := & xR_{(a,1)}y \vee xR_{(a,2)}y \vee [x = c \wedge (s_1 R_{(a,1)} y \vee s_2 R_{(a,2)} y)] \\ \phi_{\surd}(x) & := & \surd_1(x) \vee \surd_2(x) \vee [x = c \wedge (\surd_1(s_1) \vee \surd_2(s_2))] \\ \sigma & := & c \end{array}$$

Free Merge $(\|)$ is defined with $m = 2$, $n = 2$ and CONS $= \emptyset$.

$$\begin{array}{lcl} \phi_a(x,y) & := & \exists x_1 x_2 y_1 y_2. \left(\begin{array}{l} C(x,x_1,x_2) \wedge C(y,y_1,y_2) \wedge \\ ([x_1 R_{(a,1)} y_1 \wedge x_2 = y_2] \vee [x_1 = y_1 \wedge x_2 R_{(a,2)} y_2]) \end{array} \right) \\ \phi_{\surd}(x) & := & \exists x_1 x_2. (C(x,x_1,x_2) \wedge \surd_1(x_1) \wedge \surd_2(x_2)) \\ \sigma & := & s_1 s_2 \end{array}$$

To complete the picture, we will mention some natural process operations that are *not* definable in our setting.

First, first-order definable operations always have a finite arity (namely m). *Infinitary* operations like the infinite sum (or infinite alternative composition) of μCRL ([13]), do therefore not fit within our framework.

Next, we give an example of a unary (thus finitary) operation that is not first-order definable. We say that a process graph $\mathfrak{S}$ *terminates* if there is a path from the root to a success-state, i.e. a path $s \xrightarrow{a_1} s_1 \dots \xrightarrow{a_k} s_k$ such that $s_k \Vdash \surd$. Define TERMINATE as follows:

$$\text{TERMINATE}(\mathfrak{S}) := \left\{ \begin{array}{ll} \varepsilon & \text{if } \mathfrak{S} \text{ terminates} \\ \delta & \text{else} \end{array} \right.$$

where ε is the process graph consisting of just a root, which is also a success-state, and no arrows, and δ is similar, except here the root is not a success-state. We sketch the proof that this operation is not first-order definable on page 29.

As we've mentioned before, [6] also contains a notion of first-order definable operations. There, process graphs are identified with states in a transition system. Then it is possible to state what it means for a state to be (for instance) the alternative composition of two other states. An operation like the merge, however, seems to require the possibility of speaking about sequences of states (or pairs, or multisets...). This is precisely what the current framework offers.

3 Respecting bisimulation

Many notions of *process equivalence* exist (see [2], [7]), but strong bisimulation (which we will simply call *bisimulation* in this paper) rates high among them.

Definition 3.1 *A* **bisimulation** Z *between two process graphs* $\mathfrak{S} := (S, R_a, V, s)_{a\in A}$ *and* $\mathfrak{T} := (T, R'_a, V', t)_{a\in A}$ *is a relation between* S *and* T *such that:*

1. sZt.
2. *The* **zig-condition***: if* xZx' *and* xR_ay *then there is a* $y' \in T$ *with* $x'R'_ay'$ *and* yZy'.
3. *The* **zag-condition***: if* xZx' *and* $x'R'_ay'$ *then there is a* $y \in S$ *with* xR_ay *and* yZy'.
4. *If* xZx' *then* [$x \in V(p)$ *iff* $x' \in V'(p)$].

'Z *is a bisimulation between* $\mathfrak{S}$ *and* $\mathfrak{T}$*' is denoted by* $Z : \mathfrak{S} \leftrightarrows \mathfrak{T}$*. When such a* Z *exists we call* $\mathfrak{S}$ *and* $\mathfrak{T}$ **bisimilar***: we write* $\mathfrak{S} \leftrightarrows \mathfrak{T}$*. A bisimulation that is a total function is called a* **zigzagmorphism**. □

Taking bisimulation as our notion of equivalence, process operations should not be able to distinguish between bisimilar process graphs. That is, if O is an m-ary process operation and whenever $\mathfrak{S}_i \leftrightarrows \mathfrak{T}_i$ for every $i \in \{1, \ldots, m\}$ then $O(\vec{\mathfrak{S}})$ should be bisimilar to $O(\vec{\mathfrak{T}})$. When this is true for an operation, we say that the operation **respects bisimulation**.

From now on, we will only be interested in operations **modulo bisimulation**. So when we say that an m-ary operation O is not first-order definable, we mean that for no first-order definable operation O' is it the case that $O(\mathfrak{S}_1, \ldots, \mathfrak{S}_m) \leftrightarrows O'(\mathfrak{S}_1, \ldots, \mathfrak{S}_m)$ for every sequence of process graphs $\mathfrak{S}_1, \ldots, \mathfrak{S}_m$. Even in this stronger sense, the unary operation TERMINATE of the previous section is not first-order definable. But the standard modal operation of unravelling (see also definition 5.4) *is* first-order definable in this sense, as it is the same as the identity operator, modulo bisimulation.

It is clear that not all first-order definable operations respect bisimulation. A natural question is therefore: which *do*?

Inspection of common process operations reveals that these not only respect bisimulation, but the bisimulations between the inputs can be transformed into bisimulations between the outputs in a uniform way.[1] For example, consider the definitions of the operations $\cdot$, $+$ and $\|$ as in the previous section. If $Z_i : \mathfrak{S}_i \leftrightarrows \mathfrak{T}_i$ (for $i = 1, 2$) then:

1. $Z_1 \cup Z_2 : (\mathfrak{S}_1 \cdot \mathfrak{S}_2) \leftrightarrows (\mathfrak{T}_1 \cdot \mathfrak{T}_2)$.
2. $Z_1 \cup Z_2 \cup \{(c, c)\} : (\mathfrak{S}_1 + \mathfrak{S}_2) \leftrightarrows (\mathfrak{T}_1 + \mathfrak{T}_2)$.
3. $\{(u_1u_2, v_1v_2) \mid u_1Z_1v_1 \text{ and } u_2Z_2v_2\} : (\mathfrak{S}_1 \parallel \mathfrak{S}_2) \leftrightarrows (\mathfrak{T}_1 \parallel \mathfrak{T}_2)$.

Extrapolating a pattern, we define the following:

Definition 3.2 *Let* $Z_i : \mathfrak{S}_i \leftrightarrows \mathfrak{T}_i$ *(for* $1 \leq i \leq m$*) be bisimulations. Then their* **sequence extension** [2] *is the smallest relation between* $\mathbb{S}(\vec{\mathfrak{S}})$ *and* $\mathbb{S}(\vec{\mathfrak{T}})$ *such that:*

1. $Z_i \subseteq Z$*, for every* $i \in \{1, \ldots, m\}$.
2. $(c, c) \in Z$*, for every* $c \in$ CONS.
3. $(\lambda, \lambda) \in Z$.

[1] This uniformity is also remarked upon in [5].

[2] This notion is inspired by the notion of *bisimulation extension* in [6], where only extensions to fresh constants are considered. 'Sequence and constant extension' would be a more accurate name, but somewhat lengthy.

4. If $(u_1, u_2), (v_1, v_2) \in Z$ *and* $|u_1 v_1| \leq n$ *then* $(u_1 v_1, u_2 v_2) \in Z$. □

In the above definition we assume of course that the operation $\mathbb{S}$ is given relative to some fixed set CONS and a fixed $n \in \mathbb{N}$. Thus $\mathbb{S}(\vec{\mathfrak{S}})$ and $\mathbb{S}(\vec{\mathfrak{T}})$ both have sequences of length at most n and they both contain the same fresh constants.

The sequence extension Z as in the definition above connects two sequences only when they are of the same length. Two sequences $x_1 \dots x_k$ and $y_1 \dots y_k$ are connected precisely when $x_i Z y_i$ for each $i \in \{1, \dots, k\}$. In turn, these singleton sequences are connected precisely when they are already connected by one of the bisimulations Z_i, or when both of them are the same element c of CONS (recall that elements of CONS are interpreted as themselves in superstructures).

We say that a first-order definable operation O **respects sequence extension** if, whenever we have bisimulations $Z_i : \mathfrak{S}_i \underline{\leftrightarrow} \mathfrak{T}_i$ (for $1 \leq i \leq m$) and Z is their sequence extension, then $Z : O(\vec{\mathfrak{S}}) \underline{\leftrightarrow} O(\vec{\mathfrak{T}})$. Clearly first-order operations that respect sequence extension must respect bisimulation. We can thus fine-tune our main problem a little. The question now is: which first-order definable operations respect sequence extension?

The solution to this problem is still out of our reach. But the notion of sequence extension suggests some other problems.

Let us call a formula $\phi(x)$ of $\mathcal{L}_m$ **invariant for sequence extension** if whenever we have bisimulations $Z_i : \mathfrak{S}_i \underline{\leftrightarrow} \mathfrak{T}_i$ then their sequence extension Z preserves ϕ. That is, Z satisfies the zig- and zag-conditions with respect to ϕ:

If uZv then $\mathbb{S}(\vec{\mathfrak{S}}) \models \phi[u]$ iff $\mathbb{S}(\vec{\mathfrak{T}}) \models \phi[v]$.

A question we may ask is: which formulas are invariant for sequence extension? This is akin to asking which first-order formulas are invariant for bisimulation (see Van Benthem [4]).

Let us call a formula $\phi(x, y)$ of $\mathcal{L}_m$ **safe for sequence extension** if whenever we have bisimulations $Z_i : \mathfrak{S}_i \underline{\leftrightarrow} \mathfrak{T}_i$ then their sequence extension Z is a bisimulation for the relation defined by ϕ:

If uZv and $\mathbb{S}(\vec{\mathfrak{S}}) \models \phi[u, u']$ then there is a v' such that $u'Zv'$ and $\mathbb{S}(\vec{\mathfrak{T}}) \models \phi[v, v']$.

And if uZv and $\mathbb{S}(\vec{\mathfrak{T}}) \models \phi[v, v']$ then there is a u' such that $u'Zv'$ and $\mathbb{S}(\vec{\mathfrak{S}}) \models \phi[u, u']$.

We could ask ourselves which formulas are safe for sequence extension. Again, this is similar to the safety-question for bisimulation as stated in Van Benthem [5].

Some elementary connections between safe and invariant formulas are the following. If $\phi(x)$ is invariant, then $\phi \wedge x = y$ is safe. If $\phi(x, y)$ is safe then $\exists y.\phi$ is invariant. Thus, if Γ is the set of safe formulas, then:

$$\Delta = \{\psi(x) \mid \phi \in \Gamma \text{ and } \models \psi \leftrightarrow \exists y.\phi\}$$

is the set of invariant formulas. For if $\phi(x)$ is invariant then $\phi \wedge x = y \in \Gamma$, so $\exists y.(\phi \wedge x = y)$ is in Δ and this formula is equivalent to ϕ.

If we define a first-order operation by means of formulas $\phi_a(x, y)$ which are safe for sequence extension and formulas $\phi_p(x)$ which are invariant for sequence extension (let us call such an operation **safe/invariant**), we get an operation that respects sequence extension, as can easily be verified. The converse does not hold, however: a first-order operation that respects sequence extension may use formulas in its definition that are not invariant or safe. As an example consider the operation O defined as follows:

- Let $A = \{a, b\}$, $n = m = 1$, CONS $= \{c\}$.
- $\phi_a = \phi_b := xR_{(a,1)}y \wedge xR_{(b,1)}y$.
- $\phi_{\vee} := S_1(x) \wedge \neg(xR_{(a,1)}x)$.
- $\sigma := c$.

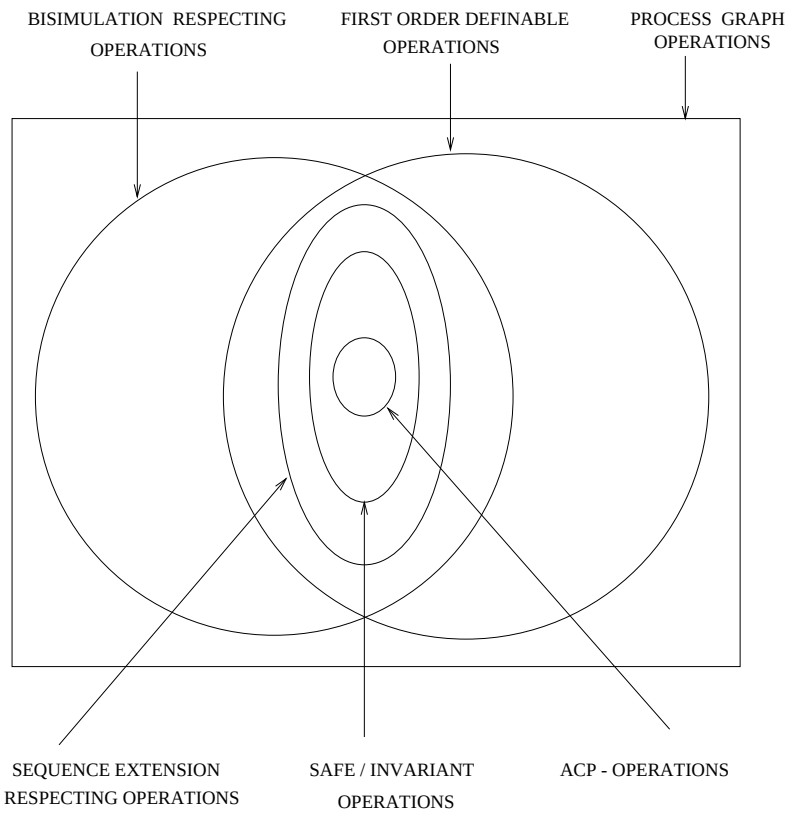


Figure 1: Classes of process operations.

Clearly this operation sends any model to a model (modulo bisimulation) consisting of a single point c, where $\surd$ is false and no arrows occur (i.e. the δ we've seen before). Furthermore, it respects sequence extension, as any sequence extension connects c to c. But *none* of the defining formulas are safe or invariant.

Maybe this is not such a good example however, as we can easily make a safe/invariant operation that is the same as O, modulo bisimulation. Simply let $\phi_a = \phi_b = \phi_{\surd} := \bot$ and $\sigma := c$. An open question is the following:

> Is there an operation O that respects sequence extension such that it is not safe/invariant, not even modulo bisimulation?

Figure 1 depicts the situation we are in. First of all, we are concerned with operations on process graphs. From a process algebra point of view, the subclass of bisimulation respecting operations is of interest. We have introduced the class of first-order definable operations. Within the intersection of the last two classes we find another class we have discussed: the class of first-order definable operations that respect sequence extension. This contains the subclass of operations whose defining formulas are safe or invariant. *This* is the class that we will characterise in the current paper. Finally, to convince process algebraists of the usefulness of the safe/invariant format, we mention that the operations of ACP are without exception safe/invariant (see the examples on page 14).

4 Invariance for sequence extension

In this section we will characterise the subclass of formulas that are invariant for sequence extension, in a way similar to the characterisation of formulas that are invariant for bisimulation. In the section after this we characterise the set of safe formulas. As remarked upon in the previous section, a

characterisation of the safe formulas automatically yields one for the invariant formulas, so the reader may wonder: why bother? First, the characterisation of the invariant formulas is much easier than that of the safe formulas, so it may serve as a warming-up for the next section. Second, the proof of the present section uses lemmas which we will need in the next section, so from a modularity point of view this section is also not superfluous.

Definition 4.1 *MOD_m is the following (modal) language:*

$$\phi ::= \top \mid p_i \mid S_i \mid c \mid \lambda \mid \langle a, i\rangle\phi \mid \phi \vee \phi \mid \neg\phi \mid GOTO_{s_i}(\phi) \mid \phi \bullet \phi$$

where $a \in A$, $1 \leq i \leq m$, $c \in$ CONS and $p \in$ PROP. □

Formulas in MOD_m are interpreted at points in superstructures: this is why we call it a *modal* language. In the following truth definition, let $\mathcal{M}$ be the superstructure $\mathbb{S}(\vec{\mathfrak{S}})$ and u some element of this superstructure.

$\mathcal{M}, u \Vdash \top$		always.
$\mathcal{M}, u \Vdash p_i$	iff	$\mathfrak{S}_i, u \Vdash p$.
$\mathcal{M}, u \Vdash S_i$	iff	$u \in S_i$.
$\mathcal{M}, u \Vdash c$	iff	$u = c$.
$\mathcal{M}, u \Vdash \lambda$	iff	$u = \lambda$.
$\mathcal{M}, u \Vdash \langle a, i\rangle\phi$	iff	there is a v such that $u \stackrel{a}{\rightarrow} v$ in $\mathfrak{S}_i$ and $\mathcal{M}, v \Vdash \phi$.
$\mathcal{M}, u \Vdash \phi \vee \psi$	iff	$\mathcal{M}, u \Vdash \phi$ or $\mathcal{M}, u \Vdash \psi$.
$\mathcal{M}, u \Vdash \neg\phi$	iff	$\mathcal{M}, u \nVdash \phi$.
$\mathcal{M}, u \Vdash GOTO_{s_i}(\phi)$	iff	$\mathcal{M}, s_i \Vdash \phi$.
$\mathcal{M}, u \Vdash \phi \bullet \psi$	iff	$u = v \cdot w$ such that $\mathcal{M}, v \Vdash \phi$ and $\mathcal{M}, w \Vdash \psi$.

In the future we will write simply $u \Vdash \phi$, if the intended superstructure is clear from the context.

From this truth definition it can easily be seen that MOD_m is nothing but a fragment of first-order $\mathcal{L}_m$. We can give a *standard translation* $(\_)^\circ$ from MOD_m to first-order $\mathcal{L}_m$-formulas in a single free variable x such that:

$$\mathbb{S}(\vec{\mathfrak{S}}), u \Vdash \phi \quad \text{iff} \quad \mathbb{S}(\vec{\mathfrak{S}}) \models (\phi)^\circ[x := u].$$

as follows:

$(\top)^\circ$	$=$	$\top$
$(p_i)^\circ$	$=$	$P_i(x)$
$(S_i)^\circ$	$=$	$S_i(x)$
$(c)^\circ$	$=$	$x = c$
$(\lambda)^\circ$	$=$	$C(x, x, x)$
$(\langle a, i\rangle\phi)^\circ$	$=$	$\exists y.(xR_{(a,i)}y \wedge (\phi)^\circ(y))$
$(\phi \vee \psi)^\circ$	$=$	$(\phi)^\circ \vee (\psi)^\circ$
$(\neg\phi)^\circ$	$=$	$\neg(\phi)^\circ$
$(GOTO_{s_i}(\phi))^\circ$	$=$	$(\phi)^\circ[x := s_i]$
$(\phi \bullet \psi)^\circ$	$=$	$\exists yz.(C(x, y, z) \wedge (\phi)^\circ(y) \wedge (\psi)^\circ(z))$

The only mysterious clause seems to be the one for λ, which is translated into $C(x, x, x)$. To understand this, notice that the only finite sequence u such that $u \cdot u = u$ is the empty sequence.

We list some remarks and some useful definitions.

1. Clearly MOD_m contains some redundancies. $\top$, obviously, but λ as well, as in superstructures λ is equivalent to $\neg((S_1 \vee \ldots \vee S_m \vee \bigvee_{c \in \mathrm{CONS}} c) \bullet \top)$. We leave them in the language for convenience.

2. MOD_m is able to measure lengths of sequences. That is, for each $k \in \mathbb{N}$ there is a formula $lth(k)$ in MOD_m such that $\mathcal{M}, u \Vdash lth(k)$ iff $|u| = k$.

$$
\begin{array}{lcll}
lth(0) & := & \lambda & \\
lth(1) & := & S_1 \vee \ldots \vee S_m \vee \bigvee_{c \in \mathrm{CONS}} c & \\
lth(k+1) & := & lth(k) \bullet lth(1) & \text{if } 1 \leq k.
\end{array}
$$

3. Consider the set of MOD_m-formulas constructed from just p_i, $\langle a, i \rangle$ and the booleans, for some fixed $i \in \{1, \ldots, m\}$. Let us call such formulas i-**formulas**. There is a clear bijection between the set of i-formulas and the ordinary modal formulas: let $(\_)^{-i}$ be the operation that removes i's from i-formulas, and $(\_)^i$ the inverse (the operation that adds i-labels to diamonds and propositional variables). Then clearly, for any $u \in S_i$ and any i-formula ϕ: $\mathbb{S}(\vec{\mathfrak{S}}), u \Vdash \phi$ iff $\mathfrak{S}_i, u \Vdash \phi^{-i}$.

Theorem 4.2 *MOD_m-formulas are invariant for sequence extension.*

Proof.
Suppose we have bisimulations $Z_i : \mathfrak{S}_i \underline{\leftrightarrow} \mathfrak{T}_i$ (for $i = 1, \ldots, m$). Let Z be their sequence extension. This is a relation between $\mathcal{M} = \mathbb{S}(\vec{\mathfrak{S}})$ and $\mathcal{N} = \mathbb{S}(\vec{\mathfrak{T}})$. We will prove that for any MOD_m-formula ϕ: if uZv then $\mathcal{M}, u \Vdash \phi$ iff $\mathcal{N}, v \Vdash \phi$, by induction on MOD_m-formulas ϕ. In the proof we only show that $\mathcal{M}, u \Vdash \phi$ implies $\mathcal{N}, v \Vdash \phi$ in each clause. The converse is always entirely analogous.

- If $\mathcal{M}, u \Vdash p_i$ then $\mathfrak{S}_i, u \Vdash p$. As uZv and $u \in S_i$, we know that uZ_iv. But Z_i is a bisimulation, so $\mathfrak{T}_i, v \Vdash p$, hence $\mathcal{N}, v \Vdash p_i$.
- If $\mathcal{M}, u \Vdash S_i$ then $u \in S_i$. As above, this, together with uZv implies that uZ_iv. But Z_i is a relation between $\mathfrak{S}_i$ and $\mathfrak{T}_i$. So $v \in T_i$ and $\mathcal{N}, v \Vdash S_i$.
- If $\mathcal{M}, u \Vdash c$ then $u = c$. Z only links c to the constant with the same name in $\mathcal{N}$, so v must also equal c. Hence $\mathcal{N}, v \Vdash c$.
- If $\mathcal{M}, u \Vdash \lambda$ then $u = \lambda$. uZv, so v must also equal λ, hence $\mathcal{N}, v \Vdash \lambda$.
- If $\mathcal{M}, u \Vdash \langle a, i \rangle \phi$ then $u \in S_i$ and there is a u' such that $u \xrightarrow{a} u'$ in $\mathfrak{S}_i$ and $\mathcal{M}, u' \Vdash \phi$. As uZv and $u \in S_i$ we must have uZ_iv. Z_i is a bisimulation, so we know that there must be a $v' \in T_i$ with $v \xrightarrow{a} v'$ and $u'Z_iv'$. But then $u'Zv'$ and thus, by the induction hypothesis, $\mathcal{N}, v' \Vdash \phi$. Hence $\mathcal{N}, v \Vdash \langle a, i \rangle \phi$.
- The case of the booleans is trivial to verify.
- Suppose $\mathcal{M}, u \Vdash GOTO_{s_i}(\phi)$. Then $\mathcal{M}, s_i \Vdash \phi$. Z_i is a bisimulation, so it must connect the root s_i of $\mathfrak{S}_i$ and the root t_i of $\mathfrak{T}_i$. Then the sequence extension Z also connects these points, so by the induction hypothesis: $\mathcal{N}, t_i \Vdash \phi$. Thus $\mathcal{N}, v \Vdash GOTO_{s_i}(\phi)$.
- If $\mathcal{M}, u \Vdash \phi \bullet \psi$ then there are sequences u_1 and u_2 such that $u = u_1u_2$, $\mathcal{M}, u_1 \Vdash \phi$ and $\mathcal{M}, u_2 \Vdash \psi$. Because u_1u_2Zv, we must be able to split v into two parts v_1 and v_2 such that u_1Zv_1 and u_2Zv_2. But then $\mathcal{N}, v_1 \Vdash \phi$ and $\mathcal{N}, v_2 \Vdash \psi$, hence $\mathcal{N}, v \Vdash \phi \bullet \psi$. □

There is a converse to theorem 4.2: any first-order $\mathcal{L}_m$-formula invariant for sequence extension is equivalent (in the class of superstructures SUP) to a MOD_m-formula. We will proceed by proving the important lemma 4.3, to prove this converse statement. The same lemma will prove useful in the characterisation of the safe formulas as well.

Lemma 4.3 *Let $Z_i = \{(u, v) \in S_i \times T_i \mid$ u satisfies the same modal formulas as v $\}$ be a bisimulation for every $i \in \{1, \ldots, m\}$. (We say that the relation of* modal equivalence *is a bisimulation between $\mathfrak{S}_i$ and $\mathfrak{T}_i$.)*

Let $\mathcal{M} := \mathbb{S}(\vec{\mathfrak{S}})$ and $\mathcal{N} := \mathbb{S}(\vec{\mathfrak{T}})$ with domains respectively M and N. Then

$$\equiv := \{(u, v) \in M \times N \mid \text{ } u \text{ and } v \text{ satisfy the same } MOD_m\text{-formulas}\}$$

is the sequence extension of the Z_i.

Proof.
Let Z be the sequence extension of the Z_i. Clearly $Z \subseteq \equiv$, as MOD_m-formulas are invariant for sequence extension.

Now suppose $u \equiv v$. We prove that uZv, by induction on $|u|$.

- $|u| = 0$. Then $u = \lambda$, hence $u \Vdash \lambda$. Thus $v \Vdash \lambda$, so uZv.
- $|u| = 1$. Then one of the following cases holds:
  1. $u \in S_i$. Then $u \Vdash S_i$, thus $v \Vdash S_i$, hence $v \in T_i$. As u and v satisfy the same i-formulas (see page 9), they satisfy the same modal formulas (when considered as points in respectively $\mathfrak{S}_i$ and $\mathfrak{T}_i$). Thus uZ_iv, hence uZv.
  2. $u = c \in$ CONS. Then $u \Vdash c$, hence $v \Vdash c$, thus $v = c$. So uZv.
- $|u| = k{+}1$. Then $u \Vdash lth(k{+}1)$ (see page 8 for the definition of lth) and thus $v \Vdash lth(k{+}1)$, hence $|v| = k + 1$. So we can write u as u_1u_2 and v as v_1v_2, with $|u_1| = |v_1| = k$ and $|u_2| = |v_2| = 1$. Now if $u_1 \Vdash \phi$ then $u \equiv v \Vdash (lth(k) \wedge \phi) \bullet \top$, hence $v_1 \Vdash \phi$. Using reasoning like this, we can prove that $u_1 \equiv v_1$ and $u_2 \equiv v_2$. By the induction hypothesis, u_1Zv_1 and u_2Zv_2, hence uZv.

We have hereby shown that $Z = \equiv$. That is: $\equiv$, the relation of satisfying the same MOD_m-formulas, is the sequence extension. □

Classes K of process graphs where for any two models in K modal equivalence of the roots implies that the relation of modal equivalence is a bisimulation (a condition needed for the above lemma) are called *Hennessy-Milner classes* ([11, 16]). An important example of such a class is the class of **m(odally)-saturated** models. This class is defined to be the class of all ω-saturated models, modulo bisimulation.

Theorem 4.4 *If $\phi(x)$ is invariant for sequence extension then $\phi(x)$ is SUP-equivalent to a MOD_m-formula, where SUP is the class of superstructures.*

Proof.
This proof is almost entirely the same as the proof that the first-order formulas that are preserved under bisimulation are precisely the modal ones (cf. [4]).

Consider the set of formulas:

$$\Gamma := \{(\psi)^\circ \mid \psi \in MOD_m \text{ and } \phi \models_{\mathsf{SUP}} (\psi)^\circ\}.$$

We will prove that $\Gamma \models_{\mathsf{SUP}} \phi$.

So suppose $\mathcal{M} = \mathbb{S}(\vec{\mathfrak{S}}) \models \Gamma[u]$. Take an ω-saturated elementary extension $\mathcal{M}^*$ of $\mathcal{M}$ (to see that this is always possible, see [18]). $\mathcal{M}^*$ will still satisfy ϕ_{SUP} (the sentence axiomatising SUP) and therefore be of the form $\mathbb{S}(\vec{\mathfrak{S}}^*)$. As each $\mathfrak{S}_i^*$ can be *interpreted* within $\mathcal{M}^*$ (see [14]), each $\mathfrak{S}_i^*$ is also ω-saturated. Furthermore, we still have $\mathcal{M}^* \models \Gamma[u]$ (as $\mathcal{M}^*$ is an elementary extension of $\mathcal{M}$). We must prove that $\mathcal{M}^* \models \phi[u]$, for then $\mathcal{M} \models \phi[u]$, by elementary descension (i.e. by the fact that $\mathcal{M}$ is an elementary submodel of $\mathcal{M}^*$).

Consider the set:

$$\Psi := \{(\psi)^\circ \mid \psi \in MOD_m \text{ and } \mathcal{M}^*, u \Vdash \psi\} \cup \{\phi\}.$$

$\{\phi_{\mathsf{SUP}}\} \cup \Psi$ is satisfiable. For suppose otherwise. Then, by compactness, there must be $\psi_1, \ldots, \psi_k \in MOD_m$ such that $\phi(x) \models_{\mathsf{SUP}} (\neg(\psi_1 \wedge \ldots \wedge \psi_k))^\circ$ and $\mathcal{M}^*, u \Vdash \psi_1 \wedge \ldots \wedge \psi_k$. But then $(\neg(\psi_1 \wedge \ldots \wedge \psi_k))^\circ \in \Gamma$, so $\mathcal{M}^*, u \Vdash \neg(\psi_1 \wedge \ldots \wedge \psi_k)$. Contradiction!

This gives us a model $\mathcal{N} = \mathbb{S}(\vec{\mathfrak{T}})$ and a point v in $\mathcal{N}$ such that $\mathcal{N} \models \Psi[v]$. As above, we can assume that each $\mathfrak{T}_i$ is ω-saturated. We will prove that the relation of MOD_m-equivalence, $\equiv$, is a sequence extension. This relation connects u and v. As ϕ is invariant for sequence extension and v satisfies this formula, u must also satisfy it.

Clearly, for any modal formula ψ:

$$\mathfrak{S}_i^*, s_i \Vdash \psi \Leftrightarrow \mathcal{M}^*, u \Vdash GOTO_{s_i}(\psi^i) \Leftrightarrow \mathcal{N}, v \Vdash GOTO_{s_i}(\psi^i) \Leftrightarrow \mathfrak{T}_i, t_i \Vdash \psi$$

by the truth definition of $GOTO_{s_i}$ and the fact that $u \equiv v$. Thus the roots of $\mathfrak{S}_i^*$ and $\mathfrak{T}_i$ satisfy the same modal formulas. As both $\mathfrak{S}_i^*$ and $\mathfrak{T}_i$ were assumed to be ω-saturated, the remarks above the statement of this theorem give us that the relation of modal equivalence is a bisimulation between $\mathfrak{S}_i^*$ and $\mathfrak{T}_i$. So the conditions of lemma 4.3 are satisfied and $\equiv$ is a sequence extension.

We have now proved that $\Gamma \models_{\mathsf{SUP}} \phi$. By compactness there must be $\psi_1, \ldots, \psi_k \in MOD_m$ such that $\phi \models_{\mathsf{SUP}} (\psi_j)^\circ$ for every $j \in \{1, \ldots, k\}$ (i.e. $\phi \models_{\mathsf{SUP}} (\psi_1 \wedge \ldots \wedge \psi_k)^\circ$) and $(\psi_1)^\circ, \ldots, (\psi_k)^\circ \models_{\mathsf{SUP}} \phi$. Hence $\mathsf{SUP} \models \phi \leftrightarrow (\psi_1 \wedge \ldots \wedge \psi_k)^\circ$. □

Theorem 4.2 and 4.4 together give us a characterisation of first-order $\mathcal{L}_m$-formulas invariant for sequence extension: this notion conincides with equivalence (in SUP) to MOD_m-formulas. The next section gives a similar characterisation of the formulas safe for sequence extension.

Remark: Something more can be said about the link between first-order definable operations and Hennessy-Milner classes. Hennessy-Milner classes of process graphs are useful in that in such classes nonbisimilarity of two process graphs is always *witnessed* by a modal formula. Such a modal witness would be true at only one of the two process graphs, thus distinguishing them. See Cleaveland [10] for a method of automatically generating such witnesses on finite process graphs.

A Hennessy-Milner class could be viewed as a workspace within which to work. As long as we stay in this space, we need never worry that a failure of bisimilarity is not witnessed by any modal formula. However, certain process operations, when applied to process graphs from a certain Hennessy-Milner class K may take us outside this class. Hollenberg [15] proves for a notion of first-order definable operation very similar to the one in the present paper that such operations always preserve ω-saturation. The proof of [15] can easily be fitted to the present notion. It implies that first-order definable operations that respect bisimulation will preserve m-saturation. Thus, the Hennessy-Milner class of all m-saturated process graphs can be used as a workspace as long as we only consider safe/invariant operations, which seems enough for most purposes.

Actually, first-order definable operations trivially preserve finiteness, so the class of finite process graphs may also be used as a workspace. The class of m-saturated process graphs is more general however: in this workspace we can model infinite computations and even infinite choice, both of which are very useful abstractions, unavailable in the class of finite models. □

5 Safety for sequence extension

This section contains a characterisation of the $\mathcal{L}_m$-formulas that are safe for sequence extension. This notion was defined on page 6.

Definition 5.1 *REL_m is the following language:*

$$\theta ::= R_{(a,i)} \mid \phi? \mid \Delta \mid \textsc{permute} \mid RES_{s_i} \mid RES_c \mid RES_\lambda \mid \theta \cup \theta \mid \theta;\theta \mid \sim\theta \mid \theta \bullet \theta$$

where $a \in A$, $1 \leq i \leq m$, $c \in \textsc{cons}$ and $\phi \in MOD_m$. □

We can give the meaning of REL_m-formulas in a superstructure $\mathcal{M} = \mathbb{S}(\vec{\mathfrak{S}})$ in terms of binary relations on M, the universe of $\mathcal{M}$.

$$
\begin{array}{lcl}
R_{(a,i)}^{\mathcal{M}} & := & \{(u,v) \in S_i \times S_i \mid u \xrightarrow{a} v \text{ in } \mathfrak{S}_i\} \\
\phi?^{\mathcal{M}} & := & \{(u,u) \in M \times M \mid \mathcal{M}, u \Vdash \phi\} \\
\Delta^{\mathcal{M}} & := & \{(xu, xxu) \in M \times M \mid |x| = 1\} \\
\textsc{permute}^{\mathcal{M}} & := & \{(xyu, yxu) \in M \times M \mid |x| = |y| = 1\} \\
RES_{s_i}^{\mathcal{M}} & := & \{(u, s_i) \mid u \in M\} \\
RES_c^{\mathcal{M}} & := & \{(u, c) \mid u \in M\} \\
RES_\lambda^{\mathcal{M}} & := & \{(u, \lambda) \mid u \in M\} \\
(\theta_1 \cup \theta_2)^{\mathcal{M}} & := & \theta_1^{\mathcal{M}} \cup \theta_2^{\mathcal{M}} \\
(\theta_1; \theta_2)^{\mathcal{M}} & := & \theta_1^{\mathcal{M}} \circ \theta_2^{\mathcal{M}} \\
(\sim \theta)^{\mathcal{M}} & := & \{(u,u) \in M \times M \mid \neg \exists v. u \theta^{\mathcal{M}} v\} \\
(\theta_1 \bullet \theta_2)^{\mathcal{M}} & := & \{(u_1u_2, v_1v_2) \in M \times M \mid \ u_1 \theta_1^{\mathcal{M}} v_1 \text{ and } u_2 \theta_2^{\mathcal{M}} v_2\}
\end{array}
$$

where $\circ$ is relation composition. Note that some of these relations are always partial functions. The resetting relations (RES_λ, RES_c and RES_{s_i}) are *total* functions. PERMUTE is a partial function only defined on sequences in the domain of $\mathcal{M}$ whose length lies between 2 and n (where n is the maximum length of sequences in our superstructures). Δ gives a partial function defined precisely on sequences of length between 1 and $n-1$. It is not defined on sequences of length n, because this would require sequences of length $n+1$ to exist in $\mathcal{M}$.

These relations are all first-order definable in $\mathcal{M}$, as the following standard translation of REL_m-formulas will testify. Each REL_m-formula is thus an abbreviation for a $\mathcal{L}_m$-formula $\phi(x,y)$.

$$
\begin{array}{lcl}
(R_{(a,i)})^\circ & := & xR_{(a,i)}y \\
(\phi?)^\circ & := & x = y \wedge (\phi)^\circ \\
(\Delta)^\circ & := & \exists uv.(|u| = 1 \wedge C(x,u,v) \wedge C(y,u,x)) \\
(\textsc{permute})^\circ & := & \exists uvwz_1z_2. \left(\begin{array}{l} |u| = 1 \wedge |v| = 1 \wedge C(x,u,z_1) \wedge C(z_1,v,w) \wedge \\ C(y,v,z_2) \wedge C(z_2,u,w) \end{array} \right) \\
(RES_{s_i})^\circ & := & y = s_i \\
(RES_c)^\circ & := & y = c \\
(RES_\lambda)^\circ & := & C(y,y,y) \\
(\theta_1 \cup \theta_2)^\circ & := & (\theta_1)^\circ \vee (\theta_2)^\circ \\
(\theta_1; \theta_2)^\circ & := & \exists z.((\theta_1)^\circ(x,z) \wedge (\theta_2)^\circ(z,y)) \\
(\sim \theta)^\circ & := & x = y \wedge \neg \exists y.(\theta)^\circ \\
(\theta_1 \bullet \theta_2)^\circ & := & \exists uvwz. \left(\begin{array}{l} C(x,u,v) \wedge C(y,w,z) \wedge (\theta_1)^\circ(u,w) \wedge \\ (\theta_2)^\circ(v,z) \end{array} \right)
\end{array}
$$

Note that the standard translation of $\phi?$ uses that of ϕ, defined in the previous section. $|u| = 1$, as used in the definition of PERMUTE and Δ, is of course not an $\mathcal{L}_m$-formula, but we use it as an abbreviation for $S_1(x) \vee \ldots \vee S_m(x) \vee \bigvee_{c \in \mathrm{CONS}}(x = c)$. Clearly for any $\theta \in REL_m$:

$$\theta^{\mathcal{M}} = \{(u,v) \in M \mid \mathcal{M} \models (\theta)^\circ[u,v]\}.$$

$R_{(a,i)}$, ? (test), RES_{s_i} (resetting to the root), $\cup$, ; and $\sim$ (strong negation) are taken from [5]. There a setting without sequences and without extra constants is considered. The extra operations of REL_m allow us to do some manipulation of sequences: copying (Δ), permuting (PERMUTE) and deleting (RES_λ).

Let us acquaint ourselves with REL_m somewhat by writing some useful REL_m-*programs*. By this we mean REL_m-formulas whose interpretation is always a partial function.

Examples: programming in REL_m.

1. $\textsc{id}_k := lth(k)?$ tests whether the input sequence is of length k.

2. LROTATE_k transforms a sequence $y_1 \dots y_k x$ into $x y_1 \dots y_k$, where we use y_j and x as names for sequences of length 1.

$$\begin{aligned} \text{LROTATE}_0 &:= \text{ID}_1 \\ \text{LROTATE}_{k+1} &:= (\text{ID}_k \bullet \text{PERMUTE}); (\text{LROTATE}_k \bullet \text{ID}_1). \end{aligned}$$

Like ID_k this program is very specific about the length of its input sequence: LROTATE_k succeeds precisely on sequences of length $k+1$. LROTATE_0 does nothing but test whether its input sequence is of the right length (that is, 1). The first part of LROTATE_{k+1} (i.e. $(\text{ID}_k \bullet \text{PERMUTE})$) succeeds on a sequence if it is of the form $y_1 \dots y_k y_{k+1} x u$ (where u is a possibly empty sequence) in which case it will return the sequence $y_1 \dots y_k x y_{k+1} u$. The second part of LROTATE_{k+1} is more specific however and will now only succeed if $u = \lambda$, in which case the desired $x y_1 \dots y_k y_{k+1}$ is returned.

3. RROTATE_k sends $x y_1 \dots y_k$ to $y_1 \dots y_k x$.

$$\begin{aligned} \text{RROTATE}_0 &:= \text{ID}_1 \\ \text{RROTATE}_{k+1} &:= \text{PERMUTE}; (\text{ID}_1 \bullet \text{RROTATE}_k). \end{aligned}$$

4. $\text{TRANSPOSE}(i,j)$ (with $1 \leq i < j$) sends $x_1 \dots x_i \dots x_j u$ to $x_1 \dots x_j \dots x_i u$. Thus it permutes the i-th and the j-th position in the sequence. Such an operation is called a *transposition*. As any permutation (i.e. any program sending $x_1 \dots x_k$ to $x_{\pi(1)} \dots x_{\pi(k)}$ for a fixed bijection on $\{1, \dots, k\}$) is a composition of such transpositions, we have programs in REL_m that can carry out any permutation.

   $\text{TRANSPOSE}(i,j)$ is defined to be:

$$(\text{ID}_i \bullet \text{LROTATE}_{(j-i)-1} \bullet \top?); (\text{ID}_{i-1} \bullet \text{PERMUTE}); (\text{ID}_i \bullet \text{RROTATE}_{(j-i)-1} \bullet \top?)$$

   As $\bullet$ and ; are interpreted as associative operations, $\text{TRANSPOSE}(i,j)$ is well-defined.

5. Another useful program is one that sends $x_1 \dots x_i \dots x_k$ to $x_1 \dots x_i \dots x_k x_i$. That is, one that copies the i-th element of a k-sequence and attaches this copy to the end.

$$\text{COPY}(k,i) := (\text{ID}_{i-1} \bullet \Delta); (\text{ID}_i \bullet \text{RROTATE}_{k-i})$$

   where $1 \leq i \leq k < n$.

6. Finally, we define a program that sends any sequence u to uc for some constant $c \in \text{CONS}$ (if $|u| < n$ of course).

$$\text{ADD}(c) := \top? \bullet (\lambda?; RES_c).$$

Here endeth the programming lesson.

Theorem 5.2 *REL_m-formulas are safe for sequence extension.*

Proof.
Suppose $Z_i : \mathfrak{S}_i \underline{\leftrightarrow} \mathfrak{T}_i$ are bisimulations and Z is their sequence extension. Let $\mathcal{M} := \mathbb{S}(\vec{\mathfrak{S}})$ and $\mathcal{N} := \mathbb{S}(\vec{\mathfrak{T}})$. We prove that Z is a bisimulation with respect to each $\theta \in REL_m$, by induction on θ.

- If uZv and $u R^{\mathcal{M}}_{(a,i)} u'$ then $u \in S_i$, so $u Z_i v$. Also $u \xrightarrow{a} u'$ in $\mathfrak{S}_i$, so by the fact that Z_i is a bisimulation, there must be a $v' \in T_i$ with $v \xrightarrow{a} v'$ and $u' Z_i v'$. Thus $v R^{\mathcal{N}}_{(a,i)} v'$ and $u' Z v'$, as desired.
- If uZv and $u\ \phi?^{\mathcal{M}} u$ then $\mathfrak{M}, u \Vdash \phi$. MOD_m-formulas are invariant for sequence extension (theorem 4.2), so $\mathcal{N}, v \Vdash \phi$, hence $v\ \phi?^{\mathcal{N}} v$. As we already had uZv, we are done.

- Suppose $xuZv$ and $xu\Delta^{\mathcal{M}}xxu$, where $|xu| = k < n$. Z only connects sequences of the same length, so v can be written as yw, where $|y| = 1$ and $|w| = k-1$. Furthermore, the definition of Z ensures that xZy and uZw. Now $v\Delta^{\mathcal{N}}yyw$ (we know that the latter element exists in $\mathcal{N}$, because $|yyw| = k+1 \leq n$). But we also have that $xxuZyyw$, so we are done.
- The PERMUTE case is dealt with similarly.
- The cases of $\cup$, ; and $\sim$ are standard (see [5]): we skip the proofs here.
- Suppose u_1u_2Zv, $u_1\theta_1^{\mathcal{M}}u_1'$ and $u_2\theta_2^{\mathcal{M}}u_2'$ with $|u_1'u_2'| \leq n$ (implying $u_1u_2(\theta_1 \bullet \theta_2)^{\mathcal{M}}u_1'u_2'$). We can split v into parts v_1 and v_2 such that u_1Zv_1 and u_2Zv_2. By the induction hypothesis, there must be v_1' and v_2' such that $u_1'Zv_1'$, $u_2'Zv_2'$, $v_1\theta_1^{\mathcal{N}}v_1'$ and $v_2\theta_2^{\mathcal{N}}v_2'$. Thus $u_1'u_2'Zv_1'u_2'$ and $v_1v_2(\theta_1 \bullet \theta_2)^{\mathcal{N}}v_1'v_2'$. □

For any REL_m-formula θ, $\sim\theta$ is SUP-equivalent to $\phi?$, for some MOD_m-formula. For above we have seen that $\sim\theta$ is safe, hence

$$\begin{aligned}\exists y.(\sim\theta)^\circ &\equiv \exists y.(x = y \wedge \neg\exists y.(\theta)^\circ)\\ &\equiv \neg\exists y.(\theta)^\circ\end{aligned}$$

is invariant for sequence extension. By the characterisation of invariant formulas, it must thus be SUP-equivalent to a MOD_m-formula ϕ. But then:

$$\begin{aligned}(\phi?)^\circ &\equiv x = y \wedge \neg\exists y.(\theta)^\circ\\ &\equiv (\sim\theta)^\circ.\end{aligned}$$

Thus $\sim$ is superfluous.

We mentioned earlier that many common process algebra operations, in fact all the operations of ACP (the Algebra of Communicating Processes, [9]) are safe/invariant. To show this, it suffices to show that they are first-order definable using MOD_m and REL_m-formulas in their definition.

Examples: ACP operations.
In the following assume fixed some set A of atomic actions (as usual) and also a commutative and associative partial *communication function* $\gamma : A \times A \to A$.

- **Alternative composition $\mathfrak{S} + \mathfrak{T}$:** $m = 2$, CONS $= \{c\}$, $n = 1$.

  $\phi_a := R_{(a,1)} \cup R_{(a,2)} \cup (c?; RES_{s_1}; R_{(a,1)}) \cup (c?; RES_{s_2}; R_{(a,2)})$.
  $\phi_{\surd} := \surd_1 \vee \surd_2 \vee [c \wedge (GOTO_{s_1}(\surd_1) \vee GOTO_{s_2}(\surd_2))]$.
  $\sigma := c$.

- **Sequential Composition $\mathfrak{S} \cdot \mathfrak{T}$:** $m = 2$, CONS $= \emptyset$, $n = 1$.

  $\phi_a := R_{(a,1)} \cup R_{(a,2)} \cup (\surd_1?; RES_{s_2}; R_{(a,2)})$.
  $\phi_{\surd} := \surd_2 \vee (\surd_1 \wedge GOTO_{s_2}(\surd_2))$.
  $\sigma := s_1$.

The next three ACP-operations are all merges of some sort. Each of them employs the following REL_2-formula:

$$\theta_a := (R_{(a,1)} \bullet S_2?) \cup (S_1? \bullet R_{(a,2)}) \cup \bigcup_{\gamma(b_1,b_2)=a} (R_{(b_1,1)} \bullet R_{(b_2,2)}).$$

The first two formulas in this union are for the independent steps, the final formula is for *communication*-steps. Of course, for θ_a to be a proper REL_2-formula, the union contained in it must be finite. That is, we assume that for each $a \in A$: $\gamma^{-1}(\{a\})$ is a finite set. This is not such an outlandish demand, as this set usually *is* taken to be finite. For example, it is satisfied when A is finite. But even when it is infinite, often $\gamma^{-1}(\{a\})$ is either empty (when a is not a communication action) or consists of two elements (r, s) and (s, r), where s is some *send*-action, r some *read*-action and a is their communication.

- **Merge $\mathfrak{S} \parallel \mathfrak{T}$:** $m = 2$, $\textsc{cons} = \emptyset$, $n = 2$.

$$\begin{array}{lcl} \phi_a & := & \theta_a. \\ \phi_{\surd} & := & \surd_1 \bullet \surd_2. \\ \sigma & := & s_1 s_2. \end{array}$$

- **Left-merge $\mathfrak{S} \mathbin{\underline{\parallel}} \mathfrak{T}$:** $m = 2$, $\textsc{cons} = \{c\}$, $n = 2$. This is like $\parallel$, except that the first action of the combined process has to be a step in $\mathfrak{S}$.

$$\begin{array}{lcl} \phi_a & := & \theta_a \cup (c?; [(RES_{s_1}; R_{(a,1)}) \bullet RES_{s_2}]) \\ \phi_{\surd} & := & \surd_1 \vee \surd_2. \\ \sigma & := & c. \end{array}$$

- **Communication merge $\mathfrak{S}|\mathfrak{T}$:** $m = 2$, $\textsc{cons} = \{c\}$, $n = 2$. Here we demand that the first action is a communication step.

$$\begin{array}{lcl} \phi_a & := & \theta_a \cup [c?; \bigcup_{\gamma(b_1,b_2)=a} [(RES_{s_1}; R_{(b_1,1)}) \bullet (RES_{s_2}; R_{(b_2,2)})]] \\ \phi_{\surd} & := & \surd_1 \vee \surd_2. \\ \sigma & := & c. \end{array}$$

- **Encapsulation $\partial_H(\mathfrak{S})$, for $H \subseteq A$:** $m = 1$, $\textsc{cons} = \emptyset$, $n = 1$. This operator eliminates all H-actions.

$$\begin{array}{lcl} \phi_a & := & \begin{cases} R_{(a,1)} & \text{if } a \notin H. \\ \bot? & \text{if } a \in H. \end{cases} \\ \phi_{\surd} & := & \surd_1 \\ \sigma & := & s_1 \end{array}$$

  Note that this is the first operation we have seen that is not uniform in its definition of the new transitions.

- **Atomic processes $a \in A$:** $m = 0$, $\textsc{cons} = \{c, d\}$, $n = 1$.

$$\begin{array}{lcl} \phi_b & := & \begin{cases} c?; RES_d & \text{if } b = a. \\ \bot? & \text{else.} \end{cases} \\ \phi_{\surd} & := & d. \\ \sigma & := & c. \end{array}$$

- **Termination ε:** $m = 0$, $\textsc{conc} = \{c\}$, $n = 1$.

$$\begin{array}{lcl} \phi_a & := & \bot?. \\ \phi_{\surd} & := & c. \\ \sigma & := & c. \end{array}$$

- **Deadlock δ:** $m = 0$, $\textsc{conc} = \{c\}$, $n = 1$.

$$\begin{array}{lcl} \phi_a & := & \bot?. \\ \phi_{\surd} & := & \bot. \\ \sigma & := & c. \end{array}$$

As the ACP-operations are not very involved, the final example we give is the *projection operator*, which must use Δ. In the modal logic literature, this operator is known under the name of *k-unravelling*. We need to assume that A is finite.

- **Projection $\pi_k(\mathfrak{S})$:** $m = 1$, $\textsc{cons} = A$, $n = 2k + 1$.

$$\begin{array}{lcl} \phi_a & := & \top? \bullet [S?; \Delta; (S? \bullet (\lambda?; RES_a) \bullet R_a)] \\ \phi_{\surd} & := & \top \bullet \surd \\ \sigma & := & s \end{array}$$

We left the labels out of S, R_a and s, as there is only one candidate here (π_k is a unary operation), so there can be no confusion. The elements of $\pi_k(\mathfrak{S})$ reachable from its root are of the form $sa_1s_1 \dots a_is_i$ with $s \stackrel{a_1}{\to} s_1 \stackrel{a_2}{\to} \dots \stackrel{a_i}{\to} s_i$. Two such sequences us_i and v are connected by an a-step precisely if $v = us_ias_{i+1}$ for some s_{i+1} with $s_i \stackrel{a}{\to} s_{i+1}$. Note that when a sequence has length $n = 2k+1$ (denoting a path of length k) it can have no successors, because successors increase in length. So π_k transforms a process graph into a tree-like structure, cut off at depth k.

End of examples.

REL_m is not just any subset of the set of safe formulas:

Theorem 5.3 (Main Theorem) *Fix n, m and* CONS. *If $\phi(x,y)$ is safe for sequence extension then ϕ is* SUP*-equivalent to a REL_m-formula.*

The proof of this theorem will simulate the safety proof of Van Benthem ([5]). His proof uses a characterisation of *continuous* modal formulas. We can do without an analogous characterisation here.

We need some definitions. The first of these is implicitly defined already in [1].

Definition 5.4 *Let $\mathfrak{S}$ be a rooted Kripke model. Then the* **multiplied, unravelled** *model $\mathfrak{S}^\diamond = (S^\diamond, R_a^\diamond, V^\diamond, s^\diamond)_{a\in A}$ is defined as follows:*

1. $S^\diamond := \{(s_0 \dots s_n, a_1 \dots a_n, \sigma) \in S^+ \times A^* \times \mathbb{N}^* \mid s = s_0 \stackrel{a_1}{\to} \dots \stackrel{a_n}{\to} s_n$ *and* $|\sigma| = n\}$. *Here S^+ is the set of all nonempty sequences built from elements of S, A^* is the set of all A-sequences (including λ, the empty sequence) and similarly for $\mathbb{N}^*$.*
2. $R_a^\diamond := \{((\vec{s}s', \vec{a}, \sigma), (\vec{s}s's'', \vec{a}a, \sigma n)) \in S^\diamond \times S^\diamond \mid s' \stackrel{a}{\to} s''$ *and* $n \in \mathbb{N}\}$.
3. $V^\diamond(p) := \{(\vec{s}s', \vec{a}, \sigma) \in S^\diamond \mid s' \in V(p)\}$.
4. $s^\diamond := (s, \lambda, \lambda)$. □

It is easy to see that the function Z that takes $(\vec{s}\cdot s', \vec{a}, \sigma)$ to s' is a zigzagmorphism (that is, a totally functional bisimulation) from $\mathfrak{S}^\diamond$ to $\mathfrak{S}$. The operation $(\_)^\diamond$ thus takes a rooted model to a bisimilar one, but transmogrified into an intransitive tree, with moreover the properties that:

1. It is generated by the root $s^\diamond$ (this need not have been the case for $\mathfrak{S}$).
2. If there is an a-transition between two points in $\mathfrak{S}^\diamond$ then there is no b-transition between them, for any $b \in A - \{a\}$.
3. If $xR_a^\diamond y$ then there are infinitely many y' also reachable from x via an a-step, such that the trees generated by y and y' are isomorphic. This explains the predicate *multiplied*.

Definition 5.5 *Let $(\mathcal{S}_i)_{i\in I}$ be a nonempty I-sequence of unrooted Kripke models. Then their* **disjoint union** *$\sum_{i\in I}\mathcal{S}_i$ is the following model:*

1. *Domain:* $\{(i,x) \mid i \in I$ *and* $x \in S_i\}$.
2. $(i,x)R_a(j,y)$ *iff* $i = j$ *and* $xR_a^{\mathcal{S}_i}y$.
3. $V(p) := \{(i,x) \mid \mathcal{S}_i, x \Vdash p\}$. □

Now for a less standard notion:

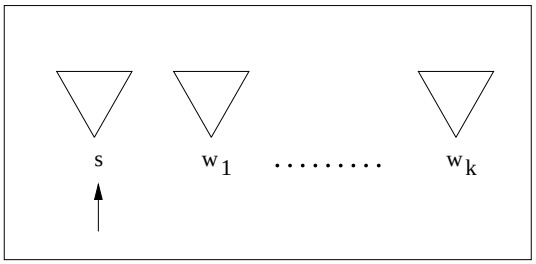


Figure 2: The model $\mathfrak{S}^\diamond(w_1, \ldots, w_k)$.

Definition 5.6 *Let $\mathfrak{S} = (\mathcal{S}, s)$ again be a rooted model ($\mathcal{S}$ is its unrooted part and s its root) and let $w_1, \ldots, w_k$ be a possibly empty sequence of elements of its domain (possibly containing repetitions). Then $\mathfrak{S}^\diamond(w_1, \ldots, w_k)$ is the model constructed as follows. Multiply and unravel each of $\mathfrak{S}, (\mathcal{S}, w_1), \ldots, (\mathcal{S}, w_k)$. This gives us, respectively, $\mathfrak{T}_0 = (\mathcal{T}_0, t_0)$, $\mathfrak{T}_1 = (\mathcal{T}_1, t_1)$, $\ldots$, $\mathfrak{T}_k = (\mathcal{T}_k, t_k)$. Define $\mathfrak{S}^\diamond(w_1, \ldots, w_k)$ to be the model $(\sum_{0 \leq i \leq k} \mathcal{T}_i, (0, t_0))$.* □

So $\mathfrak{S}^\diamond(w_1, \ldots, w_k)$ is the disjoint union of $\mathfrak{S}^\diamond, (\mathcal{S}, w_1)^\diamond, \ldots, (\mathcal{S}, w_k)^\diamond$, with the root of $\mathfrak{S}^\diamond$ as the root of the disjoint union (see figure 2). There is again a zigzagmorphism from $\mathfrak{S}^\diamond(w_1, \ldots, w_k)$ to $\mathfrak{S}$ (use the zigzagmorphism between a process graph and its unravelling).

Definition 5.7 *If w is an element in the domain of some model $\mathcal{M} = \mathbb{S}(\mathfrak{S}_1, \ldots, \mathfrak{S}_m)$, then w is some sequence $w_1 \ldots w_k$, a result of interleaving sequences $w_1^i \ldots w_{k_i}^i$ $(1 \leq i \leq m)$ of elementents in S_i and some sequence of constants from* CONS*. Relative to this decomposition of w, define:*

$$\mathcal{M}^\diamond := \mathbb{S}(\mathfrak{S}_1^\diamond(w_1^1, \ldots, w_{k_1}^1), \ldots, \mathfrak{S}_m^\diamond(w_1^m, \ldots, w_{k_m}^m)),$$

□

Let Z be the (functional) sequence extension of the zigzagmorphisms from $\mathfrak{S}_i^\diamond(w_1^i, \ldots, w_{n_i}^i)$ to $\mathfrak{S}_i$. If $\alpha(w_j^i)$ is set to $(j, (w_j^i, \lambda, \lambda))$ (recall that $(w_j^i, \lambda, \lambda)$ is the root of the j-th part of $\mathfrak{S}_i^\diamond(w_i^1, \ldots, w_{k_i}^i)$) and $\alpha(c)$ to c (for $c \in$ CONS) then Z connects $\alpha(w_1) \ldots \alpha(w_n)$ to $w_1 \ldots w_n$. So if we are interested in a particular point w in $\mathcal{M}$ only modulo sequence extensions, then we may assume the following:

1. Each $\mathfrak{S}_i$ is the disjoint union of a sequence of multiplied unravelled models, rooted in $s_i, w_1^i, \ldots, w_{k_i}^i$ respectively.
2. $w_1^i, \ldots, w_{k_i}^i$ are distinct points in $\mathfrak{S}_i$, different from the root s_i.

Let us call superstructures that have this form with respect to some sequence w **unravelled with respect to** w.

Proof of the Main Theorem:
Only the 'only if' part is nontrivial. So suppose $\phi(x, y)$ is safe for sequence extension. We will prove that:

$$\phi \models_{\mathsf{SUP}} \bigvee \{\psi \in REL_m \mid \psi \models_{\mathsf{SUP}} \phi\}.$$

So suppose that $\mathcal{M} = \mathbb{S}(\vec{\mathfrak{S}}) \models \phi[w, v]$. We may assume that $\mathcal{M}$ is unravelled with respect to w and that each constituent process graph $\mathfrak{S}_i$ is m-saturated, without loss of generality. For consider an ω-saturated elementary extension $\mathcal{M}'$ of $\mathcal{M}$, which can be found with the aid of [18]. Each constituent process graph $\mathfrak{S}_i'$ of this superstructure must then also be ω-saturated. Now unravel $\mathcal{M}'$ with respect to w, by means of the above trick. Then we obtain ansuperstructure $\mathcal{M}''$, unravelled with respect to some sequence w'. Furthermore, there is a sequence extension Z between $\mathcal{M}''$ and $\mathcal{M}'$ such that $w' Z w$. When restricted to the components of $\mathcal{M}''$, Z gives zigzagmorphisms to the corresponding

components in $\mathcal{M}'$. These latter were all ω-saturated, so the components of $\mathcal{M}''$ must be m-saturated. By elementary extension $\mathcal{M}' \models \phi[w, v]$. As ϕ is safe and $w'Zw$, $\mathcal{M}'' \models \phi[w', v']$ for some v' with $v'Zv$. If we could now show that $\mathcal{M}'' \models \psi[w', v']$ for some $\psi \in REL_m$ with $\psi \models_{\mathsf{SUP}} \phi$, then we would be done, for then by the fact that REL_m-formulas are safe for sequence extension, $\mathcal{M}' \models \psi[w, u]$ for some u in $\mathcal{M}'$ with $v'Zu$ As Z is functional (it is the sequence extension of a collection of zigzagmorphisms), this u must be v. So $\mathcal{M}' \models \psi[w, v]$, hence $\mathcal{M} \models \psi[w, v]$ by elementary descension. So we may assume safely that $\mathcal{M}$ is itself unravelled with respect to w and that its constituent process graphs are m-saturated.

We will construct a set of first-order formulas $\Psi(x, y, \vec{y})$ describing how v can be reached from w and what MOD_m-formulas are satisfied along the way. We will then show that $\Psi \models_{\mathsf{SUP}} \phi$. By compactness, for some finite subset $\Psi_0(x, y, \vec{y})$ of $\Psi(x, y, \vec{y})$ it is the case that $\Psi_0 \models_{\mathsf{SUP}} \phi$, i.e. $\exists \vec{y}. \bigwedge \Psi_0 \models_{\mathsf{SUP}} \phi$. $\exists \vec{y}. \bigwedge \Psi_0$ will turn out to be equivalent to some REL_m-formula ψ. As the description Ψ will be built in such a way that it is satisfiable in $\mathcal{M}$ with x bound to w and y to v, $\mathcal{M} \models \psi[w, v]$ will hold.

The set Ψ is quite complicated, but we will give an example after the definition, so bear with us.

First, assign a distinct variable $\overline{u}$ to every element u of $\mathcal{M}$. If $u \in \mathcal{M}$, let Φ_u be the set of MOD_m-formulas true at u. By convention, we confuse these with their standard translations. Let w be the sequence $w_1 \dots w_g$ and v the sequence $v_1 \dots v_h$. Now construct Ψ as follows:

1. Start with $\Phi_w(\overline{w}) \cup \Phi_v(\overline{v})$.
2. For every $w_j \in S_i$, list the v_l that can be reached from it by a path in $\mathfrak{S}_i$. If this list is empty, add nothing. If the list is nonempty we describe the minimal subtree rooted in w_j, whose points include all the points in the list, as a set of formulas of the form $\{\overline{w_j} R_{(a,i)} \overline{u}, \dots\}$. Furthermore, for all the intermediate points u in this subtree, add the set $\Phi_u(\overline{u})$.
3. For every root s_i, do the same, but instead of the variable $\overline{w_j}$, use the constant s_i.
4. Add a formula stating '$w = w_1 \dots w_g$', using C. For instance: '$w = w_1 w_2 w_3$' is expressed by $\exists z.(C(z, \overline{w_1}, \overline{w_2}) \wedge C(\overline{w}, z, \overline{w_3}))$ and '$w = \lambda$' is expressed by $C(\overline{w}, \overline{w}, \overline{w})$.
5. Similarly, add a formula stating '$v = v_1 \dots v_h$'.
6. Finally, add $\{x = \overline{w}, y = \overline{v}\}$.

Example:
Let $w = w_1 c w_3$ and $v = v_1 v_2 v_3 v_2$ where these are related as depicted in figure 3 and where $c \in$ CONS. Then Ψ is the union of the following sets of formulas:

1. $\Phi_w(\overline{w}) \cup \Phi_v(\overline{v})$.
2. $\{\overline{w_1} R_{(a,1)} \overline{u},\ \overline{u} R_{(b,1)} \overline{v_1},\ \overline{u} R_{(a,1)} \overline{v_3}\} \cup \Phi_u(\overline{u})$.
3. $\{s_2 R_{(b,2)} \overline{v_2}\}$.
4. $\{\exists z.(C(z, \overline{w_1}, c) \wedge C(\overline{w}, z, \overline{w_2}))\}$.
5. $\{\exists z z'.(C(z, \overline{v_1}, \overline{v_2}) \wedge C(z', \overline{v_3}, \overline{v_2}) \wedge C(\overline{v}, z, z'))\}$.
6. $\{x = \overline{w}, y = \overline{v}\}$.

End of example.

We will prove that Ψ SUP-implies ϕ. So suppose we have a model $\mathcal{N} = \mathbb{S}(\vec{\mathfrak{T}}) \models \Psi$ under some assignment σ to the variables. Say $\sigma(\overline{w})$ is $x_1 \dots x_g$. $x_1 \dots x_g$ is again the result of interleaving sequences $x^i_1 \dots x^i_{k_i}$ from $\mathfrak{T}_i$ and elements of CONS in the same way as w is constructed, as $x_1 \dots x_g$ and w satisfy the same MOD_m-formulas, by definition of Ψ.

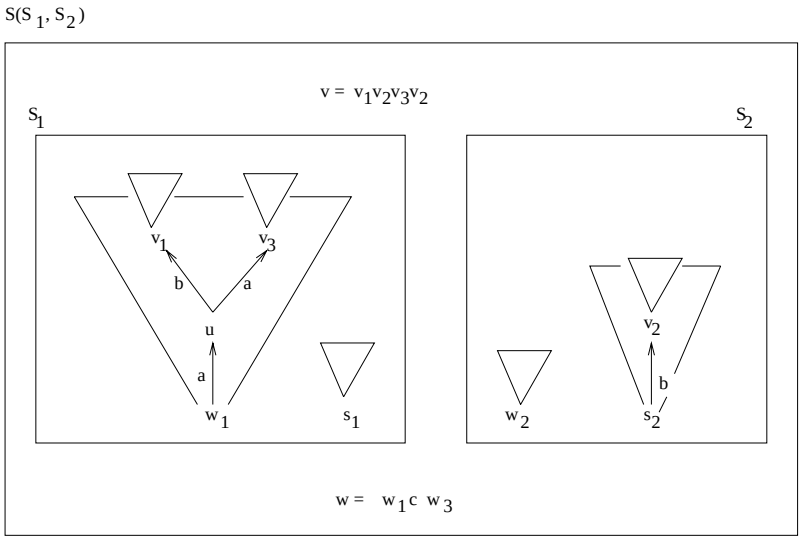


Figure 3: Part of a superstructure.

We will show that we may assume that $\mathcal{N}$ is unravelled with respect to $x_1 \dots x_g$, that each $\mathfrak{T}$ is m-saturated and that distinct free variables in Ψ are not assigned the same value by σ. This last condition is to ensure that Ψ describes subtrees in $\mathcal{N}$, isomorphic to those it describes in $\mathcal{M}$, under the map $\iota : u \mapsto \sigma(\overline{u})$. There is some work to be done here, for even if $\mathcal{N}$ is unravelled and multiplied, ι may not be injective.

First, as we are just concerned with first-order formulas, we may assume that each $\mathfrak{T}_i$ is m-saturated. For if they were not, as before we could find an ω-saturated elementary extension $\mathcal{N}^*$. Each component $\mathfrak{T}_i^*$ of this superstructure would also be ω-saturated, hence m-saturated. By the fact that $\mathcal{N}^*$ is an elementary extension of $\mathcal{N}$, $\mathcal{N}^* \models \Psi[\sigma]$. Furthermore, if we could show that $\mathcal{N}^* \models \phi[\sigma]$, then $\mathcal{N} \models \phi[\sigma]$ (which is what we are trying to demonstrate), again using the definition of elementary extensions. So we may assume, without loss of generality, that the components of $\mathcal{N}$ themselves are m-saturated.

Next define an assignment τ from the free variables in Ψ to elements of $\mathcal{N}^\diamond$ (definition 5.7), where $\mathcal{N}^\diamond$ is calculated with respect to $x_1 \dots x_g$. Each free variable of Ψ is of the form $\overline{u}$ for some $u \in \mathcal{M}$. Either $u \in \{w, v\}$ or u is reachable from either one of the w_j^i or one of the roots s_i. . Furthermore, Ψ has only finitely many free variables. τ can thus be defined in steps as follows:

- $\tau(\overline{w_j^i}) = (j, (\sigma(\overline{w_j^i}), \lambda, \lambda)) = (j, (x_j^i, \lambda, \lambda))$.
- If $\tau(\overline{u})$ has been defined as $(j, (\vec{s}, \vec{a}, \vec{n}))$ and $u_1, \dots, u_k$ are all the elements of S_i such that $\overline{u_1}, \dots, \overline{u_k}$ are free in Ψ and $u \xrightarrow{b_l} u_l$ for some $b_l \in A$ (for each $l \in \{1, \dots, k\}$), then define:

$$\tau(\overline{u_l}) = (j, (\vec{s} \cdot \sigma(\overline{u_l}), \vec{a} \cdot b_l, \vec{n} \cdot l)).$$

- Likewise, if $s_i \xrightarrow{b_l} u_l$ for each $l \in \{1, \dots, k\}$, $u_1, \dots, u_k$ are all elements of $\mathcal{M}$ such that they can be reached in one step from the root s_i and $\overline{u_1}, \dots, \overline{u_k}$ are free in Ψ, then:

$$\tau(\overline{u_l}) = (j, (t_i \cdot \sigma(\overline{u_l}), b_l, l))$$

  where t_i is of course the root of $\mathfrak{T}_i$.

- Define $\tau(\overline{w}) = \tau(x) := \tau(\overline{w_1}) \cdot \ldots \cdot \tau(\overline{w_g})$ and $\tau(\overline{v}) = \tau(y) := \tau(\overline{v_1}) \cdot \ldots \cdot \tau(\overline{v_h})$.

Clearly then the functional sequence extension Z between $\mathcal{N}^\diamond$ and $\mathcal{N}$ (mentioned below definition 5.7) connects $\tau(\overline{u})$ to $\sigma(\overline{u})$ for every free variable $\overline{u}$ of Ψ. Now the following facts are easily verified:

1. $\tau(\overline{u})$ and $\sigma(\overline{u})$ satisfy the same MOD_m-formulas, as they are connected by the sequence extension Z.

2. $\sigma(\overline{u})R_{(a,i)}\sigma(\overline{u'})$ in $\mathcal{N}$ iff $\tau(\overline{u})R_{(a,i)}\tau(\overline{u'})$ in $\mathcal{N}^\diamond$.

3. $\tau(\overline{w}) = \tau(x)$ and $\tau(\overline{v}) = \tau(y)$.

It follows that $\mathcal{N}^\diamond \models \Psi[\tau]$. As there is a zigzagmorphism from each constituent process graph of $\mathcal{N}^\diamond$ to one of $\mathcal{N}$ and as the latter were assumed to be m-saturated, the constituent process graphs of $\mathcal{N}^\diamond$ are m-saturated. Furthermore, we made sure that τ assigns distinct values to distinct variables. Now if we could prove that $\mathcal{N}^\diamond \models \phi[\tau(\overline{w}), \tau(\overline{v})]$ then, as ϕ is safe for sequence extension and as $\tau(\overline{w})Z\sigma(\overline{w})$, $\mathcal{N} \models \phi[\sigma(\overline{w}), u]$, for some u with $\tau(\overline{v})Zu$. But we know that $\tau(\overline{v})Z\sigma(\overline{v})$, so as Z is functional, $u = \sigma(\overline{v})$, hence $\mathcal{N} \models \phi[\sigma]$ (as σ must agree on the values for x, y and $\overline{w}, \overline{v}$ respectively, by the definition of Ψ), which is what we are after. Henceforth, we make no further mention of $\mathcal{N}^\diamond$, but simply assume that $\mathcal{N}$ and σ itself have all the mentioned properties. We have shown that this can be done without loss of generality.

So we have two superstructures $\mathcal{M}$ and $\mathcal{N}$ whose constituent process graphs are m-saturated. By definition of Ψ we know that w and $x_1 \dots x_g$ satisfy the same MOD_m-formulas. This implies that the roots s_i and t_i of $\mathfrak{S}_i$ and $\mathfrak{T}_i$ respectively satisfy the same modal formulas (see the proof of theorem 4.4). Thus the relation of modal equivalence, Z_i, must be a bisimulation between $\mathfrak{S}_i$ and $\mathfrak{T}_i$, for each $1 \leq i \leq m$. By lemma 4.3, the sequence extension of these bisimulations, Z, is the relation of satisfying the same MOD_m-formulas. Z thus connects w to $x_1 \dots x_g$.

If $\sigma(\overline{v})$ were the only point in $\mathcal{N}$ linked via Z to v and vice versa, then we would be done. For then, as $\mathcal{M} \models \phi[w, v]$ and ϕ is safe for sequence extension, there must be a point u in $\mathcal{N}$ such that vZu and $\mathcal{N} \models \phi[\sigma(\overline{w}), u]$. By our assumption, u could only be $\sigma(\overline{v})$ so we would be done.

The problem is thus reduced to transforming the bisimulations Z_i such that their sequence extension links v only to $\sigma(\overline{v})$ and vice versa. This is done by repeatedly removing Z_i-links, in such a way as to ensure that for any u in $\mathfrak{S}_i$ such that $\overline{u}$ is free in Ψ, Z_i still connects u to $\sigma(\overline{u})$. In this way, their sequence extension will still link w to $\sigma(\overline{w})$ and v to $\sigma(\overline{v})$. In the end, we want the link between u and $\sigma(\overline{u})$ to be unique: neither u nor $\sigma(\overline{u})$ should then be connected to anything else. This should hold for any u with $\overline{u}$ free in Ψ.

We have made sure that $\mathfrak{S}_i = (\mathcal{S}, s_i)$ is the disjoint union of $(\mathcal{S}_{s_i}, s_i), (\mathcal{S}_{w_1^i}, w_1^i), \dots, (\mathcal{S}_{w_{k_i}^i}, w_{k_i}^i)$.[3] Likewise $\mathfrak{T}_i = (\mathcal{T}, t_i)$ is the disjoint union of $(\mathcal{T}_{t_i}, t_i), (\mathcal{T}_{x_1^i}, x_1^i), \dots, (\mathcal{T}_{x_{k_i}^i}, x_{k_i}^i)$, where $x_1^i \dots x_{k_i}^i$ is the T_i-part of $x_1 \dots x_g$.

First make sure that Z_i only connects $(\mathcal{S}_{s_i}, s_i)$ to $(\mathcal{T}_{t_i}, t_i)$, $(\mathcal{S}_{w_1^i}, w_1^i)$ to $(\mathcal{T}_{x_1^i}, x_1^i)$, etc., by removing all other links. Now we can concentrate on the bisimulation between $(\mathcal{S}_{w_l^i}, w_l^i)$ and $(\mathcal{T}_{x_l^i}, x_l^i)$, say. For convenience, let us call these $\mathfrak{A} = (\mathcal{A}, \alpha)$ and $\mathfrak{B} = (\mathcal{B}, \beta)$ respectively and let the restiction of Z_i to these process graphs be B.

Ψ describes a certain subtree in $\mathfrak{B}$, originating in β, that also occurs in $\mathfrak{A}$ (but here emerging from α). Our assumptions give us that these subtrees are isomorphic and that B links these subtrees in the sense that if u occurs in the subtree in $\mathfrak{A}$ then B links u to $\sigma(\overline{u})$ in $\mathfrak{B}$. But there may be *too many* links.

By standard techniques from modal logic, we can limit B further in such a way that:

1. B only connects points at the same *distance* from α and β.

2. If B connects z_1 and z_2 then either $z_1 = \alpha$ and $z_2 = \beta$ or there are unique $a \in A$, z_3 and z_4 such that Z_i connects z_3 and z_4, while $z_3 \stackrel{a}{\to} z_1$ and $z_4 \stackrel{a}{\to} z_2$.

For if we define

$$\begin{array}{lcl} B_0 & := & \{(\alpha, \beta)\} \\ B_{k+1} & := & \{(x, y) \in B \mid \exists (u, v) \in B_k . \exists a \in A . (u \stackrel{a}{\to} x \wedge v \stackrel{a}{\to} y)\} \end{array}$$

[3] If $\mathcal{S}$ is an unrooted Kripke model, and s is an element in its domain, then $\mathcal{S}_s$ is the submodel generated by s.

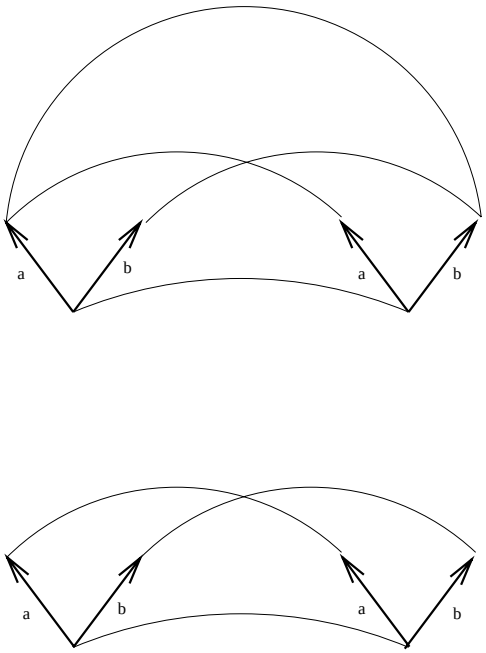


Figure 4: B before, and after.

then $\bigcup_{k\in\mathbb{N}} B_k$ is still a bisimulation between $\mathfrak{A}$ and $\mathfrak{B}$ (here we must use that $\mathfrak{A}$ and $\mathfrak{B}$ are unravelled), which moreover has the desired properties.

We may *still* have too many links. What we want is the following. We have two isomorphic subtrees (described by Ψ), one in $\mathfrak{A}$ and one in $\mathfrak{B}$. The corresponding points (corresponding via the isomorphism) on these subtrees are linked via B, as can easily be seen. But B may link points on the subtrees to other points besides the one to which it corresponds. We want to remove such extra, troublesome, links. This can be done by applying the following procedure from the roots up.

1. The roots (i.e. α and β) are on the subtrees and because of our earlier transformation of B are only linked to each other. So suppose we are considering points in the subtrees higher up.

2. A point on the subtree may be linked to a point on the other subtree, besides the point to which it corresponds. We may simply remove this link (and all links connecting the trees above these two points). The resulting relation is still a bisimulation that connects the subtrees. (For clarification, consult figure 4: the thick arrows denote part of the subtree, the other lines are B-links.)

3. A more serious problem occurs when B connects a point u on the subtree to points $(y_i)_{i\in I}$ that are not on the subtree. Because of the way we restricted B we know that the predecessor of u (call this point z and say $z \xrightarrow{a} u$) is linked to those of $(y_i)_{i\in I}$. We may assume that the work has been done for these lower points, so that B can only link a single point to z (because z is on the subtree) and this point, z', will be on the other subtree. As our models are multiplied, we can find an a-successor u' of z that is not on the subtree (which after all is finite) and whose generated tree is isomorphic to that of u. Now we remove the links from u to $(y_i)_{i\in I}$ (and those *above* these points) and replace them by links from u' to $(y_i)_{i\in I}$ (actually, these links will already be in place, as the bisimulation we started out with is the maximal one). The resulting relation still satisfies all our needs, as can easily be seen. See figure 5.

After performing all these lesions on Z_i, we may consider the sequence extension of the new bisimulations (call this Z again). This Z will have the property that v is only linked to $\sigma(\overline{v})$, and vice versa, as desired.

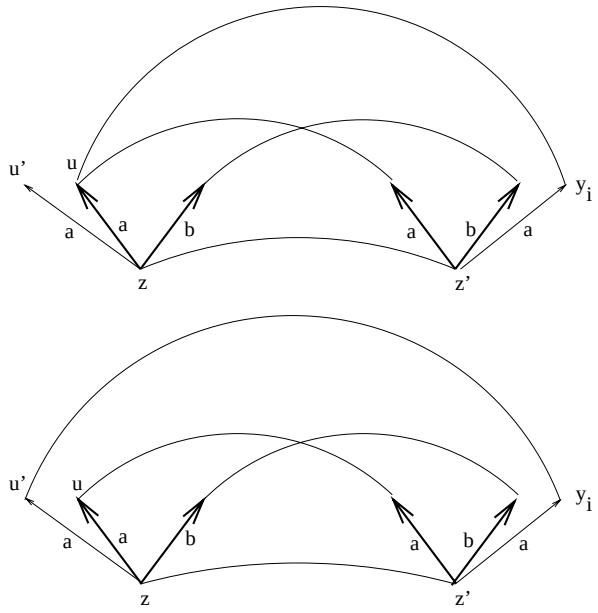


Figure 5: B before, and after.

So now we know that $\Psi \models_{\mathsf{SUP}} \phi$, thus by compactness there is a finite subset $\Psi_0(x, y, \vec{y})$ of Ψ such that $\Psi_0 \models_{\mathsf{SUP}} \phi$. We may assume that Ψ_0 is obtained from Ψ by simply restricting the MOD_m-descriptions $\Phi_u(\overline{u})$ to finite subsets of these. Now $\exists \vec{y}. \bigwedge \Psi_0 \models_{\mathsf{SUP}} \phi$. What remains to be seen is that $\exists \vec{y}. \bigwedge \Psi_0$ is SUP-equivalent to a REL_m-formula.

Let us consider our example from page 18 again. We may now assume each $\Phi_u(\overline{u})$ to be a finite set of (standard translations of) MOD_m-formulas. As $\Phi_u(\overline{u})$ is closed under conjunctions, let us equate $\Phi_u(\overline{u})$ with $\phi_u(\overline{u})$, some MOD_m-formula in the variable $\overline{u}$.

We will construe a REL_m-formula ψ, SUP-equivalent to this concrete example of $\exists \vec{y}. \bigwedge \Psi_0$.

First define

$$\theta := R_{(a,1)}; \phi_u?; \Delta; (R_{(b,1)} \bullet R_{(a,1)})$$

Then

$$\psi := \phi_w?; [\theta \bullet RES_\lambda \bullet (\lambda?; RES_{s_2}; R_{(b,2)})]; [\mathrm{ID}_2 \bullet \Delta]; \textsc{transpose}(2,3); \phi_v?.$$

where $\textsc{transpose}(2,3)$ is the program defined on page 13 that permutes the second and third position of a sequence. The reader may verify that for any superstructure $\mathcal{M}$: $\mathcal{M} \models \exists \vec{y}. \bigwedge \Psi_0[u_1, u_2]$ iff $u_1 \psi^{\mathcal{M}} u_2$.

In fact, similar tricks can be applied to prove that $\exists \vec{y} \bigwedge \Psi_0$ is SUP-equivalent to a REL_m-formula for an arbitrary Ψ_0 built in the manner described: simply find REL_m-formulas encoding the subtrees (possibly 'remembering' intermediate states), put these together, add some constants, copy some items and finally permute until everything is in the right order. To do this formally of course requires some tedious work, which does not seem didactic at this point.

We have argued that $\exists \vec{y}. \bigwedge \Psi_0$ is SUP-equivalent to a REL_m-formula ψ. As Ψ is satisfiable in $\mathcal{M}$ under the assignment taking $\overline{u}$ to u, x to w and y to v, $\mathcal{M} \models \psi[w, v]$. Furthermore, $\psi \models_{\mathsf{SUP}} \phi$. So we have proved that $\phi \models_{\mathsf{SUP}} \bigvee \{\psi \in REL_m \mid \psi \models_{\mathsf{SUP}} \phi\}$. By compactness we have $\psi_1, \ldots \psi_k \in REL_m$ which SUP-imply ϕ such that $\phi \models_{\mathsf{SUP}} \psi_1 \vee \ldots \vee \psi_k$. Thus $\mathsf{SUP} \models \phi \leftrightarrow (\psi_1 \vee \ldots \vee \psi_k)$. REL_m is closed under finite disjunction, as $\theta_1 \vee \theta_2 \equiv \theta_1 \cup \theta_2$ and the empty disjunction $\bot$ is equivalent to the REL_m-formula $(\bot?)^\circ$. $(\psi_1 \vee \ldots \vee \psi_k)$, and thus ϕ is SUP-equivalent to a REL_m-formula.

End of the proof of the Main Theorem

6 Modal preconditions

In [8] the following question is posed:

> How can modal statements about some process algebra construct be reduced to statements about its components?

In the current setting, the question can be stated as follows. Given an operation O defined using REL_m- and MOD_m-formulas and a modal formula ϕ, under what conditions is $O(\vec{\mathfrak{S}}), \sigma \Vdash \phi$ true (where σ is the root)? We are not asking for a simple truth condition here, but an answer in terms of modal truth at the roots of the input models.

Some examples (the process-operations are defined as on page 14):

- $\mathfrak{S} + \mathfrak{T}, c \Vdash [a]\surd$ iff $\mathfrak{S}, s \Vdash [a]\surd$ and $\mathfrak{T}, t \Vdash [a]\surd$.
- $\mathfrak{S} \cdot \mathfrak{T}, s \Vdash [a]\surd$ iff
$$\left(\begin{array}{l} \mathfrak{S}, s \Vdash \neg\surd \wedge [a]\bot \\ \text{or} \\ \mathfrak{S}, s \Vdash [a]\bot \text{ and } \mathfrak{T}, t \Vdash [a]\surd \\ \text{or} \\ \mathfrak{S}, s \Vdash [a]\surd \text{ and } \mathfrak{T}, t \Vdash \surd \wedge [a]\surd \\ \text{or} \\ \mathfrak{S}, s \Vdash \neg\surd \wedge [a]\surd \text{ and } \mathfrak{T}, t \Vdash \surd. \end{array}\right)$$

This section will discuss an algorithm for calculating such *modal preconditions*.

Definition 6.1 *A* **root-description** *is a MOD_m-formula of the form:*

$$GOTO_{s_1}(\phi_1) \wedge \ldots \wedge GOTO_{s_m}(\phi_m)$$

where each ϕ_i is an i-formulas (see page 9). □

A root-description is really nothing but a statement about modal satisfaction of certain formulas at the roots of the input models. That is, if $\vec{\mathfrak{S}}$ is an m-sequence of process graphs, then:

$$\begin{array}{c} \mathbb{S}(\vec{\mathfrak{S}}), u \Vdash GOTO_{s_1}(\phi_1) \wedge \ldots \wedge GOTO_{s_m}(\phi_m) \\ \text{iff} \\ \mathfrak{S}_1, s_1 \Vdash (\phi_1)^{-1} \text{ and } \ldots \text{ and } \mathfrak{S}_m s_m \Vdash (\phi_m)^{-m} \end{array}$$

where u is any element of the superstructure (recall that if ϕ is an i-formula, ϕ^{-i} is the result of removing all the i-labels). Thus any boolean combination of root-descriptions is really a statement about the input models.

Theorem 6.2 (Modal Precondition Computation) *If O is a safe/invariant operation and ϕ is an ordinary modal formula, then we may compute a disjunction of root-descriptions ψ such that for any m-sequence of process graphs $\vec{\mathfrak{S}}$:*

$$O(\vec{\mathfrak{S}}), \sigma \Vdash \phi \Longleftrightarrow \mathbb{S}(\vec{\mathfrak{S}}), \sigma \Vdash \psi$$

where σ is the root of $O(\vec{\mathfrak{S}})$.

Proof.
Suppose O is defined by REL_m-formulas ϕ_a (for each $a \in A$), MOD_m-formulas ϕ_p (for each $p \in$ PROP) and a sequence of constants $t_1 \ldots t_k$. We can translate ϕ into a MOD_m-formula ϕ' (lemma 6.8) such that, if $\mathcal{M} = \mathbb{S}(\vec{\mathfrak{S}})$:

$$O(\vec{\mathfrak{S}}), \sigma \Vdash \phi \text{ iff } \mathcal{M}, \sigma \Vdash \phi'$$

where $\sigma = t_1^{\mathcal{M}} \dots t_k^{\mathcal{M}}$. But for any MOD_m-formula χ we can compute a disjunction of root-descriptions $GOTO_{t_1 \dots t_k}(\chi)$ (see lemma 6.13) such that for any superstructure $\mathcal{N}$:

$$\mathcal{N}, x \Vdash GOTO_{t_1 \dots t_k}(\chi) \text{ iff } \mathcal{N}, t_1^{\mathcal{N}} \dots t_k^{\mathcal{N}} \Vdash \chi$$

In the present case:

$$\mathcal{M}, \sigma \Vdash \phi' \text{ iff } \mathcal{M}, \sigma \Vdash GOTO_{t_1 \dots t_k}(\phi').$$

Therefore $GOTO_{t_1 \dots t_k}(\phi')$ is the desired formula. □

The rest of the section is devoted to proving the lemmas mentioned in the above proof.

Definition 6.3 *For $k \leq n$, a* **k-formula** *is a MOD_m-formula ϕ of the form:*

$$(\lambda \wedge \phi_0) \bullet (1 \wedge \phi_1) \bullet \dots \bullet (1 \wedge \phi_k)$$

where 1 is an abbreviation for the formula $lth(1)$. □

Lemma 6.4 *For every $\phi \in MOD_m$ there is a disjunction of k-formulas $\alpha_k(\phi)$ such that whenever u is an element of a superstructure $\mathcal{M}$ and $|u| = k$ then $\mathcal{M}, u \Vdash \phi$ iff $\mathcal{M}, u \Vdash \alpha_k(\phi)$.*

Proof.

- $\alpha_0(\phi) := \lambda \wedge \phi$.
- $\alpha_1(\phi) := (\lambda \wedge \top) \bullet (1 \wedge \phi)$.

From now on, assume $k \geq 2$.

- $\alpha_k(\top) := (\lambda \wedge \top) \bullet (1 \wedge \top) \bullet \dots \bullet (1 \wedge \top)$.
- $\alpha_k(p_i) = \alpha_k(S_i) = \alpha_k(c) = \alpha_k(\lambda) = \alpha_k(\langle a, i \rangle \phi) := \bot$. As we assume $\bot$ to be the empty disjunction, this is of the right form.
- $\alpha_k(\phi \vee \psi) := \alpha_k(\phi) \vee \alpha_k(\psi)$.
- $\alpha_k(\neg\phi)$ is calculated as follows. First calculate $\alpha_k(\phi)$. Negate this and bring the negation inwards, with De Morgan laws. This gives us a conjunction of formulas of the form:

  $$\neg[(\lambda \wedge \phi_0) \bullet (1 \wedge \phi_1) \bullet \dots \bullet (1 \wedge \phi_k)].$$

  On sequences of length k such formulas are equivalent to:

  $$\begin{array}{l} [(\lambda \wedge \neg\phi_0) \bullet (1 \wedge \top) \bullet \dots \bullet (1 \wedge \top)] \\ \vee \\ [(\lambda \wedge \top) \bullet (1 \wedge \neg\phi_1) \bullet \dots \bullet (1 \wedge \top)] \\ \vee \\ \vdots \\ \vee \\ [(\lambda \wedge \top) \bullet (1 \wedge \top) \bullet \dots \bullet (1 \wedge \neg\phi_k)]. \end{array}$$

  This gives us a conjunction of disjunctions of k-formulas. Distribution transforms this into a disjunction of conjunctions of k-formulas. Conjunctions of k-formulas are equivalent to k-formulas:

  $$\begin{array}{c} [(\lambda \wedge \phi_0) \bullet (1 \wedge \phi_1) \bullet \dots \bullet (1 \wedge \phi_k)] \wedge [(\lambda \wedge \psi_0) \bullet (1 \wedge \psi_1) \bullet \dots \bullet (1 \wedge \psi_k)] \\ \equiv \\ (\lambda \wedge \phi_0 \wedge \psi_0) \bullet (1 \wedge \phi_1 \wedge \psi_1) \bullet \dots \bullet (1 \wedge \phi_k \wedge \psi_k). \end{array}$$

  Thus we obtain our desired disjunction of k-formulas.

- $\alpha_k(GOTO_{s_i}(\phi)) := (\lambda \wedge GOTO_{s_i}(\phi)) \bullet (1 \wedge \top) \bullet \ldots \bullet (1 \wedge \top)$.
- $\alpha_k(\phi \bullet \psi)$ is arrived at by the following procedure. Start off with

$$[\alpha_0(\phi) \bullet \alpha_k(\psi)] \vee \ldots \vee [\alpha_k(\phi) \bullet \alpha_0(\psi)].$$

If we could show that, whenever χ is a disjunction of k-formulas and ρ is a disjunction of l-formulas, then $\chi \bullet \rho$ is equivalent to a disjunction of $k+l$-formulas, we would be done. So suppose we have such $\chi \bullet \rho$. Disjunctions can be brought outward by the following laws:

$$\begin{aligned} (\alpha \vee \beta) \bullet \gamma &\equiv (\alpha \bullet \gamma) \vee (\beta \bullet \gamma) \\ \alpha \bullet (\beta \vee \gamma) &\equiv (\alpha \bullet \beta) \vee (\alpha \bullet \gamma) \end{aligned}$$

This gives us a disjunction of formulas of the form:

$$[(\lambda \wedge \chi_0) \bullet (1 \wedge \chi_1) \bullet \ldots \bullet (1 \wedge \chi_k)] \bullet [(\lambda \wedge \rho_0) \bullet (1 \wedge \rho_1) \bullet \ldots \bullet (1 \wedge \rho_l)].$$

But such formulas are equivalent to:

$$(\lambda \wedge \chi_0 \wedge \rho_0) \bullet (1 \wedge \chi_1) \bullet \ldots \bullet (1 \wedge \chi_k) \bullet (1 \wedge \rho_1) \bullet \ldots \bullet (1 \wedge \rho_l).$$

This is due to the following two equivalences concerning λ, which may easily be verified.

$$\begin{aligned} (\lambda \wedge \alpha) \bullet \beta &\equiv \beta \bullet (\lambda \wedge \alpha) \\ (\lambda \wedge \alpha) \bullet (\lambda \wedge \beta) &\equiv \lambda \wedge \alpha \wedge \beta \end{aligned}$$

□

Lemma 6.5 *For every $\phi \in MOD_m$ there is a MOD_m-formula $GOTO_\lambda(\phi)$ such that $u \Vdash GOTO_\lambda(\phi)$ iff $\lambda \Vdash \phi$, for any u in a superstructure.*

Proof.
Notice that theorem 4.4 already gives us that such a formula exists: if $\phi \in MOD_m$ then $\exists x.(C(x,x,x) \wedge (\phi)^\circ)$ is invariant for sequence extension, so it must be SUP-equivalent to a MOD_m-formula. Theorem 4.4 does not give us an effective method of finding this formula, however. The following gives us such a method.

- $GOTO_\lambda(\top) := \top$.
- $GOTO_\lambda(p_i) = GOTO_\lambda(S_i) = GOTO_\lambda(c) = GOTO_\lambda(\langle a, i\rangle \phi) := \bot$.
- $GOTO_\lambda(\lambda) := \top$.
- $GOTO_\lambda(\phi \vee \psi) := GOTO_\lambda(\phi) \vee GOTO_\lambda(\psi)$.
- $GOTO_\lambda(\neg\phi) := \neg GOTO_\lambda(\phi)$.
- $GOTO_\lambda(GOTO_{s_i}(\phi)) := GOTO_{s_i}(\phi)$.
- $GOTO_\lambda(\phi \bullet \psi) := GOTO_\lambda(\phi) \wedge GOTO_\lambda(\psi)$. □

Lemma 6.6 *For every $\phi \in MOD_m$ and $c \in$ CONS there is a MOD_m-formula $GOTO_c(\phi)$ such that $u \Vdash GOTO_c(\phi)$ iff $c \Vdash \phi$, for any u in a superstructure.*

Proof.

- $GOTO_c(\top) := \top$.

- $GOTO_c(p_i) = GOTO_c(S_i) = GOTO_c(\lambda) = GOTO_c(\langle a, i\rangle\phi) := \bot$.
- $GOTO_c(d) := \begin{cases} \top & \text{if } c = d. \\ \bot & \text{else.} \end{cases}$ (where $d \in$ CONS).
- $GOTO_c(\phi \vee \psi) := GOTO_c(\phi) \vee GOTO_c(\psi)$.
- $GOTO_c(\neg\phi) := \neg GOTO_c(\phi)$.
- $GOTO_c(GOTO_{s_i}(\phi)) := GOTO_{s_i}(\phi)$.
- $GOTO_c(\phi \bullet \psi) := [GOTO_\lambda(\phi) \wedge GOTO_c(\psi)] \vee [GOTO_c(\phi) \wedge GOTO_\lambda(\psi)]$. □

Lemma 6.7 *Let $\theta \in REL_m$ and $\phi \in MOD_m$. Then there is a MOD_m-formula $\langle\theta\rangle\phi$ such that if $\mathcal{M}$ is a superstructure:*

$$\mathcal{M}, u \Vdash \langle\theta\rangle\phi \quad \textit{iff} \quad \textit{there is a } v \textit{ such that } u\theta^{\mathcal{M}} v \textit{ and } \mathcal{M}, v \Vdash \phi.$$

Proof.
We prove this by first adding diamonds $\langle\theta\rangle$ for every $\theta \in REL_m$ and then showing that these can be effectively eliminated, by induction on θ.

Notice that if ϕ is some disjunction $\phi_1 \vee \ldots \vee \phi_k$ then $\langle\theta\rangle\phi$ is equivalent to $\langle\theta\rangle\phi_1 \vee \ldots \vee \langle\theta\rangle\phi_k$. We will refer to this operation as 'bringing the disjunctions outside'.

- $\langle R_{(a,i)}\rangle\phi \equiv \langle a, i\rangle\phi$.
- $\langle\phi?\rangle\psi \equiv \phi \wedge \psi$.
- $\langle\Delta\rangle\phi$ is calculated as follows. Notice that $\langle\Delta\rangle\phi$ is equivalent to a formula $\langle\Delta\rangle[\alpha_2(\phi) \vee \ldots \vee \alpha_n(\phi)]$ (with α_i as in lemma 6.4) After bringing the disjunctions outside, what is left is a disjunction of formulas of the form:
$$\langle\Delta\rangle[(\lambda \wedge \phi_0) \bullet (1 \wedge \phi_1) \bullet \ldots \bullet (1 \wedge \phi_k)]$$
for $2 \leq k \leq n$. Such a formula is equivalent to:
$$(\lambda \wedge \phi_0) \bullet (1 \wedge \phi_1 \wedge \phi_2) \bullet (1 \wedge \phi_3) \bullet \ldots \bullet (1 \wedge \phi_k)$$
so we have our desired MOD_m-formula.
- $\langle$PERMUTE$\rangle\phi$ is calculated likewise. We may again assume that ϕ is a disjunction of k-formulas, with k ranging over $2, \ldots, n$. Thus we may bring the disjunction outside to get a disjunction of formulas of the form:
$$\langle\text{PERMUTE}\rangle[(\lambda \wedge \phi_0) \bullet (1 \wedge \phi_1) \bullet \ldots \bullet (1 \wedge \phi_k)]$$
for $2 \leq k \leq n$. Such a formula is equivalent to:
$$(\lambda \wedge \phi_0) \bullet (1 \wedge \phi_2) \bullet (1 \wedge \phi_1) \bullet (1 \wedge \phi_3) \bullet \ldots \bullet (1 \wedge \phi_k).$$
- $\langle RES_{s_i}\rangle\phi \equiv GOTO_{s_i}(\phi)$.
- $\langle RES_c\rangle\phi \equiv GOTO_c(\phi)$ (see lemma 6.6).
- $\langle RES_\lambda\rangle\phi \equiv GOTO_\lambda(\phi)$ (see lemma 6.5).
- $\langle\theta_1 \cup \theta_2\rangle\phi \equiv \langle\theta_1\rangle\phi \vee \langle\theta_2\rangle\phi$.
- $\langle\theta_1; \theta_2\rangle\phi \equiv \langle\theta_1\rangle\langle\theta_2\rangle\phi$.

- $\langle \sim \theta \rangle \phi \equiv \phi \wedge \neg \langle \theta \rangle \top$.
- Finally, we calculate $\langle \theta_1 \bullet \theta_2 \rangle \phi$. Assume that ϕ is a disjunction of k-formulas, with k ranging over $0, \ldots, n$. Now bring the disjunctions outside. This gives us a disjunction of formulas of the form:

$$\langle \theta_1 \bullet \theta_2 \rangle [(\lambda \wedge \phi_0) \bullet (1 \wedge \phi_1) \bullet \ldots \bullet (1 \wedge \phi_k)].$$

As this is equivalent to:

$$\begin{array}{l} [\langle \theta_1 \rangle (\lambda \wedge \phi_0)] \bullet [\langle \theta_2 \rangle ((1 \wedge \phi_1) \bullet \ldots \bullet (1 \wedge \phi_k))] \\ \vee \\ [\langle \theta_1 \rangle ((\lambda \wedge \phi_0) \bullet (1 \wedge \phi_1))] \bullet [\langle \theta_2 \rangle ((1 \wedge \phi_2) \bullet \ldots \bullet (1 \wedge \phi_k))] \\ \vee \\ \ldots \\ \vee \\ [\langle \theta_1 \rangle ((\lambda \wedge \phi_0) \bullet (1 \wedge \phi_1) \bullet \ldots \bullet (1 \wedge \phi_{k-1}))] \bullet [\langle \theta_2 \rangle (1 \wedge \phi_k)] \\ \vee \\ [\langle \theta_1 \rangle ((\lambda \wedge \phi_0) \bullet (1 \wedge \phi_1) \bullet \ldots \bullet (1 \wedge \phi_k))] \bullet [\langle \theta_2 \rangle \lambda] \end{array}$$

we are done: $\langle \theta_1 \bullet \theta_2 \rangle \phi$ is then a disjunction of such formulas. □

Lemma 6.8 *Let O be a safe/invariant operation and ϕ any modal formula. Let ϕ' be the result of substituting each p by ϕ_p and each $\langle a \rangle$ by $\langle \phi_a \rangle$. By lemma 6.7, ϕ' is (equivalent to) a MOD_m-formula. Furthermore:*

$$O(\vec{\mathfrak{S}}), u \Vdash \phi \quad \textit{iff} \quad \mathbb{S}(\vec{\mathfrak{S}}), u \Vdash \phi'$$

For any m-sequence of process graphs and any $u \in \mathbb{S}(\vec{\mathfrak{S}})$ (recall that $O(\vec{\mathfrak{S}})$ and $\mathbb{S}(\vec{\mathfrak{S}})$ share the same universe). □

Definition 6.9 *An i-**description** (with $1 \leq i \leq m$) is a MOD_m-formula of the form $\phi \wedge \psi$, where ϕ is an i-formula and ψ is a root-description.*

Notice that any root-description ψ is equivalent to an i-description, for any i, as $\psi \equiv (\top \wedge \psi)$.

Lemma 6.10 *If ϕ is a disjunction of i-descriptions then $GOTO_{s_i}(\phi)$ is equivalent to a disjunction of root-descriptions.*

Proof.
As $GOTO_{s_i}$ distributes over disjunctions, bring these disjunctions outside first. This gives us a disjunction of formulas of the form:

$$GOTO_{s_i}(\phi_0 \wedge GOTO_{s_1}(\phi_1) \wedge \ldots \wedge GOTO_{s_i}(\phi_i) \wedge \ldots \wedge GOTO_{s_m}(\phi_m))$$

where ϕ_0 is an i-formula and for any $1 \leq j \leq m$, ϕ_j is a j-formula. But this is equivalent to:

$$GOTO_{s_1}(\phi_1) \wedge \ldots \wedge GOTO_{s_i}(\phi_i \wedge \phi_0) \wedge \ldots \wedge GOTO_{s_m}(\phi_m))$$

so we have our disjunction of root-descriptions. □

Lemma 6.11 *Any boolean combination of i-descriptions is equivalent to a disjunction of i-descriptions. The same goes for root-descriptions.*

Proof.
By standard propositional logic, we may assume that a boolean combination ϕ of i-descriptions is a disjunction of formulas of the form $\phi_1 \wedge \ldots \wedge \phi_k$, where each ϕ_j is either an i-description or a negation of such. Negations of i-descriptions are equivalent to disjunctions of i-descriptions:

$$\begin{array}{c} \neg(\phi_0 \wedge GOTO_{s_1}(\phi_1) \wedge \ldots \wedge GOTO_{s_m}(\phi_m)) \\ \equiv \\ [\neg\phi_0 \wedge GOTO_{s_1}(\top) \wedge \ldots \wedge GOTO_{s_m}(\top)] \\ \vee \\ [\top \wedge GOTO_{s_1}(\neg\phi_1) \wedge GOTO_{s_2}(\top) \wedge \ldots \wedge GOTO_{s_m}(\top)] \\ \vee \\ \vdots \\ \vee \\ [\top \wedge GOTO_{s_1}(\top) \wedge \ldots \wedge GOTO_{s_{k-1}}(\top) \wedge GOTO_{s_m}(\neg\phi_m)] \end{array}$$

So ϕ is equivalent to a formula consisting solely of i-descriptions, conjunctions and disjunctions. After applying distribution laws, ϕ can be seen to be equivalent to a disjunction of conjunctions of i-descriptions. But a finite conjunction of i-descriptions is again an i-description, as the following equalities will testify:

$$\begin{array}{c} [\phi_0 \wedge GOTO_{s_1}(\phi_1) \wedge \ldots \wedge GOTO_{s_m}(\phi_m)] \wedge \\ [\psi_0 \wedge GOTO_{s_1}(\psi_1) \wedge \ldots \wedge GOTO_{s_m}(\psi_m)] \\ \equiv \\ (\phi_0 \wedge \psi_0) \wedge GOTO_{s_1}(\phi_1 \wedge \psi_1) \wedge \ldots \wedge GOTO_{s_m}(\phi_m \wedge \psi_m) \end{array}$$

$$\top \quad \equiv \quad \top \wedge GOTO_{s_1}(\top) \wedge \ldots \wedge GOTO_{s_m}(\top)$$

So ϕ is equivalent to a disjunction of i-formulas.

Using the same reasoning any boolean combination of root-descriptions must be equivalent to a disjunction of root-descriptions. □

Lemma 6.12 *There is a (computable) function β_i on MOD_m-formulas, such that for any $\phi \in MOD_m$, $\beta_i(\phi)$ is a disjunction of i-descriptions and for any superstructure $\mathcal{M} = \mathbb{S}(\vec{\mathfrak{S}})$ and $u \in S_i$:*

$$\mathcal{M}, u \Vdash \phi \;\; \textit{iff} \;\; \mathcal{M}, u \Vdash \beta_i(\phi).$$

Proof.
The proof goes by induction on MOD_m-formulas ϕ.

- $\beta_i(\top) := \top$. $\top$ is not strictly a disjunction of i-descriptions, but it *is* a boolean combination of such, so by the above lemma, it is equivalent to a disjunction of i-descriptions. In the rest of the proof we will often leave out these details.
- $\beta_i(p_j) := \begin{cases} p_i \wedge GOTO_{s_1}(\top) \wedge \ldots \wedge GOTO_{s_m}(\top) & \text{if } i = j. \\ \bot & \text{else.} \end{cases}$
- $\beta_i(S_i) := \begin{cases} \top & \text{if } i = j. \\ \bot & \text{else.} \end{cases}$
- $\beta_i(c) = \beta_i(\lambda) := \bot$.
- $\beta_i(\langle a, j\rangle\phi) := \begin{cases} \langle a, i\rangle\beta_i(\phi) & \text{if } i = j. \\ \bot & \text{else.} \end{cases}$

  $\langle a, i\rangle\beta_i(\phi)$ is not a disjunction of i-descriptions, so our definition is clearly wrong here. We fix it as follows. $\beta_i(\phi)$ is a disjunction of i-descriptions. As $\langle a, i\rangle$ distributes over disjunction,

$\langle a,i\rangle\beta_i(\phi)$ is equivalent to $\langle a,i\rangle\psi_1 \vee \ldots \vee \langle a,i\rangle\psi_k$, where each $\psi_1,\ldots,\psi_k$ is an i-description. Using:

$$\langle a,i\rangle(\phi_0 \wedge GOTO_{s_1}(\phi_1) \wedge \ldots \wedge GOTO_{s_m}(\phi_m)\rangle)$$
$$\equiv$$
$$(\langle a,i\rangle\phi_0) \wedge GOTO_{s_1}(\phi_1) \wedge \ldots \wedge GOTO_{s_m}(\phi_m)\rangle)$$

we get our desired disjunction of i-descriptions.

- $\beta_i(\phi \vee \psi) := \beta_i(\phi) \vee \beta_i(\psi)$.
- $\beta_i(\neg\phi) := \neg\beta_i(\phi)$. Using lemma 6.11, $\neg\beta_i(\phi)$ is equivalent to a formula of the desired form.
- $\beta_i(GOTO_{s_j}(\phi)) := GOTO_{s_j}(\beta_j(\phi))$. With lemma 6.10 this gives us a disjunction of root-descriptions, and hence a disjunction of i-descriptions.
- $\beta_i(\phi \bullet \psi) := (GOTO_\lambda(\beta_i(\phi)) \wedge \beta_i(\psi)) \vee (GOTO_\lambda(\beta_i(\psi)) \wedge \beta_i(\phi))$. Inspecting the definition of $GOTO_\lambda$ it is easily seen that $GOTO_\lambda(\beta_i(\phi))$ and $GOTO_\lambda(\beta_i(\psi))$ give us disjunctions of i-descriptions, hence we have a boolean combination of i-descriptions. □

Lemma 6.13 *Let $t_1 \ldots t_k$ be a sequence of $\mathcal{L}_m$-constants with $k \leq n$. Let ϕ be a MOD_m-formula. Then there is a MOD_m-formula $GOTO_{t_1\ldots t_k}(\phi)$ equivalent to a disjunction of root-descriptions such that:*

$$\mathcal{M},u \Vdash GOTO_{t_1\ldots t_k}(\phi) \Longleftrightarrow \mathcal{M},\sigma \Vdash \phi$$

where $\mathcal{M}$ is some superstructure, u is any element in its domain and $\sigma = t_1^{\mathcal{M}} \ldots t_k^{\mathcal{M}}$.

Proof.
If we change the definition of $GOTO_\lambda$ such that

$$GOTO_\lambda(GOTO_{s_i}(\phi)) := GOTO_{s_i}(\beta_i(\phi))$$

then the operation $GOTO_\lambda$ can easily be seen to always give us a boolean combination of root-descriptions. A similar remark suffices for $GOTO_c$.

$GOTO_{s_i}(\phi)$ is equivalent to $GOTO_{s_i}(\beta_i(\phi))$, which is equivalent to a disjunction of root-descriptions (lemma 6.10).

Now for the general case, where k may be greater than 1. Clearly $GOTO_{t_1\ldots t_k}(\phi)$ will be equivalent to $GOTO_{t_1\ldots t_k}(\alpha_k(\phi))$. $GOTO_{t_1\ldots t_k}$ will also distribute over disjunctions, so the problem is reduced to finding a formula equivalent to:

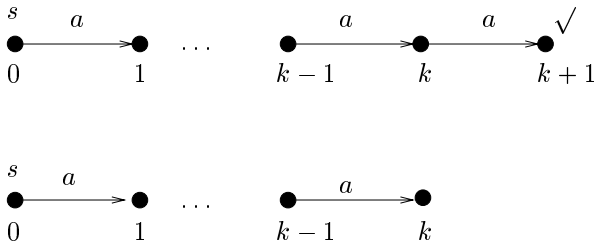
$$GOTO_{t_1\ldots t_k}((\lambda \wedge \phi_0) \wedge (1 \wedge \phi_1) \wedge \ldots \wedge (1 \wedge \phi_k)).$$

This is of course: $GOTO_\lambda(\phi_0) \wedge GOTO_{t_1}(\phi_1) \wedge \ldots \wedge GOTO_{t_k}(\phi_k)$. □

We have seen how to reduce modal truth in the output of safe/invariant operations to modal truth in the input process graphs. In [15] it is shown that something similar can be done for first-order sentences in general. The notion of first-order definable operations employed in the latter paper is slightly different from the one employed here, but the method can easily be applied to the present setting.

This gives us a method for determining whether an operation is first-order definable or not. To demonstrate this, consider again the operation TERMINATE, as defined on page 4. We will show that TERMINATE is not first-order definable, not even modulo bisimulation. To obtain a contradiction, suppose that TERMINATE *is* first-order definable, employing σ as its root and $\phi_\surd$ as the defining formula for success. Furthermore, suppose that $a \in A$ (if A is nonempty, TERMINATE becomes trivially first-order definable). Note that $\mathfrak{S}$ terminates iff TERMINATE$(\mathfrak{S}) \models \exists x.(x = \sigma \wedge \phi_\surd)$, where '$x = \sigma$' is an obvious abbreviation if σ is not of length one. By the reduction mentioned above, we can reduce the truth of $\exists x.(x = \sigma \wedge \phi_\surd)$ to the truth of a first-order sentence in $\mathfrak{S}$. That is, there must be a first-order sentence ψ such that TERMINATE$(\mathfrak{S}) \models \exists x.(x = \sigma \wedge \phi_\surd)$ iff $\mathfrak{S} \models \psi$. But then ψ

characterises termination of a process graph. Furthermore, ψ must be invariant for bisimulation, as two bisimilar process graphs either both terminate or neither of them do. By the familiar invariance theorem of [3], ψ must then be equivalent to a modal formula. This modal formula comes equipped with a certain *degree*: it can only distinguish up to certain depth from the root. Suppose this degree is k. Then ψ cannot distinguish between the following two models:



The top model terminates, while the bottom one does not, yet modal formulas of degree k cannot distinguish between the two. So, in particular, ψ does not express termination and we have reached our contradiction: TERMINATE is not first-order definable.

7 Further research

We have characterised those first-order operations whose defining formulas are safe or invariant for sequence extension within the larger class of first-order operations. For process algebraists this class is of interest, because all ACP-operations reside in this class. The format also allows a method for uniformly reducing truth of modal formulas ('Hennessy-Milner formulas' in the process algbebra tradition) in outputs of process operations to truth in their inputs, as demonstrated in the last section.

Further research could consist of studying other formats besides the first-order one. Examples are infinitary languages, which would enable us to define infinitary operations, and second order languages, which would make TERMINATE definable. Also, we may consider other important notions of process-equivalence, such as (rooted) branching bisimulation. Finally, it would be of interest to compare the expressivity of the present framework to that of restricted formats of TSS's (Transition System Specifications, [12]), which is originally a way of specifying term-based transition systems, but may also be viewed as a way of specifying operations on process graphs.

References

[1] H. Andréka, J.F.A.K. van Benthem, and I. Németi. Back and Forth between Modal Logic and Classical Logic. To appear in *Bulletin of the Interest Group in Pure and Applied Logics*, 1994.

[2] J.C.M. Baeten and W.P. Weijland. *Process Algebra*, volume 18 of *Cambridge Tracts in Theoretical Computer Science*. Cambridge University Press, 1990, 1994.

[3] J.F.A.K. van Benthem. *Modal Correspondence Theory*. PhD thesis, Mathematisch Instituut & Instituut voor Grondslagenonderzoek, University of Amsterdam, 1976.

[4] J.F.A.K. van Benthem. *Language in Action. Categories, Lambdas and Dynamic Logic*, volume 130 of *Studies in Logic*. North-Holland, Amsterdam, 1991.

[5] J.F.A.K. van Benthem. Programming Operations that are Safe for Bisimulation. CSLI Report 93-179, Center for the Study of Language and Information, Stanford University, 1993. to appear in *Studia Logica*.

[6] J.F.A.K. van Benthem. A Modal Perspective on Process Operations. draft in progress, 1994.

[7] J.F.A.K. van Benthem and J.A. Bergstra. Logic of Transition Systems. Report P9308, Programming Research Group, University of Amsterdam, 1993.

[8] J.F.A.K. van Benthem, J. van Eijck, and V. Stebletsova. Modal Logic, Transition Systems and Processes. *Journal of Logic and Computation*, 4(5):811–855, 1994.

[9] J.A. Bergstra and J.W. Klop. Process Algebra for Synchronous Communication. *Information and Control*, 60:82–95, 1984.

[10] R. Cleaveland. On Automatically Distinguishing Inequivalent Processes. In R. Kursham and E.M. Clarke, editors, *1990 Workshop on Computer-Aided Verification*. DIMACS technical report 90-31, Vol. 2, New Jersey, 1990.

[11] R. Goldblatt. Saturation and the Hennessy-Milner Property. Research report 94-145, Victoria University of Wellington, 1994.

[12] J.F. Groote. Transition System Specifications with Negative Premises. *Theoretical Computer Science*, 118:261–299, 1993.

[13] J.F. Groote and A. Ponse. The syntax and semantics of μCRL. Report CS-R9076, CWI, Amsterdam, 1990.

[14] W. Hodges. *Model Theory*, volume 42 of *Encyclopedia of Mathematics and Its Applications*. Cambridge University Press, 1993.

[15] M.J. Hollenberg. Bisimulation, Saturation and First Order Operations. In S. Fischer and M. Trautwein, editors, *Proceedings Accolade '95*, pages 75–90, Department of Mathematics and Computer Science, University of Amsterdam, 1995. Dutch Graduate School in Logic.

[16] M.J. Hollenberg. Hennessey-Milner Classes and Process Algebra. In M. de Rijke A. Ponse and Y. Venema, editors, *Modal Logic and Process Algebra: a Bisimulation Perspective*, volume 53 of *CSLI Lecture Notes*, pages 187–216. CSLI Publications, 1995.

[17] M.J. Hollenberg. Safety for Bisimulation in General Modal Logic. unpublished manuscript, to appear in the proceedings of the Tenth Amsterdam Colloquium, 1995.

[18] H.J. Keisler. Ultraproducts and elementary classes. *Indagationes Mathematicae*, 23:477–495, 1961.

[19] D. Park. Concurrency and Automata on Infinite Sequences. In *Proceedings 5th GI Conference*, pages 167–183. Springer, 1981.

[20] M. de Rijke. *Extending Modal Logic*. PhD thesis, ILLC-dissertation series 1993-4, 1993.