

Foundations of compositional program refinement (second version)

Citation for published version (APA):

Gerth, R. T. (1990). *Foundations of compositional program refinement (second version)*. (Computing science notes; Vol. 9003). Technische Universiteit Eindhoven.

Document status and date:

Published: 01/01/1990

Document Version:

Publisher's PDF, also known as Version of Record (includes final page, issue and volume numbers)

Please check the document version of this publication:

- A submitted manuscript is the version of the article upon submission and before peer-review. There can be important differences between the submitted version and the official published version of record. People interested in the research are advised to contact the author for the final version of the publication, or visit the DOI to the publisher's website.
- The final author version and the galley proof are versions of the publication after peer review.
- The final published version features the final layout of the paper including the volume, issue and page numbers.

[Link to publication](#)

General rights

Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

- Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
- You may not further distribute the material or use it for any profit-making activity or commercial gain
- You may freely distribute the URL identifying the publication in the public portal.

If the publication is distributed under the terms of Article 25fa of the Dutch Copyright Act, indicated by the "Taverne" license above, please follow below link for the End User Agreement:

www.tue.nl/taverne

Take down policy

If you believe that this document breaches copyright please contact us at:

openaccess@tue.nl

providing details and we will investigate your claim.

Foundations of Compositional Program Refinement
(second version)

by

Rob Gerth

90/3

September, 1990

COMPUTING SCIENCE NOTES

This is a series of notes of the Computing Science Section of the Department of Mathematics and Computing Science Eindhoven University of Technology. Since many of these notes are preliminary versions or may be published elsewhere, they have a limited distribution only and are not for review. Copies of these notes are available from the author or the editor.

Eindhoven University of Technology
Department of Mathematics and Computing Science
P.O. Box 513
5600 MB EINDHOVEN
The Netherlands
All rights reserved
editors: prof.dr.M.Rem
 prof.dr.K.M.van Hee.

Foundations of Compositional Program Refinement*

(second version)

Rob Gerth[†]

Eindhoven University of Technology[‡]

September, 1990

Abstract

The aim of this paper is twofold: first is to formulate a foundation for refinement of parallel programs that may synchronously communicate and/or share variables; *programs rendered as 1st order transition systems*. The second aim is to bring closer and to show the relevance of the algebraic theory of parallel processes to that of the refinement of such 1st order systems. We do this by first developing a notion of refinement and a complete verification criterion for it for algebraic, uninterpreted transition systems—basing ourselves on already existing theory. This refinement notion is sensitive to both safety properties and absence of deadlock and divergence. Then we show how 1st order transition systems can be translated—while preserving those aspects of their semantics that we are interested in—into uninterpreted transition systems. Since this translation is canonical, it is used to lift the algebraic refinement and verification criteria to the level of 1st order systems. Specifically, we show that they yield *assertional methods* for refinement of such systems that resemble the methods used in Z. Manna and A. Pnueli’s temporal logic proof system. The results in this paper also apply to the problem of proving refinement of systems based on *trace inclusion*.

Keywords: refinement, implementation, concurrency, compositionality, algebraic process theory, transition system, simulation, assertional methods, inductive assertions, communication, shared variables, completeness, safety, liveness, (pre-)congruence, behavior, full abstractness.

Contents

1	Introduction	2
2	Uninterpreted transition systems	6
3	A refinement notion: failure refinement	13
4	A verification criterion: failure simulation	16
5	First order transition systems	21
6	Verifying first order refinement	25
7	Conclusions	34

*A first version of the paper appeared in the Proceedings of the NFI/REX workshop on Stepwise Refinement of Distributed Systems: Models, Formalisms, Correctness; LNCS 430, pp. 777–808, Springer Verlag.

[†]The author is currently working in and partially supported by ESPRIT project P3096: “Formal Methods and Tools for the Development of Distributed and Real-Time Systems (SPEC)”.

[‡]Department of Computing Science, Eindhoven University of Technology, P.O. Box 513, 5600 MB Eindhoven, The Netherlands. Email: wsinrobg@win.tue.nl.

1 Introduction

There are a number of reasons to use transition systems as a specification tool. One is their ease of use and their intuitiveness. Many of the existing refinement and verification theories are based on transition systems. Also, it seems hard to argue against the principle that the prime characteristics of any discrete system are the states it can be in and the way the state evolves through the actions (transitions) that the system executes or participates in; at least this seems true when one is interested in implementing such systems. Transition systems concentrate on just these aspects of systems. They are the standard tools for specifying and analyzing communication protocols [LS84] and have been used to specify and develop distributed algorithms [FLS87, SdeR87, Sto89] and real-time controllers [JM88, Lyn90]. The idea generated Harel's statecharts [Har87] and Berry's Esterel [BC85]. Also, it is the motivation for algebraic process calculi such as CCS [Mil80, Mil89], CSP [Hoa85] and ACP [BK84]

Dijkstra's stepwise development paradigm [Dij76] resurfaces in this area as program-refinement. It has been used, e.g., in [WLL88] on some quite complicated algorithms and of course by Chandy and Misra in UNITY [CM88, GP89]. The most forceful and convincing proponent for its use in system specification undoubtedly has been L. Lamport, who in a series of papers [Lam83, Lam89, AL88, AL90] has turned the use of transition systems and of refinement into a practical specification method.

One important aspect of system design is missing in transition systems and that is the idea of *composition* of systems. For this reason, we introduce combinators with which (transition) systems, π^0 and π^1 , can be composed; the most important one being parallel composition: $\pi^0 \parallel \pi^1$.

Most of the existing research and methods — algebraic process calculi excepted — use a notion of refinement that is based on *trace semantics*, and use a verification criterion that is based on *Milner-simulation* [Mil71]; cf. *refinement-mapping* in [AL88], *possibility-function* in [LT87] and *multi-valued possibility mapping* in [Lyn90]. To us, the decision whether to use refinement based on trace semantics or not cannot be made arbitrarily but, rather, should follow from an analysis of what properties refinement should satisfy. Likewise, whether or not to use (a variant of) Milner simulation as a verification criterion should depend on its suitability for the chosen notion of refinement.

At this point, the most vital question is

Exactly what should it mean that a system P refines another system Q : $P \sqsupseteq Q$?

To us, this question has two aspects. The first concerns the system properties that are of interest to us, sometimes called the *observable behavior*, and that we want preserved. The second aspect concerns the (meta) properties, such as *transitivity*, that we want refinement, \sqsupseteq , to satisfy. These questions we shall discuss now.

System properties. There is consensus among researchers that $P \sqsupseteq Q$ at least should entail that the computations of P are all allowed (i.e., also occur) in Q :

$$P \sqsupseteq Q \quad \Rightarrow \quad \neg\phi P \phi \subseteq \neg\phi Q \phi, \tag{1}$$

where $\neg\phi P \phi$ denotes the set of computations or the *observable behavior* of P . Of course, there is still room for different decisions as to what such behaviors make visible about a system's execution. Is it only the start and terminal states of an execution that one can see or can one also observe intermediate states? Even if so, maybe one should only be able to observe the values of the shared variables of the system. Then again, shouldn't one be able to observe the actions that a system participates in; or at least its communication actions?

There are no clearcut answers to these questions and rather than making a decision in these matters—a decision that will have to be arbitrary—in this paper, on the one hand, we allow the complete state to be visible at any point during execution as well as every action that the system executes but, on the other hand, we also introduce operators that allow parts of states and actions to be 'hidden'; i.e., to be made unobservable. Finally, we allow the fact that a system has terminated or is diverging to be observed.

Systems having infinitary behavior raise an additional issue. This is illustrated by the program $P \equiv b := true; x := 0; * [b \rightarrow x := x + 1 \square b \rightarrow b := false]$, where \square denotes non-deterministic choice between

the branches and $\star[\cdot \cdot]$ iterates until the guard of either branch is false. Obviously, P admits an infinite computation; namely, the one in which the second branch is never selected. Let P^{fair} be the program with the same text as P but with the additional assumption that the choice between the branches should be 'fair' in the intuitive sense in which the infinite computation in P is not (i.e., the second branch can always be chosen but never is). Should P be considered a refinement of P^{fair} ? The answer seems obvious: no. Yet, this paper takes the opposite view. The reason is the difference in techniques and methods between those that deal with program properties that are determined by the finitary behavior of programs—the so-called safety properties—and those, such as fairness, that do not—the so-called liveness properties.

In Leslie Lamport's intuitive characterization, safety properties state that along every computation path a 'bad' thing does not happen whereas liveness properties state that along every computation path some 'good' thing does happen. This description suggests invariants to describe safety properties ('a bad thing has not happened yet') and well-foundedness arguments to deal with liveness properties ('it takes less than that many more steps until the good thing happens').

For this reason and for the fact that liveness arguments usually depend on safety arguments of the system, this paper restricts itself to refinement w.r.t. safety properties and ignores liveness. We stress that our refinement notion is sensitive to deadlock and divergence of systems.

Intuitively, the only difference between P and P^{fair} is the liveness property that every computation of P^{fair} terminates (i.e., along every computation the good thing 'termination' happens). The 'bad' thing of non-termination does not happen along every computation path of P . Hence, we shall have $P \sqsupseteq P^{fair}$.

Technically speaking, safety properties can be characterized as those properties for which, given that they hold for every finite, initial part of a computation, they also hold for the whole, possibly infinite, computation; see also [AS85]. When we define the *observable behavior* of systems, we shall also include such 'limits' of finite computations. The effect is that $\neg P^{fair} \phi = \neg P \phi$. Left to right inclusion is obvious. Because, all initial parts of the infinite computation of P are in $\neg P^{fair} \phi$, the infinite 'limit' computation is included, too.

Meta properties. Next, what meta properties should \sqsupseteq satisfy? It seems obvious that \sqsupseteq should at least be transitive: if P_0 refines P_1 and P_1 refines P_2 then clearly P_0 should refine P_2 . As it seems quite unreasonable not to allow a system to be a refinement of itself, this makes \sqsupseteq a *pre order*:

$$\begin{array}{l} P_0 \sqsupseteq P_1, P_1 \sqsupseteq P_2 \implies P_0 \sqsupseteq P_2 \\ P \sqsupseteq P \end{array} \quad (2)$$

One should not expect \sqsupseteq to be a partial order: obviously, even if both P_0 refines P_1 and P_1 refines P_0 , it does not follow that P_0 and P_1 are (syntactically) identical.

A second desirable meta property concerns the composition of systems. The advantage of system composition is that one has the option to develop parts of a system independently and of integrating them at a later stage. As refinement is the formalization of system development (at least in this paper) this advantage implicitly constrains the notion of refinement that one uses: *whenever parts of a system are refined then if the refined parts are put together again, the resulting system should be a refinement of the original one*. If one's refinement notion does not satisfy this constraint, then one can only develop a system as a monolithic whole. Let $C[\cdot]$ stand for a system with a 'hole' in it (a *context*), into which another system, P , can be plugged: $C[P]$. Then the above constraint is written as follows:

$$\text{for every context } C[\cdot]: P \sqsupseteq Q \implies C[P] \sqsupseteq C[Q] \quad (3)$$

This makes \sqsupseteq a *pre congruence* w.r.t. the program combinators; i.e., it makes $\sqsupseteq \cap \sqsubseteq$ a congruence. In the absence of other equally obvious requirements on a refinement relation, we let \sqsupseteq be *determined* by the above properties 1, 2 and 3.

Perhaps this whole discussion seems a bit trite. After all, these principles and constraints all are rather obvious. Still, it is worth stressing these points, since they very much constrain adequate notions of refinement which often forces one to consider aspects of system behavior that, a priori, one was not interested in. For example, in this paper it will force us to consider the precise deadlock behavior of

systems [Hen88]. Had we allowed communication action to time out—which we have not—even more detailed behavior would have had to be known [GB87].

The point of view embodied in the above discussion is not common in the area of program refinement. On the other hand, it is the point of departure in algebraic process theory [Mil80, Hen88, Hoa85, BK84]. There one starts with a (context free) programming language (or term algebra) and one looks for programming equivalences or pre orders satisfying some properties that are congruences or pre congruences and that are, if possible, *fully abstract*. A (pre) congruence is fully abstract w.r.t. some properties precisely if it is the coarsest one satisfying these properties. Our stipulating that \sqsupseteq be determined by equations (1), (2) and (3), makes \sqsupseteq fully abstract w.r.t. inclusion of observable behavior; in fact, it is equivalent with it.

Languages that allow synchronization have been investigated intensively in this area [Dar82, NH84, BHR84, BKO86]; for an overview of various proposed congruences see [Nic87]. One conclusion of this research is that *if the observable behavior of a program allows the state of terminated (or maximal) computations to be observed as such, then any (pre) congruence for such a language should be sensitive to the precise way in which systems may deadlock*. The standard example illustrating this (using a CSP-like language) compares $P :: [true \rightarrow R!1 \square true \rightarrow S!1]$ and $Q :: [R!1 \rightarrow skip \square S!1 \rightarrow skip]$. Here, $R!1$ is a send command that waits until process R wants to receive a value and then sends the value 1. In process Q send commands appear in the guards of the (I/O) guarded command. This means that process Q will suspend until *either R or S is ready to receive a value* and then will execute the appropriate send command. Although, intuitively, the computations of both programs are the same, there is a context, namely $C \equiv (R :: x := 0; ?x) \parallel [.]$, such that if we run P and Q within C , we do observe a difference ($;$ $?x$ is the program that waits until it receives a value and then assigns it to x): $C[Q]$ does not have a maximal computation that ends in a state in which $x = 0$ holds, whereas $C[P]$ does; namely the one in which P chooses the second branch and subsequently deadlocks.

The same research indicates that a (pre-)congruence should not be ‘sensitive’ to more than the computations of a program and its deadlock possibilities during execution. This is the *testing pre-order* of [NH84], the *failure set semantics* of [BHR84] and the *fully observable congruence* of [Dar82]. The refinement pre-order that we develop in Section 3 will be based on this. In contrast with most researchers in this area, we shall use (uninterpreted) transition systems (with parallel composition and renaming of actions) as programming language instead of CCS or TCSP. In this sense, our set up is closer in spirit to that of [BKO86]: their process graphs correspond to our uninterpreted transition systems.

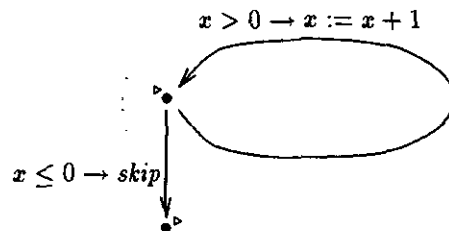
Once the notion of refinement is elucidated (in Section 3), the next question is how to prove it. Now, a transition system is specified in terms of vertices, edges and actions (associated with the edges). Hence, a usable verification criterion should be cauched in terms of these primitives. In Section 4 we develop such a criterion, called *failure simulation* and written as \hookrightarrow , by generalizing the notion of Milner-simulation [Mil71]. This criterion is then proven sound and complete; i.e. we prove that

$$\pi^0 \sqsupseteq \pi^1 \iff \pi^0 \hookrightarrow \pi^1,$$

or, equivalently, that $\sqsupseteq = \hookrightarrow$. There are some connections here with [Dar82].

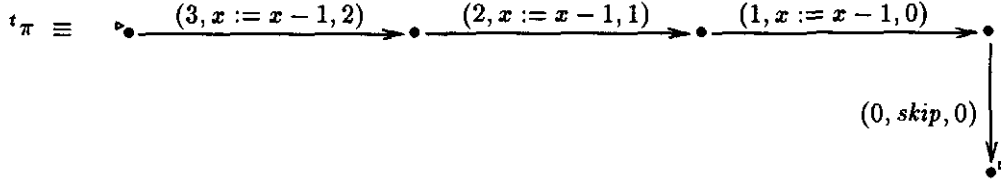
Upto this point we have looked at algebraic, uninterpreted transition systems. The rest of the paper concentrates on 1st order transition systems. If we want to use the results of the first part, we need a correspondence between such 1st order systems and the uninterpreted ones. Section 5 defines such a correspondence and also shows how various operators, such as parallel composition and the hiding of variables, can be defined in terms of uninterpreted systems. This correspondence is, in fact, quite simple

and can be illustrated with the system $\pi \equiv$



, that corresponds to

the program **while** $x > 0$ **do** $x := x - 1$ **od**, and respectively “ \triangleright •” and “• \triangleright ” are the initial and terminal vertex. Now, π implicitly depends on a notion of *state* and on the *meaning* of the ‘labels’ on the transitions. The translation makes these dependencies explicit. If we assume that x is the only variable, then we may represent a state as just a value; namely, the value of x . Let us take the intuitive meaning of π and let us assume that the system always starts in the state 3 (i.e., in the state in which x has the value 3). Then π ’s translation would be



where now the labels on the edges record a state, an action and another state; each pair of states is in the input-output relation of the corresponding action; and ${}^t\pi$ contains a vertex for every vertex-state pair that can occur during an execution of π . Note that the labels in ${}^t\pi$ are just that: labels; no further interpretation of them is necessary. The translation also unfolds the loop but this does not happen in general: the translated system contains a cycle whenever it is possible to arrive more than once in the same vertex with the same state.

Section 6 develops on the basis of this translation a verification criterion for 1st order systems. First we define for 1st order systems, π^0 and π^1 , $\pi^0 \sqsupseteq \pi^1$ to mean that ${}^t\pi^0 \sqsupseteq {}^t\pi^1$; i.e. a system refines another one if its translation refines the other one’s. If one accepts the translation as reasonable in the sense that it preserves the meaning of a system, then this is the correct notion of refinement.

Now, suppose that $\pi^0 \sqsupseteq \pi^1$ where π^0 and π^1 are 1st order systems. Then we know that ${}^t\pi^0 \hookrightarrow {}^t\pi^1$, by completeness of the criterion of Section 4. So, “all” that has to be done is to find some conditions on π^0 and π^1 that imply that ${}^t\pi^0 \hookrightarrow {}^t\pi^1$. This we do, using techniques from Floyd’s inductive assertion method [Flo67].

The methods in Sections 3 and 6 also yield verification techniques for a notion of refinement based on *trace inclusion*, as used, e.g., in [Lam89, Lyn90]. The necessary simplifications are indicated.

Finally, Section 7 contains some discussion.

We end the introduction with establishing some notation.

Notation.

Functions, f , unless stated otherwise, are partial and have domain, $dom(f)$, and range, $ran(f)$. Composition, \circ , is defined by $(f \circ g)(x) = f(g(x))$. Functions are sometimes defined as lists, like $[1 \mapsto 2, 2 \mapsto 3]$ which denotes the (partial) function, f , such that $f(1) = 2$ and $f(2) = 3$. Define $f\{a/x\}$, a variant of f , by $f \circ [x \mapsto a]$. In general, any function f is pointwise extended to $2^{dom(f)}$: if $V \subseteq 2^{dom(f)}$ then $f(V) = \{f(v) \mid v \in V\}$. The restriction of f to V , $f \upharpoonright V$ is the function with $dom(f \upharpoonright V) = dom(f) \cap V$ and $(f \upharpoonright V)(v) = f(v)$ for $v \in dom(f \upharpoonright V)$. Finally, $f^{-1}(w) = \{v \mid f(v) = w\}$.

Relations over some set A are usually denoted by letters $R, M, \dots \subseteq A \times A$. Instead of $(a, b) \in R$ we often write $a R b$. Also, the *post image* of a under R , $a R$, is $\{b \mid a R b\}$ and the *pre image* of b under R , $R b$, is $\{a \mid a R b\}$. This extends to the post image and pre image of sets. Instead of $a R$ we occasionally write $R(a)$. We usually identify functions with the relations that they denote.

The class of ordinals is Ord . Letters α, β, \dots range over the ordinals; letters n, m, \dots range over the finite ordinals (i.e., the natural numbers). As usual, ω denotes the first transfinite ordinal.

If A is some set of symbols, then A^+ is the set of finite, non empty sequences over A ; A^* is the set of finite sequences over A ; and A^ω the set of infinite such sequences. Then $A^\dagger = A^* \cup A^\omega$. Sequences in A^\dagger are denoted by s, t, \dots ; the elements of such an s are s_i for $i < |s|$, the length of s (if $s \in A^\omega$ then $|s| = \omega$); hence, s_0 is the first element of s . As alternative sequence notation, there is $s = (s_i)_{i < |s|}$; hence, the empty string, ϵ , is $(s_i)_{i < 0}$ (for any s). Write $s \prec t$ if $|s| < |t|$ and $s = (t_i)_{i < |s|}$. Any function $h: A \mapsto A$ extends to A^\dagger by having it act pointwise on sequences: $h(s) = (h(s_i))_{i < |s|}$. Write $\mathcal{A}(s)$ for the set of symbols in s : $\mathcal{A}(s) = \{s_i \mid i < |s|\}$. As a matter of notation, $\mathcal{A}_{xyz\dots}$ will stand for the set $\mathcal{A} \cup \{x, y, z, \dots\}$.

There is a denumerable set of variables Var . If Σ is some signature or similarity type, then $Tm(\Sigma)$ and $L(\Sigma)$ denote the set of terms and first order formulae over Σ (and Var). Elements of $L(\Sigma)$ are usually denoted by letters p, q, r ; terms by e, t and variables by u, v, x, y, z . As usual, $p[e/x]$ ($t[e/x]$) stands for the formula (term) obtained by substituting the term or expression e for every free occurrence of x in p (t). The set of free variables of a formula or term ϕ is denoted by $FV(\phi)$.

Σ -structures or Σ -algebra's, \mathbf{A} , are defined as usual and give interpretations of the symbols in Σ . Given a Σ -structure \mathbf{A} , states σ, τ, ν are total functions from Var into $|\mathbf{A}|$, the universe of \mathbf{A} . Let \mathcal{S} be the set of states.

The value of a term t in a state σ , $\sigma(t)$, and the truth-value or satisfaction of a formula p in a state σ , $\sigma \models p$ or $\mathbf{A}, \sigma \models p$, are defined as usual. Write $\mathbf{A} \models p$ if $p \in L(\Sigma)$ is valid in \mathbf{A} , i.e., if $\mathbf{A}, \sigma \models p$ holds for every state σ ; write $\sigma \models p$ if $\mathbf{A}, \sigma \models p$ and \mathbf{A} is clear from the context; and $\models p$ if p is valid, i.e., if $\mathbf{A} \models p$ holds for every Σ -structure \mathbf{A} . As usual, any variable that appears free in some formula lives in the scope of an implicit universal quantification (of that formula).

By convention, all super- and subscripts, bars, underscores etc. of composite objects are inherited by the components: e.g., if $K = (a, b)$ then ${}^r\hat{K}_1 = ({}^r\hat{a}_1, {}^r\hat{b}_1)$.

2 Uninterpreted transition systems

In this section, the basic framework of uninterpreted transition systems is defined and the basic notation established.

- There is some infinite, enumerable, alphabet, \mathcal{A} of actions, which is partitioned into a set of local actions, \mathcal{A}^l , and a set of global actions, \mathcal{A}^g . The intention is that local actions never synchronize with actions in other systems, whereas global action may do so. To formalize the synchronization of global actions, we interpret \mathcal{A}^g as the generator of the free, commutative semigroup denoted by $(\mathcal{A}^g, |)$. The commutativity condition ensures that the parallel composition itself is commutative (see below). Unless stated otherwise, \mathcal{A}^g stands for the semigroup it generates.

There are two special 'actions': δ and \dagger , which are not in \mathcal{A} . The first one, δ , denotes the *undoable* action; the second one, \dagger , signifies *divergence*. Intuitively, \dagger means that the system is engaged in an infinite, internal or hidden computation that does not yield any visible behavior.

The domain of $|$ is extended to $\mathcal{A}_{\delta\dagger} \times \mathcal{A}_{\delta\dagger}$ by mapping pairs of actions that are *not* in $dom(|)$ to δ . Instead of $a|b$ we shall often write ab .

- A *directed graph* is an object $(\partial_s, \partial_t: \mathcal{E} \mapsto \mathcal{V})$, where
 - \mathcal{V} is a set of *vertices*,
 - \mathcal{E} is a set of *edges*, and
 - ∂_s , respectively, ∂_t is the function that associates each edge with a *source*, respectively, *target* vertex.

In the sequel we shall often write *e instead of $\partial_s(e)$ or $\partial_s e$, and e^* instead of $\partial_t(e)$ or $\partial_t e$. Moreover this notation dualizes and we write *v for the set $\{e \in \mathcal{E} \mid {}^*e = v\}$ and v^* for the set $\{e \in \mathcal{E} \mid e^* = v\}$.

- An *uninterpreted transition system* (uts) over alphabet \mathcal{A} , π , is an object

$$(\partial_s, \partial_t: \mathcal{E} \mapsto \mathcal{V}, I, T, \mathcal{L}: \mathcal{E} \mapsto \mathcal{A}_{\delta\dagger}) ,$$

where

- $(\partial_s, \partial_t: \mathcal{E} \mapsto \mathcal{V})$ is a directed graph,
- $I \subseteq \mathcal{V}$ is a set of *initial* vertices with $I \neq \emptyset$,

- $T \subseteq \mathcal{V}$ is a set of *terminal* vertices, and
- \mathcal{L} is a *labelling function* that associates actions with the edges.

The class of uts over an alphabet \mathcal{A} is $\text{UTS}_{\mathcal{A}}$ or just UTS^1 .

Notice that that there are no cardinality constraints on the vertex and edge sets of uts's and that any uts has at least one initial vertex. Also note that an uts can express the occurrence of divergence and that it may have disabled actions that will never occur.

Some notation: the canonical uts is denoted by π and has components $(\partial_s, \partial_t: \mathcal{E} \mapsto \mathcal{V}, I, T, \mathcal{L}: \mathcal{E} \mapsto \mathcal{A}_{\delta\uparrow})$. The alphabet, $\mathcal{A}(\pi)$, of π is the set of labels appearing on its edges: $\mathcal{A}(\pi) = \text{ran}(\mathcal{L})$. If $V \subseteq \mathcal{V}$ then $V; \pi$ is the uts $(\partial_s, \partial_t: \mathcal{E} \mapsto \mathcal{V}, V, T, \mathcal{L}: \mathcal{E} \mapsto \mathcal{A}_{\delta\uparrow})$, with initial vertices in V . Finally, $\mathcal{L}_g(\cdot)$ is the function $\mathcal{L}(\cdot) \cap \mathcal{A}^g$.

- The set of *sequences* of π is

$$\text{Seq}(\pi) = \{ (v, (e_i)_{i < \alpha}) \mid \alpha \leq \omega, v \in I, \alpha > 0 \Rightarrow v = \bullet e_0, \forall 0 < i < \alpha e_{i-1} = \bullet e_i \}$$

Hence, a sequence is a possibly infinite path in the underlying directed graph of π that starts in an initial vertex.

- The *behavior* or *computations* of an uts, $\pi, \dashv\pi\Phi^o$, consists of the maximal sequences of actions that π can perform together with an indication, \uparrow, \downarrow or δ , whether the system may diverge at that point, has terminated or is stable. The letter ρ ranges over $\{\uparrow, \downarrow, \delta\}$.

First define for any $v \in \mathcal{V}$ the predicates:

$$\uparrow(v) \iff \uparrow \in \mathcal{L}(\bullet v), \quad \downarrow(v) \iff v \in T \quad \text{and} \quad \delta(v) \iff \mathcal{L}(\bullet v) \cap \mathcal{A}_{\uparrow} = \emptyset \ \& \ v \notin T.$$

So, $\uparrow(v)$ holds if v has an outgoing edge labelled with \uparrow —i.e., if v may diverge—; $\downarrow(v)$ holds if v is a terminal vertex; and $\delta(v)$ holds if v can neither diverge nor perform an action (in \mathcal{A}) and is not terminal. Write $\uparrow(\pi, v)$, $\downarrow(\pi, v)$ and $\delta(\pi, v)$ if π is not determined by the context. Now

$$\dashv\pi\Phi^o = \left\{ s\rho \mid \begin{array}{l} \exists (v, (e_i)_{i < \alpha}) \in \text{Seq}(\pi), \alpha \leq \omega, s = (\mathcal{L}(e_i))_{i < \alpha} \in \mathcal{A}_{\uparrow}, \rho \in \{\uparrow, \downarrow, \delta\}, \\ \alpha = \omega \Rightarrow \rho = \downarrow, \alpha = 0 \Rightarrow \rho(v), 0 < \alpha < \omega \Rightarrow \rho(e_{\alpha-1}^{\bullet}) \end{array} \right\}.$$

So, a finite sequence of actions is maximal if the corresponding path ends in a vertex from which it is possible to diverge or which is terminal or which is stable.

If $s\rho \in \dashv\pi\Phi^o$ then any $\bar{s} \in \mathcal{A}^*$ with $\bar{s} \prec s$ is called a *partial computation*.

There is an alternative way to define the computations of an uts. Namely, by defining for every $a \in \mathcal{A}_{\uparrow}$ a *transition relation*, $\xrightarrow{a} \in \text{UTS} \times \text{UTS}$:

$$\pi \xrightarrow{a} \bar{\pi} \iff \exists e \in \bullet I \ \mathcal{L}(e) = a \ \& \ \bar{\pi} = e^{\bullet}; \pi.$$

So, each uts defines, through its initial vertices, a set of actions that it can perform; namely those actions that appear as label on an edge starting in an initial state: $\mathcal{L}(\bullet I)$. Performing such an action transforms the uts π by changing its set of initial vertices to the target vertex of the corresponding edge. This generalizes and for $s \in \mathcal{A}^*$ we write:

$$\pi \xrightarrow{s} \bar{\pi} \iff \forall i < |s| \exists \pi^i \in \text{UTS} \ \pi^0 = \pi, \pi^{i-1} = \bar{\pi}, \forall 0 < i < |s| \ \pi^{i-1} \xrightarrow{s_i} \pi^i.$$

Also, write $\pi \xrightarrow{s}$ if there is a $\bar{\pi}$ such that $\pi \xrightarrow{s} \bar{\pi}$; if $s \in \mathcal{A}^{\omega}$, $\pi \xrightarrow{s}$ has the obvious meaning. Finally, $\pi \not\xrightarrow{s} \bar{\pi}$ and $\pi \not\xrightarrow{s}$ stand for the negations of the above relations; $\pi \not\xrightarrow{a}$ means $\pi \not\xrightarrow{a}, \forall a \in \mathcal{A}$.

¹To be pedantic, it is the isomorphism class generated by uts that is of interest rather than the uts itself. Isomorphism, here, means graph isomorphism that maintains the edge labelling and initial vertices.

In this setup, the behavior of a $\pi \in \text{UTS}$ can be described by

$$\langle \pi \rangle^o = \{ s\rho \mid s \in \mathcal{A}^*, \exists \bar{\pi} \pi \xrightarrow{s} \bar{\pi}, \exists v \in \bar{I} \rho(v) \} \cup \{ s\downarrow \mid s \in \mathcal{A}^\omega, \pi \xrightarrow{s} \}$$

The *observable behavior* of π can now be defined by

$$\langle \pi \rangle = \langle \pi \rangle^o \cup \{ s\downarrow \mid s \in \mathcal{A}^\omega, \forall \bar{s} \prec s \pi \xrightarrow{\bar{s}} \},$$

in accordance with the discussion of liveness in the introduction.

As a final piece of notation, define π after s ($s \in \mathcal{A}^*$) by

$$\pi \text{ after } s = V; \pi, \text{ where } V = \{ v \mid \pi \xrightarrow{s} v; \pi \}.$$

We shall often write expressions such as $v \in \pi$ after s instead of the formally correct statement that π after $s = V; \pi$ and $v \in V$.

2.1 An algebra of unlabeled transition systems

Here the operators to compose uts's are defined: the *parallel composition*, $\cdot \parallel \cdot$; the *renaming*, $h(\cdot)$; the *choice*, $\cdot + \cdot$; and the *sequential composition operator*, $\cdot ; \cdot$.

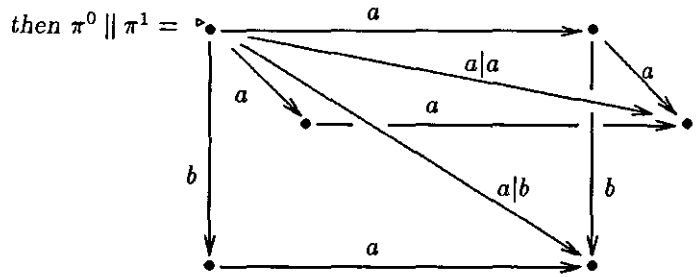
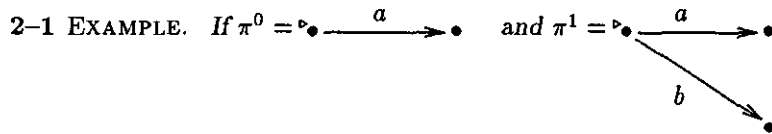
2.1.1 Parallel composition

For any $\pi^0, \pi^1 \in \text{UTS}$, the *parallel composition*

$$\pi^0 \parallel \pi^1 = \pi \in \text{UTS}$$

is the product of π^0 and π^1 with the 'diagonals' filled in according to \mid . Formally, it is defined by²

- $\mathcal{V} = \mathcal{V}^0 \times \mathcal{V}^1$,
- $\mathcal{E} = \mathcal{E}^0 \times \mathcal{V}^1 + \mathcal{E}^1 \times \mathcal{V}^0 + \mathcal{E}^0 \times \mathcal{E}^1$ (+ denotes disjoint union),
- $I = I^0 \times I^1$,
- $T = T^0 \times T^1$,
- for $(e, v) \in \mathcal{E}^i \times \mathcal{V}^{1-i}$ $\partial_s(e, v) = (\partial_s^i(e), v)$, $\partial_t(e, v) = (\partial_t^i(e), v)$,
 $\mathcal{L}(e, v) = \mathcal{L}^i(e)$ ($i < 2$) and
- for $(e^0, e^1) \in \mathcal{E}^0 \times \mathcal{E}^1$ $\partial_s(e^0, e^1) = (\partial_s^0(e^0), \partial_s^1(e^1))$, $\partial_t(e^0, e^1) = (\partial_t^0(e^0), \partial_t^1(e^1))$,
 $\mathcal{L}(e^0, e^1) = \mathcal{L}^0(e^0) \mid \mathcal{L}^1(e^1)$.



²Remember, it is the isomorphism class that is of interest.

In terms of transition relations, one has

$$\begin{aligned} \pi^0 \parallel \pi^1 \xrightarrow{a} \bar{\pi}^0 \parallel \pi^1 & \text{ if } \pi^0 \xrightarrow{a} \bar{\pi}^0, \\ \pi^0 \parallel \pi^1 \xrightarrow{a} \pi^0 \parallel \bar{\pi}^1 & \text{ if } \pi^1 \xrightarrow{a} \bar{\pi}^1 \text{ and} \\ \pi^0 \parallel \pi^1 \xrightarrow{a^0 | a^1} \bar{\pi}^0 \parallel \bar{\pi}^1 & \text{ if } \pi^i \xrightarrow{a^i} \bar{\pi}^i \text{ (} i < 2 \text{)}. \end{aligned}$$

The three clauses correspond to the edges in $\mathcal{E}^0 \times \mathcal{V}^1$, $\mathcal{E}^1 \times \mathcal{V}^0$ and $\mathcal{E}^0 \times \mathcal{E}^1$.

2.1.2 Renaming

Let τ be some symbol not in $\mathcal{A}_{\delta\uparrow}$. Its meaning is explained below. The set of (*renaming*) *homomorphisms*, \mathcal{H} , is

$$\{ h: \mathcal{A} \mapsto \mathcal{A}_{\delta\tau} \mid h(\mathcal{A}^i) \subseteq \mathcal{A}_{\tau\delta}^i \}.$$

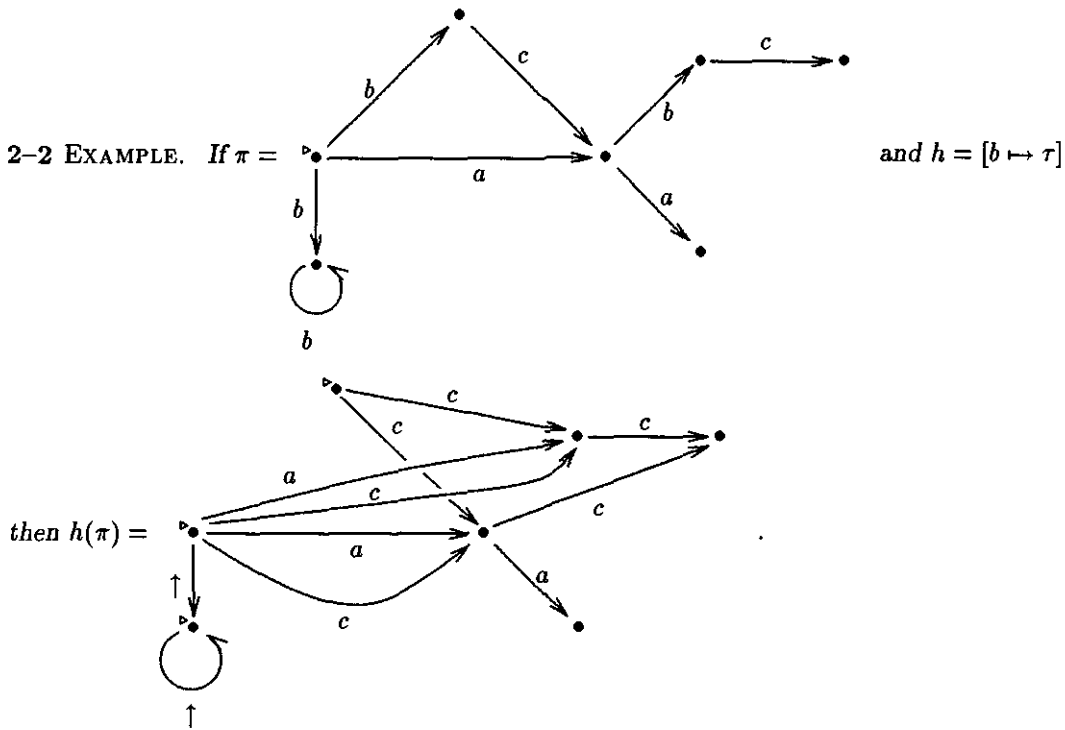
Any $h \in \mathcal{H}$ is extended to a total function on $\mathcal{A}_{\delta\uparrow}$ by having $h(a) = a$ for any $a \notin \text{dom}(h)$. Let δ also stand for the function $[a \mapsto \delta, a \in \mathcal{A}]$. Homomorphisms *rename* actions; except that local actions never rename to global actions and that δ and \uparrow rename to themselves. If $h(a) = \delta$ (i.e., the symbol) this means that the actions a can no longer occur in $h(\pi)$. If $h(a) = \tau$ this means that although a can still occur it is no longer possible to *observe* this—the action is *hidden*. As expected, h acts pointwise on sequences of actions, when τ is interpreted as the empty sequence, ε .

For $\bar{\pi} \in \text{UTS}$ and $h \in \mathcal{H}$, the *renamed* system

$$h(\bar{\pi}) = \pi \in \text{UTS}$$

is obtained by redefining the set of edges and of initial vertices: an edge in $h(\bar{\pi})$ is any finite path in $\bar{\pi}$ on which precisely one non-erased action occurs or those infinite paths on which every action is erased. The initial vertices are the original ones, together with those vertices that can be reached from an initial vertex by a finite number of erased actions. Formally, it is defined by

- $\mathcal{V} = \bar{\mathcal{V}}$,
- $\mathcal{E} = \left\{ (\bar{e}_i)_{i < \alpha} \left| \begin{array}{l} 0 < \alpha \leq \omega, \forall 0 < i < \alpha \bar{\partial}_t(\bar{e}_{i-1}) = \bar{\partial}_s(\bar{e}_i), \\ \alpha < \omega \Rightarrow h((\bar{\mathcal{L}}(\bar{e}_i))_{i < \alpha}) \in \mathcal{A}_{\delta\uparrow}, \\ \alpha = \omega \Rightarrow h((\bar{\mathcal{L}}(\bar{e}_i))_{i < \alpha}) = \varepsilon \end{array} \right. \right\}$,
- $\partial_s((\bar{e}_i)_{i \leq n}) = \bar{\partial}_s(\bar{e}_0)$, $\partial_t((\bar{e}_i)_{i \leq n}) = \bar{\partial}_t(\bar{e}_n)$,
 $\partial_s((\bar{e}_i)_{i < \omega}) = \bar{\partial}_s(\bar{e}_0)$, $\partial_t((\bar{e}_i)_{i < \omega}) = \bar{\partial}_t(\bar{e}_0)$ (sic),
- $I = \bar{I} \cup \{ \partial_t((\bar{e}_i)_{i \leq n}) \mid \exists v \in \bar{I} (v, (e_i)_{i \leq n}) \in \text{Seq}(\bar{\pi}), h((\bar{\mathcal{L}}(e_i))_{i \leq n}) = \varepsilon \}$,
- $T = \bar{T}$,
- if $e = (\bar{e}_i)_{i \leq n}$ then $\mathcal{L}(e) = h((\bar{\mathcal{L}}(e_i))_{i \leq n})$ and
if $e = (\bar{e}_i)_{i < \omega}$ then $\mathcal{L}(e) = \uparrow$.



Although the construction of $h(\pi)$ may seem overly complicated, it induces the following simple and intuitive correspondence between the transitions in π and those in $h(\pi)$:

$$h(\pi) \xrightarrow{a} h(\bar{\pi}) \quad \text{if} \quad \exists s \in \mathcal{A}_{\delta\uparrow}^* \pi \xrightarrow{s} \bar{\pi} \ \& \ h(s) = a \in \mathcal{A}_{\delta\uparrow} \quad \text{and}$$

$$h(\pi) \xrightarrow{\uparrow} h(\bar{\pi}) \quad \text{if} \quad \exists s \in \mathcal{A}^\omega \pi \xrightarrow{s} \bar{\pi} \ \& \ h(s) = \varepsilon \ \& \ \pi \xrightarrow{s_0} \bar{\pi} .$$

Observe that the renaming in the above example yields two additional initial vertices. This is the technical motivation for allowing sets of initial vertices.

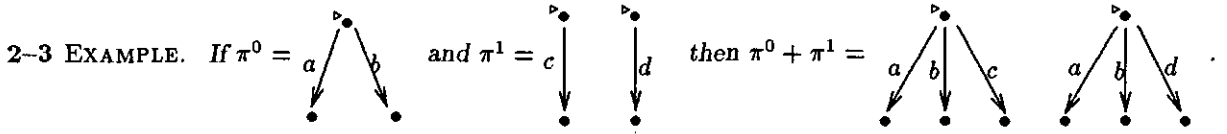
2.1.3 Choice


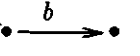
For any two $\pi^0, \pi^1 \in UTS$, with $I^0 \bullet \cup I^1 \bullet = \emptyset$, their *choice composition*,

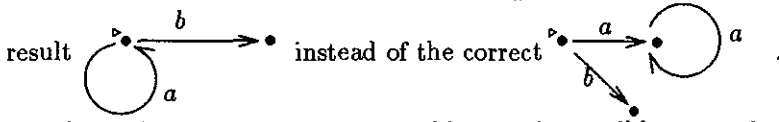
$$\pi^0 + \pi^1 = \pi \in UTS ,$$

is obtained by connecting all pairs of initial vertices. Formally it is given by

- $\mathcal{V} = \mathcal{V}^0 \setminus I^0 + \mathcal{V}^1 \setminus I^1 + I^0 \times I^1$,
- $I = I^0 \times I^1$,
- $T = (T^0 + T^1 + T^0 \times I^1 + T^1 \times I^0) \cap \mathcal{V}$,
- $\mathcal{E} = \mathcal{E}^0 \setminus \bullet I^0 + \mathcal{E}^1 \setminus \bullet I^1 + \bar{\mathcal{E}}^0 + \bar{\mathcal{E}}^1$, with $\bar{\mathcal{E}}^i = \bullet I^i \times I^{1-i} \quad (i < 2)$,
- for $e \in \mathcal{E}^i \setminus \bullet I^i \quad \partial_s(e) = \partial_s^i(e), \partial_t(e) = \partial_t^i(e),$
 $\mathcal{L}(e) = \mathcal{L}^i(e) \quad (i < 2) \quad \text{and}$
- for $(e, v) \in \bar{\mathcal{E}}^i \quad \partial_s(e, v) = \partial_s^i(e), \partial_t(e, v) = \partial_t^i(e),$
 $\mathcal{L}(e, v) = \mathcal{L}^i(e) \quad (i < 2)$

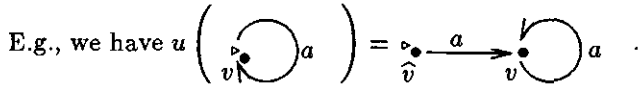


If an UTS admits edges that are incident on its initial vertices, there is a complication: if we were to apply the above construction to $\pi^0 =$  and $\pi^1 =$  then we would obtain the wrong



This is in fact a well-known problem with a well-known solution; see e.g., [BK84]. For the general construction, first add fresh copies of $\bullet I^i$ to π^i (hence, also fresh initial nodes) so as to defer cycles until the initial choice has been made. Define for $\pi \in \text{UTS}$, $u(\pi) = \bar{\pi} \in \text{UTS}$ by

- $\bar{\mathcal{V}} = \mathcal{V} + \widehat{I}^3$,
- $\bar{\mathcal{E}} = \mathcal{E} + \widehat{I}$,
- $\bar{I} = \widehat{I}$,
- $\bar{T} = T$, and
- for $\widehat{e} \in \widehat{I}$ $\bar{\partial}_s(\widehat{e}) = \widehat{\partial}_s(e)$, $\bar{\partial}_t(\widehat{e}) = \partial_t(e)$,
 $\bar{\mathcal{L}}(\widehat{e}) = \mathcal{L}(e)$



For arbitrary $\pi^0, \pi^1 \in \text{UTS}$, define

$$\pi^0 + \pi^1 = u(\pi^0) + u(\pi^1).$$

The induced transition relation is

$$\begin{aligned} \pi^0 + \pi^1 &\xrightarrow{a} \bar{\pi}^0 && \text{if } \pi^0 \xrightarrow{a} \bar{\pi}^0 \\ \pi^0 + \pi^1 &\xrightarrow{a} \bar{\pi}^1 && \text{if } \pi^1 \xrightarrow{a} \bar{\pi}^1 \end{aligned}$$

Because there are no cardinality constraints on vertex and edge sets and because $+$ is commutative, we can easily define an infinitary version of choice composition on arbitrary sets of uts's.

2.1.4 Sequential composition

For $\pi^0, \pi^1 \in \text{UTS}$ the sequential composition

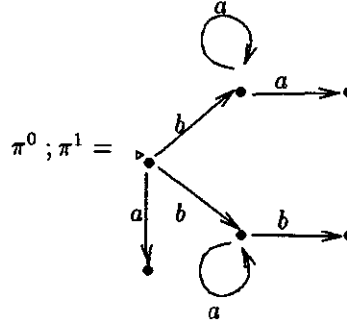
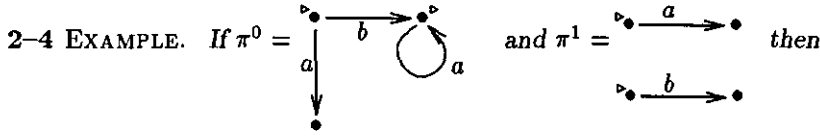
$$\pi^0 ; \pi^1 = \pi \in \text{UTS},$$

is obtained, basically, by identifying the pairs in $T^0 \times I^1$. Formally, it is defined as

$$- \mathcal{V} = \mathcal{V}^0 \setminus T^0 + \mathcal{V}^1 \setminus I^1 + T^0 \times I^1,$$

³I.e., $\widehat{I} = \{\widehat{v} \mid v \in I\}$.

- $\mathcal{E} = \mathcal{E}^0 \setminus (\bullet T^0 \cup T^0 \bullet) + \mathcal{E}^1 \setminus (\bullet I^1 \cup I^1 \bullet) + \widehat{\mathcal{E}}$, where
 $\widehat{\mathcal{E}} = (\bullet T^0 \cup T^0 \bullet) \times I^1 + (\bullet I^1 \cup I^1 \bullet) \times T^0$,
- $I = (I^0 \cup I^0 \times T^1) \cap \mathcal{V}$,
- $T = (T^1 \cup T^0 \times I^1) \cap \mathcal{V}$,
- for $e \in \mathcal{E} \cap \mathcal{E}^i$ $\partial_s(e) = \partial_s^i(e)$, $\partial_t(e) = \partial_t^i(e)$,
 $\mathcal{L}(e) = \mathcal{L}^i(e)$ ($i < 2$) and
- for $(e, v) \in \widehat{\mathcal{E}}$, $e \in \mathcal{E}^i$ $\partial_s(e, v) = \partial_s^i(e)$, $\partial_t(e, v) = \partial_t^i(e)$,
 $\mathcal{L}(e, v) = \mathcal{L}^i(e)$ ($i < 2$)



The transition relation of $\pi^0 ; \pi^1$ satisfies

$$\begin{aligned} \pi^0 ; \pi^1 \xrightarrow{a} \bar{\pi}^0 ; \bar{\pi}^1 & \text{ if } \pi^0 \xrightarrow{a} \bar{\pi}^0 \\ \pi^0 ; \pi^1 \xrightarrow{a} \bar{\pi}^1 & \text{ if } \pi^1 \xrightarrow{a} \bar{\pi}^1 \ \& \ T^0 \cap I^1 \neq \emptyset \end{aligned}$$

For future use, the set of *environments* or *contexts*, \mathcal{C} , with typical element $C[\cdot]$, is defined as the smallest set such that

- $[\cdot] \in \mathcal{C}$,
- $h \in \mathcal{H}$, $\circ \in \{||, +, ;\}$, $\pi \in \text{UTS}$, $C[\cdot] \in \mathcal{C} \implies h(C[\cdot]), \pi \circ C[\cdot], C[\cdot] \circ \pi \in \mathcal{C}$.

Hence, a context $C[\cdot]$ is a system with a 'hole' in it. Then, $C[\pi]$ is the system with π plugged into the hole so that $C[\pi] \in \text{UTS}$. Obviously,

$$C[\cdot], \bar{C}[\cdot] \in \mathcal{C} \implies C[\bar{C}[\cdot]] \in \mathcal{C}.$$

I.e., contexts compose.

3 A refinement notion: failure refinement

As argued in the introduction, the pre order, \sqsupseteq , should be determined by

$$\begin{aligned} \dashv\pi^0\clubsuit &\subseteq \dashv\pi^1\clubsuit \text{ and} \\ \forall C[\cdot] \in \mathcal{C} \quad C[\pi^0] \sqsupseteq C[\pi^1]. \end{aligned} \quad (4)$$

The second part of this condition implies that \sqsupseteq is a pre congruence w.r.t. the combinators with which contexts can be constructed. The rest of the section develops an explicit characterization of \sqsupseteq .

Based on (4), \sqsupseteq can be defined as a greatest fixed point: first let $\mathcal{R} = \mathcal{P}(\text{UTS} \times \text{UTS})$ —the set of relations over UTS. Define a functional $\mathcal{F}: \mathcal{R} \mapsto \mathcal{R}$ by

$$\mathcal{F}(\mathcal{R}) = \{ (\pi^0, \pi^1) \in \mathcal{R} \mid \dashv\pi^0\clubsuit \subseteq \dashv\pi^1\clubsuit, \forall C[\cdot] \in \mathcal{C} (C[\pi^0], C[\pi^1]) \in \mathcal{R} \}.$$

Then, we have

$$\pi^0 \sqsupseteq \pi^1 \iff \exists \mathcal{R} \in \mathcal{R} \quad (\pi^0, \pi^1) \in \mathcal{R} \ \& \ \mathcal{R} \subseteq \mathcal{F}(\mathcal{R}),$$

whence

$$\sqsupseteq = \bigcup \{ \mathcal{R} \in \mathcal{R} \mid \mathcal{R} \subseteq \mathcal{F}(\mathcal{R}) \}.$$

This characterizes \sqsupseteq as the greatest fixed point, $\nu \mathcal{F}$, in the standard lattice of relations (over UTS). Obviously, \mathcal{F} is monotonic in this lattice so that the fixed point exists, indeed. Standard theory also implies that

$$\sqsupseteq = \bigcap_{\alpha \in \text{Ord}} \mathcal{F}^\alpha(\mathcal{R}),$$

where the approximants, \mathcal{F}^α , to the function \mathcal{F} are defined by

$$\begin{aligned} \mathcal{F}^0 &= \lambda x.x \quad (\text{i.e., the identity function}) \text{ and} \\ \mathcal{F}^\alpha &= \lambda x.\mathcal{F}\left(\bigcap_{\beta < \alpha} \mathcal{F}^\beta(x)\right) \quad \text{for } \alpha > 0. \end{aligned}$$

We compute the first few approximants to \sqsupseteq , $\mathcal{F}^\alpha(\mathcal{R})$:

$$\mathcal{F}^0(\mathcal{R}) = \mathcal{R}; \quad \mathcal{F}^1(\mathcal{R}) = \mathcal{F}(\mathcal{F}^0(\mathcal{R})) = \{ (\pi^0, \pi^1) \mid \dashv\pi^0\clubsuit \subseteq \dashv\pi^1\clubsuit \};$$

and

$$\begin{aligned} \mathcal{F}^2(\mathcal{R}) = \mathcal{F}(\mathcal{F}^1(\mathcal{R})) &= \left\{ (\pi^0, \pi^1) \in \mathcal{F}^1(\mathcal{R}) \mid \begin{array}{l} \dashv\pi^0\clubsuit \subseteq \dashv\pi^1\clubsuit, \\ \forall C[\cdot] \in \mathcal{C} (C[\pi^0], C[\pi^1]) \in \mathcal{F}^1(\mathcal{R}) \end{array} \right\} \\ &= \{ (\pi^0, \pi^1) \mid \dashv\pi^0\clubsuit \subseteq \dashv\pi^1\clubsuit, \forall C[\cdot] \in \mathcal{C} \dashv C[\pi^0]\clubsuit \subseteq \dashv C[\pi^1]\clubsuit \} \\ &= \{ (\pi^0, \pi^1) \mid \forall C[\cdot] \in \mathcal{C} \dashv C[\pi^0]\clubsuit \subseteq \dashv C[\pi^1]\clubsuit \}, \end{aligned}$$

since $[\cdot] \in \mathcal{C}$ and $[\pi] = \pi$. Finally, $\mathcal{F}^3(\mathcal{R}) = \mathcal{F}^2(\mathcal{R})$ because contexts can be composed.

From this we obtain a new characterization of \sqsupseteq :

$$\pi^0 \sqsupseteq \pi^1 \iff \forall C[\cdot] \in \mathcal{C} \dashv C[\pi^0]\clubsuit \subseteq \dashv C[\pi^1]\clubsuit. \quad (5)$$

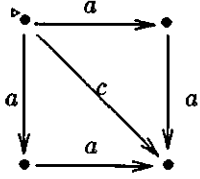
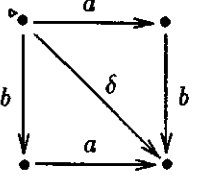
This shows that whether or not $\pi^0 \sqsupseteq \pi^1$ holds, depends not only on the observable behavior of the components, π^0 and π^1 , themselves but also on what a system in which they are embedded can ‘sense’ about them.

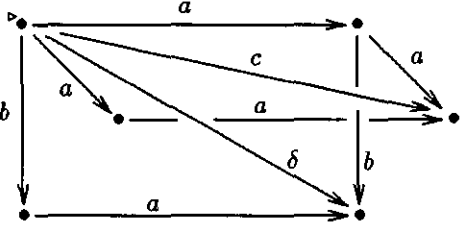
The following is the standard example showing some of the nature of \sqsupseteq as defined by (5):

Let $\pi^0 = \begin{array}{c} \bullet \\ \downarrow a \\ \bullet \end{array} \quad \begin{array}{c} \bullet \\ \downarrow b \\ \bullet \end{array}$ and $\pi^1 = \begin{array}{c} \bullet \\ \downarrow a \\ \bullet \end{array} \quad \begin{array}{c} \bullet \\ \searrow b \\ \bullet \end{array}$ ($a, b \in \mathcal{A}^g$). Clearly, $\dashv\pi^0\clubsuit \subseteq \dashv\pi^1\clubsuit$. We claim that $\pi^0 \not\sqsupseteq \pi^1$.

For this we must construct a context, $C[\cdot]$ such that $\dashv C[\pi^0]\clubsuit \not\subseteq \dashv C[\pi^1]\clubsuit$: take $C[\cdot] = h_0 h_1 (\pi \parallel [\cdot])$

with $\pi = \triangleright \bullet \xrightarrow{a} \bullet$, $h_0 = [a \mapsto \delta, b \mapsto \delta]$ and $h_1 = \delta\{c/aa\}$.

Then $h_1(\pi \parallel \pi^0) =$  and 

$h_1(\pi \parallel \pi^1) =$  , so that $\varepsilon\delta \in \diamond C[\pi^0]\diamond$ but $\varepsilon\delta \notin \diamond C[\pi^1]\diamond$.

Thus we see that \sqsubseteq depends on the actions that a system may refuse to do at some point in a computation. A context tests this by offering synchronization and disabling the (non-synchronized) actions. This construction is canonical and, granting this, it implies impossibility of such a test in case the tested system can still do a *local* action—as these do not synchronize—or the system can diverge—as this cannot be prevented.

Sequential composition allows additional differentiations. Consider $\pi^0 = \triangleright \bullet \xrightarrow{a} \bullet$ and $\pi^1 = \triangleright \bullet \xrightarrow{b} \bullet$

$\triangleright \bullet \xrightarrow{a} \bullet$ and take a context $\overline{C}[\cdot] = C[\cdot; \triangleright \bullet \xrightarrow{a} \bullet]$ ($C[\cdot]$ is the context of the previous example).

$\triangleright \bullet \xrightarrow{b} \bullet$
Then $\varepsilon\delta \in \diamond \overline{C}[\pi^0]\diamond$ and $\varepsilon\delta \notin \diamond \overline{C}[\pi^1]\diamond$ so that $\pi^0 \not\sqsubseteq \pi^1$ although $\diamond \pi^0 \diamond = \diamond \pi^1 \diamond$.

Before offering an explicit characterization of \sqsubseteq we need some definitions.

3-1 DEFINITION. Let $\pi \in \text{UTS}$, $v \in \mathcal{V}$ and $F \subseteq \mathcal{A}^g \cup \{\downarrow\}$. Then

- $\text{stable}(v) \iff \mathcal{L}(\bullet v) \subseteq \mathcal{A}_\delta^g$; write $\text{stable}(\pi, v)$ if π is not determined by the context,
- $\pi \text{ fails } F \iff \exists v \in I \text{ stable}(v), F \cap \mathcal{L}(\bullet v) = \emptyset \ \& \ \downarrow \in F \Rightarrow \neg \downarrow(v)$.

So, a node, v , in π is stable if it cannot diverge and there are no transitions with a *local* action out of it. It means that a suitable context can *control* the behavior of π if it is “at” v . In particular, π can be caused to deadlock and, more precisely, can be caused to fail certain global actions; namely, any set of actions $F \in \mathcal{A}^g$ that satisfies $\pi \text{ fails } F$.

Now we can state

3-2 THEOREM. Define the relation $\sqsubseteq \subseteq \text{UTS} \times \text{UTS}$ by

$$\pi^0 \sqsubseteq \pi^1 \iff \diamond \pi^0 \diamond \subseteq \diamond \pi^1 \diamond \ \& \ \forall s \in \mathcal{A}^*, F \subseteq \mathcal{A}^g \ \pi^0 \text{ after } s \text{ fails } F \Rightarrow \pi^1 \text{ after } s \text{ fails } F .$$

Then $\sqsubseteq = \sqsubseteq$.

The proof, which is given in the appendix, is by showing two-sided inclusion. The case $\sqsubseteq \subseteq \sqsubseteq$ is proved by assuming that $\pi^0 \not\sqsubseteq \pi^1$ and then constructing a context, $C[\cdot] \in \mathcal{C}$, that makes this explicit: $\diamond C[\pi^0]\diamond \not\subseteq \diamond C[\pi^1]\diamond$ (whence $\pi^0 \not\sqsubseteq \pi^1$). The case $\diamond \pi^0 \diamond \subseteq \diamond \pi^1 \diamond$ is trivial. Otherwise, for some $s \in \mathcal{A}^*$ and $F \subseteq \mathcal{A}_\downarrow^g$, we have π^0 after s fails F but not π^1 after s fails F . Let $\alpha \in F \setminus \{\downarrow\}$ if $\downarrow \in F$ and let

$\alpha = \delta$ otherwise. Take a set $A = \{x, y, z\} \cup \bar{A}$ with $\bar{A} = \{\bar{a} \mid a \in \mathcal{A}(s)\}$ such that $A \cap (\mathcal{A}(\pi^0) \cup \mathcal{A}(\pi^1)) = \emptyset$.

Construct a context $C[\cdot] = h_0 h_1 h_2 \left(\begin{array}{c} \begin{array}{c} \bullet \\ \downarrow \\ v \\ \downarrow \\ w \end{array} \begin{array}{c} \bullet \\ \downarrow \\ x \\ \downarrow \\ w \end{array} \end{array} \parallel \begin{array}{c} \bullet \\ \downarrow \\ \alpha \\ \downarrow \\ \bullet \end{array} \right)$, with $h_0 = [a \mapsto \delta, a \notin \bar{A} \cup \{z\}]$,

$h_1 = [a \mapsto \bar{a}, a \in \mathcal{A}(s) \cap \mathcal{A}^l]$ and $h_2 = \delta \circ [(a, x) \mapsto \bar{a}, a \in \mathcal{A}(s) \cap \mathcal{A}^g; (a, y) \mapsto z, a \in F \setminus \{\downarrow\}]$. We have $\bar{s}\delta \in \dashv C[\pi^0] \dashv$ but $\bar{s}\delta \notin \dashv C[\pi^1] \dashv$.

The proof of the other direction, $\sqsubseteq \subseteq \sqsupseteq$, uses induction on the structure of contexts and is an easy corollary of four lemma's which express the behaviors and failure possibilities of $\pi^0 \parallel \pi^1$, $h(\pi^0)$, $\pi^0 + \pi^1$ and $\pi^0; \pi^1$ in terms of the behaviors of, respectively, π^0 and π^1 .

First define for notational ease

$$\llbracket \pi \rrbracket = \dashv \pi \dashv \cup \{(s, F) \mid \pi \text{ after } s \text{ fails } F\}.$$

Obviously, $(s, F) \in \llbracket \pi \rrbracket$ and $\bar{F} \subseteq F$ imply that $(s, \bar{F}) \in \llbracket \pi \rrbracket$.

We may interpret $\llbracket \cdot \rrbracket$ as the semantics corresponding to \sqsubseteq ; we have $\pi^0 \sqsubseteq \pi^1 \iff \llbracket \pi^0 \rrbracket \subseteq \llbracket \pi^1 \rrbracket$. In fact, the lemma's below show that $\llbracket \cdot \rrbracket$ is compositional.

3-3 LEMMA.

$$\bullet \llbracket \pi^0 \parallel \pi^1 \rrbracket = \{s\rho \mid \exists s^i \rho^i \in \llbracket \pi^i \rrbracket (i < 2) \ s \in s^0 \parallel s^1, \rho = \rho^0 \parallel \rho^1\} \cup \left\{ (s, F) \mid \begin{array}{l} \exists (s^i, F^i) \in \llbracket \pi^i \rrbracket (i \leq 1) \ s \in s^0 \parallel s^1, \\ F \subseteq F^0 \cap F^1, \ F \cap (\mathcal{A}^g \setminus F^0) \cap (\mathcal{A}^g \setminus F^1) = \emptyset \end{array} \right\}, \text{ where}$$

$$- \rho^0 \parallel \rho^1 = \begin{cases} \downarrow, & \text{if } \rho^0 = \rho^1 = \downarrow \\ \uparrow, & \text{if } \rho^0 = \uparrow \text{ or } \rho^1 = \uparrow \\ \delta, & \text{if } \rho^0 = \rho^1 = \delta \end{cases} \quad (\text{hence, } \rho^0 \parallel \rho^1 \text{ is partial),}$$

- $s^0 \parallel s^1$ is defined as follows:

Let $h = \delta \circ [a|X \mapsto a, a \in \mathcal{A}]$ with $X \notin \mathcal{A}(s^0) \cup \mathcal{A}(s^1)$; let $e \in \mathcal{H}$ be given by $e(X) = \tau$.

Then

$$s \in s^0 \parallel s^1 \iff \exists \bar{s}^0, \bar{s}^1 \in \mathcal{A}_X^\dagger \ s^j = e(\bar{s}^j) (j \leq 1) \ \& \ s = (h(\bar{s}_i^0, \bar{s}_i^1))_{i < |\bar{s}^0|} \in (\mathcal{A} \setminus \{X\})^\dagger.$$

In the definition of $s^0 \parallel s^1$ we should be able to modify s^0 and s^1 into \bar{s}^0 and \bar{s}^1 (of equal length) by inserting X -actions ($s^j = e(\bar{s}^j)$). An X -action in s^0 should signify a non-synchronized action in s^1 and vice versa, which is ensured by having $s \in (\mathcal{A} \setminus \{X\})^\dagger$ so that s does not contain any X -action. The parallel composition fails any action, a , that both components fail ($a \in F^0 \cap F^1$) except if a obtains from the synchronization of actions in the component $(a \notin (\mathcal{A}^g \setminus F^0) \cap (\mathcal{A}^g \setminus F^1))$.

3-4 LEMMA.

$$\bullet \llbracket h(\pi) \rrbracket = \{h(s)\rho \mid s\rho \in \dashv \pi \dashv\} \cup \{h(s)\uparrow \mid s\downarrow \in \dashv \pi \dashv, s \in \mathcal{A}^\omega, h(s) \in \mathcal{A}^*\} \cup \{h(s)\delta \mid \exists F \subseteq \mathcal{A}^g \ (s, F \cup \{\downarrow\}) \in \llbracket \pi \rrbracket, \mathcal{A}^g \setminus F \subseteq h^{-1}(\delta)\} \cup \left\{ (h(s), F) \mid \begin{array}{l} \exists \bar{F} \in \mathcal{A}^g \ (s, \bar{F}) \in \llbracket \pi \rrbracket, \\ h^{-1}(\mathcal{A}_\tau^\dagger) \cap \mathcal{A}^g \subseteq \bar{F}, \ F \subseteq (h(\bar{F}) \cup h^{-1}(\delta) \cup h^{-1}(\tau) \setminus h^{-1}(\mathcal{A}^g)) \cap \mathcal{A}^g \end{array} \right\}.$$

So, $h(\pi)$ acquires additional divergence possibilities whenever an infinite sequence is renamed to a finite one; it acquires additional (maximal) computations in case all successor actions of a stable vertex are disabled; and it fails a global action, a , only if $h(\pi)$ remains stable ($h^{-1}(\mathcal{A}_\tau^\dagger) \cap \mathcal{A}^g \subseteq \bar{F}$) and either a is the renaming of an action that π fails ($a \in h(\bar{F})$) or a is disabled ($h(a) = \delta$) or a is erased and not introduced again by a renaming ($a \in h^{-1}(\tau) \setminus h^{-1}(\mathcal{A}^g)$).

3-5 LEMMA.

$$\bullet \llbracket \pi^0 + \pi^1 \rrbracket = (\llbracket \pi^0 \rrbracket \cup \llbracket \pi^1 \rrbracket) \cap (\mathcal{A}^\dagger \times \{\uparrow, \downarrow, \delta\} \cup \mathcal{A}^+ \times 2^{\mathcal{A}^i}) \cup \{(\varepsilon, F^0 \cap F^1) \mid (\varepsilon, F^i) \in \llbracket \pi^i \rrbracket (i < 2)\}$$

After the initial choice has been made $\pi^0 + \pi^1$ behaves as one of the components; before this has happened, only actions that both components may fail can fail in the choice.

3-6 LEMMA.

$$\bullet \llbracket \pi^0 ; \pi^1 \rrbracket = \llbracket \pi^0 \rrbracket \cap (\mathcal{A}^* \times \{\uparrow, \delta\} \cup \mathcal{A}^\omega \times \{\downarrow\} \cup \mathcal{A}^* \times 2^{\mathcal{A}^i}) \cup \{s^0 s^1 \rho^1 \mid s^0 \downarrow \in \llbracket \pi^0 \rrbracket, |s^0| < \omega, s^1 \rho^1 \in \llbracket \pi^1 \rrbracket\} \cup \{(s^0 s^1, F^1) \mid s^0 \downarrow \in \llbracket \pi^0 \rrbracket, |s^0| < \omega, (s^1, F^1) \in \llbracket \pi^1 \rrbracket, s^1 \neq \varepsilon\} \cup \{(s^0, F^0 \cap F^1) \mid s^0 \downarrow \in \llbracket \pi^0 \rrbracket, (\varepsilon, F^1) \in \llbracket \pi^1 \rrbracket\}$$

As for the last clause: if (s^0, F^0) corresponds to a computation in π^0 ending at a terminal node v (which is stable in π^0), then $\pi^0 ; \pi^1$ can fail the actions that both π^0 can fail at v (F^0) and π^1 can fail at some initial node (F^1). In case there is no such correspondence, $(s^0, F^0) \in \llbracket \pi^0 ; \pi^1 \rrbracket$ by the first clause, whence we already have $(s^0, F^0 \cap F^1) \in \llbracket \pi^0 ; \pi^1 \rrbracket$ as $F^0 \cap F^1 \subseteq F^0$.

The proofs of these lemma's are tedious but not really difficult. They follow from the corresponding uts constructions.

4 A verification criterion: failure simulation

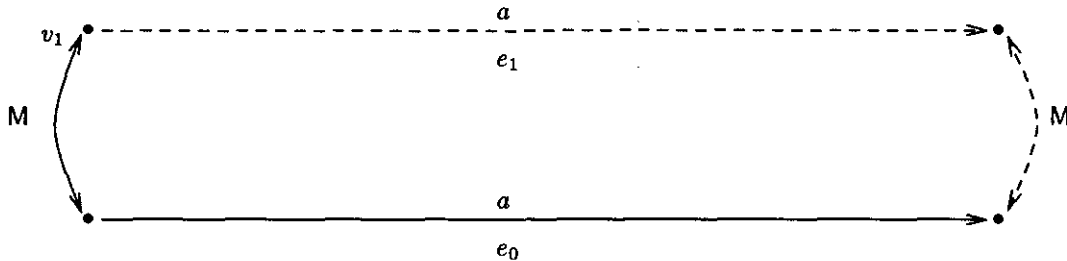
For refinement of 1st order transition systems—usually taken to be inclusion of behaviors—the paradigm of a verification criterion is that of *Milner simulation* [Mil71].

The idea behind this is that in order to prove that $\nrightarrow \pi^0 \subseteq \nrightarrow \pi^1$ one sets up a relation between the nodes of π^0 and π^1 , such that every initial node of π^0 is related to an initial node of π^1 . Such a relation should be *inductive* in the sense that if any two nodes v_0 and v_1 are related ($v^i \in \mathcal{V}^i$) then for every possible transition out of v_0 there must be a corresponding one out of v_1 (i.e., having the same effect) such that both reach nodes that again are related:

A *Milner simulation* from π^0 to π^1 (both in UTS) is a relation $M \subseteq \mathcal{V}^0 \times \mathcal{V}^1$ satisfying for every $V_0 \in \mathcal{V}^0, v_1 \in \mathcal{V}^1, e_0 \in \mathcal{E}^0, a \in \mathcal{A}$:

1. $v_0 \in I^0 \implies \exists v_1 \in I^1 v_0 M v_1$,
2. $\bullet e_0 M v_1, \mathcal{L}^0(e_0) = a \implies \exists e_1 \in \bullet v_1 e_1^\bullet M e_1^\bullet \& \mathcal{L}^1(e_1) = a$, and
3. $v_0 M v_1 \implies \uparrow(v_0) \Rightarrow \uparrow(v_1) \& \downarrow(v_0) \Rightarrow \downarrow(v_1) \& \delta(v_0) \Rightarrow \delta(v_1)$.

Clause (3) is not part of the usual definition of Milner simulation but is needed by our particular choice of behavior. The second clause is illustrated in the following picture:



where the dotted parts are the things that have to be found.

Alternatively, if we assume that the transition relation of an uts π can be interpreted as a set of relations, $(T_a)_{a \in \mathcal{A}}$ on $\mathcal{V} \times \mathcal{V}$, by

$$v T_a w \iff \exists e \in \bullet v \cap w \bullet \quad \mathcal{L}(e) = a^4,$$

then a Milner simulation, M , from π^0 to π^1 can be defined “relationally” by

$$1' \quad I^0 \subseteq I^1 M^T,$$

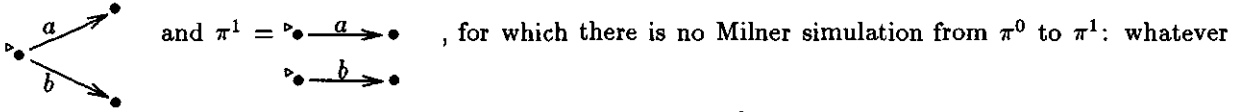
$$2' \quad M^T T_a^0 \subseteq T_a^1 M^T, \text{ and}$$

$$3' \quad \text{as (3)}.$$

Viewed in this way, it is more natural to consider relations on $\mathcal{V}^1 \times \mathcal{V}^0$.

If such a simulation can be set up, then this allows one to construct with every behavior of π^0 a corresponding one in π^1 —this is the essence of the inductiveness condition. In other words, the criterion is *sound*.

Even if $\dashv\pi^0\Phi \subseteq \dashv\pi^1\Phi$, such a simulation need not always exist. The standard example is $\pi^0 =$



initial vertices are related, either the a or the b -transition of π^0 cannot be mirrored. In other words, the criterion is *incomplete*.

This is well-known; as is well-known that if π^1 is *deterministic*, there is always a Milner simulation between π^0 and π^1 . The so-called *subset construction* is the classical way to turn an uts π into a deterministic one, ${}^d\pi$, such that $\dashv\pi\Phi = \dashv{}^d\pi\Phi$: vertices of ${}^d\pi$ are the subsets of π 's vertices; ${}^d\pi$'s initial state is I ; and two sets are related by an edge in ${}^d\pi$, labelled a , if the second set comprises all the states that can be reached via an a -labelled transition in π from some state in the first set [Eil74].

The idea behind failure simulation, which will turn out to be a sound and complete criterion for failure refinement, is to combine the idea of Milner simulation with the above subset construction to render the uts (sufficiently) deterministic. First we define a special case of failure simulation (which, incidently, already is complete):

4-1 DEFINITION. Let $\pi^0, \pi^1 \in \text{UTS}$. A simple failure simulation from π^0 to π^1 is a relation $R \in \mathcal{V}^0 \times 2^{\mathcal{V}^1}$ such that for every $e_0 \in \mathcal{E}^0$, $v_0 \in \mathcal{V}^0$, $V_1 \subseteq \mathcal{V}^1$, $a \in \mathcal{A}$:

$$1. \quad v_0 \in I^0 \implies \exists V_1 \subseteq I^1 v_0 R V_1,$$

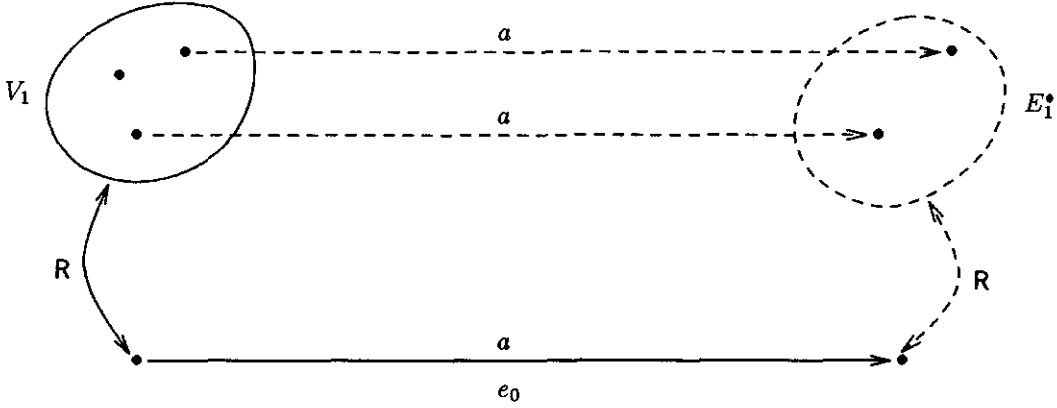
$$2. \quad \bullet e_0 R V_1, \mathcal{L}^0(e_0) = a \implies \exists E_1 \subseteq \bullet V_1 e_0^* R E_1^* \ \& \ \mathcal{L}^1(E_1) = \{a\}, \text{ and}$$

$$3. \quad v_0 R V_1 \implies \begin{aligned} & \exists v_1 \in V_1 \uparrow(v_0) \Rightarrow \uparrow(v_1) \ \& \ \downarrow(v_0) \Rightarrow \downarrow(v_1) \ \& \ \delta(v_0) \Rightarrow \delta(v_1) \ \& \\ & \exists v_1 \in V_1 \text{ stable}(v_0) \Rightarrow (\text{stable}(v_1) \ \& \ \mathcal{L}_g^1(\bullet v_1) \subseteq \mathcal{L}_g^0(\bullet v_0) \ \& \ \downarrow(v_1) \Rightarrow \downarrow(v_0)) \end{aligned}^5.$$

The second clause can be illustrated thus:

⁴Hence, if we assume that $|\bullet v \cap w \bullet \cap \mathcal{L}^{-1}(a)| \leq 1$ for any $v, w \in \mathcal{V}$ and $a \in \mathcal{A}$.

⁵By (2) and (3): $V_1 \neq \emptyset$.



Note that E_1 may be chosen a proper subset of $V_1 \cap \mathcal{L}^{-1}(a)$. I.e., not every a -labelled edge out of V_1 need be included in E_1 . This contrasts with the subset construction which would demand $E_1 = V_1 \cap \mathcal{L}^{-1}(a)$.

The third clause checks, in addition, that the deadlock and non termination possibilities "at" v_0 are allowed at some node $v_1 \in \mathcal{V}^1$: $\mathcal{L}_g^1(\bullet v_1) \subseteq \mathcal{L}_g^0(\bullet v_0)$ & $\downarrow(v_1) \Rightarrow \downarrow(v_0)$; in accordance with Theorem 3-2.

A more relational definition would render clause (2) as

$$2' \quad R^T T_a^0 \subseteq \iota^1 T_a^1 R^T, \text{ where } \iota^1 \text{ is the embedding relation on } \mathcal{V}^1 \times \mathcal{V}^1: V \iota^1 W \text{ iff } V \subseteq W .$$

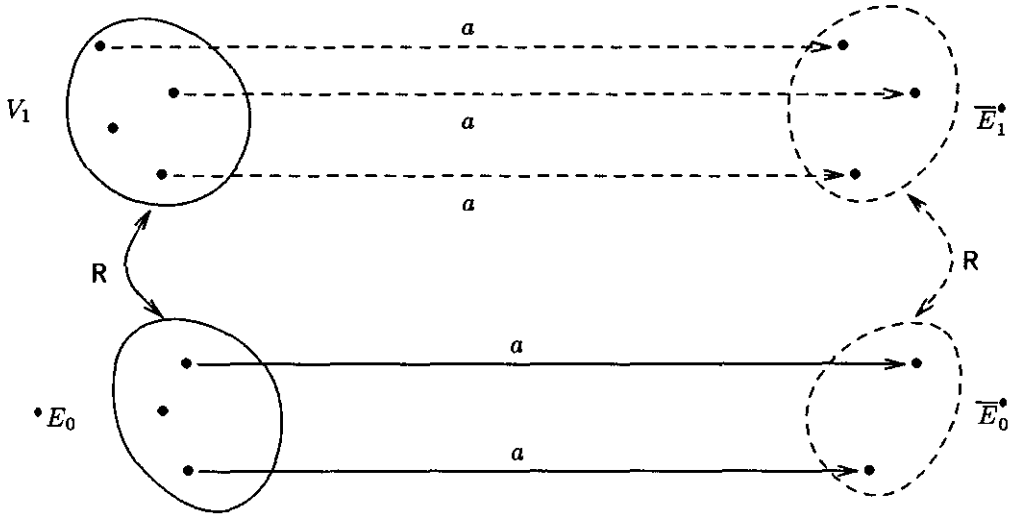
The other clauses would stay the same.

This notion of simulation can be weakened without losing completeness which has the advantage of offering more layway in proving refinement. It can also be made more symmetrical, relating subsets to subsets of nodes. This we will define now and it is what we will use in the rest of the paper.

4-2 DEFINITION. Let $\pi^0, \pi^1 \in \text{UTS}$.

- A failure simulation from π^0 to π^1 is a relation $R \in 2^{\mathcal{V}^0} \times 2^{\mathcal{V}^1}$ such that for every $E_0 \subseteq \mathcal{E}^0, V_0 \subseteq \mathcal{V}^0, V_1 \subseteq \mathcal{V}^1, a \in \mathcal{A}$:
 1. $\exists (V_i^0)_{i < \alpha} \subseteq \text{dom}(R) \quad I^0 = \bigcup_{i < \alpha} V_i^0$ & $\forall i < \alpha \exists V^1 \subseteq I^1 \quad V_i^0 R V^1$,
 2. $\bullet E_0 R V_1, \emptyset \neq \bar{E}_0 \subseteq E_0 \cap \mathcal{L}^{0^{-1}}(a), \bar{E}_0^* \in \text{dom}(R) \Rightarrow \exists \bar{E}_1 \subseteq \bullet V_1 \bar{E}_0^* R \bar{E}_1^* \text{ \& } \mathcal{L}^0(\bar{E}_0) = \mathcal{L}^1(\bar{E}_1)$,
 3. $V_0 R V_1 \Rightarrow \forall v_0 \in V_0 \exists v_1 \in V_1 \uparrow(v_0) \Rightarrow \uparrow(v_1) \text{ \& } \downarrow(v_0) \Rightarrow \downarrow(v_1) \text{ \& } \delta(v_0) \Rightarrow \delta(v_1) \text{ \& } \exists v_1 \in V_1 \text{ stable}(v_0) \Rightarrow (\text{stable}(v_1) \text{ \& } \mathcal{L}_g^1(\bullet v_1) \subseteq \mathcal{L}_g^0(\bullet v_0) \text{ \& } \downarrow(v_1) \Rightarrow \downarrow(v_0))$
 4. $\bullet E_0 \in \text{dom}(R) \Rightarrow \exists V_0 \dots V_n \in \text{dom}(R) (E_0 \cap \mathcal{L}^{0^{-1}}(a))^* = V_0 \cup \dots \cup V_n$,
- write $\pi^0 \hookrightarrow_R \pi^1$ if R is a failure simulation from π^0 to π^1 ,
- write $\pi^0 \hookrightarrow \pi^1$ if $\exists R \subseteq 2^{\mathcal{V}^0} \times 2^{\mathcal{V}^1} \pi^0 \hookrightarrow_R \pi^1$.

Clause (1) says that every initial vertex of π^0 should occur in some set of initial vertices and that every such set is related to a set of initial vertices of π^1 . Clause (2) is a direct generalization of the corresponding one in Definition 4-1: instead of a simple vertex, $\bullet e_0 \in \text{dom}(R)$, there is a set of vertices, $\bullet E_0 \in \text{dom}(R)$, and a set of edges, $\bar{E}_0 \subseteq E_0$, out of it, every edge in \bar{E}_0 labelled with a , $\bar{E}_0 \subseteq \mathcal{L}^{0^{-1}}(a)$, and $\bar{E}_0^* \in \text{dom}(R)$. This is pictured as follows:



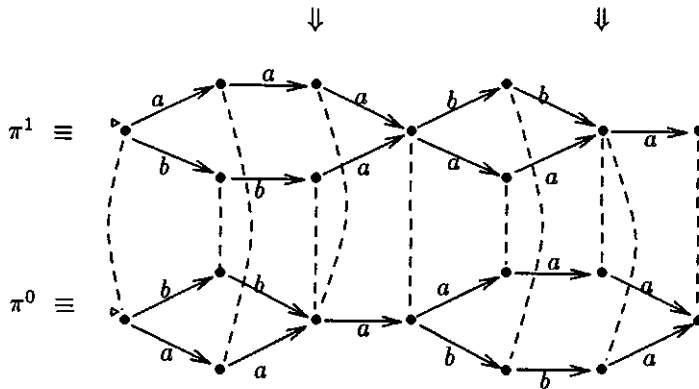
The additional clause (4) ensures that for every $a \in \mathcal{A}$ the vertices reachable by an a -labelled edge from $\bullet E_0$ are covered by sets in $\text{dom}(R)$. This is vital since we must construct for every computation in π^0 an equivalent one in π^1 .

Relationally, clause (2) would be expressed as

$$2' \quad R^T \iota^0 T_a^0 \subseteq \iota^1 T_a^1 R^T .$$

As the other clauses would stay the same, the relational view is not really worthwhile in this context.

The freedom to use simulation relations that are not functional is quite essential. In the example below, the only simulation relation, R , that can be set up between the two (trace) equivalent systems, is the one shown.



The two indicated positions show that neither R nor R^T is functional.

Note that R is a Milner simulation as well. This shows that the counter example is not a quirk of the particular simulation notion that we have shown.

The rest of the section gives the soundness and completeness proofs to show that $\sqsubseteq = \Leftarrow$. Fix some $\pi^0, \pi^1 \in \text{UTS}$ and let $R \subseteq 2^{\mathcal{V}^0} \times 2^{\mathcal{V}^1}$.

4.1 Soundness

The proof is based on the following auxiliary

4-3 LEMMA. Suppose $\pi^0 \hookrightarrow_{\mathbf{R}} \pi^1$. Then, for any $s \in \mathcal{A}^*$ and $v_0 \in \pi^0$ after s there are $V_0 \subseteq \pi^0$ after s , $V_1 \subseteq \pi^1$ after s such that $v_0 \in V_0$ and $V_0 \mathbf{R} V_1$.

PROOF. Induction on the length of s .

$s = \varepsilon$) π^0 after $\varepsilon = I^0$, hence clause (1) in the definition of failure simulation supplies a $V_0 \subseteq I^0$ with $v_0 \in V_0$ and $V_1 \subseteq I^1$ such that $V_0 \mathbf{R} V_1$.

$s = \bar{s}a$) $v_0 \in \pi^0$ after s implies there is a $\bar{v}_0 \in \mathcal{V}^0$ such that $\bar{v}_0 \in \pi^0$ after \bar{s} and $\bar{v}_0; \pi^0 \xrightarrow{a} v_0; \pi^0$. Induction gives a $\bar{V}_0 \subseteq \mathcal{V}^0$ and $\bar{V}_1 \subseteq \mathcal{V}^1$ with $\bar{v}_0 \in \bar{V}_0$ and $\bar{V}_0 \mathbf{R} \bar{V}_1$. Let $E_0 \in \bullet \bar{V}_0$ and $\tilde{E}_0 = E_0 \cap \mathcal{L}^{0-1}(a)$. By assumption, $E_0 \neq \emptyset$ and so $\tilde{E}_0 \neq \emptyset$. Since $\bullet E_0 \in \text{dom}(\mathbf{R})$, clause (4) gives a $V_0 \in \text{dom}(\mathbf{R})$ and $v_0 \in V_0 \subseteq \tilde{E}_0^*$. Let $\bar{E}_0 = \tilde{E}_0 \cap V_0^*$. Clause (2) implies existence of a set $\bar{E}_1 \subseteq \bullet \bar{V}_1$ such that $\bar{E}_0^* \mathbf{R} \bar{E}_1^*$ and $\mathcal{L}^1(\bar{E}_1) = \{a\}$. So, define $V_0 = \bar{E}_0^*$ and $V_1 = \bar{E}_1^*$. Then $v_0 \in V_0 \subseteq \pi^0$ after s , $V_1 \subseteq \pi^1$ after s and $V_0 \mathbf{R} V_1$. \square

4-4 THEOREM (Soundness). $\hookrightarrow \subseteq \sqsupseteq$.

PROOF. Choose $\pi^0 \hookrightarrow_{\mathbf{R}} \pi^1$. First take some $s\rho \in \nrightarrow \pi^0 \Phi$. If $s \in \mathcal{A}^*$, then there is some $v_0 \in \mathcal{V}^0$ and $\pi^0 \xrightarrow{s} v_0; \pi^0$ with $\rho(v_0)$ true. By Lemma 4-3, there are $V^i \subseteq \pi^i$ after s ($i \leq 1$) with $v_0 \in V_0$ and $V_0 \mathbf{R} V_1$. By clause (3) of the definition of failure simulation, $\rho(v_1)$ holds for some $v_1 \in V_1$; whence $s\rho \in \nrightarrow \pi^1 \Phi^{\circ}$. If $s \in \mathcal{A}^{\omega}$ then for every $\bar{s} \prec s$ (by definition $\bar{s} \in \mathcal{A}^*$) there is a $\bar{v}_0 \in \pi^0$ after \bar{s} and again by the previous Lemma, π^1 after $\bar{s} \neq \emptyset$. Hence, $s \downarrow \in \nrightarrow \pi^1 \Phi$.

Next, let $s \in \mathcal{A}^*$ and $F \subseteq \mathcal{A}_1^g$. Assume that π^0 after s fails F holds. This implies a $v_0 \in \pi^0$ after s such that $\text{stable}(v_0)$ holds and $F \cap \mathcal{L}_g^0(\bullet v_0) = \emptyset$. The same argument as above gives a $v_1 \in \pi^1$ after s , also with $\text{stable}(v_1)$ being true, $F \cap \mathcal{L}_g^1(\bullet v_1) = \emptyset$ and $\neg \downarrow(v_1)$ if $\downarrow \in F$. I.e., π^1 after s fails F holds, too. \square

4-5 COROLLARY. If there is a simple failure relation from π^0 to π^1 , then $\pi^0 \sqsupseteq \pi^1$.

PROOF. If \mathbf{R} is such a simple failure relation, then $\pi^0 \hookrightarrow_{\hat{\mathbf{R}}} \pi^1$ holds with

$$\hat{\mathbf{R}} = \{(\{v_0\}, V_1) \mid (v_0, V_1) \in \mathbf{R}\}.$$

\square

4.2 Completeness

Completeness can be proved directly and we state:

4-6 THEOREM (Completeness). $\sqsupseteq \subseteq \hookrightarrow$.

PROOF. Let $\pi^0 \sqsupseteq \pi^1$ and define $\mathbf{R} \subseteq 2^{\mathcal{V}^0} \times 2^{\mathcal{V}^1}$ by

$$\mathbf{R} = \{(\{v_0\}, V_1) \mid \exists s \in \mathcal{A}^* \pi^0 \xrightarrow{s} v_0; \pi^0 \ \& \ \pi^1 \text{ after } s = V_1; \pi^1\}.$$

We show that $\pi^0 \hookrightarrow_{\mathbf{R}} \pi^1$ holds:

1. By taking $s = \varepsilon$ we immediately obtain that $\{v_0\} \in \text{dom}(\mathbf{R})$ and $\{v_0\} \mathbf{R} I^1$ for all $v_0 \in I^0$.
2. Let $\bullet E_0 \mathbf{R} V_1$, $\bar{E}_0 \subseteq E_0 \cap \mathcal{L}^{0-1}(a)$ and $\bar{E}_0^* \in \text{dom}(\mathbf{R})$. By definition of \mathbf{R} , we must have $\bar{E}_0 = E_0 = \{e_0\}$ and $\mathcal{L}^0(e_0) = a$. Also, $\pi^0 \xrightarrow{s} \bullet e_0; \pi^0 \xrightarrow{a} e_0^*; \pi^0$; whence, if $\bar{E}_1 = \bullet V_1 \cap \mathcal{L}^{1-1}(a)$, $\bar{E}_0^* \mathbf{R} \bar{E}_1^*$ and $\mathcal{L}^0(\bar{E}_0) = \mathcal{L}^1(\bar{E}_1)$.

3. $V_0 \text{ R } V_1$ implies $V_0 = \{v_0\}$. Choose an $s \in \mathcal{A}^*$ such that $\pi^0 \xrightarrow{s} v_0$; π^0 and π^1 after $s = V_1$; π^1 .

If $\rho(v_0)$ is true then $s\rho \in \dashv\pi^0\clubsuit$; whence, by assumption $s\rho \in \dashv\pi^1\clubsuit$, which in its turn implies that $\rho(v_1)$ holds for some $v_1 \in V_1$ (because $\pi^1 \xrightarrow{s} v_1$; $\pi^1 \Rightarrow v_1 \in V_1$).

Next, if $\text{stable}(v_0)$ holds, $F \cap \mathcal{L}_g^0(\bullet v_0) = \emptyset$ for some $F \subseteq \mathcal{A}_1^g$ and $\neg\downarrow(v_0)$ if $\downarrow \in F$, then π^0 after s fails F is true and, again by assumption, so is π^1 after s fails F . From this conclude that $\text{stable}(v_1)$ holds, $F \cap \mathcal{L}_g^1(\bullet v_1) = \emptyset$ and $\neg\downarrow(v_1)$ if $\downarrow \in F$ for some $v_1 \in V_1$.

4. Let $\bullet E_0 \in \text{dom}(\text{R})$ and $\emptyset \neq \bar{E}_0 = E_0 \cap \mathcal{L}^{0-1}(a)$. Hence, $E_0 = \{e_0\}$, $\mathcal{L}^0(e_0) = a$ and for some $s \in \mathcal{A}^*$: $\pi^0 \xrightarrow{s} \bullet e_0$; $\pi^0 \xrightarrow{a} e_0^*$; π^0 . This implies that $\{e_0^*\} \in \text{dom}(\text{R})$. \square

4–7 COROLLARY. *If $\pi^0 \sqsupseteq \pi^1$ then there is a simple failure simulation from π^0 to π^1 .*

PROOF. If R is the relation as defined in the proof of the completeness theorem, then

$$\bar{\text{R}} = \{(v_0, V_1) \mid (\{v_0\}, V_1) \in \text{R}\}$$

is a simple failure relation from π^0 to π^1 . \square

So, both failure simulation and simple failure simulation are sound and complete verification criteria for proving failure refinement. Note that if in clause (3) of either definition we ignore the part concerning stability and the next possible moves, the resulting simulations are sound and complete criteria for *trace refinement*: $\dashv\pi^0\clubsuit \subseteq \dashv\pi^1\clubsuit$. This is an easy consequence of the above soundness and completeness proofs.

5 First order transition systems

In a first order transition system, there is an implicit notion of state and edge labels now denote tests and actions that depend on, respectively, update the state.

- Given a signature Σ , let $\text{Act}(\Sigma) = \text{Gact}(\Sigma) \cup \text{Lact}(\Sigma)$ be some set of action over Σ partitioned into $\text{Gact}(\Sigma)$ —the global actions—and $\text{Lact}(\Sigma)$ —the local actions. The ‘actions’ δ and \uparrow do not appear in $\text{Act}(\Sigma)$. The precise form of these actions does not matter at this moment. We do need that given a Σ -structure, \mathbf{A} , there is a function, $\llbracket \cdot \rrbracket^{\mathbf{A}}: \text{Act}(\Sigma)_{\uparrow} \mapsto 2^{\mathcal{S} \times \mathcal{S}}$ that gives the input-output behavior of the actions. We (arbitrarily) assume that $\llbracket \uparrow \rrbracket^{\mathbf{A}} = \text{Id}_{\mathcal{S}}$. For the rest of the section, fix some Σ -structure, \mathbf{A} .

- The class of *1st order transition systems over $\text{Act}(\Sigma)$* is $\text{TS}_{\text{Act}(\Sigma)}$ (TS_{Σ} or just TS) and is defined as

$$\{ (\partial_s, \partial_t: \mathcal{E} \mapsto \mathcal{V}, I, T, \mathcal{L}: \mathcal{E} \mapsto L(\Sigma) \times \text{Act}(\Sigma)_{\delta\uparrow}) \mid |\mathcal{E}| \text{ and } |\mathcal{V}| \text{ are finite} \}.$$

So, a 1st order transition system (ts), also denoted by π , is a transition system whose underlying graph is finite and each of whose edges is labelled with a *test* from $L(\Sigma)$ and an *action* from $\text{Act}(\Sigma)$. Write $\mathcal{L}^t(e)$, respectively, $\mathcal{L}^a(e)$ for the test, respectively, action associated with e .

- As for the meaning of a $\pi \in \text{TS}$, we can define, like for uts’s, transition relations, \xrightarrow{a} , for $a \in \text{Act}(\Sigma)$; this time as relations over $\text{TS} \times \mathcal{S}$:

$$\pi, \sigma \xrightarrow{a} \bar{\pi}, \bar{\sigma} \iff \exists e \in \bullet I \ \mathbf{A}, \sigma \models \mathcal{L}^t(e), (\sigma, \bar{\sigma}) \in \llbracket \mathcal{L}^a(e) \rrbracket^{\mathbf{A}}, \bar{\pi} = e^*; \pi. \quad (6)$$

These relations fix the behavior of a 1st order system, as they do for uts's.

- The next step is to associate with every $\pi \in \text{TS}$ an uts ${}^t\pi$ which has the same behavior as π .
The transition relation defined above gives the general idea: the vertex set of ${}^t\pi$ will also record the state; an edge label in ${}^t\pi$ will also include the state-transformation.

Given $\pi \in \text{TS}_{\mathcal{Act}(\Sigma)}$, define ${}^t\pi \in \text{UTS}_{\mathcal{S} \times \mathcal{Act}(\Sigma) \times \mathcal{S}}$ by

- ${}^t\mathcal{V} = \mathcal{V} \times \mathcal{S}$,
- ${}^t\mathcal{E} = \mathcal{S} \times \mathcal{E} \times \mathcal{S}$,
- ${}^t\partial_s(\sigma, e, \bar{\sigma}) = ({}^*e, \sigma)$ and ${}^t\partial_t(\sigma, e, \bar{\sigma}) = (e^*, \bar{\sigma})$ iff $\mathbf{A}, \sigma \models \mathcal{L}^t(e)$ and $(\sigma, \bar{\sigma}) \in \llbracket \mathcal{L}^a(e) \rrbracket^{\mathbf{A}}$,
- ${}^tI = I \times \mathcal{S}$,
- ${}^tT = T \times \mathcal{S}$, and
- ${}^t\mathcal{L}(\sigma, e, \bar{\sigma}) = (\sigma, \mathcal{L}^a(e), \bar{\sigma})$.

Hence, we shall have $\mathcal{A}^t = \mathcal{S} \times \mathcal{Act}(\Sigma) \times \mathcal{S}$ and $\mathcal{A}^g = \mathcal{S} \times \mathcal{Act}(\Sigma) \times \mathcal{S}$.

- The following easy to prove correspondence holds between π and ${}^t\pi$:

$$\pi, \sigma \xrightarrow{a} \bar{\pi}, \bar{\sigma} \iff {}^t\pi \xrightarrow{(\sigma, a, \bar{\sigma})} {}^t\bar{\pi}.$$

As to the adequacy of the translation, we can state the following.

First note that \xrightarrow{a} as defined by (6) is a relation on $\text{TS} \times \mathcal{S}$. We define bisimulation, \approx , on $\text{TS} \times \mathcal{S}$ as follows:

$\pi^0 \approx \pi^1$ iff there is an $R \subseteq (\text{TS} \times \mathcal{S})^2$ such that for every $\sigma^0, \sigma^1 \in \mathcal{S}$: $(\pi^0, \sigma^0) R (\pi^1, \sigma^1)$ and

1. $\pi^0, \sigma^0 \xrightarrow{a} \bar{\pi}^0, \bar{\sigma}^0 \Rightarrow \exists \bar{\pi}^1, \bar{\sigma}^1 \pi^1, \sigma^1 \xrightarrow{a} \bar{\pi}^1, \bar{\sigma}^1 \ \& \ (\bar{\pi}^0, \bar{\sigma}^0) R (\bar{\pi}^1, \bar{\sigma}^1)$,
2. $\pi^1, \sigma^1 \xrightarrow{a} \bar{\pi}^1, \bar{\sigma}^1 \Rightarrow \exists \bar{\pi}^0, \bar{\sigma}^0 \pi^0, \sigma^0 \xrightarrow{a} \bar{\pi}^0, \bar{\sigma}^0 \ \& \ (\bar{\pi}^0, \bar{\sigma}^0) R (\bar{\pi}^1, \bar{\sigma}^1)$, and
3. $(\pi^0, \sigma^0) R (\pi^1, \sigma^1) \Rightarrow \sigma^0 = \sigma^1 \ \& \ \forall v_0 \in I^0 \ \rho(v_0, \sigma^0) \Rightarrow \exists v_1 \in I^1 \ \rho(v_1, \sigma^1)$

Since this definition does not depend on the cardinality of ts's, it also defines bisimulation on $\text{UTS} (\cong \text{UTS} \times \{\star\})$.

Now we have

$$\text{For any } \pi^0, \pi^1 \in \text{TS}: \pi^0 \approx \pi^1 \text{ iff } {}^t\pi^0 \approx {}^t\pi^1.$$

So, ${}^t\pi$ indeed is the 'correct' translation of π . With this correspondence, we can have $\dashv\vdash\pi\vdash = \dashv\vdash{}^t\pi\vdash$ for $\pi \in \text{TS}$. Also, we can define refinement for first order systems simply by

$$\pi^0 \supseteq \pi^1 \iff {}^t\pi^0 \supseteq {}^t\pi^1 \quad (\pi^0, \pi^1 \in \text{TS}). \quad (7)$$

It would be better to write $\mathbf{A} \models \pi^0 \supseteq \pi^1$ instead of $\pi^0 \supseteq \pi^1$ so as to stress the dependence of the definition on the structure \mathbf{A} .

This canonical translation, ${}^t: \text{TS} \mapsto \text{UTS}$, can be used to formulate a notion of simulation that is defined in terms of the vertices and edges of first order systems. This is the subject of the next section.

The above gives a general, abstract set-up to deal with (refinement of) first order systems. The key step is the translation t . This allows the principles developed for uts's to be 'lifted' to first order systems.

The definition embodied in (7), which is based on it, implies that \sqsubseteq will be a pre-congruence for any program combinator on ts's, $C(\cdot, \dots, \cdot)$, such that

$$C(\pi^0, \dots, \pi^n), \sigma \xrightarrow{\alpha} C(\bar{\pi}^0, \dots, \bar{\pi}^n), \bar{\sigma} \iff {}^tC({}^t\pi^0, \dots, {}^t\pi^n) \xrightarrow{(\sigma, \alpha, \bar{\sigma})} {}^tC({}^t\bar{\pi}^0, \dots, {}^t\bar{\pi}^n),$$

where tC is defined using the combinators of the uts algebra.

The remainder of the section gives some examples. We instantiate $Act(\Sigma)$ so as to have assignments to local and shared variables and to have synchronous communication actions. We then define a *parallel composition*, $\cdot \parallel \cdot$, an *encapsulation*, $Enc(\cdot, enc)$, and a *hiding operator*, $Hide(\cdot, V)$, for such ts's. Encapsulation turns shared variables into local ones as specified by the function $enc: Svar \mapsto Pvar$; hiding makes the values of the private variables in $V \subseteq Pvar$ unobservable.

So, first partition Var into $Pvar$ and $Svar$. $Pvar$ is the set of *private* variables that can be accessed by at most one process; $Svar$ is the set of *shared* variables. Also introduce a set, $Chan$, of *channel names*: communication will occur along channels between exactly two transition systems. Then, we define $\mathcal{L}act(\Sigma)$ and $\mathcal{G}act(\Sigma)$ by

$$\begin{aligned} \mathcal{L}act(\Sigma) &= \{x := e \mid x \in Pvar, e \in Tm(\Sigma)\} \cup \\ &\quad \{C : x := e \mid x \in Pvar, e \in Tm(\Sigma), C \in Chan\}, \\ \mathcal{G}act(\Sigma) &= \{C!e \mid C \in Chan, e \in Tm(\Sigma)\} \cup \\ &\quad \{C?x \mid C \in Chan, x \in Pvar\} \cup \\ &\quad \{W : x := e \mid W \in \{E, S\}, x \in Svar, e \in Tm(\Sigma)\}. \end{aligned}$$

We shall also use $Pvar$, $Svar$ and $Chan$ as functions: $Pvar(\pi) \subseteq Pvar$ is the set of private variables that appear in π , etc. .

The intention is that $x := e$ is an ordinary assignment to a private variable; $C : x := e$ denotes a communication along channel C , causing x to obtain the value of e ; $C!e$, respectively, $C?x$ denotes the action of sending the value of e along channel C , respectively, receiving a value along channel C and assigning it to the variable x ; and $W : x := e$ is an assignment to a shared variable where W indicates whether the assignment originated in the system, $W \equiv S$, or whether the system's environment is responsible for it, $W \equiv E$. The latter type of action needs some elaboration.

As a comparison, first look at communication between processes. If C is a channel between two processes π^0 and π^1 then π^0 should only be able to perform a $C?x$ -transition if π^1 simultaneously executes a $C!e$ -transition and vice versa. Hence, $\pi^0 \parallel \pi^1$ will be defined as $h_0 h_1({}^t\pi^0 \parallel {}^t\pi^1)$, where h_1 , among other things, maps $C?x|C!e$ onto $C : x := e$, denoting a successful communication, while the renaming function h_0 , among other things, renames $C?x$ and $C!e$ to δ to enforce synchronous communication.

Now, if we look at assignments to a shared variable, $x \in Svar$, occurring in π^0 then, in principle, π^0 does not control when its environment assigns to x . Yet, in a program $Enc(\pi^0 \parallel \pi^1, [x \mapsto y])$, in which x has been made a private variable (y) of $\pi^0 \parallel \pi^1$, we must make sure that every assignment to x originates in either π^0 or π^1 . The way this is solved here is by explicitly indicating from where an assignment to a shared variable originates: in the system, S , or in its environment, E . In the definition of $\pi^0 \parallel \pi^1$, $h_0 h_1({}^t\pi^0 \parallel {}^t\pi^1)$, h_1 in combination with h_0 will also map $S : x := e|E : x := e$ into $S : x := e$ and $E : x := e|E : x := e$ into $E : x := e$. The former pair indicates that an update by π^i 's ($i \leq 1$) environment is in fact performed by π^{1-i} ; the latter pair indicates that the update originates in the environment of both π^0 and π^1 . Also, h_0 maps any "unmatched" $S : x := e$ and $E : x := e$ into δ to enforce consistency. With this set-up, encapsulating x in $\pi^0 \parallel \pi^1$ then means, among other things, that actions $E : x := e$ are renamed to δ , since π^0 and π^1 now are the only processes that can assign to x . This way of modelling shared variables was already used in [BKP84].

- Next, we must define the semantics of the actions. This is straightforward:

$$\begin{aligned} \llbracket x := e \rrbracket^{\mathbf{A}} &= \llbracket W : x := e \rrbracket^{\mathbf{A}} = \{(\sigma, \bar{\sigma}) \mid \bar{\sigma} = \sigma\{\sigma(e)/x\}\}, \quad W \in \{C, S, E\}, \\ \llbracket C!e \rrbracket^{\mathbf{A}} &= Id_{\mathcal{S}}, \\ \llbracket C?x \rrbracket^{\mathbf{A}} &= \{(\sigma, \sigma\{v/x\}) \mid v \in |\mathbf{A}|\}. \end{aligned}$$

Now we are ready to define the program combinators. The strategy will be to first define the required behavior by transition relations and then constructing a translation that will generate this behavior. We shall not give proofs of our claims below.

- **Parallel composition.** Given $\pi^0, \pi^1 \in \text{TS}$, $\pi^0 \parallel \pi^1$ is defined if $Pvar(\pi^i) \cap Var(\pi^{1-i}) = \emptyset$ for $i \leq 1$. Define $Chan(\pi^0 \parallel \pi^1) = (Chan(\pi^0) \cup Chan(\pi^1)) \setminus (Chan(\pi^0) \cap Chan(\pi^1))$. We want $\pi^0 \parallel \pi^1$ to behave as follows:

$$\begin{aligned} - \pi^0, \sigma &\xrightarrow{a} \bar{\pi}^0, \bar{\sigma} \implies \pi^0 \parallel \pi^1, \sigma \xrightarrow{a} \bar{\pi}^0 \parallel \bar{\pi}^1, \bar{\sigma} \quad \text{if } a \in \mathcal{L}act(\Sigma), \\ - \pi^0, \sigma &\xrightarrow{C!e} \bar{\pi}^0, \bar{\sigma} \implies \pi^0 \parallel \pi^1, \sigma \xrightarrow{C : x := e} \bar{\pi}^0 \parallel \bar{\pi}^1, \bar{\sigma} \quad \text{if } \bar{\sigma}(x) = \sigma(e) \text{ and } C \in Chan(\pi_i) \\ - \pi^1, \sigma &\xrightarrow{C?x} \bar{\pi}^1, \bar{\sigma} \\ & i \leq 1, \\ - \pi^0, \sigma &\xrightarrow{E : x := e} \bar{\pi}^0, \bar{\sigma} \implies \pi^0 \parallel \pi^1, \sigma \xrightarrow{W : x := e} \bar{\pi}^0 \parallel \bar{\pi}^1, \bar{\sigma} \quad \text{for } W \in \{S, E\}, \\ - \pi^1, \sigma &\xrightarrow{W : x := e} \bar{\pi}^1, \bar{\sigma} \\ - \cdot \parallel \cdot &\text{ is commutative.} \end{aligned}$$

So, in particular one sees that an update of a shared variable must be ‘allowed’ by all components. One should not interpret this as a system specifying its environment, but rather as a system that makes certain assumptions about how its environment behaves. Apart from obtaining a simple transition relation, this has the additional advantage of giving a program some control over access to a shared variable. E.g., a

program $\bullet \xrightarrow{S : x := 1} \bullet$ ($x \in Svar$) effectively says that it has exclusive access to x because it does not include any environment steps.

In order to construct ${}^t(\pi^0 \parallel \pi^1)$ so that

$$\pi^0 \parallel \pi^1, \sigma \xrightarrow{a} \bar{\pi}^0 \parallel \bar{\pi}^1, \bar{\sigma} \iff {}^t(\pi^0 \parallel \pi^1) \xrightarrow{(\sigma, a, \bar{\sigma})} {}^t(\bar{\pi}^0 \parallel \bar{\pi}^1),$$

we first define for $\sigma^0, \sigma^1 \in \mathcal{S}$ such that $\sigma^0 \upharpoonright V = \sigma^1 \upharpoonright V$ where $V = (Var(\pi^0) \cap Var(\pi^1)) \cup \mathcal{V} \setminus Var(\pi^0 \parallel \pi^1)$ ⁶:

$$(\sigma^0 + \sigma^1)(x) = \begin{cases} \sigma^0(x), & \text{if } x \in Var(\pi^0) \\ \sigma^1(x), & \text{otherwise} \end{cases}.$$

Also, for $a, b \in \mathcal{G}act(\Sigma)$: $a + b = \begin{cases} \bar{W} : x := e, & \text{if } \{a, b\} = \{E : x := e, W : x := e\} \text{ and } W \in \{E, S\} \\ C : x := e, & \text{if } \{a, b\} = \{C!e, C?x\} \end{cases}.$

Now we set ${}^t(\pi^0 \parallel \pi^1) = h_0 h_1 ({}^t\pi^0 \parallel {}^t\pi^1)$ where

$$\begin{aligned} - h_1 &= \delta \circ [(\sigma^0, a, \bar{\sigma}^0)(\sigma^1, b, \bar{\sigma}^1) \mapsto (\sigma^0 + \sigma^1, a + b, \bar{\sigma}^0 + \bar{\sigma}^1)] \quad \text{and} \\ - h_0 &((\sigma, \bar{W} : x := e, \bar{\sigma})) = (\sigma, W : x := e, \bar{\sigma}) \quad \text{for } W \in \{S, E\}, \\ &h_0((\sigma, W : x := e, \bar{\sigma})) = \delta \quad \text{for } W \in \{S, E\}, \\ &h_0((\sigma, C?x, \bar{\sigma})) = h((\sigma, C!e, \bar{\sigma})) = \delta \quad \text{if } C \in Chan(\pi^0) \cap Chan(\pi^1). \end{aligned}$$

⁶Formally, we should define what $Var(\pi^0 \parallel \pi^1)$ means: it means $Var(\pi^0) \cup Var(\pi^1)$. Such trivial extensions of $Var(\cdot)$ and the other syntactic functions, however, should be clear.

It is not difficult to see that ${}^t(\pi^0 \parallel \pi^1)$ behaves as suggested.

• **Encapsulation.** Given $\pi \in \text{TS}$ and $enc: \text{Svar} \mapsto \text{Pvar}$, in the system $Enc(\pi, enc)$ the variables in $dom(enc)$ are no longer shared with π 's environment and are renamed to private variables. Obviously, $Enc(\pi, enc)$ is only defined if enc is injective and $ran(enc) \cap \text{Pvar}(\pi) = \emptyset$.

For an action a let $enc(a)$ be the action obtained by renaming the variables (if any) according to enc . Also, for a state σ let $enc(\sigma)$ be the state $\bar{\sigma}$ such that: $\bar{\sigma}(x) = \begin{cases} \sigma(enc^{-1}(x)), & \text{if } x \in ran(enc) \\ \sigma(x), & \text{otherwise} \end{cases}$.

Then, $Enc(\pi, enc)$ should behave as follows:

$$- \pi, \sigma \xrightarrow{a} \bar{\pi}, \bar{\sigma} \implies Enc(\pi, enc), enc(\sigma) \xrightarrow{enc(a)} Enc(\bar{\pi}, enc), \tilde{\sigma} \text{ provided}$$

$$a \equiv E : x := e \Rightarrow x \notin dom(enc) \text{ and } \tilde{\sigma}(x) = \begin{cases} enc(\bar{\sigma})(x), & \text{if } x \notin dom(enc) \\ \sigma(x), & \text{otherwise} \end{cases}.$$

The translation, ${}^tEnc(\pi, enc)$, is given by $h({}^t\pi)$ where

$$- h((\sigma, a, \bar{\sigma})) = \begin{cases} \delta, & \text{if } a \equiv E : x := e \text{ and } x \in dom(enc) \\ (enc(\sigma), h(a), \tilde{\sigma}), & \text{otherwise} \end{cases},$$

and $\tilde{\sigma}$ is defined as above.

Again we have

$$Enc(\pi, enc), \sigma \xrightarrow{a} Enc(\bar{\pi}, enc), \bar{\sigma} \iff {}^tEnc(\pi, enc) \xrightarrow{(\sigma, a, \bar{\sigma})} {}^tEnc(\bar{\pi}, enc).$$

• **Hiding.** Given $\pi \in \text{TS}$ and $V \subseteq \text{Pvar}$, $Hide(\pi, V)$, makes the private variables in V unobservable. We want $Hide(\pi, V)$ to behave as follows:

$$- \pi, \sigma \xrightarrow{a} \bar{\pi}, \bar{\sigma} \implies Hide(\pi, V), \sigma \xrightarrow{a} Hide(\pi, V), \tilde{\sigma}, \text{ where } \tilde{\sigma}(x) = \begin{cases} \sigma(x), & \text{if } x \in V \\ \bar{\sigma}(x), & \text{otherwise} \end{cases}.$$

Then ${}^tHide(\pi, V) = h({}^t\pi)$, with

$$- h((\sigma, a, \bar{\sigma})) = (\sigma, a, \tilde{\sigma}), \text{ where } \tilde{\sigma} \text{ is defined as above.}$$

It is straightforward to prove that

$$Hide(\pi, V), \sigma \xrightarrow{a} Hide(\bar{\pi}, V), \bar{\sigma} \iff {}^tHide(\pi, V) \xrightarrow{(\sigma, a, \bar{\sigma})} {}^tHide(\bar{\pi}, V).$$

As we already stated, these are only examples of composition operators for 1st order systems. One can also define a hiding operator to hide actions or channels. As with encapsulation and hiding of variables, here too, this is a question of defining the appropriate renaming operator. We stress again that our notion of (1st order) refinement by definition is a pre-congruence for all such operators.

6 Verifying first order refinement

The intent, here, is not to find any old translation of failure simulation to first order systems, but rather to formulate it in terms of more or less standard assertional methods for transition systems: Floyd's inductive assertion method [Flo67]. For the remainder of the section, fix some $\pi^0, \pi^1 \in \text{TS}_{Act(\Sigma)}$ such that $\pi^0 \sqsupseteq \pi^1$ and a Σ -structure \mathbf{A} .

We need to define $\hookrightarrow_{\mathbf{R}} \subseteq \text{TS}_{Act(\Sigma)} \times \text{TS}_{Act(\Sigma)}$ and conditions on $\hookrightarrow_{\mathbf{R}}$ so that $\pi^0 \hookrightarrow_{\mathbf{R}} \pi^1$ iff ${}^t\pi^0 \hookrightarrow_{\mathbf{R}} {}^t\pi^1$ for an appropriate 'translation' of \mathbf{R} . The first thing is to decide how to represent a failure simulation, \mathbf{R} , in

terms of assertions. Obviously, R , too, will relate sets of vertices with each other and tR will be a relation over $2^{\mathcal{V}^0 \times \mathcal{S}} \times 2^{\mathcal{V}^1 \times \mathcal{S}}$, since $\mathcal{V}^i \times \mathcal{S}$ is the vertex set of ${}^t\pi^i$. Now, in an inductive assertion proof, the assertion associated with a vertex intends to describe the states with which the system can reach that vertex. *This suggests to associate assertions with sets of vertices and to relate such sets if the corresponding assertions allow both sets to be reached with the same state.* We also know from the completeness proof that which sets are related is also based on the (partial) computations. Hence, we use an assertion language, As , in which *history variables* may occur that have (partial) computations as value; i.e., that are of type $\mathcal{His} = (\mathcal{S} \times Act(\Sigma) \times \mathcal{S})^*$. Note that these history variables are additions to the assertion language and, hence, differ from what some researchers [AL88] call history variables. These are auxiliary variables that are added to a program and to which one assigns values so as to encode information about the computation history. There are some further conditions on As that we shall need later on: As will be a (weak monadic) second order language in which we allow quantification over sets of vertices and allow membership tests for vertices, edges and actions (which means that there are variables of the respective types: V, v, e and a). Also, the signature of As includes functions $\bullet, \cdot, \mathcal{L}^t(\cdot)$ and $\mathcal{L}^a(\cdot)$ with the obvious interpretation. Since we need to relate two ts's with each other, sub and superscripts are used in As to keep them apart: i.e., $\exists V_0 \forall e_0 \in \bullet V_0 \mathcal{L}^{0^a}(e_0) \in Act(\Sigma)$ expresses the existence of a set vertices in π^0 such that the outgoing edges of any vertex in this set is labelled with a local action. Any quantification over vertices, edges, and vertex sets is implicitly restricted by the ts to which these items belong. Since, these ts's are finite, such quantifications can always be replaced by finite con- and disjunctions.

Assume that assertions all use the history variable, h . Its value is neither changed by an action in $Act(\Sigma)$ nor does an action depend on the value; also, h does not appear in any test in the label of any transition in any ta : $\forall a \in Act(\Sigma), s \in \mathcal{His} \ (\sigma, \bar{\sigma}) \in \llbracket a \rrbracket^A \Rightarrow \sigma(h) = \bar{\sigma}(h) \ \& \ (\sigma\{s/h\}, \bar{\sigma}\{s/h\}) \in \llbracket a \rrbracket^A$ and $\forall \phi \in \mathcal{L}^t(\mathcal{E}) \ h \notin FV(\phi)$.

6-1 DEFINITION. Let $\pi, \pi^0, \pi^1 \in TS$.

- A set labelling of π is a (partial) function $\lambda: 2^{\mathcal{V}} \mapsto As$; by convention, let $\sigma \not\models \lambda(V)$ for every $\sigma \in \mathcal{S}$ if $V \notin dom(\lambda)$,
- Given set labellings λ^0 and λ^1 of π^0 and π^1 , the relation $[\lambda^0, \lambda^1] \subseteq 2^{\mathcal{V}^0} \times 2^{\mathcal{V}^1}$ is defined by

$$\begin{aligned} [\lambda^0, \lambda^1] &= \bigcup_{\sigma \in \mathcal{S}} [\lambda^0, \lambda^1]_{\sigma} \text{ and} \\ [\lambda^0, \lambda^1]_{\sigma} &= \{(V_0, V_1) \mid \exists s \in \mathcal{His} \ \sigma\{s/h\} \models \lambda^0(V_0) \wedge \lambda^1(V_1)\}, \end{aligned}$$

- Given a relation $[\lambda^0, \lambda^1]$, define its translation by

$${}^t[\lambda^0, \lambda^1] = \bigcup_{\sigma \in \mathcal{S}} \{(V_0 \times \{\sigma\}, V_1 \times \{\sigma\}) \mid (V_0, V_1) \in [\lambda^0, \lambda^1]_{\sigma}\}.$$

The intention is that λ^0 and λ^1 describe the states and histories with which the vertex sets in their domains can be reached.

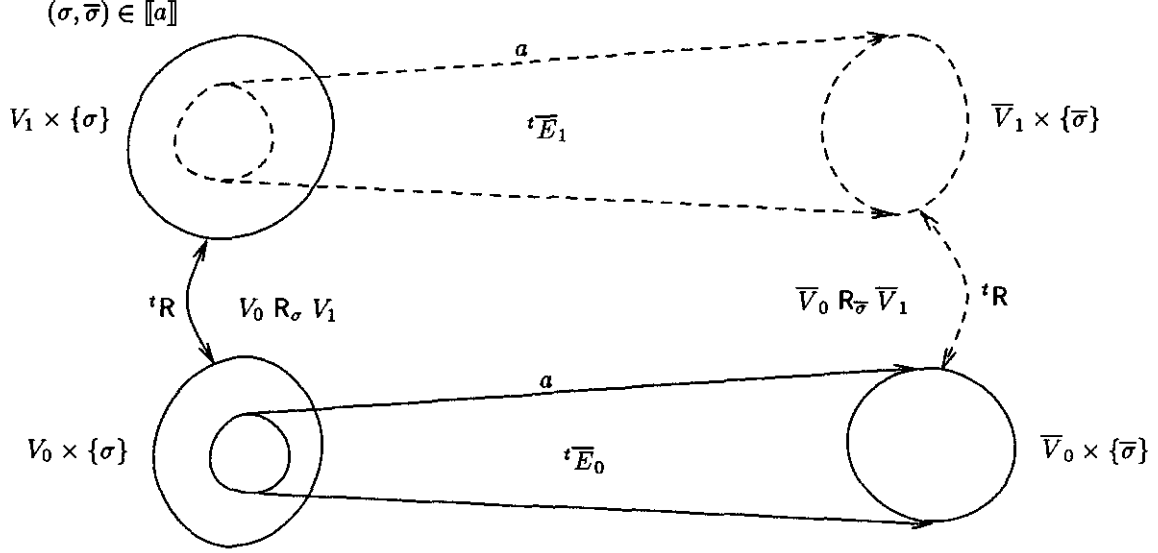
Now we ask: *what conditions should be imposed on $[\lambda^0, \lambda^1]$ so that ${}^t[\lambda^0, \lambda^1]$ is a failure simulation from ${}^t\pi^0$ to ${}^t\pi^1$?* Note that the particular representation influences such conditions and other choices are possible. This is discussed at the end of the section.

In tackling this, we first concentrate on clause (2) of Definition 4-2, formulated below for ${}^t\pi^0$ and ${}^t\pi^1$ where R stands for $[\lambda^0, \lambda^1]$:

$$\bullet {}^tE_0 \ {}^tR \ {}^tV_1, \emptyset \neq {}^t\bar{E}_0 \subseteq {}^tE_0 \cap \mathcal{L}^{0^{-1}}(a), \ {}^t\bar{E}_0 \in dom({}^tR) \implies \exists {}^t\bar{E}_1 \subseteq \bullet {}^tV_1 \ {}^t\bar{E}_0 \ {}^tR \ {}^t\bar{E}_1 \ \& \quad (8) \\ {}^t\mathcal{L}^0({}^t\bar{E}_0) = {}^t\mathcal{L}^1({}^t\bar{E}_1),$$

Let ${}^*E_0 = {}^*V_0$ ⁷. Now, ${}^tV_0 \text{ } {}^tR \text{ } {}^tV_1$ implies there are $V_0 \subseteq \mathcal{V}^0$, $V_1 \subseteq \mathcal{V}^1$ and a $\sigma \in \mathcal{S}$ such that ${}^tV_i = V_i \times \{\sigma\}$ and $V_0 R_\sigma V_1$. The action, a , is of the form $(\sigma, \bar{a}, \bar{\sigma})$ for some $\bar{a} \in \text{Act}(\Sigma)$ and $\bar{\sigma} \in \mathcal{S}$. By assumption, there is a set of edges ${}^tE_0 \subseteq {}^tE_0$, all labelled with a . So, ${}^tE_0^* = \bar{V}_0 \times \{\bar{\sigma}\}$ for some $\bar{V}_0 \in \text{dom}(R_{\bar{\sigma}})$.

We need to find a set of edges tE_1 , also labelled with a and such that ${}^tE_1 \subseteq {}^*V_1$ and ${}^tE_0^* \text{ } {}^tR \text{ } {}^tE_1^*$. If we have such a set, there is a $\bar{V}_1 \in \text{ran}(R_{\bar{\sigma}})$ such that ${}^tE_1^* = \bar{V}_1 \times \{\bar{\sigma}\}$. The following figure sketches the situation:



Again, the dotted parts have to be constructed.

Now, $V_0 \in \text{dom}(R_\sigma)$ means that $\sigma\{s/h\} \models \lambda^0(V_0)$ for some $s \in \mathcal{H}is$. Likewise, $\bar{V}_0 \in \text{dom}(R_{\bar{\sigma}})$ is equivalent with $\bar{\sigma}\{\bar{s}/h\} \models \lambda^0(\bar{V}_0)$ for an $\bar{s} \in \mathcal{H}is$. One would expect $\bar{s} = sa$ but this does not follow from the definition of R . This expectation is based on the implicit assumption that if some set of vertices V_0 is reachable by some history s and state σ , then actually $\sigma\{s/h\} \models \lambda^0(V_0)$ should hold. Not unreasonable and we define

6-2 DEFINITION (safe labelling). Given $\pi \in \text{TS}$ and a set labelling λ if π .

- The labelling λ is safe for π if for any $V \in \text{dom}(\lambda)$, $s \in \mathcal{H}is$, $\sigma \in \mathcal{S}$:

$$V \times \{\sigma\} \subseteq {}^t\pi \text{ after } s \implies \sigma\{s/h\} \models \lambda(V).$$

So, let us assume now that

$$\lambda^0 \text{ is a safe set labelling for } \pi^0. \quad (9)$$

Also, assume w.l.o.g. that $V_0 \times \{\sigma\} \subseteq {}^t\pi^0$ after s . Now we have $\bar{V}_0 \times \{\bar{\sigma}\} \subseteq {}^t\pi^0$ after sa and, hence, $\bar{\sigma}\{sa/h\} \models \lambda^0(\bar{V}_0)$. Next, consider the tV_1 that we need to construct. We must have $\bar{V}_0 R_{\bar{\sigma}} \bar{V}_1$ and, therefore, $\bar{\sigma}\{sa/h\} \models \lambda^1(\bar{V}_1)$. Well, not exactly: we know that there has to be a common computation with which both tV_0 and tV_1 are reachable but we do not know that sa is the one. Still, this too is a reasonable assumption and we demand that

$$A \models \lambda^0(V_0) \rightarrow \exists V_1 \lambda^1(V_1). \quad (10)$$

⁷Note that ${}^*E_0 = {}^*V_0$

Now, we can take a \bar{V}_1 such that $\bar{\sigma}\{sa/h\} \models \lambda^1(\bar{V}_1)$. If we can show that ${}^t\bar{V}_1^* \subseteq {}^*\bar{V}_1 \cap \mathcal{L}^{1-1}(a)$ we are done. Why should this be? Well, basically from the idea that since $\bar{V}_1 \times \{\bar{\sigma}\}$ is reachable in ${}^t\pi^1$ via a computation sa , it has to pass through $V_1 \times \{\sigma\}$ since that set is reachable via s . Again, two implicit assumptions have been made here: the first is that because $\bar{\sigma}\{sa/h\} \models \lambda^1(\bar{V}_1)$, these vertexes are in fact reachable via sa ; the second assumption is that therefore the computations have to pass through $V_1 \times \{\sigma\}$.

Let us define

6-3 DEFINITION (sure and history determined (hd) labelling). *Given $\pi \in \text{TS}$ and a set labelling λ if π .*

- The labelling λ is sure for π if for any $V \in \text{dom}(\lambda)$, $s \in \mathcal{H}is$, $\sigma \in \mathcal{S}$:

$$\sigma\{s/h\} \models \lambda(V) \quad \Rightarrow \quad V \times \{\sigma\} \subseteq {}^t\pi \text{ after } s ,$$

- The labelling λ is history determined (hd) if for any $\bar{V} \in \text{dom}(\lambda)$ and $sa \in \mathcal{H}is$ with $a = (\sigma, \bar{a}, \bar{\sigma})$:

$$\begin{aligned} \bar{\sigma}\{sa/h\} \models \lambda(\bar{V}) \quad \Rightarrow \quad & \text{there is a } V \in \text{dom}(\lambda) \text{ such that } \sigma\{s/h\} \models \lambda(V) \text{ and} \\ & \text{for all such } V : \bar{V} \times \{\bar{\sigma}\} \subseteq ({}^*(V \times \{\sigma\}) \cap \mathcal{L}^{a-1}(a))^* \end{aligned}$$

We demand that

$$\lambda^1 \text{ is a sure set labelling for } \pi^1 \text{ and is hd .} \quad (11)$$

Then, because λ^1 is sure, $\bar{\sigma}\{sa/h\} \models \lambda^1(\bar{V}_1)$ implies that $\bar{V}_1 \times \{\bar{\sigma}\} \subseteq {}^t\pi^1$ after sa . We also have, by assumption, that $\sigma\{s/h\} \models \lambda^1(V_1)$. Hence, history determinedness of λ^1 gives that ${}^t\bar{V}_1^* \subseteq {}^*\bar{V}_1$.

We have shown that if λ^0 and λ^1 satisfy conditions (9), (10) and (11), then ${}^t[\lambda^0, \lambda^1]$ satisfies (8). This derivation clearly show the different roles played by the set labellings of π^0 and of π^1 . As π^0 is the implementation, we must make sure that every computation of it is also a computation of π^1 . If computations are characterized using assertions, then it is essential that such assertions hold along every computation of π^0 . Otherwise, not every computation of π^0 is mapped onto one of π^1 . So, such assertions may ‘err’ on the safe side by being satisfied along computations that are not generated by π^0 . Of course we may not be able to prove refinement because of this, but that is a different issue. By the same token, assertions characterizing computations in π^1 , the system that is to be implemented, must be sure in the sense that they should never hold on computations that do not occur in π^1 . Otherwise, some computation of π^0 might be mapped onto a computation that does not occur in π^1 . Hence, these assertions may ‘err’ on the sure side by being invalid along certain computations of π^1 .

The other conditions of failure simulation are easier to satisfy. Clause (1) reads

$$\exists ({}^tV_i^0)_{i < \alpha} \subseteq \text{dom}({}^tR) \quad {}^tI^0 = \bigcup_{i < \alpha} {}^tV_i^0 \ \& \ \forall i < \alpha \exists {}^tV^1 \subseteq {}^tI^1 \quad {}^tV_i^0 \ {}^tR \quad {}^tV^1 . \quad (12)$$

We have ${}^tI^0 = I^0 \times \mathcal{S}$ and it seems natural to find a covering V_0^0, \dots, V_n^0 of I^0 and to have $V_i^0 \times \{\sigma\} \in \text{dom}({}^tR)$ for any $\sigma \in \mathcal{S}$ and $i \leq n$ (; remember that $\pi^0 \in \text{TS}$ so that I^0 is finite). Similarly for the V_i^1 . So, let us demand that

$$\begin{aligned} \exists V_0^0 \dots V_n^0 \in \text{dom}(\lambda^0) \quad I^0 = V_0^0 \cup \dots \cup V_n^0 \ \& \ \exists V^1 \in \text{dom}(\lambda^1) \quad V^1 \subseteq I^1 \ \& \\ \forall V^1 \in \text{dom}(\lambda^1) \quad V^1 \subseteq I^1 \ \Rightarrow \ \mathbf{A} \models h = \langle \rangle \ \rightarrow \ \lambda^1(V^1) \end{aligned} , \quad (13)$$

where $\langle \rangle$ stands for the empty sequence; i.e., $\sigma\{\varepsilon/h\} \models h = \langle \rangle$.

Choose a $\sigma \in \mathcal{S}$ and consider $V_i^0 \times \{\sigma\}$ for any $i \leq n$. As $V_i^0 \subseteq I^0$, we have that $V_i^0 \times \{\sigma\} \subseteq {}^t\pi^0$ after ε and so by safeness of λ^0 that $\sigma\{\varepsilon/h\} \models \lambda^0(V_i^0)$. There is a $V^1 \in \text{dom}(\lambda^1)$ such that $V^1 \subseteq I^1$ and (hence) $\sigma\{\varepsilon/h\} \models \lambda^1(V^1)$. This means that $V_i^0 \in \text{dom}(R_\sigma)$ and that $V_i^0 \ R_\sigma \ V^1$. Hence,

$$\begin{aligned} {}^tI^0 = \bigcup \{V_i^0 \times \{\sigma\} \mid i \leq n, \sigma \in \mathcal{S}\} \ \text{and} \\ \forall i \leq n, \sigma \in \mathcal{S}, \exists V^1 \in \text{dom}(\lambda^1) \quad V^1 \times \{\sigma\} \subseteq {}^tI^1 \ \& \ V_i^0 \times \{\sigma\} \ {}^tR \ V^1 \times \{\sigma\} , \end{aligned}$$

so that (12) holds.

Next, clause (3):

$${}^tV_0 \text{ } {}^tR \text{ } {}^tV_1 \implies \forall {}^tv_0 \in {}^tV_0 \exists {}^tv_1 \in {}^tV_1 \forall \rho \in \{\uparrow, \downarrow, \delta\} \rho({}^tv_0) \Rightarrow \rho({}^tv_1) \ \& \ \exists {}^tv_1 \in {}^tV_1 \text{ stable}({}^tv_0) \Rightarrow (\text{stable}({}^tv_1) \ \& \ {}^t\mathcal{L}_g^1({}^*{}^tv_1) \subseteq {}^t\mathcal{L}_g^0({}^*{}^tv_0) \ \& \ \downarrow({}^tv_1) \Rightarrow \downarrow({}^tv_0)) .$$

We want to check $\uparrow(\cdot)$, etc. on the level of the first order systems. So we define them as auxiliary predicates in As:

6-4 DEFINITION. Let $\pi \in \text{TS}$, $v \in \mathcal{V}$ and $\sigma \in \mathcal{S}$.

- $\sigma \models \uparrow(v)$ iff $\sigma \models \exists e \in {}^*v \ \mathcal{L}^t(e) \wedge \mathcal{L}^a(e) = \uparrow$,
- $\sigma \models \downarrow(v)$ iff $v \in T$,
- $\sigma \models \delta(v)$ iff $\sigma \models \forall e \in {}^*v \ \mathcal{L}^t(e) \rightarrow \mathcal{L}^a(e) = \delta \wedge v \notin T$,
- $\sigma \models \text{stable}(v)$ iff $\sigma \models \forall e \in {}^*v \ \mathcal{L}^t(e) \rightarrow \mathcal{L}^a(e) \in \mathcal{Gact}(\Sigma)_\delta$,
- $\sigma \models \text{enabled}(v) = E$ iff $\sigma \models \forall e \in \mathcal{E} \ e \in E \leftrightarrow ({}^*e = v \wedge \mathcal{L}^t(e))$.

There is a straightforward correspondence between these predicates in As and their informal counterparts for ${}^t\pi$. For $\pi \in \text{TS}$ and $v \in \mathcal{V}$ we have ' $\sigma \models \rho(v)$ iff $\rho((v, \sigma))$ (in ${}^t\pi$)' for $\rho \in \{\downarrow, \uparrow, \delta\}$ and ' $\sigma \models \text{stable}(v)$ iff $\text{stable}((v, \sigma))$ (in ${}^t\pi$)'. For the function $\text{enabled}(\cdot)$ we have that ' $\sigma \models \text{enabled}(v) = E$ iff ${}^*(v, \sigma) \subseteq \{\sigma\} \times E \times \mathcal{S}$ '. This, too, follows directly from the construction of ${}^t\pi$.

Hence, we demand that the following holds:

$$\mathbf{A} \models \lambda^0(V_0) \wedge \lambda^1(V_1) \rightarrow \forall v_0 \in V_0 (\exists v_1 \in V_1 \forall \rho \in \{\uparrow, \downarrow, \delta\} \rho(v_0) \Rightarrow \rho(v_1) \wedge \exists v_1 \in V_1 \text{ stable}(v_0) \rightarrow (\text{stable}(v_1) \wedge \mathcal{L}_g^1(\text{enabled}(v_1)) \subseteq \mathcal{L}_g^0(\text{enabled}(v_0)) \wedge \downarrow(v_1) \rightarrow \downarrow(v_0))) .$$

The only moot point concerns the test $\mathcal{L}_g^1(\text{enabled}(v_1)) \subseteq \mathcal{L}_g^0(\text{enabled}(v_0))$. Choose some σ with $\sigma \models \lambda^0(V_0) \wedge \lambda^1(V_1)$; take some $v_0 \in V_0$ and let v_1 be the picked vertex in V_1 ; let $\sigma \models \text{stable}(v_0)$. We must prove that ${}^t\mathcal{L}_g^1({}^*(v_1, \sigma)) \subseteq {}^t\mathcal{L}_g^0({}^*(v_0, \sigma))$ holds. Let $\sigma \models \text{enabled}(v_i) = E_i$ ($i = 0, 1$). Then, by the construction of ${}^t\pi$:

$${}^*(v_i, \sigma) = \{(\sigma, e_i, \bar{\sigma}) \mid e_i \in E_i, (\sigma, \bar{\sigma}) \in \llbracket \mathcal{L}^a(e_i) \rrbracket^{\mathbf{A}}\} .$$

Now, take an edge ${}^te_1 = (\sigma, e_1, \bar{\sigma}) \in {}^*(v_1, \sigma)$ with $\mathcal{L}^1 a(e_1) = a$. Then, ${}^t\mathcal{L}^1({}^te_1) = (\sigma, a, \bar{\sigma})$ and $(\sigma, \bar{\sigma}) \in \llbracket [a] \rrbracket^{\mathbf{A}}$. As $e_1 \in E_1$, there is an edge $e_0 \in E_0$ with $\mathcal{L}^0 a(e_0) = a$, whence ${}^te_0 = (\sigma, e_0, \bar{\sigma}) \in {}^*(v_0, \sigma)$ and ${}^t\mathcal{L}^0({}^te_0) = (\sigma, a, \bar{\sigma})$.

Finally, clause (4):

$$\exists {}^tE_0 \in \text{dom}({}^tR) \implies \exists {}^tV_0 \dots {}^tV_n \in \text{dom}({}^tR) ({}^tE_0 \cap {}^t\mathcal{L}^{0^{-1}}(a))^* = {}^tV_0 \cup \dots \cup {}^tV_n . \quad (14)$$

We define

6-5 DEFINITION (cover). Let λ be a set labeling for π .

- λ covers π if for any $V \in \text{dom}(\lambda)$, $\bar{a} \in \text{Act}(\Sigma)$ and $\sigma \in \mathcal{S}$:

$$\sigma \models \lambda(V) \implies \exists V_0, \dots, V_n \in \text{dom}(\lambda) \forall \bar{\sigma} \in \llbracket [\bar{a}] \rrbracket^{\mathbf{A}}(\sigma) \ \bar{\sigma}\{s_{\bar{\sigma}}/h\} \models \bigwedge_{i \leq n} \lambda(V_i) \ \& \ \sigma \models v \in \bigcup_{i \leq n} V_i \leftrightarrow \exists e \in {}^*V \cap v^* \ \mathcal{L}^t(e) \wedge \mathcal{L}^a(e) = a ,$$

where $s_{\bar{\sigma}} = \sigma(h) \sim (\sigma, \bar{a}, \bar{\sigma})$.

Assume that

$$\lambda^0 \text{ covers } \pi^0 .$$

Take some ${}^*tE_0 = {}^*V_0 \in \text{dom}({}^*tR)$; let ${}^*t\bar{V}_0 = ({}^*tE_0 \cap {}^*t\mathcal{L}^{0^{-1}}(a))^*$, ${}^*tV_0 = V_0 \times \{\sigma\}$ and ${}^*t\bar{V}_0 = \bar{V}_0 \times \{\bar{\sigma}\}$. Hence, $a = (\sigma, \bar{a}, \bar{\sigma})$ for some $\bar{a} \in \text{Act}(\Sigma)$ and $(\sigma, \bar{\sigma}) \in \llbracket \bar{a} \rrbracket^{\mathbf{A}}$. Then we have $\sigma \models \lambda^0(V_0)$. Because λ^0 covers π^0 , there are $V_0^0, \dots, V_n^0 \in \text{dom}(\lambda^0)$ such that

$$\begin{aligned} \bar{\sigma}\{\sigma(h)a/h\} &\models \bigwedge_{i \leq n} \lambda^0(V_i^0) \text{ and} \\ \sigma \models v \in \bigcup_{i \leq n} V_i^0 &\leftrightarrow \exists e_0 \in E_0 \cap v^* \mathcal{L}^{0^{-1}}(e_0) \wedge \mathcal{L}^{0^a}(e_0) = \bar{a} . \end{aligned}$$

Let ${}^*tV_i^0 = V_i^0 \times \{\bar{\sigma}\}$ ($i \leq n$). Then ${}^*tV_0^0 \dots {}^*tV_n^0 \in \text{dom}({}^*tR)$ since $V_0^0 \dots V_n^0 \in \text{dom}(R_{\bar{\sigma}})$. Finally, $({}^*tE_0 \cap {}^*t\mathcal{L}^{0^{-1}}(a))^* = \{(e_0^*, \bar{\sigma}) \mid e_0 \in E_0, \sigma \models \mathcal{L}^{0^{-1}}(e_0), \mathcal{L}^{0^a}(e_0) = \bar{a}\}$ and (14) follows.

We have derived a *proof rule* to establish refinement of first order transition systems. Note that we have split up clause (13) among the premisses of the rule:

FO – REF

Given $\pi^0, \pi^1 \in \text{TS}$ with set labellings λ^0 and λ^1 :

1. λ^0 is safe for π^0 and covers π^0 , $\exists V_0^0 \dots V_n^0 \in \text{dom}(\lambda^0) I^0 = V_0^0 \cup \dots \cup V_n^0$,
 2. λ^1 is sure for π^1 and is hd, $\exists V^1 \in \text{dom}(\lambda^1) V^1 \subseteq I^1$,
 $\forall V^1 \in \text{dom}(\lambda^1) V^1 \subseteq I^1 \Rightarrow \mathbf{A} \models h = \langle \rangle \rightarrow \lambda^1(V^1)$,
 3. $\mathbf{A} \models \lambda^0(V_0) \rightarrow \exists V_1 \lambda^1(V_1)$,
 4. $\mathbf{A} \models \lambda^0(V_0) \wedge \lambda^1(V_1) \rightarrow \forall v_0 \in V_0 (\exists v_1 \in V_1 \forall \rho \in \{\uparrow, \downarrow, \delta\} \rho(v_0) \Rightarrow \rho(v_1) \wedge$
 $\exists v_1 \in V_1 \text{ stable}(v_0) \rightarrow (\text{stable}(v_1) \wedge \mathcal{L}_g^1(\text{enabled}(v_1)) \subseteq \mathcal{L}_g^0(\text{enabled}(v_0)) \wedge \downarrow(v_1) \rightarrow \downarrow(v_0)))$
-
- $$\mathbf{A} \models \pi^0 \sqsupseteq \pi^1$$

Note that removing the last premiss gives a rule for proving trace inclusion. Write $\vdash_{\text{FO-REF}} \pi^0 \sqsupseteq \pi^1$ if $\mathbf{A} \models \pi^0 \sqsupseteq \pi^1$ can be derived using the rule.

The derivation of the rule establishes soundness. Completeness depends—as usual—on the expressiveness of the assertion language. Specifically, we make the following assumption:

For any $\pi \in \text{TS}$ and $V \subseteq \mathcal{V}$, there is a formula $\text{COMP}(\pi, V) \in \text{As}$, such that for any $\sigma \in \mathcal{S}$ and $s \in \text{His}$

$$\sigma\{s/h\} \models \text{COMP}(\pi, V) \iff V \times \{\sigma\} = {}^*t\pi \text{ after } s .$$

Note that π and V are *not* parameters of the formula. I.e., we do not need a formula *uniformly* in π and V . Existence of such formulae means that the semantics of the actions should be definable in As and that it must be possible to describe the i th record (in $\mathcal{S} \times \text{Act}(\Sigma) \times \mathcal{S}$) of a sequence for any i .

Now, define set labellings for π^0 and π^1 as follows:

$$\begin{aligned} \lambda^0 &= \left[\{v_0\} \mapsto \bigvee \{ \text{COMP}(\pi^0, V_0) \mid v_0 \in V_0 \subseteq \mathcal{V}^0 \}, v_0 \in I^0 \right] , \\ \lambda^1 &= \left[V_1 \mapsto \text{COMP}(\pi^1, V_1), V_1 \subseteq \mathcal{V}^1 \right] . \end{aligned}$$

Observe that

$$[\lambda^0, \lambda^1]_{\sigma} = \left\{ (\{v_0, V_1\}) \mid \exists s \in \text{His } {}^*t\pi^0 \xrightarrow{s} (v_0, \sigma); {}^*t\pi^0, {}^*t\pi^1 \text{ after } s = V_1 \times \{\sigma\} \right\} .$$

Hence, $[\lambda^0, \lambda^1]$ is precisely the failure simulation (from ${}^t\pi^0$ to ${}^t\pi^1$) used in the proof of the completeness theorem 4-6.

We show that λ^0 and λ^1 satisfy the premisses of FO – REF, in case $\pi^0 \sqsupseteq \pi^1$ holds.

By definition, λ^0 and λ^1 are both safe and sure.

To establish that λ^0 covers π^0 , take some $v_0 \in \mathcal{V}^0$, $a \in \mathcal{Act}(\Sigma)$ and $\sigma \in \mathcal{S}$ with $\sigma \models \lambda^0(v_0)$. Define

$$\{v_0^0, \dots, v_n^0\} = \{e_0^* \mid e_0 \in {}^*v_0, \sigma \models \mathcal{L}^0({}^t e_0) \wedge \mathcal{L}^0 a(e_0) = a\}$$

and let $\bar{\sigma} \in \llbracket \bar{a} \rrbracket^{\mathbf{A}}(\sigma)$. By definition of λ^0 we have $\bar{\sigma}\{s/h\} \models \bigwedge_{i \leq n} \lambda^0(v_i^0)$ where $s = \sigma(h)(\sigma, \bar{a}, \bar{\sigma})$. The second part of the definition of covering, (6-5), immediately follows from the definition of $\{v_0^0, \dots, v_n^0\}$.

For history determinedness of λ^1 , take some $\bar{V}_1 \in \text{dom}(\lambda^1)$ such that $\bar{\sigma}\{sa/h\} \models \lambda^1(\bar{V}_1)$ with $s \in \mathcal{His}$ and $a = (\sigma, \bar{a}, \bar{\sigma})$. By definition of λ^1 , this means that $\bar{V}_1 \times \{\bar{\sigma}\} = {}^t\pi^1$ after sa . Let $V_1 \times \{\sigma\} = {}^t\pi^1$ after s . Then $\sigma\{s/h\} \models \lambda^1(V_1)$ holds and V_1 is the unique set with that property. Moreover, $({}^*(V_1 \times \{\sigma\}) \cap \mathcal{L}^{a^{-1}}(a))^* = \bar{V}_1 \times \{\sigma\}$

If $I^1 = \{v_0^0, \dots, v_n^0\}$, then $\{v_i^0\} \in \text{dom}(\lambda^0)$ for $i \leq n$. Also, $I^1 \in \text{dom}(\lambda^1)$ and $\mathbf{A} \models h = \langle \rangle \rightarrow \lambda^1(I^1)$ by definition of λ^1 . Finally, note that $V_1 \in \text{dom}(\lambda^1)$ and $V_1 \subseteq I^1$ implies that $V_1 = I^1$.

To show the third premiss, take some $\sigma\{s/h\} \models \lambda^0(v_0)$. This means that $V_0 \times \{\sigma\} \subseteq {}^t\pi^0$ after s and, since $\diamond^t \pi^0 \diamond \subseteq \diamond^t \pi^1 \diamond$, that $\sigma\{s/h\} \models \lambda^1(V_1)$ with ${}^t\pi^1$ after $s = V_1 \times \{\sigma\}$.

Finally, the fourth premiss. Take a $\sigma \in \mathcal{S}$, $s \in \mathcal{His}$ and assume that $\sigma\{s/h\} \models \lambda^0(v_0) \wedge \lambda^1(V_1)$ for some $v_0 \in \mathcal{V}^0$ and $V_1 \subseteq \mathcal{V}^1$. Hence, $(v_0, \sigma) \in {}^t\pi^0$ after s and $V_1 \times \{\sigma\} \in {}^t\pi^1$ after s . As ${}^t\pi^0 \sqsupseteq {}^t\pi^1$ there is a $v_1 \in V_1$ such that $\rho((v_0, \sigma)) \Rightarrow \rho((v_1, \sigma))$ for $\rho \in \{\downarrow, \uparrow, \delta\}$ and such that $\text{stable}((v_0, \sigma)) \Rightarrow (\text{stable}((v_1, \sigma)) \ \& \ \mathcal{L}_g^1({}^*(v_1, \sigma)) \subseteq \mathcal{L}_g^0({}^*(v_0, \sigma)) \ \& \ \downarrow(v_1, \sigma) \Rightarrow \downarrow(v_0, \sigma))$. This immediately implies that

$$\sigma \models \text{stable}(v_0) \Rightarrow (\text{stable}(v_1) \ \& \ \mathcal{L}_g^1({}^*v_1) \subseteq \mathcal{L}_g^0({}^*v_0) \ \& \ \downarrow(v_1) \Rightarrow \downarrow(v_0)) .$$

We have obtained the following

6-6 THEOREM (Soundness and completeness of FO – REF). *Let $\pi^0, \pi^1 \in \text{TS}$, then*

$$\mathbf{A} \models \pi^0 \sqsupseteq \pi^1 \quad \text{iff} \quad \vdash_{\text{FO-REF}} \pi^0 \sqsupseteq \pi^1 .$$

We get a sound and complete rule for trace refinement of 1st order systems if the fourth premiss of FO-REF is dropped.

It remains to show how to prove safeness, sureness, covering and history determinedness of labellings. The next two subsections address, respectively, verifying safeness+covering and sureness+history determinedness of labellings. The intention is to show how the proof of these properties can be reduced to verifying properties of (sets of) transitions. As such, these proof principles are formulated on a level analogous to that of Floyd's inductive assertion method and Manna and Pnueli's temporal logic proof rules [MP81, MP84].

6.1 Proving safeness and covering

Proving safeness is a straightforward generalization of proving so-called *local correctness* of a (Floyd) labelling in an inductive assertion proof. A Floyd labelling, ϕ , associates assertions, $\phi(v)$ to the vertices, v , of a transition system $\pi \in \text{TS}$. Local correctness of a labelling entails (in our notation) that $(v, \sigma) \in {}^t\pi$ after $s \Rightarrow \sigma \models \phi(v)$ for all $s \in \mathcal{His}$ and $\sigma \in \mathcal{S}$. It is proved by showing that

$$1. v \in I \Rightarrow \mathbf{A} \models \phi(v) \quad \& \quad 2. \forall e \in \mathcal{E} \ \sigma \models \phi(e^*) \wedge \mathcal{L}^t(e) \ \& \ \bar{\sigma} \in \llbracket \mathcal{L}^a(e) \rrbracket^{\mathbf{A}}(\sigma) \Rightarrow \bar{\sigma} \models \phi(e^*) .$$

Property (2) is also called “ a leads from $\phi(e^*)$ to $\phi(e^*)$ ” and is written $\phi(e^*) \xrightarrow{a} \phi(e^*)$ (if $a = \mathcal{L}(e)$).

Our case is more complicated because we deal with set labellings and we have history variables; however, the principle is the same. By convention, bold face letters, \mathbf{V}, \dots will stand for sets of sets of vertices.

6-7 DEFINITION (leads to). Let $\pi \in \text{TS}$ and let λ be a set labelling for π . Take some $V \in \text{dom}(\lambda)$ and $V_0, \dots, V_n \subseteq \text{dom}(\lambda)$. Assume that $\emptyset \neq E = {}^*V \cap \mathcal{L}^a{}^{-1}(a)$ and $E^* = \bigcup_{i \leq n} V_i$.

- $V \xrightarrow{a} V_0, \dots, V_n$ (a leads from V to V_0, \dots, V_n) if for all $(\sigma, \bar{\sigma}) \in \llbracket a \rrbracket^{\mathbf{A}}$, $s \in \mathcal{H}$ is:

$$\begin{aligned} \sigma\{s/h\} \models \lambda(V) \wedge \bigvee \{\mathcal{L}^t(e) \mid e \in E\} &\implies \\ \exists i \leq n \quad \sigma\{s/h\} \models v \in \bigcup V_i \leftrightarrow \exists e \in E \quad v = e^* \wedge \mathcal{L}^t(e) \ \& \ \bar{\sigma}\{s(\sigma, a, \bar{\sigma})/h\} \models \bigwedge \{\lambda(\bar{V}) \mid \bar{V} \in V_i\}. \end{aligned}$$

As a convention, $V \xrightarrow{a} V_0, \dots, V_n$ holds vacuously if either $V \notin \text{dom}(\lambda)$, or $V_i \not\subseteq \text{dom}(\lambda)$ for some $i \leq n$, or $E = \emptyset$, or $E^* \neq \bigcup_{i \leq n} V_i$.

We have the following proof principle:

<p>SACO</p> <p>Given $\pi \in \text{TS}$ and a set labelling λ for it:</p> $\forall V \in \text{dom}(\lambda) \quad V \subseteq I \Rightarrow \mathbf{A} \models h = \langle \rangle \rightarrow \lambda(V),$ $\forall a \in \text{Act}(\Sigma), \quad V \in \text{dom}(\lambda) \quad \exists V_0, \dots, V_n \subseteq \text{dom}(\lambda) \quad (V \cap \mathcal{L}^a{}^{-1}(a))^* = \bigcup_{i \leq n} V_i \ \& \ \forall V_0, \dots, V_n \subseteq V \quad V \xrightarrow{a} V_0, \dots, V_n$ <hr style="width: 80%; margin: 10px auto;"/> <p style="text-align: center;">λ is safe for π and covers π</p>

The proof rule needs one assumption concerning the vertex sets with which assertions are associated in order to be sound:

$$\forall a \in \text{Act}(\Sigma) \quad \forall \bar{V} \in \text{dom}(\lambda) : \quad \forall \bar{v} \in \bar{V} \quad a \in \mathcal{L}^a(\bar{v}) \Rightarrow \exists V \in \text{dom}(\lambda) \quad ({}^*V \cap \mathcal{L}^a{}^{-1}(a))^* = \bar{V}. \quad (15)$$

So, if some set of vertices in $\text{dom}(\lambda)$ is reachable by a -transitions, then there must be another set in $\text{dom}(\lambda)$ from which every vertex in the former set is reachable by an a -transition.

Now we can prove

6-8 THEOREM. SACO is complete. Also, if $\exists V_0, \dots, V_n \in \text{dom}(\lambda) \quad I = V_0 \cup \dots \cup V_n$ and λ satisfies (15), then SACO is sound.

PROOF.

Completeness Take some $V \in \text{dom}(\lambda)$ and $a \in \text{Act}(\Sigma)$. If $V \subseteq I$ then safeness gives $\mathbf{A} \models h = \langle \rangle \rightarrow \lambda(V)$ since $V \times \{\sigma\} \subseteq {}^t\pi \text{ after } \varepsilon$ holds for every $\sigma \in \mathcal{S}$. Next, let $E = {}^*V \cap \mathcal{L}^a{}^{-1}(a)$. Because λ covers π , there are $V_0, \dots, V_m \in \text{dom}(\lambda)$ for every $\sigma \in \mathcal{S}$ and $v \in \bigcup_{i \leq m} V_i$ iff for some $e \in {}^*v \cap E$ we have $\sigma \models \mathcal{L}^t(e)$. Since, π is finite, there are only a finite number of such coverings. Let V_0, \dots, V_n be these coverings. Clearly, $E^* = \bigcup_{i \leq n} V_i$ and $V_i \subseteq \text{dom}(\lambda)$ for $i \leq n$. Now, take some $(\sigma, \bar{\sigma}) \in \llbracket a \rrbracket$ with $\sigma\{s/h\} \models \lambda(V) \wedge \bigvee \{\mathcal{L}^t(e) \mid e \in E\}$. By construction there is an $i \leq n$ such that $\sigma \models v \in \bigcup V_i \leftrightarrow \exists e \in {}^*V \cap v^* \mathcal{L}^t(e)$. As $h \notin FV(\pi)$ this formula is also valid in $\sigma\{s/h\}$. By the same token, $\bar{\sigma}\{s(\sigma, a, \bar{\sigma})/h\} \models \bigwedge \{\lambda(\bar{V}) \mid \bar{V} \in V_i\}$. Hence, $V \xrightarrow{a} V_0, \dots, V_n$.

Soundness That λ covers π is clear. Safeness of λ is proven with induction on the length of the computation, s .

$s = \varepsilon$) Since $V \times \{\sigma\} \subseteq {}^t\pi \text{ after } \varepsilon \Rightarrow V \subseteq I$, this case is covered by the first premiss and the condition in the theorem.

$s = \hat{s}a$) Let $a = (\sigma, \bar{a}, \bar{\sigma})$ and let $\bar{V} \times \{\bar{\sigma}\} \subseteq {}^t\pi \text{ after } s$ with $\bar{V} \in \text{dom}(\lambda)$ (and hence $\bar{V} \neq \emptyset$). By (15), there is a $V \in \text{dom}(\lambda)$ such that $({}^*V \cap \mathcal{L}^a{}^{-1}(a))^* = \bar{V}$. By definition, $V \times \{\sigma\} \subseteq {}^t\pi \text{ after } \hat{s}$, whence $\sigma\{\hat{s}/h\} \models \lambda(V)$

by induction. Let $E = \bullet V \cap \mathcal{L}^{a-1}(a)$. Then, the second premiss gives $V_0, \dots, V_n \subseteq \text{dom}(\lambda)$ such that $E^\bullet = \bigcup_{i \leq n} V_i$ and if $\sigma\{\bar{s}/h\} \models \bigvee \{\mathcal{L}^t(e) \mid e \in E\}$, there is an $i \leq n$ such that $\sigma\{\bar{s}/h\} \models v \in \bigcup V_i \leftrightarrow \exists e \in E v = e^\bullet \wedge \mathcal{L}^t(e)$. W.l.o.g. we may assume that $\bar{V} \in V_i$. Hence, we obtain that $\bar{\sigma}\{s/h\} \models \lambda(\bar{V})$. \square

Note that the condition in Theorem 6–8 is one of the premisses of FO – REF. Assumption (15) is satisfied by the set labelling λ^0 used in the completeness proof.

As we know that the existence of a simple failure simulation, too, is necessary for failure refinement, the rule can be simplified (by requiring that the labelling is a Floyd labelling). Thus, covering would not need to be checked. We mention without proof, that for Floyd-type labellings (i.e., labellings whose domain only contain singleton sets) the above proof rule simplifies to

Given $\pi \in \text{TS}$ and a Floyd labelling ϕ for it:

$$\begin{array}{l} \forall v \in I: \mathbf{A} \models h = \langle \rangle \rightarrow \phi(\{v\}), \\ \forall e \in \mathcal{E}, (\sigma, \bar{\sigma}) \in \llbracket \mathcal{L}^a(e) \rrbracket^{\mathbf{A}}, h \in \mathcal{H}is: \sigma\{s/h\} \models \phi(\{e^\bullet\}) \wedge \mathcal{L}^t(e) \Rightarrow \\ \qquad \qquad \qquad \bar{\sigma}\{s(\sigma, \mathcal{L}^a(e), \bar{\sigma})/h\} \models \phi(\{e^\bullet\}) \end{array}$$

ϕ is safe for π and covers π

I.e., one just proves local correctness!

6.2 Proving sureness and history determinedness

The property of sureness is dual to that of safeness. Therefore it should not come as a surprise that the proof rule below is based on a notion “comes from” (the dual of “leads to”).

6–9 DEFINITION (comes from). Let $\pi \in \text{TS}$ and let λ be a set labelling for π . Take some $\bar{V} \in \text{dom}(\lambda)$ and $V \subseteq \text{dom}(\lambda)$. Let $E = \bullet \bar{V} \cap \mathcal{L}^{a-1}(a)$.

- $\bar{V} \overset{a}{\leftarrow} V$ (*a comes from V to \bar{V}*) if either $E = \emptyset$ or $E^\bullet = \bigcup V$ and for all $(\sigma, \bar{\sigma}) \in \llbracket a \rrbracket^{\mathbf{A}}$ and $s \in \mathcal{H}is$

$$\bar{\sigma}\{s(\sigma, \bar{a}, \bar{\sigma})/h\} \models \lambda(\bar{V}) \quad \Rightarrow \quad \exists V \in V \sigma\{s/a\} \models \lambda(V) \ \& \ \forall V \in \text{dom}(\lambda) \\ \sigma\{s/h\} \models \lambda(V) \wedge (\forall e \in \bar{V}^\bullet \rightarrow \exists v \in V e^\bullet = v \wedge \mathcal{L}^t(e)).$$

We have the following proof principle:

SUHD

Let $\pi \in \text{TS}$ and let λ be a set labelling for it:

$$\begin{array}{l} \forall V \in \text{dom}(\lambda): \sigma\{e/h\} \models \lambda(V) \Rightarrow V \subseteq I, \\ \forall a \in \text{Act}(\Sigma), \bar{V} \in \text{dom}(\lambda): \exists V \subseteq \text{dom}(\lambda) \bar{V} \overset{a}{\leftarrow} V \end{array}$$

λ is sure for π and is hd

6–10 THEOREM. SUHD is sound and complete.

PROOF.

Soundness History determinedness is obvious. Sureness is proved by an induction on the length of the computation sequence, s .

$s = \varepsilon$) This case follows from the first premiss.

$s = \widehat{sa}$) Let $a = (\sigma, \bar{a}, \bar{\sigma})$ and take some $\bar{V} \in \text{dom}(\lambda)$. Suppose that $\bar{\sigma}\{s\} \models \lambda(\bar{V})$. History determinedness implies that $\bullet V \cap \mathcal{L}^{a-1}(a) \neq \emptyset$. By the second premiss of the rule, there is a set $\mathbf{V} \subseteq \text{dom}(\lambda)$ and a set $V \in \mathbf{V}$ such that $\sigma\{\widehat{s}/h\} \models \lambda(V)$. By induction this means that $V \times \{\sigma\} \subseteq {}^t\pi$ after \widehat{s} . We also have that $\sigma \models e \in \bar{V}^\bullet \rightarrow \exists v \in V v = \bullet e \wedge \mathcal{L}^t(e)$ because h does not appear in any test in π . This immediately implies that $\bar{V} \times \{\bar{\sigma}\} \subseteq {}^t\pi$ after s .

Completeness The first premiss directly follows from sureness of λ . For the second premiss, take an $a \in \text{Act}(\Sigma)$ and some $\bar{V} \in \text{dom}(\lambda)$ such that $\emptyset \neq \bullet \bar{V} \cap \mathcal{L}^{a-1}(a)$. As λ is history determined, for every $s \in \mathcal{H}is$ and $(\sigma, \bar{\sigma}) \in \llbracket a \rrbracket^{\mathbf{A}}$ such that $\bar{\sigma}\{s(\sigma, a, \bar{\sigma}/h)\} \models \lambda(\bar{V})$ there is a $V \in \text{dom}(\lambda)$ with $\sigma\{s/h\} \models \lambda(V)$ and for any such V : $\bar{V} \times \{\bar{\sigma}\} \subseteq (\bullet(V \times \{\sigma\}) \cap {}^t\mathcal{L}^{-1}((\sigma, a, \bar{\sigma})))^\bullet$. Because \mathcal{V} is finite there are only finitely many such $V \in \text{dom}(\lambda)$. Collect them in the set \mathbf{V} . We claim that $\bar{V} \stackrel{a}{\leftarrow} \mathbf{V}$. So, take some $(\sigma, \bar{\sigma}) \in \llbracket a \rrbracket^{\mathbf{A}}$ and $s \in \mathcal{H}is$ such that $\bar{\sigma}\{s(\sigma, \bar{a}, \bar{\sigma})/h\} \models \lambda(\bar{V})$. By construction there is a $V \in \mathbf{V}$ such that $\sigma\{s/h\} \models \lambda(V)$. Finally, $\bar{V} \times \{\bar{\sigma}\} \subseteq (\bullet(V \times \{\sigma\}) \cap {}^t\mathcal{L}^{-1}((\sigma, a, \bar{\sigma})))^\bullet$ is equivalent with $\sigma\{s/h\} \models \forall e \in \bar{V}^\bullet \rightarrow \exists v \in V \bullet e = v \wedge \mathcal{L}^t(e)$ (; remember that h does not appear free in π). \square

In this section we have lifted failure refinement to 1st order systems by first defining a way to describe a failure simulation in terms of (set) labellings: $[\lambda^0, \lambda^1]$. Then we translated the conditions of failure simulation in Definition 4–2 to that of conditions on the 1st order systems, using the translation, ${}^t\cdot$, defined in Section 5. The resulting rule FO – REF was formulated in terms of a number of primitive notions: safeness, sureness, covers and hd. The last two subsections developed proof principles to prove these properties.

We stress again that the resulting proof rules depend on the particular choice of representing the simulation relation. Here, we decided to use pairs of set labelings and other choices are possible. One is to use labelings of type $\lambda: \mathcal{V}^0 \times \mathcal{V}^1 \mapsto \mathbf{As}$. This would allow a very direct lifting of the conditions, which would become easier to state. On the other hand, our use of set labelings is more modular and in a refinement sequence $\pi^0 \sqsupseteq \pi^1 \sqsupseteq \pi^2$ allows π^1 's labeling to be reused.

7 Conclusions

We have developed a theory of refinement for 1st order programs that is grounded in algebraic process theory. This allowed us to make use of existing results and—by virtue of a canonical embedding of 1st order transition systems into uninterpreted ones—then lift these results to 1st order systems. In the formulation of the 1st order verification criteria, we have tried to stay close to the spirit of the inductive assertion method.

This paper has laid some foundations, but there remains a lot to be done. The existing framework falls short of a refinement calculus. E.g., it is not possible to directly prove that $\mathbf{A} \models \pi^0 \parallel \pi^1 \sqsupseteq \pi$, with $\pi^0, \pi^1, \pi \in \text{TS}$, because π is not a parallel composition. This would require a special proof rule that basically expresses how $\pi^0 \parallel \pi^1$ can be seen as a 1st order transition system; e.g., a location in $\pi^0 \parallel \pi^1$ would be given by a pair of vertices, one in π^0 and one in π^1 .

Also, process algebra suggests to try and lift axiomatizations of the algebraic pre order to the interpreted level.

The 1st order systems that we introduced are still one step away from a usable specification language. Instead of explicitly describing the underlying directed (control) graph, one would like to specify the

behavior of the actions and have these specifications implicitly define the control graph. This is the approach taken by Lamport [Lam83, Lam89].

This brings us to another issue: the comparison with other verification techniques for program refinement. For example, in [AL88, Mer90] it is shown how Milner simulation becomes a complete criterion for trace refinement if one allows programs to be augmented with various types of auxiliary variables. It should be investigated how these ideas relate to the methods of this paper.

Finally, the paper provides a solid basis to investigate the refinement of general liveness properties, e.g., using ω -automata [Eil74].

Acknowledgments

I thank Willem P. de Roever for his comments on previous versions of the paper and also the participants of the EUT-Weizmann DESCARTES seminar for their indulgence of my strive for perfect obfuscation.

References

- [AL88] M. ABADI, L. LAMPORT (1988), "The Existence of Refinement Mappings", Proc. 3d IEEE Conf. on Logic in Computer Science (LICS), pp. 165–175.
- [AL90] M. ABADI, L. LAMPORT (1990), "Composing Specifications", Proc. of the NFI/REX workshop on Stepwise Refinement of Distributed Systems: Models, Formalisms, Correctness, LNCS 430, pp. 1–42, Springer Verlag.
- [AS85] B. ALPERN, F.B. SCHNEIDER (1985), Defining Liveness, *Information Processing Letters*, Vol. 21, No. 4, pp. 181–185.
- [BC85] G. BERRY, L. COSSERAT (1985), The Synchronous Programming Language ESTEREL and its Mathematical Semantics, LNCS 197, pp. 389–449, Springer Verlag.
- [BHR84] S. BROOKES, C.A.R. HOARE, A. ROSCOE (1984), A Theory of Communicating Sequential Processes, *Journal of the ACM*, Vol. 31, No. 7, pp. 560–599.
- [BK84] J. BERGSTRA, J.W. KLOP (1984), Process Algebra for Synchronous Communication, *Information and Computation*, Vol. 60, pp. 109–137.
- [BKO86] J. BERGSTRA, J.W. KLOP, E.-R. OLDEROG (1986), "Failure semantics with fair abstraction", Report CS-R8609, Center for Mathematics and Computer Science (CWI), Amsterdam.
- [BKP84] H. BARRINGER, R. KUIPER, A. PNUELI (1984), Now You May Compose Temporal Logic Specifications, Proc. of the 16th Annual ACM Symposium on Theory of Computing (STOC), pp. 51–64, ACM.
- [CM88] K.M. CHANDY, J. MISRA (1988), **Parallel Program Design**, Addison-Wesley.
- [Dar82] PH. DARONDEAU (1982), "An Enlarged Definition and Complete Axiomatization of Observational Congruence of Finite Processes", LNCS 137, pp. 47–62, Springer Verlag.
- [Dij76] E.W. DIJKSTRA (1976), **A Discipline of Programming**, Prentice-Hall.
- [Eil74] S. EILENBERG (1974), **Automata, Languages and Machines, Volume A**, Academic Press.
- [FLS87] A. FEKETE, N. LYNCH, L. SHRIRA (1987), "A Modular Proof of Correctness for a Network Synchronizer", Proc. 2nd International Workshop on Distributed Algorithms, LNCS 312, Springer Verlag.

- [GB87] R. GERTH, A. BOUCHER (1987), "A Timed Failures Model for Extended Communicating Processes", Proc. 14th ICALP, LNCS 267, pp. 95–115, Springer Verlag.
- [GP89] R. GERTH, A. PNUELI (1989), "Rooting UNITY", Proc. 5th IEEE International Workshop on Software Specification and Design, pp. 11–19.
- [Flo67] R. FLOYD (1967), "Assigning Meaning to Programs", Proc. Sympos. in Appl. Math. 19, pp.19–32, American Mathematical Society.
- [Har87] D. HAREL (1987), Statecharts: a visual approach to complex systems, *Science of Computer Programming*, Vol. 8, No. 3.
- [Hen88] M. HENNESSY (1988), *Algebraic Theory of Processes*, The MIT press.
- [Hoa85] C.A.R. HOARE (1985), *Communicating Sequential Processes*, Prentice-Hall.
- [JM88] F. JAHANIAN, A. MOK (1988), Modecharts: a specification language for real-time systems, *IEEE Transactions on Software Engineering*, to appear.
- [Lam83] L. LAMPORT (1983), Specifying concurrent program modules, *ACM Transactions on Programming Languages and Systems*, Vol. 5, No. 2, pp. 190–222.
- [Lam89] L. LAMPORT (1989), A simple approach to specifying concurrent systems, *Communications of the ACM*, Vol. 32, No. 1, pp.32–45.
- [Lyn90] N. LYNCH (1990), "Multivalued Possibilities Mappings", Proc. of the NFI/REX workshop on Stepwise Refinement of Distributed Systems: Models, Formalisms, Correctness, LNCS 430, pp. 519–544, Springer Verlag.
- [LS84] S.S. LAM, A.U. SHANKAR (1984), Protocol verification via projection, *IEEE Transactions on Software Engineering*, Vol. 10, No. 4, pp. 325–342.
- [LT87] N. LYNCH, M. TUTTLE (1987), "Hierarchical correctness proofs for distributed algorithms", Proc. 6th ACM Sympos. Principles of Distributed Computing (PODC), pp. 137–151, ACM.
- [Mer90] M. MERRIT (1990), "Completeness Theorems for Automata", Proc. of the NFI/REX workshop on Stepwise Refinement of Distributed Systems: Models, Formalisms, Correctness, LNCS 430, pp. 544–561, Springer Verlag.
- [Mil71] R. MILNER (1971), "An algebraic definition of simulation between programs", Proc. 2nd Joint Conf. on Artificial Intelligence, British Computer Society, pp. 481–489. Also as Report No. CS-205, Computer Science Department, Stanford University.
- [Mil80] R. MILNER (1980), *A Calculus of Communicating Systems*, LNCS 94, Springer-Verlag, New York.
- [Mil83] R. MILNER (1983), Calculi for Synchrony and Asynchrony, *Theoretical Computer Science*, Vol. 25, pp. 267–310.
- [Mil89] R. MILNER (1989), *Communication and Concurrency*, Prentice Hall.
- [MP81] Z. MANNA, A. PNUELI (1981), "Verification of Concurrent Programs: The Temporal Framework", *The Correctness Problem in Computer Science* (R. S. Boyer, J. S. Moore, eds.), pp. 215–274, Academic Press.
- [MP84] Z. MANNA, A. PNUELI (1984), Adequate Proof Principles for Invariance and Liveness Properties of Concurrent Programs, *Science of Computer Programming*, Vol. 4, pp. 257–289.
- [Nic87] R. DE NICOLA (1987), Extensional Equivalences for Transition Systems, *Acta Informatica*, Vol. 24, pp. 211–237.

- [NH84] R. DE NICOLA, M. HENNESSY (1984), Testing Equivalences for Processes, *Theoretical Computer Science*, Vol. 34, pp. 83–133.
- [SdeR87] F. STOMP, W.P. DE ROEVER (1987), “A correctness proof of a distributed minimum-weight spanning tree algorithm”, *Proc. 7th IEEE International Conference on Distributed Computer Systems (ICDCS)*, pp. 440–448.
- [Sto89] F. STOMP (1989), *Design and Verification of Distributed Network Algorithms: Foundations and Applications*, Ph.D. thesis, Eindhoven University of Technology.
- [WLL88] J. WELCH, L. LAMPORT, N. LYNCH (1988), “A lattice-structured proof of a minimum spanning tree algorithm”, *Proc. ACM Symposium on Principles of Distributed Computing (PODC)*.

Index of notation

$dom(\cdot)$ 5
 $ran(\cdot)$ 5
 $f\{a/x\}$ 5
 A^+ 5
 A^* 5
 A^ω 5
 A^\dagger 5
 $\cdot \prec \cdot$ 5
 $\mathcal{A}(\cdot)$ 5, 7
 $\mathcal{A}_{xyz\dots}$ 5
 Var 6
 Σ 6
 $Tm(\Sigma)$ 6
 $L(\Sigma)$ 6
 $p[e/x]$ 6
 $FV(\cdot)$ 6
 \mathbf{A} 6
 \mathcal{S} 6
 $\sigma(t)$ 6
 $\mathbf{A}, \sigma \models p$ 6
 \mathcal{A} 6
 \mathcal{A}^l 6
 \mathcal{A}^g 6
 $(\mathcal{A}^g, |)$ 6
 δ 6
 \uparrow 6
 \mathcal{V} 6
 \mathcal{E} 6
 $\partial_s(\cdot)$ 6
 $\partial_t(\cdot)$ 6
 \cdot^* 6, 21
 \cdot^\bullet 6, 21
 π 6, 21
 I 6
 T 7
 \mathcal{L} 6
 π 6, 17
 UTS 6
 $V; \pi$ 7
 $\mathcal{L}_g(\cdot)$ 7
 $Seq(\cdot)$ 7
 behavior 7
 computation 7
 ρ 7
 $\uparrow(\cdot)$ 7, 29
 $\downarrow(\cdot)$ 7, 29
 $\delta(\cdot)$ 7, 9
 $\rightarrow \cdot \rightarrow^\circ$ 7
 partial computation 7
 \xrightarrow{a} 7, 21
 $\rightarrow \cdot \rightarrow$ 8
 \cdot after \cdot 8
 $\cdot || \cdot$ 8
 τ 9
 \mathcal{H} 9
 $h(\cdot)$ 9
 $\cdot + \cdot$ 10
 $\cdot ; \cdot$ 11
 \mathcal{C} 12
 \supseteq 13, 22
 $stable(\cdot)$ 14, 29
 \cdot fails \cdot 14
 \sqsubseteq 14
 \hookrightarrow_R 18
 \hookleftarrow 18
 $Act(\Sigma)$ 21
 $Gact(\Sigma)$ 21, 23
 $\mathcal{L}act(\Sigma)$ 21, 23
 $\llbracket \cdot \rrbracket^{\mathbf{A}}$ 21
 TS 21
 $\mathcal{L}^t(\cdot)$ 21, 26
 $\mathcal{L}^a(\cdot)$ 21, 26
 \dagger 22
 $\mathbf{A} \models \pi^0 \sqsubseteq \pi^1$ 22
 $Pvar$ 23
 $Svar$ 23
 $Chan$ 23
 $\cdot || \cdot$ 24
 $Enc(\cdot, enc)$ 25
 $Hide(\cdot, V)$ 25
 As 26
 His 26
 h 26
 set labelling 26
 λ 26
 $[\lambda^0, \lambda^1]$ 26
 $[\lambda^0, \lambda^1]_\sigma$ 26
 $\dagger[\lambda^0, \lambda^1]$ 26
 safe 27
 sure 28
 history determined 28
 $enabled(\cdot)$ 29
 cover 29
 $FO - REF$ 30
 $\vdash FO - REF$ 30
 local correctness 31
 leads to 32
 $SACO$ 32

In this series appeared :

No.	Author(s)	Title
85/01	R.H. Mak	The formal specification and derivation of CMOS-circuits.
85/02	W.M.C.J. van Overveld	On arithmetic operations with M-out-of-N-codes.
85/03	W.J.M. Lemmens	Use of a computer for evaluation of flow films.
85/04	T. Verhoeff H.M.L.J.Schols	Delay insensitive directed trace structures satisfy the foam the foam rubber wrapper postulate.
86/01	R. Koymans	Specifying message passing and real-time systems.
86/02	G.A. Bussing K.M. van Hee M. Voorhoeve	ELISA, A language for formal specification of information systems.
86/03	Rob Hoogerwoord	Some reflections on the implementation of trace structures.
86/04	G.J. Houben J. Paredaens K.M. van Hee	The partition of an information system in several systems.
86/05	J.L.G. Dietz K.M. van Hee	A framework for the conceptual modeling of discrete dynamic systems.
86/06	Tom Verhoeff	Nondeterminism and divergence created by concealment in CSP.
86/07	R. Gerth L. Shira	On proving communication closedness of distributed layers.
86/08	R. Koymans R.K. Shyamasundar W.P. de Roever R. Gerth S. Arun Kumar	Compositional semantics for real-time distributed computing (Inf.&Control 1987).
86/09	C. Huizing R. Gerth W.P. de Roever	Full abstraction of a real-time denotational semantics for an OCCAM-like language.
86/10	J. Hooman	A compositional proof theory for real-time distributed message passing.
86/11	W.P. de Roever	Questions to Robin Milner - A responder's commentary (IFIP86).
86/12	A. Boucher R. Gerth	A timed failures model for extended communicating processes.
86/13	R. Gerth W.P. de Roever	Proving monitors revisited: a first step towards Verifying object oriented systems (Fund. Informatica

86/14	R. Koymans	Specifying passing systems requires extending temporal logic.
87/01	R. Gerth	On the existence of sound and complete axiomatizations of the monitor concept.
87/02	Simon J. Klaver Chris F.M. Verberne	Federatieve Databases.
87/03	G.J. Houben J.Paredaens	A formal approach to distributed information systems.
87/04	T.Verhoeff	Delay-insensitive codes - An overview.
87/05	R.Kuiper	Enforcing non-determinism via linear time temporal logic specification.
87/06	R.Koymans	Temporele logica specificatie van message passing en real-time systemen (in Dutch).
87/07	R.Koymans	Specifying message passing and real-time systems with real-time temporal logic.
87/08	H.M.J.L. Schols	The maximum number of states after projection.
87/09	J. Kalisvaart L.R.A. Kessener W.J.M. Lemmens M.L.P. van Lierop F.J. Peters H.M.M. van de Wetering	Language extensions to study structures for raster graphics.
87/10	T.Verhoeff	Three families of maximally nondeterministic automata.
87/11	P.Lemmens	Eldorado ins and outs. Specifications of a data base management toolkit according to the functional model.
87/12	K.M. van Hee and A.Lapinski	OR and AI approaches to decision support systems.
87/13	J.C.S.P. van der Woude	Playing with patterns - searching for strings.
87/14	J. Hooman	A compositional proof system for an occam-like real-time language.
87/15	C. Huizing R. Gerth W.P. de Roever	A compositional semantics for statecharts.
87/16	H.M.M. ten Eikelder J.C.F. Wilmont	Normal forms for a class of formulas.
87/17	K.M. van Hee G.-J.Houben J.L.G. Dietz	Modelling of discrete dynamic systems framework and examples.

87/18	C.W.A.M. van Overveld	An integer algorithm for rendering curved surfaces.
87/19	A.J.Seebregts	Optimalisering van file allocatie in gedistribueerde database systemen.
87/20	G.J. Houben J. Paredaens	The R^2 -Algebra: An extension of an algebra for nested relations.
87/21	R. Gerth M. Codish Y. Lichtenstein E. Shapiro	Fully abstract denotational semantics for concurrent PROLOG.
88/01	T. Verhoeff	A Parallel Program That Generates the Möbius Sequence.
88/02	K.M. van Hee G.J. Houben L.J. Somers M. Voorhoeve	Executable Specification for Information Systems.
88/03	T. Verhoeff	Settling a Question about Pythagorean Triples.
88/04	G.J. Houben J.Paredaens D.Tahon	The Nested Relational Algebra: A Tool to Handle Structured Information.
88/05	K.M. van Hee G.J. Houben L.J. Somers M. Voorhoeve	Executable Specifications for Information Systems.
88/06	H.M.J.L. Schols	Notes on Delay-Insensitive Communication.
88/07	C. Huizing R. Gerth W.P. de Roever	Modelling Statecharts behaviour in a fully abstract way.
88/08	K.M. van Hee G.J. Houben L.J. Somers M. Voorhoeve	A Formal model for System Specification.
88/09	A.T.M. Aerts K.M. van Hee	A Tutorial for Data Modelling.
88/10	J.C. Ebergen	A Formal Approach to Designing Delay Insensitive Circuits.
88/11	G.J. Houben J.Paredaens	A graphical interface formalism: specifying nested relational databases.
88/12	A.E. Eiben	Abstract theory of planning.
88/13	A. Bijlsma	A unified approach to sequences, bags, and trees.
88/14	H.M.M. ten Eikelder R.H. Mak	Language theory of a lambda-calculus with recursive types.

88/15	R. Bos C. Hemerik	An introduction to the category theoretic solution of recursive domain equations.
88/16	C.Hemerik J.P.Katoen	Bottom-up tree acceptors.
88/17	K.M. van Hee G.J. Houben L.J. Somers M. Voorhoeve	Executable specifications for discrete event systems.
88/18	K.M. van Hee P.M.P. Rambags	Discrete event systems: concepts and basic results.
88/19	D.K. Hammer K.M. van Hee	Fasering en documentatie in software engineering.
88/20	K.M. van Hee L. Somers M.Voorhoeve	EXSPECT, the functional part.
89/1	E.Zs.Lepoeter-Molnar	Reconstruction of a 3-D surface from its normal vectors.
89/2	R.H. Mak P.Struik	A systolic design for dynamic programming.
89/3	H.M.M. Ten Eikelder C. Hemerik	Some category theoretical properties related to a model for a polymorphic lambda-calculus.
89/4	J.Zwiers W.P. de Roever	Compositionality and modularity in process specification and design: A trace-state based approach.
89/5	Wei Chen T.Verhoeff J.T.Udding	Networks of Communicating Processes and their (De-)Composition.
89/6	T.Verhoeff	Characterizations of Delay-Insensitive Communication Protocols.
89/7	P.Struik	A systematic design of a parallel program for Dirichlet convolution.
89/8	E.H.L.Aarts A.E.Eiben K.M. van Hee	A general theory of genetic algorithms.
89/9	K.M. van Hee P.M.P. Rambags	Discrete event systems: Dynamic versus static topology.
89/10	S.Ramesh	A new efficient implementation of CSP with output guards.
89/11	S.Ramesh	Algebraic specification and implementation of infinite processes.
89/12	A.T.M.Aerts K.M. van Hee	A concise formal framework for data modeling.

89/13	A.T.M.Aerts K.M. van Hee M.W.H. Hesen	A program generator for simulated annealing problems.
89/14	H.C.Haesen	ELDA, data manipulatie taal.
89/15	J.S.C.P. van der Woude	Optimal segmentations.
89/16	A.T.M.Aerts K.M. van Hee	Towards a framework for comparing data models.
89/17	M.J. van Diepen K.M. van Hee	A formal semantics for Z and the link between Z and the relational algebra.
90/1	W.P.de Roever-H.Barringer C.Courcoubetis-D.Gabbay R.Gerth-B.Jonsson-A.Pnueli M.Reed-J.Sifakis-J.Vytopil P.Wolper	Formal methods and tools for the development of distributed and real time systems, pp. 17.
90/2	K.M. van Hee P.M.P. Rambags	Dynamic process creation in high-level Petri nets, pp. 19.
90/3	R. Gerth	Foundations of Compositional Program Refinement - safety properties - , p. 38.
90/4	A. Peeters	Decomposition of delay-insensitive circuits, p. 25.
90/5	J.A. Brzozowski J.C. Ebergen	On the delay-sensitivity of gate networks, p. 23.
90/6	A.J.J.M. Marcelis	Typed inference systems : a reference document, p. 17.
90/7	A.J.J.M. Marcelis	A logic for one-pass, one-attributed grammars, p. 14.
90/8	M.B. Josephs	Receptive Process Theory, p. 16.
90/9	A.T.M. Aerts P.M.E. De Bra K.M. van Hee	Combining the functional and the relational model, p. 15.
90/10	M.J. van Diepen K.M. van Hee	A formal semantics for Z and the link between Z and the relational algebra, p. 30. (Revised version of CSNotes 89/17).
90/11	P. America F.S. de Boer	A proof system for process creation, p. 84.
90/12	P.America F.S. de Boer	A proof theory for a sequential version of POOL, p. 110.
90/13	K.R. Apt F.S. de Boer E.R. Olderog	Proving termination of Parallel Programs, p. 7.
90/14	F.S. de Boer	A proof system for the language POOL, p. 70.
90/15	F.S. de Boer	Compositionality in the temporal logic of concurrent systems, p. 17.

- 90/16 F.S. de Boer C. Palamidessi A fully abstract model for concurrent logic languages, p. 23.
- 90/17 F.S. de Boer C. Palamidessi On the asynchronous nature of communication in concurrent logic languages: a fully abstract model based on sequences, p. 29.
- 90/18 J.Coenen E.v.d.Sluis E.v.d.Velden Design and implementation aspects of remote procedure calls, p. 15.
- 90/19 M.M. de Brouwer P.A.C. Verkoulen Two Case Studies in ExSpect, p. 24.