Math. Struct. in Comp. Science (1997), *vol.* 7, *pp.* 241–282. Printed in the United Kingdom
© 1997 Cambridge University Press

Structured transition systems with parametric observations: observational congruences and minimal realizations†

GIANLUIGI FERRARI‡, UGO MONTANARI‡
and MIRANDA MOWBRAY§

‡*Dipartimento di Informatica, Università di Pisa, Corso Italia 40, I-56100 Pisa, Italy*
§*Hewlett-Packard Labs, Filton Road, Stoke Gifford, Bristol BS126QZ, UK*

Received 20 September 1995; revised 10 September 1996

A large number of observational semantics for process description languages have been developed, many of which are based on the notion of bisimulation. In this paper, we consider in detail the problem of defining a semantic framework to unify these. The discussion takes place in a purely algebraic setting. We introduce a special class of algebras called *Structured Transition Systems*. A structured transition system can be viewed as a transition system with an algebraic structure both on states and transitions. In this framework, observations of behaviours are dealt with by means of maps from the transitions to some algebra of observations.

Using several examples, we show that this framework allows us to describe a range of observational semantics within a single underlying presentation: it is enough to consider different mappings and algebras of observations. Furthermore, we introduce a notion of bisimulation that is parameterized with respect to the choice of the algebra of observations, and we find circumstances under which a Structured Transition System has good properties with respect to this parameterized bisimulation.

First, some general syntactic constraints, independent from the choice of the algebra of the observations, are given for Structured Transition System presentations. We show that these constraints ensure that parameterized bisimulation is always a congruence. Next, we address the problem of *Minimal Realizations*. We show that when the presentation satisfies the syntactic constraints there exists a minimal realization, *i.e.*, there is a model of the presentation whose elements fully characterize congruence classes under bisimulation.

1. Introduction

In recent years, many process description languages have been proposed, including *CCS* (Milner 1989), *CSP* (Hoare 1985) and *ACP* (Bergstra and Klop 1984). The starting point for the operational semantics of such languages is the idea of a *Labelled Transition*

† This research was supported by, and collaboratively carried out at, Hewlett-Packard Laboratories, Pisa Science Centre, and by CNR project 'Strumenti per la specifica e la verifica di proprietà critiche di sistemi concorrenti e distribuiti'.

https://doi.org/10.1017/S0960129597002284 Published online by Cambridge University Press

System (Keller 1976). Labelled transition systems provide a rather general computational model: in order to give operational definitions in terms of them it is enough to identify the collections of *configurations* or *states*, and then to describe how one state evolves into another – labels of transitions describe the (observable) effect of the evolution in the external environment. In the case of process description languages, states are endowed with an algebraic structure, because states are terms of the language, and labelled transitions between states are defined by inference rules driven by the syntactic structure of states. This method is often referred to as *SOS* (*Structured Operational Semantics*) (Plotkin 1981).

Labelled transition systems are well suited to handle computational systems that are characterized by their on-going behaviour. However, this elegant computational model fails to capture some useful properties of computations associated with spatial distribution or concurrency. For instance, there is no way of expressing independence between parallel transitions. As a consequence, computations that differ just by the order in which two independent parallel transitions are performed cannot be identified. This limitation has led several authors to extend the notion of transition systems by enriching either the structure of states (*e.g.*, Boudol *et al.* (1993), Castellani and Hennessy (1989), Degano *et al.* (1988), Degano *et al.* (1989) and Olderog (1987)), or the structure of transitions (*e.g.*, Bednarczyk (1988), Baduel and Darondeau (1992), Boudol and Castellani (1989), Nielsen and Winskel (1992), and Stark (1989)).

In this paper we attempt to define a general and uniform framework for describing the semantics of process description languages. The characterizing feature of our proposal is that we consider transition systems that have an algebraic structure both on states and on transitions, namely *Structured Transition Systems* (STS).

An STS has three types of component: *states*, *transitions* and *computations*. STSs are defined in a purely algebraic style as particular *Equational Type Algebras* (Manca *et al.* 1990). Equational type algebras are one-sorted algebras equipped with a typing relation that provides a discipline for the use of the operations. The typing information is used to identify which elements are considered to be states, transitions or computations. Hence, a specific STS is defined by giving its operators and prescribing the typing discipline.

In order to illustrate the expressive power of the approach, our main focus in this paper is on the semantics of process description languages (we consider as a running example Milner's *Calculus of Communicating Systems* (CCS) (Milner 1989)). However, the range of applications of STS is not restricted to this field. For instance, Petri Nets (Reisig 1985) and Horn Clause Logic (Lloyd 1987) can be provided with an algebraic semantics in terms of STS (Corradini *et al.* 1990). The STS approach generalizes the algebraic treatment of Petri Nets as proposed by Meseguer and Montanari (Meseguer and Montanari 1990).

In the STS presentation of CCS – $\mathscr{CCS}$ – the states are CCS agents, so the algebraic structure on states is provided by the language itself; this is in the line of the SOS approach. The algebraic structure on transitions is obtained instead by considering the SOS inference rules as operations. Finally, the computations are sequences of transitions, generated from the transitions by an operation of sequential composition. This is in the style of the classical categorical interpretation of logical systems: there, propositions are the objects of a category and the morphisms are the proofs generated by the inference rules.

https://doi.org/10.1017/S0960129597002284 Published online by Cambridge University Press

Instead of defining a single model for CCS, we introduce a collection of models, technically the category $\mathscr{STS}[\mathscr{CCS}]$, whose objects are algebras satisfying the presentation $\mathscr{CCS}$, and where the morphisms preserve the algebraic structure and the typing discipline.

The algebras that satisfy the presentation $\mathscr{CCS}$ are certain algebras for CCS where the transitions are unlabelled. In contrast, the standard semantics for CCS are based on labelled transitions. In our framework we deal with the labelling of the transitions by using a labelling map from the transitions to some algebra of observations; the map abstracts away those aspects of behaviour that are considered unimportant.

A fundamental property of concurrency theory is the great variety of possible domains of observations. Because many diverse aspects of process behaviour can be important in different situations, we believe that concurrency models should not be based on a single domain of observations. Instead, they should be equipped with a uniform mechanism to deal with different facets of process behaviour. The existence of a uniform mechanism that allows us to deal with different domains of observations is one of the main properties of our proposal. In contrast, the SOS approach fixes the labels of transitions once and for all.

The algebras satisfying an STS presentation are in general not free algebras: nonfreeness reflects a particular interpretation of the operations. For instance by suitably interpreting the operations, we can construct an algebra satisfying the presentation (*i.e.*, an object of the category of STS) such that the interpretation of the operations reflects a certain domain of observation. We call this algebra the *model of observations*. Hence, defining a labelling operation corresponds to selecting a morphism from each object of the category to the model of observations. The labelling operation constructs a category where the objects are labelled structured transition systems, and where the morphisms preserve the labels. Notice that this operation is internal to the category of structured transition systems satisfying the presentation.

One advantage of using our algebraic setting is that it is possible to have different interpretations of the operators with a single underlying presentation. As a consequence, there is a clean division between the *control* level and the *observation* level. The control level is completely determined by the presentation; whereas which aspects of the behaviour are observable (the observation level) is determined by the interpretation of the operations. By choosing different interpretations we can consider a wide range of domains of observations without modifying the control structure. For our running example, we show that the category $\mathscr{STS}[\mathscr{CCS}]$ has enough structure to handle both a model of observations that gives the standard *interleaving* semantics for CCS, and a model giving a *truly concurrent* semantics. We also provide models of observations that abstract away from internal moves.

There is general agreement amongst concurrency theorists that the semantical objects that really matter are equivalence classes of processes under some observational equivalence. A considerable number of observational equivalences have been proposed. Bisimulation (Park 1981; Milner 1989) is the most stable and studied mathematical device for defining equivalences that captures the idea of similarity of processes.

A particularly useful property that an equivalence may have is that of being a congruence with respect to the operators of the language, which means that the associated

https://doi.org/10.1017/S0960129597002284 Published online by Cambridge University Press

semantics is compositional. For instance, strong bisimulation (the simplest bisimulation equivalence) is a congruence for CCS.

Since De Simone's work on the expressiveness of SCCS-MEIJE calculi (De Simone 1985) several works have been devoted to understand the observational semantics of process description languages in terms of *meta-theoretic properties* of the SOS method. It has been shown that some basic theorems always hold if the SOS inference rules fit in some format (they satisfy certain syntactic constraints) (Bloom *et al.* 1995; Baeten and Verhoef 1993; Groote 1990; Groote and Vaandrager 1992).

This brings us to one of the main topics of the paper. We first extend the notion of bisimulation to deal with our algebraic framework. In particular, we introduce a notion of bisimulation that is parameterized with respect to the model of observations. Then, we introduce some syntactic constraints on the format of STS presentations, and we prove that under these constraints, parameterized bisimulation is always a congruence. An important feature of this set of constraints is that it is *completely independent* of the model of observations chosen.

The fact that we have bisimulations that are also congruences leads us to an interesting question. Are our semantics characterized by an abstract structured transition system? In other words, is there a structured transition system satisfying the presentation that plays the role of the classical minimal state realization of automata? If such a structured transition system exists, it will be the *minimal realization* of the language with respect to the semantics. The answer in the case when the presentation satisfies the constraints is that such a minimal realization does exist.

To construct the minimal realization, we introduce a special class of morphisms between labelled structured transition systems called *transition preserving homomorphisms*. These morphisms can be understood as program transformations that take a system and yield a structurally simpler but equivalent system. We then consider the category of *Admissible Realizations*, whose objects are models of the presentation that are reachable from the initial object via transition preserving homomorphisms, and whose arrows are transition preserving homomorphisms. We prove that this category always has a terminal object, and, moreover, that when the presentation satisfies the syntactic constraints, the terminal object is a minimal realization.

2. Structured Transition Systems

Structured Transition Systems are introduced in a purely algebraic style in terms of *Equational Type Algebras* (Manca *et al.* 1990). Equational type algebras are one-sorted algebras enriched with a typing relation that assigns types to elements of the algebra (types are themselves elements of the algebra). Formally, an STS is an equational type algebra over a certain signature equipped with three fixed constant types: the type `state`, the type `trans` and the type `comp`. There are some typing judgments expressing the conditions for an element to have a certain type. This typing information expresses when an element of the algebra should be considered a *correct* state (respectively, transition, computation). Notice that this is the standard notion of typing in programming languages: only well-typed programs are relevant.

https://doi.org/10.1017/S0960129597002284 Published online by Cambridge University Press

The use of type algebras is motivated mainly by the need to discriminate between states, transitions and computations. As an example, synchronization will be interpreted as an operation on transitions, *i.e.*, it is well typed only on elements of type `trans` and not on elements of type `state` or `comp`. Another issue that can be elegantly addressed by the use of equational type algebras is the construction of recursive processes.

Here we recall the basic definitions of equational type algebras (refer to Manca *et al.* (1990) for a more comprehensive introduction). Let Σ be a one-sorted signature, *i.e.*, a set of operator symbols together with their arities.

Definition 1. (Type Algebras and Morphisms) A *type algebra* $\mathscr{A}$ over the signature Σ is a pair $(A, :_A)$, where A is a one-sort total algebra over Σ, and $:_A$ is a binary relation, the typing relation, on the carrier of A.

A morphism from the type algebra $\mathscr{A}$ into the type algebra $\mathscr{B}$ is a mapping h from A to B that respects the operations and the typing:

— if σ is an r-ary operator in Σ then $h(\sigma(a_1, \ldots, a_r)) = \sigma(h(a_1), \ldots, h(a_r))$,
— if $a :_A t$ then $h(a) :_B h(t)$.

Notice that the definition above requires that morphisms preserve the typing relation. However, there is nothing to stop a morphism from mapping an untyped element to a typed element.

To specify the properties that characterize a certain class of type algebras, a generalization of the standard equational logic (called Equational Type Logic) is introduced. Let V be a set of variables. We use $T_\Sigma(V)$ to indicate the set of Σ terms over V.

Definition 2. (Equational Type Logic) The *equational type logic* (ET logic) over the signature Σ and variables V is given as follows.

Atomic Formulae

$$e_1 = e_2 \quad \text{(Equations)}$$
$$e_1 : e_2 \quad \text{(Type Assignments)}$$

where e_1, e_2 are elements of $T_\Sigma(V)$

Formulae or Judgments

$$\frac{\varphi_1, \ldots, \varphi_n}{\varphi} \quad \text{(Judgment)}$$

where φ is an atomic formula (called conclusion), and $\varphi_1, \ldots, \varphi_n$ is a finite, possibly empty, set of atomic formulae (called assumptions).

In the following we will use $\Gamma \vdash e_1 = e_2$ (respectively, $\Gamma \vdash e_1 : e_2$) to indicate that the equality (respectively, the type assignment) is derivable from the set of formulae Γ. Soundness and completeness of ET logic have been proved in Manca *et al.* (1990).

Definition 3. An *equational type algebra presentation* is a triple (Σ, V, Γ) where Γ is a set of ET formulae over the signature Σ and set of variables V.

The example below gives a presentation of the natural numbers as an equational type algebra. We assume that n ranges over the set of variables of the presentation.

$$0 : nat \qquad \frac{n : nat}{succ(n) : nat} \qquad \frac{n : nat}{pred(succ(n)) = n}$$

https://doi.org/10.1017/S0960129597002284 Published online by Cambridge University Press

No type can be assigned to the term *pred*(0) by applying the above judgments. However, it is easy to see that *pred*(0) is an (untyped) element, furthermore, *succ*(*t*) and *pred*(*t*) are untyped whenever t is untyped. This is a typical feature of equational type algebras: the untyped elements are elements of the algebra, but they have to be considered as underdefined.

Definition 4. (Structured Transition System Presentations) A *Structured Transition System Presentation* (STS presentation) $\mathscr{T}$ is an equational type algebra presentation (Σ, V, Γ) where:

1 `state`, `trans` and `comp` are constants in Σ,
2 The typing judgments are either of the form:

$$\frac{e : e_1}{e : e_2}$$

where e is a term and e_1, e_2 are constants, or of the form

$$\frac{\varphi_1(e_1), \ldots, \varphi_n(e_n)}{\sigma(e_1, \ldots, e_n) : e}$$

where σ is a operator symbol of arity n, $\varphi_i(e_i)$ is an atomic formula containing the term e_i, and e is a constant.
3 Γ includes the judgment

$$\frac{e : \texttt{trans}}{e : \texttt{comp}}$$

4 ∂_0, ∂_1 are unary operator symbols in Σ, such that

$$\Gamma \vdash \partial_i(e) : \texttt{state} \Leftrightarrow \Gamma \vdash e : \texttt{comp}, \text{ for } i \in \{0, 1\}.$$

5 ';' is a binary operation in Σ such that

$$\frac{\partial_1(e_1) : \texttt{state}, \partial_0(e_2) : \texttt{state}, \partial_1(e_1) = \partial_0(e_2)}{e_1 ; e_2 : \texttt{comp}}$$

$$\partial_0(e_1 ; e_2) = \partial_0(e_1) \text{ and } \partial_1(e_1 ; e_2) = \partial_1(e_2)$$

$$e_1 ; (e_2 ; e_3) = (e_1 ; e_2) ; e_3$$

A *Structured Transition System* is an equational type algebra that satisfies an STS presentation.

An STS presentation is an equational type algebra presentation that necessarily has the constants `state`, `trans` and `comp` that can be used as types. Point (2) ensures that multityping is allowed, and that typing judgments are expressed by structural induction. Point (3) states that any element of type `trans` is also of type `comp`. By point (4), the unary operators ∂_0, ∂_1 associate origins and destinations (which are elements of the algebra of type `state`) to elements of type `comp`. Finally, point (5) prescribes the behaviour of the operation of sequential composition.

In the case of elements of type `trans`, the typing judgments can be also interpreted as conditional rules in the SOS style, defining the transitions starting from each state. The

https://doi.org/10.1017/S0960129597002284 Published online by Cambridge University Press

element $\sigma(e_1,\ldots,e_n)$: `trans` can be understood as representing a labelled transition of the form

$$\partial_0(\sigma(e_1,\ldots,e_n)) \overset{\sigma(e_1,\ldots,e_n)}{\longrightarrow} \partial_1(\sigma(e_1,\ldots,e_n))$$

expressing the fact that the state $\partial_0(\sigma(e_1,\ldots,e_n))$ evolves into $\partial_1(\sigma(e_1,\ldots,e_n))$ by performing the instruction $\sigma(e_1,\ldots,e_n)$. Hence, an STS can be thought of as a graph whose nodes are the elements of type `state`, and whose edges are the elements of type `comp` (which include the elements of type `trans`). The unary operators ∂_0, ∂_1 give the source and target of edges. Strictly speaking, these operators are defined on all elements, but we require that $\partial_0(e)$, $\partial_1(e)$ are typed `state` if and only if e has type `comp`.

As a matter of notation, we will use $t : u \to v$ as shorthand for 't has type `trans`, $\partial_0(t) = u$, $\partial_1(t) = v$'. Similarly, we will use $c : u \Rightarrow v$ to indicate that c has type `comp`, $\partial_0(c) = u$, and $\partial_1(c) = v$. It should be remarked that when the conclusion of a judgments is of the form $t : u \to v$ (respectively, $c : u \Rightarrow v$), the judgment stands for the set of three judgments with the same assumptions whose conclusions are $\partial_0(t) = u$, $\partial_1(t) = v$, and t : `trans` (respectively, $\partial_0(c) = u$, $\partial_1(c) = v$, and c : `comp`).

In the following we will identify an STS presentation with its judgments. Moreover, in giving a particular STS presentation, we will not concentrate on the basic ingredients (constants and operators) but on the judgments of the specific operators, for example, we will always assume that `state` is a constant, that $t : u \to v$ implies $t : u \Rightarrow v$, and, finally, the judgments of the operator of sequential composition will always be omitted.

Definition 5. Let $\mathscr{T} = (\Sigma, V, \Gamma)$ be an STS presentation. The category $\mathscr{SIS}[\Gamma]$ has as objects the class of equational type algebras satisfying the presentation $\mathscr{T}$, and as arrows morphisms of equational type algebras.

The usual constructions of universal algebra have been extended to equational type algebras as well, and therefore to structured transition systems. For instance, the *initial* equational type algebra always exists. In the category of structured transition systems the initial algebra is the equational type algebra whose elements are the elements of the word algebra $T_\Sigma(V)$, modulo the congruence generated by the equational judgments, and whose typing relation is the minimal relation generated by the typing judgments.

Proposition 6. Let $\mathscr{T} = (\Sigma, V, \Gamma)$ be an STS presentation. The category $\mathscr{SIS}[\Gamma]$ has an initial object $\mathscr{I}$

Structured Transition Systems will be used mainly as a tool for giving operational semantics to process description languages. We will use as a running example an STS presentation for Milner's *Calculus of Communicating Systems* (CCS) (Milner 1989), which we present below.

We first recall the syntax of CCS. Let Δ be the alphabet for basic actions (which is ranged over by α) and $\overline{\Delta}$ the alphabet of complementary actions ($\Delta = \overline{\overline{\Delta}}$ and $\Delta \cap \overline{\Delta} = \varnothing$); the set $\Lambda = \Delta \cup \overline{\Delta}$ will be ranged over by λ. Let $\tau \notin \Lambda$ be a special action and let $\Lambda \cup \{\tau\}$ (ranged over by μ) be the set of CCS actions. Finally, let R (ranged over by x) be a set of *recursion variables* that is disjoint from the set of variables V of the presentation.

https://doi.org/10.1017/S0960129597002284 Published online by Cambridge University Press

Definition 7. (A Calculus of Communicating Systems) A CCS expression E has the following syntax:

$$\begin{array}{rcl} E & := & nil \\ & | & x \\ & | & \mu.E \\ & | & E\backslash\alpha \\ & | & E[\Phi] \\ & | & E + E \\ & | & E \mid E \\ & | & rec\ x.E \end{array}$$

where Φ is a permutation of $\Lambda \cup \{\tau\}$ preserving τ and complementation.

The prefix *rec x* binds the variable x; the concepts of free and bound variables can be defined in the usual way. A CCS agent is a CCS expression without free variables. A guarded CCS agent is a CCS agent where each occurrence of a variable x in *rec x.E* is within a subexpression $\mu.E'$.

The STS presentation of CCS, $\mathscr{CCS}$, is given in Table 1. In the presentation, generic elements (sometimes called terms) will be denoted by e, elements of type `state` by u, v, w, elements of type `trans` by t and elements of type `comp` by c. More precisely, these are the variables of the presentation and whenever (for instance) the variable u appears in a judgment there is an implicit assumption u : `state`.

We can comment briefly on the presentation $\mathscr{CCS}$. The elements of the algebra that are guarded CCS agents have type `state`. CCS expressions may have free and unguarded variables, thus we use (X, Y), where $X, Y \subseteq R$, to indicate the type of a CCS expression whose free variables are X and whose free unguarded variables are Y. These elements (X, Y) are constants of the algebra. Notice that any element with type `state` also has type (ϕ, ϕ), and that if an element has type (X, Y), then $Y \subseteq X$.

A crucial part of the presentation is the handling of recursive processes. In order to show the construction of recursive processes, consider the elements $x : (\{x\}, \{x\})$ and $\mu.x : (\{x\}, \phi)$. Applying the judgment of recursion we have that $rec\ x.\mu.x : (\phi, \phi)$. On the other hand, this judgment cannot even be applied to $\mu.x + x$: the unwinding of recursion can only be applied to guarded recursion variables. The intuitive idea is that a recursive element of type `state` can be characterized by the class of its finite approximations, *i.e.*, the unwinding of the recursive subterms. The operator *rec* can be unwound at any depth of recursive subterms, then in the initial object, $\mathscr{I}_{\mathscr{CCS}}$, we can write any element of type `state` as a term whose topmost operator is not *rec x* for any x. Hence, we can safely remove the recursive operators from the set of operators yielding elements of type `state`.

This begs the question of whether we can assign meaning to recursive terms in the class of algebras satisfying the presentation $\mathscr{CCS}$. The interpretation of recursive terms as solution of fixed point equations (see Part II of Hennessy (1988)) can be extended to cover our algebraic framework. In this perspective the guardness requirement plays a major role, since guarded recursive equations are guaranteed to have a unique solution, and the structure of the typing judgments ensures that the recursive elements of type `state` are indeed guarded.

https://doi.org/10.1017/S0960129597002284 Published online by Cambridge University Press

The first set of judgments gives conditions for an element to have type `state`.

$$nil : (\phi, \phi) \qquad \frac{u : (\phi, \phi)}{u : \texttt{state}}$$

$$x : (\{x\}, \{x\}) \qquad \frac{e : (X, Y)}{\mu.e : (X, \phi)}$$

$$\frac{e : (X, Y)}{e[\Phi] : (X, Y)} \qquad \frac{e : (X, Y)}{e\backslash\alpha : (X, Y)}$$

$$\frac{e_i : (X_i, Y_i), i = 1, 2}{e_1 + e_2 : (\cup_i X_i, \cup_i Y_i)} \qquad \frac{e_i : (X_i, Y_i), i = 1, 2}{e_1 \mid e_2 : (\cup_i X_i, \cup_i Y_i)}$$

$$\frac{e : (X, Y), x \notin Y}{rec\ x.e : (X \cap \overline{\{x\}}, Y)} \qquad \frac{rec\ x.e : \texttt{state}}{rec\ x.e = e[rec\ x.e/x]}$$

The final set of judgments deals with the typing of the operations on transitions.

$$[\mu, v\rangle : \mu.v \to v$$

$$\frac{t : u \to v}{t[\Phi] : u[\Phi] \to v[\Phi]} \qquad \frac{t : u \to v}{t\backslash\alpha : u\backslash\alpha \to v\backslash\alpha}$$

$$\frac{t : u \to v}{t \lhd w : u + w \to v} \qquad \frac{t : u \to v}{w \rhd t : w + u \to v}$$

$$\frac{t : u \to v}{t\rfloor w : u \mid w \to v \mid w} \qquad \frac{t : u \to v}{w\lfloor t : w \mid u \to w \mid v}$$

$$\frac{t_1 : u_1 \to v_1, t_2 : u_2 \to v_2}{t_1 \mid t_2 : u_1 \mid u_2 \to v_1 \mid v_2}$$

Table 1. *The Structured Transition System presentation of CCS*

Basically, CCS agents are the elements of type `state` of the initial object $\mathscr{I}_{\mathscr{CCS}}$. Hereafter, when it is clear from context, we will use CCS agents to indicate the corresponding elements of type `state`.

The STS description of the operational semantics of CCS differs from the standard SOS semantics. In that approach, states are CCS agents, and transitions among states are defined by structural induction on the syntax of agents using the SOS style. Here, instead, inference rules are replaced by operations.

This construction has several analogies with the construction of the *Proved Transition System* of Boudol and Castellani (Boudol and Castellani 1989). There, the transitions are labelled with their proofs, *i.e.*, inference rules are proof constructors. In our approach,

https://doi.org/10.1017/S0960129597002284 Published online by Cambridge University Press

we are interested in studying the properties of a class of equational type algebras whose algebraic structure on elements of type `trans` is determined by interpreting the inference rules as operations. The proved transition system will turn out to be the initial object, $\mathscr{I}_{\mathscr{CCS}}$, of this class of algebras. Notice that the elements of type `comp` in $\mathscr{I}_{\mathscr{CCS}}$ are all of the form $t_1; t_2; \ldots; t_n$ for some $n \geqslant 1$ and elements $t_1, \ldots t_n$ of type `trans`.

To illustrate how the STS semantics of CCS works, let us consider the element of type `state` of $\mathscr{I}_{\mathscr{CCS}}$ corresponding to the CCS agent $E = (\beta.\alpha.nil + \gamma.nil) \mid \overline{\beta}.nil$. Applying the judgments of the presentation, we have

$$\frac{\dfrac{[\beta, \alpha.nil\rangle : \beta.\alpha.nil \longrightarrow \alpha.nil}{[\beta, \alpha.nil\rangle \lhd \gamma.nil : \beta.\alpha.nil + \gamma.nil \longrightarrow \alpha.nil} \qquad [\overline{\beta}, nil\rangle : \overline{\beta}.nil \longrightarrow nil}{([\beta, \alpha.nil\rangle \lhd \gamma.nil) \mid [\overline{\beta}, nil\rangle : (\beta.\alpha.nil + \gamma.nil) \mid \overline{\beta}.nil \longrightarrow \alpha.nil \mid nil}$$

Roughly speaking, the transition above corresponds to the derivation

$$(\beta.\alpha.nil + \gamma.nil) \mid \overline{\beta}.nil \xrightarrow{\tau} \alpha.nil \mid nil.$$

Indeed, the term $([\beta, \alpha.nil\rangle \lhd \gamma.nil) \mid [\overline{\beta}, nil\rangle$ represents the proof of this derivation.

We have already remarked that in the initial object, computations are obtained just by concatenating sequences of transitions. In our example we can derive the following transition:

$$\frac{[\alpha, nil\rangle : \alpha.nil \longrightarrow nil}{[\alpha, nil\rangle\rfloor nil : \alpha.nil \mid nil \longrightarrow nil \mid nil}$$

Now, by applying the judgment of the sequential composition, we have

$$([\beta, \alpha.nil\rangle \lhd \gamma.nil \mid [\overline{\beta}, nil\rangle); [\alpha, nil\rangle\rfloor nil : (\beta.\alpha.nil + \gamma.nil) \mid \overline{\beta}.nil \Rightarrow nil \mid nil.$$

Our final example shows that the range of applications of STS is not restricted to the operational semantics of process description languages. For instance, Petri Nets (Reisig 1985) can be viewed as structured transition systems. Here we consider the formulation of (place/transition) Petri nets as graphs where the nodes form a free commutative monoid (Meseguer and Montanari 1990), and we show how the $\mathscr{T}[N]$ construction fits in our framework (see also (Montanari and Yankelevich 1994)). $\mathscr{T}[N]$ is the strictly symmetric monoidal category freely generated by the net N (Meseguer and Montanari 1990). The morphisms of $\mathscr{T}[N]$ have been proved (Degano *et al.* 1989) to be isomorphic to Best-Devillers (Best and Devillers 1987) net processes.

Definition 8. (Petri Nets as Graphs with a Monoidal Structure) A Place/Transition Petri Net is a quadruple $N = (S^{\otimes}, T, pre, post)$, where $S^{\otimes}$ is the free commutative monoid over the set of places S, T is a set of transitions, and $pre, post : T \longrightarrow S^{\otimes}$ are functions associating to each transition a source and a target node, respectively.

The elements of the free commutative monoid $S^{\otimes}$ are called *markings* (ranged over by u). For any transition $t \in T$ we shall write $t : u \to v$ to indicate that $pre(t) = u$ and $post(t) = v$. We now introduce an STS presentation whose generators are the elements (places and transitions) of the net.

The elements of type `state` are the elements of the free commutative monoid over the set of places. The elements of type `trans` are the transitions of the net. Finally, the

https://doi.org/10.1017/S0960129597002284 Published online by Cambridge University Press

The first set of judgments details the conditions for an element to have type `state`.

$$0 : \texttt{state}$$

$$\frac{s \in S}{s : \texttt{state}}$$

$$\frac{u : \texttt{state},\ v : \texttt{state}}{u \otimes v : \texttt{state}}$$

The next judgments detail the conditions for an element to have type `trans` or `comp`.

$$t : u \to v,\ for\ t \in T$$

$$\frac{u : \texttt{state}}{id(u) : u \Rightarrow u}$$

$$\frac{\alpha : u \Rightarrow v, \alpha' : u' \Rightarrow v'}{\alpha \otimes \alpha' : u \otimes u' \Rightarrow v \otimes v'}$$

The next judgments define the equations the elements of the algebra must satisfy.

$$(e_1 \otimes e_2) \otimes e_3 = e_1 \otimes (e_2 \otimes e_3)$$

$$e_1 \otimes e_2 = e_2 \otimes e_1$$

$$\frac{u : \texttt{state}}{u \otimes 0 = u}$$

$$\frac{\alpha : \texttt{comp}}{\alpha \otimes id(0) = \alpha}$$

$$\frac{\alpha : \texttt{comp}}{\alpha; id(\partial_1(\alpha)) = \alpha = id(\partial_0(\alpha)); \alpha}$$

$$\frac{\alpha, \beta, \gamma, \delta : \texttt{comp}}{(\alpha \otimes \gamma); (\beta \otimes \delta) = (\alpha; \beta) \otimes (\gamma; \delta)}$$

$$\frac{u, v : \texttt{state}}{id(u \otimes v) = id(u) \otimes id(v)}$$

Table 2. *Petri nets as STS*

elements of type `comp` are obtained by closing the set of elements of type `trans` with respect to the operations of parallel $(- \otimes -)$ and sequential $(-;-)$ composition. The STS presentation is given in Table 2.

Now, let $\mathscr{I}_{\mathscr{N}}$ be the initial model of the category $\mathscr{SMS}[\mathscr{N}]$. It is easy to see that the elements of type `state` of $\mathscr{I}_{\mathscr{N}}$ correspond to the objects of the category $\mathscr{T}[N]$, and the elements of type `comp` correspond to the morphisms of $\mathscr{T}[N]$. In fact, the judgments of the presentation express the fact that the transitions of the net are the generators of the algebra, and the elements of type `comp` are obtained by closing the generators with respect to parallel and sequential composition. Furthermore, the axioms express the fact

https://doi.org/10.1017/S0960129597002284 Published online by Cambridge University Press

that $- \otimes -$ is a (strictly) monoidal operation, $-;-$ gives the composition of morphisms and $id(u)$ is the identity morphism.

As a final remark in this section we want to point out that, in the case of the presentation of the category $\mathscr{STS}[\mathscr{CCS}]$, the only operations yielding elements that are typed as computations but not as transitions is the operation of sequential composition; whereas in the case of $\mathscr{STS}[\mathscr{N}]$ the typing information on transitions is used just to identify the generators, and the operations are meaningful only on computations.

3. Observing behaviours

Much work in the semantics of process description languages has been based on notions of observations of behaviour, *i.e.*, abstract descriptions of the behaviour of processes. A fundamental property of concurrency theory is the great variety of possible observations of behaviour, such as strings of actions, labelled partial orders, and so on. The large variety of possible observations depends on the fact that at present there is no widely accepted metalanguage to describe the behaviour of processes in an abstract way, and that different aspects of process behaviour are important in different applications. Because of the variety of aspects of process behaviour that can be important, we think that a model of concurrency should not be based on a single notion of behaviour, but, instead, should be equipped with a uniform mechanism to deal with the different facets of process behaviour. This issue is the main concern of this section.

Let $\mathscr{T} = (\Sigma, V, \Gamma)$ be an STS presentation. As it stands, the objects of the category $\mathscr{STS}[\Gamma]$ do not include any mechanism for observing computations, in that the elements of type `comp` represent computations but there is no further abstraction corresponding to what can be detected by an external observer. For instance, let us consider the category $\mathscr{STS}[\mathscr{CCS}]$. In the initial object $\mathscr{I}_{\mathscr{CCS}}$ there is an element

$$[\alpha, nil\rangle \mid [\beta, nil\rangle : \alpha.nil \mid \beta.nil \rightarrow nil \mid nil,$$

which is considered to be a legitimate computation, since it is of type `trans`. However, it is not a legitimate computation if we assume the standard CCS synchronization rule.

In this paper, we follow the approach introduced in Ferrari (1990) and Ferrari and Montanari (1990): the observation mechanism is handled by an operation of labelling that is internal to the category of structured transition systems $\mathscr{STS}[\Gamma]$. The idea is the following. We select a specific object in the category $\mathscr{STS}[\Gamma]$ to be the structured transition system of behaviours or *model of observations*. Then, defining an observation mechanism corresponds to selecting a morphism from each object of the category to the model of observations.

Definition 9. Let $\mathscr{B}$ be any object of $\mathscr{STS}[\Gamma]$. Then $\mathscr{STS}[\Gamma] \gg \mathscr{B}$ is the category defined as follows:

— The objects are pairs $(T, \ell : T \rightarrow \mathscr{B})$ where
  – T is an object of $\mathscr{STS}[\Gamma]$;
  – ℓ is a morphism of $\mathscr{STS}[\Gamma]$.

https://doi.org/10.1017/S0960129597002284 Published online by Cambridge University Press

— The morphisms are maps $f : (T_1, \ell_1 : T_1 \to \mathscr{B}) \to (T_2, \ell_2 : T_2 \to \mathscr{B})$ such that:
  – $f : T_1 \longrightarrow T_2$ is a morphism of $\mathscr{STS}[\Gamma]$;
  – $\ell_2(f(e)) = \ell_1(e)$, for all elements of T_1.

The construction above labels each element of a structured transition system with an element taken from the model of observations $\mathscr{B}$. Moreover, the morphisms of the category $\mathscr{STS}[\Gamma] \gg \mathscr{B}$ preserve the labelling. In particular, we have defined a mechanism to observe computations. In fact, let (T, ℓ) be an element of $\mathscr{STS}[\Gamma] \gg \mathscr{B}$, and let c : `comp` an element of T, then $\ell(e)$: `comp` is an element of $\mathscr{B}$: it is the observation of the computation c.

The labelling construction can seem *ad hoc*, but it can be completely formulated in the language of category theory: it is a special case of the *Comma Category* construction (Mac Lane 1971). The category $\mathscr{STS}[\Gamma] \gg \mathscr{B}$ is the comma category $(Id \downarrow \mathscr{B})$, where Id is the identity functor on $\mathscr{STS}[\Gamma]$ and the functor $\mathscr{B}$ is the constant functor always returning the structured transition system $\mathscr{B}$. Comma categories of this form are also known under the name of *slice categories.*

Proposition 10. Let **C** be a category with an initial object i, and let b be another object of **C**. Then (i, η_b) is the initial object of the slice category $(Id \downarrow b)$, where η_b is the unique morphism from i to b.

One consequence of this general categorical result is that $(\mathscr{I}, B : \mathscr{I} \to \mathscr{B})$ is the initial object of the category $\mathscr{STS}[\Gamma] \gg \mathscr{B}$, where B is the unique morphism from $\mathscr{I}$ to $\mathscr{B}$.

The main advantage of this construction resides in its flexibility: there is a great variety of possibilities as to the choice of the model of observations. However, even though it is sound from a mathematical point of view to do so, we do not use any structured transition system as the model of observations. Our choice for the model of observations will be restricted to structured transition systems where the algebraic structure has been suitably interpreted to embody the chosen notion of what can be observed. This approach means that we can change the notion of observability just by changing the interpretation of the operations in the model of observations. In other words, the observation of computations is not forced by the STS presentation.

More specifically, we require that the carrier of the model of observations will include a subset $*$ of elements of type `trans`; we also require that the operations yielding elements of type `trans` or `comp` are strict on this subset. The idea is that the elements in $*$ indicate the occurrence of an error; hence the operations when applied to errors always return errors. In other words, a computation is labelled with an element not in $*$ if and only if it is a legitimate computation with respect to the chosen notion of observability.

Definition 11. (Model of Observations) Let $\mathscr{O}$ be an object of $\mathscr{STS}[\Gamma]$, and let O be the unique morphism from $\mathscr{I}$ to $\mathscr{O}$.
$\mathscr{O}$ is a model of observations provided that the following hold:

— The carrier of $\mathscr{O}$ includes a subset $*$ of elements of type `trans`, such that the operators yielding elements of type `trans` or `comp` are strict on $*$.

https://doi.org/10.1017/S0960129597002284 Published online by Cambridge University Press

— There is no junk and no confusion on types: if $O(e) : O(b)$, there are some e', b' such that $e' : b'$ and $O(e') = O(e)$, $O(b) = O(b')$ (no junk), and if b is on the right hand side of a typing relation in Γ and $O(e) = O(b)$, then $e = b$ (no confusion).

In the theory of algebraic specifications, the *no junk* condition means that every data can be constructed using only the operations in the signature. Sometimes this property is also known under the name of *reachability*. *No confusion* means that two data can be identified only if the equality can be proved by using the equations (of the presentation). Hence, the second condition ensures that there is a bijection between the sets of elements that appear on the right-hand side of a typing relation in the initial model and in the model of the observations.

We now give some examples of models of observations for the category $\mathscr{STS}[\mathscr{CCS}]$. These examples will also allow us to discuss some of the problems that arise in observational semantics for process description languages. The semantics of process description languages has been developed following two main alternative lines, often referred to as the *interleaving* and the *true concurrency* approach.

In the interleaving approach (Milner 1989) the observer interacts with a machine viewed as a black-box. The machine is equipped with a button for each basic experiment or action. The observer performs the experiment by pushing the corresponding button. Observable behaviours are described in terms of which buttons can be pushed in which order. The main consequence is that there is no visible difference between an action performed by a sequential machine and the same action performed by a system of parallel machines. This assumption is the basis of the interleaving semantics: the parallel execution of actions is represented by their non-deterministic interleaving, so that a parallel machine can always be simulated by a sequential but non-deterministic machine.

As a consequence, strings of actions are the main tool used to describe behaviour. This has the advantage of giving algebras of observations that are easy to understand and manage. However, a total ordering between causally and/or spatially independent actions is imposed. This is the main criticism made by those who favour the true concurrency approach: the peculiar features of concurrent/distributed systems are not faithfully represented.

Instead, *true concurrency* models (*e.g.*, Boudol and Castellani (1989), Degano and Montanari (1987), Mazurkiewicz (1989), Nielsen *et al.* (1981), Pratt (1986), and Reisig (1985)) describe the behaviour of distributed systems in terms of the events they may perform and the constraints on their occurrence: a partial ordering represents the causal dependencies among events, while concurrency is represented by the absence of ordering. True concurrency models provide a faithful account of distributed computations, but they are more complex to describe within an algebraic framework. For instance, there is no obvious operation of sequential composition of partial orders that preserves all the information on causal dependencies, except for the special case of series parallel pomsets (Gisher 1988). This means that observation mechanisms for such models usually have a *global* nature. They describe the global outcome of computations in a way that is not *incremental*, that is, the observation of a computation is not obtained by composing more elementary observations.

https://doi.org/10.1017/S0960129597002284 Published online by Cambridge University Press

It should be remarked that since the operation of sequential composition is one of the basic operations of Structured Transition Systems, all the models of observations we will introduce here have the property of describing computations incrementally.

In the following we introduce two different models of observation for CCS. The first model allows us to define the classical interleaving semantics of CCS, while the second handles truly concurrent observations. These models of observations are obtained by introducing a set of elements (the carrier of the algebra) and by interpreting the operations of the presentation of structured transition systems for CCS on that set of elements. Alternatively, the clauses expressing the interpretation of the operators can be understood as equations of equational type algebras. Thus the model of observations can be viewed as the initial object of the extended presentation obtained by adding these equations to the original presentation of the category $\mathscr{STS}[\mathscr{CCS}]$. The first model of observations is based on the simple algebra of strings: the concatenation of strings gives sequential composition (*i.e.*, it is incremental). The second model is more complex and is based on the *Algebra of Concatenable Concurrent Histories* (Ferrari *et al.* 1991). Concatenable Concurrent Histories are essentially labelled partial orders of events enriched with origins and destination processes that allow sequential composition to be defined. Hence, Concatenable Concurrent Histories provides an algebra of truly concurrent observations where computations are described incrementally.

3.1. *An interleaving model of observations*

In this section we define the interleaving model of observations $\mathscr{M}$. This is done by chosing an object in the category $\mathscr{STS}[\mathscr{CCS}]$ that has just one element of type `state`, denoted by **1**, whose transitions are actions, and for which the computations not in $*$ are strings over $\Delta \cup \overline{\Delta} \cup \{\tau\}$. In the following we will use $\circ$ to indicate the operation of concatenation. The set $*$ has a single element, also, for simplicity, called $*$. The interpretation of the operations of the model of observations $\mathscr{M}$ are given in Table 3.

Theorem 12. (Soundness of $\mathscr{M}$)

(a) $\mathscr{M}$ is an object of the category $\mathscr{STS}[\mathscr{CCS}]$.
(b) $\mathscr{M}$ is a model of observations.

Proof.
(a) $\mathscr{M}$ satisfies the presentation $\mathscr{CCS}$ since it is obtained just by adding the judgments giving the interpretation of the operations.

(b) The judgments that are added to the original presentation do not modify the typing relation. Furthermore, the judgments of $\mathscr{M}$ ensure the strictness condition of the operation with respect to $*$. □

We can now apply the labelling construction, thus obtaining the category of the structured transition systems satisfying the STS presentation of CCS with observations in $\mathscr{M}$, that is, the category $\mathscr{STS}[\mathscr{CCS}] \gg \mathscr{M}$. The pair $(\mathscr{I}_{\mathscr{CCS}}, M : \mathscr{I}_{\mathscr{CCS}} \to \mathscr{M})$ is the initial object of this category.

https://doi.org/10.1017/S0960129597002284 Published online by Cambridge University Press

$$\frac{e : (X, Y)}{e = \mathbf{1}_{X,Y}}$$

$$\mathbf{1}_{\phi,\phi} = \mathbf{1}$$

$$* : \mathbf{1} \to \mathbf{1}$$

$$[\mu, u\rangle = \mu$$

$$\mu\backslash\alpha = \text{ if } \mu \neq \{\alpha, \overline{\alpha}\} \text{ then } \mu \text{ else } *$$

$$*\backslash\alpha = *$$

$$\mu[\Phi] = \Phi(\mu)$$

$$*[\Phi] = *$$

$$t \lhd u = t$$

$$u \rhd t = t$$

$$t \rfloor u = t$$

$$u \lfloor t = t$$

$$\lambda_1 \mid \lambda_2 = \text{ if } \lambda_1 = \overline{\lambda_2} \text{ then } \tau \text{ else } *$$

$$t \mid * = * = * \mid t$$

$$c; c' = \text{ if } c \text{ and } c' \neq * \text{ then } c \circ c' \text{ else } *$$

Table 3. *The interleaving model* $\mathscr{M}$

Let u : `state` be an element of $\mathscr{I}_{\mathscr{CCS}}$. Clearly, $M(u) = \mathbf{1}$, that is, any agent is seen as a global state. This is a known property of the interleaving approach: the (distributed) structure of states is irrelevant and processes evolve through sequences of global states.

Let us now consider the transition $t = [\alpha, u\rangle \rfloor v : \alpha.u \mid v \to u \mid v$ of $\mathscr{I}_{\mathscr{CCS}}$. The transition describes a move of a process within a context with some idle subsystem v on the right. Evaluating the transition in $\mathscr{M}$ we have

$$M(t) = \alpha.$$

As expected, the observation of the transition t is just the action α. The interleaving interpretation forgets about the physical distribution of agents: an external observer cannot see any difference between an action performed by a sequential agent and the same action performed by a system composed by several parallel agents. For instance, the observation of the transition $[\alpha, u\rangle$ is again the action α.

The model $\mathscr{M}$ characterizes the standard interleaving operational semantics of CCS. As an example we have that $M(([\alpha, nil\rangle)\backslash\alpha) = *$. Any attempt to perform an α-move within a context where α-moves are not allowed yields an error. On the other hand, $M(([\beta, nil\rangle)\backslash\alpha) = \beta$. Finally, the interpretation of the synchronization follows the laws of CCS synchronization algebra. Recall that in CCS only transitions performing comple-

https://doi.org/10.1017/S0960129597002284 Published online by Cambridge University Press

mentary actions may synchronize. In fact, because of the axioms $M([\alpha, nil\rangle \mid [\overline{\alpha}, nil\rangle) = \tau$, while $M([\alpha, nil\rangle \mid [\beta, nil\rangle) = *$.

Remark 13. An important issue of the operational semantics of process description languages that is naturally handled by STS concerns the side conditions of the inference rules. As an example, consider the inference rule of the CCS restriction operator

$$\frac{E \xrightarrow{\mu} E'}{E\backslash\alpha \xrightarrow{\mu} E'\backslash\alpha} \quad \mu \notin \{\alpha, \overline{\alpha}\}.$$

In the STS setting, this side condition is internalized in the definition of the model of the observation. For instance, the elements of type `trans`

$$[\alpha, nil\rangle\backslash\alpha : \alpha.nil\backslash\alpha \longrightarrow nil\backslash\alpha \qquad [\beta, nil\rangle\backslash\alpha : \beta.nil\backslash\alpha \longrightarrow nil\backslash\alpha$$

under the interpretation of the model $\mathscr{M}$ become

$$M(([\alpha, nil\rangle)\backslash\alpha) = * \text{ and } M(([\beta, nil\rangle)\backslash\alpha) = \beta.$$

The role played by the set of distinguished constants $*$ should now be clearer: each transition labelled with an element in the set $*$ is a 'gedanken' transition whose applicability conditions have not been satisfied.

Clearly, the satisfaction of the applicability conditions depends on the actual definition of the model of observations. Notice that we require that the operations of the model of observations are strict over the set $*$ meaning that a gedanken transition cannot be transformed into a real transition. This property will always hold in any model of observations we consider in this paper.

Remark 14. The initial object of the category $\mathscr{STS}[\mathscr{CCS}] \gg \mathscr{M}$ does not correspond to the labelled transition system of CCS as defined by Milner (Milner 1989). As an example, transitions $[\alpha, nil\rangle \lhd \alpha.nil$ and $\alpha.nil \rhd [\alpha, nil\rangle$ evaluate to α in $\mathscr{M}$ and they *both* correspond to the same CCS derivation

$$\alpha.nil + \alpha.nil \xrightarrow{\alpha} nil.$$

However, Milner's transition system is an object of the category $\mathscr{STS}[\mathscr{CCS}] \gg \mathscr{M}$: the *Standard Model* (Ferrari 1990). Roughly speaking, the standard model is obtained by making the quotient of the initial model with respect to a congruence that identifies the transitions having the same source, the same target and the same label.

Example 15. Let us consider a distributed system with two autonomous processes P and Q that may progress independently or communicate with an arbiter A. This system can be specified in CCS by the the agent $(P \mid A \mid Q)\backslash\alpha$ where

$$P = rec\ x.\alpha.x + \lambda.x$$
$$A = rec\ x.\overline{\alpha}.x$$
$$Q = rec\ x.\alpha.x + \rho.x\ .$$

The term $c = t_1; t_2; t_3; t_4$ where

$$t_1 = ((\alpha.P \rhd [\lambda, P\rangle)\rfloor A \mid Q)\backslash\alpha \qquad t_2 = (P \lfloor [\overline{\alpha}, A\rangle \mid [\alpha, Q\rangle \lhd \rho.Q)\backslash\alpha$$

$$t_3 = ([\alpha, P\rangle \lhd \lambda.P \mid [\overline{\alpha}, A\rangle\rfloor Q)\backslash\alpha \quad t_4 = (P \mid A\lfloor\alpha.Q \rhd [\rho, Q\rangle)\backslash\alpha$$

https://doi.org/10.1017/S0960129597002284 Published online by Cambridge University Press

describes a computation of the system where the two processes communicate with the arbiter. In this computation, while one process is communicating with the arbiter, the other process performs a local operation. The observation in $\mathscr{M}$ of the computation c is the string $\lambda\tau\tau\rho$. In the observation there is no way of detecting the parallelism between the local actions or the synchronizations.

3.2. *Observing causality, incrementally*

In this section, we define an observation mechanism for CCS that takes causal information into account. This is done by choosing as the model of observations an object of the category $\mathscr{SIS}[\mathscr{CCS}]$ for which the computations that are not mapped into $*$ have a partial ordering structure. For the following definitions we fix two nonintersecting alphabets P, A.

Definition 16. (Concurrent Histories (Degano and Montanari 1987)) Concurrent Histories are labelled partial orders $(V, \leqslant, \ell)$ where the labelling function $\ell : V \to P \cup A$ sends the set of maximal and minimal elements (of the partial order $(V, \leqslant)$) to P and the set of other elements to A. Concurrent Histories are considered up to label preserving partial ordering isomorphisms.

The elements of V with labels in P are called *processes* and those with labels in A are called *events*.

Definition 17. (Label Indexed Ordering Functions) Suppose S is a set with a labelling function $\ell : S \to P$. A label indexed ordering function on the labelled set S is a function α from S to the set of natural numbers such that for each $p \in P$ the restriction of α to the set of elements labelled p is a bijection onto the set $\{1, 2, \ldots, n_p\}$ where $n_p = |\{s \in S : \ell(s) = p\}|$.

Definition 18. (Concatenatable Concurrent Histories) A Concatenable Concurrent History is a triple (h, β, γ) where $h = (V, \leqslant, \ell)$ is a Concurrent History for which no element is both maximal and minimal, and β, γ are label indexed ordering functions on the labelled sets of minimal and maximal elements of V (called *origins* and *destinations*), respectively. Concatenable Concurrent Histories are defined up to isomorphisms of labelled partial orders that preserve the label indexed ordering functions.

The introduction of the label indexed ordering functions allows us to discriminate between different maximal elements (and minimal elements) with the same label. Figure 1 illustrates two concatenable concurrent histories ch_1 and ch_2((a) and (b)); the order relation is depicted through its Hasse diagram growing downwards. Processes (respectively, events) are represented as circles (boxes): all processes have the same label. Finally, the label indexed ordering functions are represented by positive numbers on processes.

The algebra of Concatenable Concurrent Histories has two operations: parallel composition and sequential composition. Let $ch_1 = (h_1, \beta_1, \gamma_1)$ and $ch_2 = (h_2, \beta_2, \gamma_2)$ be Concatenable Concurrent Histories, where $h_1 = (\mathrm{V}_1, \leqslant_1, \ell_1)$ and $h_2 = (\mathrm{V}_2, \leqslant_2, \ell_2)$. Without loss of generality, V_1 and V_2 are disjoint. Let $\mathrm{Min}(h_1)$, $\mathrm{Max}(h_1)$, $\mathrm{Min}(h_2)$, $\mathrm{Max}(h_2)$ be the origins and destinations of h_1 and h_2, respectively.

https://doi.org/10.1017/S0960129597002284 Published online by Cambridge University Press

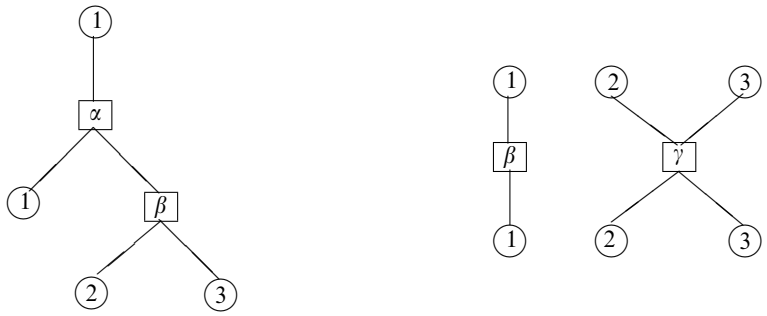


Fig. 1. Two concatenable concurrent histories

Definition 19. (The Algebra of Concatenable Concurrent Histories) The parallel composition $ch_1 \otimes ch_2$ is the history $((V, \leqslant, \ell), \beta, \gamma)$ where

1 $V = V_1 \cup V_2$
2 $\leqslant$ is $\leqslant_1 \cup \leqslant_2$
3 $\ell(v)$ is $\ell_1(v)$ if $v \in V_1$ and is $\ell_2(v)$ if $v \in V_2$
4 $\beta(v)$ is $\beta_1(v)$ if $v \in \text{Min}(h_1)$ and is $n_1(v) + \beta_2(v)$ if $v \in \text{Min}(h_2)$, where $n_1(v)$ is the number of origins of ch_1 with label equal to $\ell_2(v)$
5 $\gamma(v)$ is $\gamma_1(v)$ if $v \in \text{Max}(h_1)$ and is $m_1(v) + \gamma_2(v)$ if $v \in \text{Max}(h_2)$, where $m_1(v)$ is the number of destinations of ch_1 with label equal to $\ell_2(v)$.

The sequential composition $ch_1 \odot ch_2$ is defined if and only if $\ell_1(Max(h_1)) = \ell_2(Min(h_2))$, intended as multisets. When defined, the operation of sequential composition yields the history $((V, \leqslant, \ell)\ \beta, \gamma)$ where

1 $V = V_1 \cup V_2 \setminus (\text{Max}(h_1) \cup \text{Min}(h_2))$
2 $\leqslant$ is the restriction to $V \times V$ of the transitive closure of
$\leqslant_1 \cup \leqslant_2 \cup \{(w, v) : w \in \text{Max}(h_1), v \in \text{Min}(h_2), \ell_1(w) = \ell_2(v), \gamma_1(w) = \beta_2(v)\}$
3 $\ell(v)$ is $\ell_1(v)$ if $v \in V_1$ and is $\ell_2(v)$ if $v \in V_2$
4 $\beta(v) = \beta_1(v)$ and $\gamma(v) = \gamma_2(v)$.

Figure 2 shows the result of the parallel composition $ch_1 \otimes ch_2$ (a) and the result of the sequential composition $ch_1 \odot ch_2$ (b) of the two histories depicted in Figure 1(a) and 1(b). Notice that $\otimes$ is associative but not commutative. In fact, because histories describe the causal relations between events performed during a computation, $\otimes$ cannot be commutative, otherwise the operation of sequential composition would not preserve the information on causal dependencies. As an illustration of this fact, let h_μ be the Concatenable Concurrent History with three linearly ordered elements, the middle one of which is labelled μ. Then in the Concatenable Concurrent History $(h_\alpha \otimes h_\beta) \odot (h_\gamma \otimes h_\delta)$ the event labelled α causes the event labelled γ, but in $(h_\beta \otimes h_\alpha) \odot (h_\gamma \otimes h_\delta)$ this is not the case.

https://doi.org/10.1017/S0960129597002284 Published online by Cambridge University Press

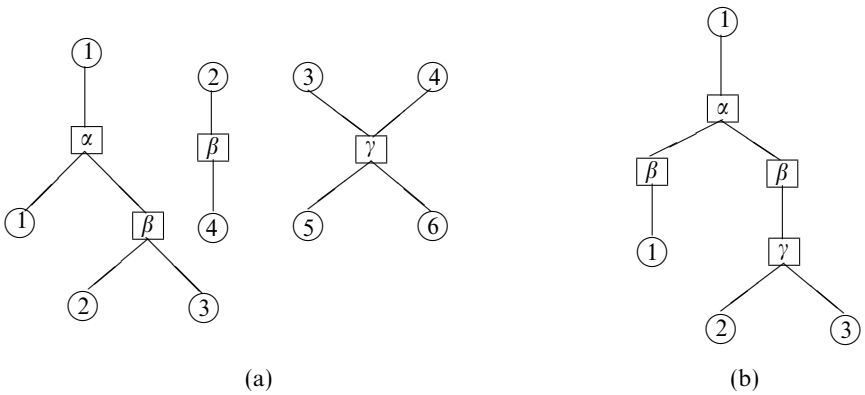


Fig. 2. Parallel (a) and sequential composition (b) of the histories of Figure 1

We now consider the algebra of Concatenable Concurrent Histories with the alphabet P of processes being a singleton, and the alphabet A of events being the set $\Lambda \cup \{\tau\}$ of actions. We say that a history is *bipartite* if all its elements are either minimal or maximal. We write $idle_1$ for the bipartite history with just two elements and $idle_n$ for the parallel composition of n copies of this. We write $\wedge_n$ for the bipartite history with $n+1$ elements, n of which are maximal, and $\vee_n$ for the bipartite history with $n+1$ elements, n of which are minimal. As in the last paragraph we write h_μ for the history with three linearly ordered elements where the middle one is labelled μ.

We define the model of observation $\mathscr{H}$. The set of elements of $\mathscr{H}$ of type `comp` is the disjoint union of the algebra of Concatenable Concurrent Histories with $*$. There is one element $*_{n,m} : [\mathbf{n}] \to [\mathbf{m}]$ of $*$ for each pair of natural numbers n, m. We extend the operations $\otimes, \odot$ to the set of elements of type `comp`, in such a way that the set $*$ is absorbent for the operations of parallel and sequential composition. Finally, with $\Phi(h)$ we indicate the history that is obtained from h by relabelling each event labelled μ with $\Phi(\mu)$, and with $\pi[h_\mu]$ we indicate the set of histories of the form $\eta \odot (idle_k \otimes h_\mu) \odot \gamma$, where η, γ are bipartite histories. Ferrari *et al.* (1991) proved that any history with one element can always be reduced to a term with such form.

Definition 20. (The Model of Observations $\mathscr{H}$) The interpretation of the operations of the model $\mathscr{H}$ of causal observations is given in Table 4. The interpretation makes use of the function $N(e)$, which is defined as follows:

— $\mathrm{N}(\mathrm{nil}) = \mathrm{N}(\mathrm{x}) = \mathrm{N}(\mu.e) = \mathrm{N}(e + e') = 1$,
— $\mathrm{N}(e[\Phi]) = \mathrm{N}(e\backslash\alpha) = \mathrm{N}(e)$,
— $\mathrm{N}(e \mid e') = \mathrm{N}(e) + \mathrm{N}(e')$.

We can comment briefly on the interpretation of the operators. The intuition behind the interpretation of the elements of type `state` is that the number associated with an element is the number of sequential components of the element that can be considered autonomous processes (that is, it measures the amount of top-level paral-

https://doi.org/10.1017/S0960129597002284 Published online by Cambridge University Press

$$\frac{e : (X, Y)}{e = [X, Y, N(e)]}$$

$$[\phi, \phi, \mathbf{n}] = [\mathbf{n}]$$

$$*_{m,n} : [\mathbf{m}] \to [\mathbf{n}]$$

$$[\mu, u\rangle = h_\mu \odot \wedge_k, \text{where } N(u) = [\mathbf{k}]$$

$$\frac{t \notin \pi[h_\alpha] \text{ and } t \notin \pi[h_{\bar{\alpha}}]}{t \backslash \alpha = t}$$

$$\frac{t : [\mathbf{m}] \to [\mathbf{n}], t \in \pi[h_\alpha] \text{ or } t \in \pi[h_{\bar{\alpha}}]}{t \backslash \alpha = *_{m,n}}$$

$$\frac{t : [\mathbf{m}] \to [\mathbf{n}], t \neq *_{m,n}}{t[\Phi] = \Phi(t)}$$

$$*_{m,n}[\Phi] = *_{m,n}$$

$$\frac{t : [\mathbf{m}] \to [\mathbf{n}]}{t \lhd u = \wedge_m \odot t}$$

$$\frac{t : [\mathbf{m}] \to [\mathbf{n}]}{u \rhd t = \wedge_m \odot t}$$

$$\frac{t : [\mathbf{m}] \to [\mathbf{n}], N(u) = k}{t \rfloor u = t \otimes idle_k}$$

$$\frac{t : [\mathbf{m}] \to [\mathbf{n}], N(u) = k}{u \lfloor t = idle_k \otimes t}$$

$$\frac{t_1 \in \pi[h_\lambda], t_1 = (\eta_1 \odot (idle_{k_1} \otimes h_\lambda) \odot \gamma_1), t_2 \in \pi[h_{\bar{\lambda}}], t_2 = (\eta_2 \odot (h_{\bar{\lambda}} \otimes idle_{k_2}) \odot \gamma_2)}{t_1 \mid t_2 = (\eta_1 \otimes \eta_2) \odot (idle_{k_1} \otimes (\vee_2 \odot h_\tau \odot \wedge_2) \otimes idle_{k_2}) \odot (\gamma_1 \otimes \gamma_2)}$$

$$\frac{t_i : [\mathbf{m}_i] \to [\mathbf{n}_i], (i = 1, 2), \nexists \lambda \text{ such that } t_1 \in \pi[h_\lambda], t_2 \in \pi[h_{\bar{\lambda}}]}{t_1 \mid t_2 = *_{m_1+m_2, n_1+n_2}}$$

$$\frac{c : [\mathbf{m}] \Rightarrow [\mathbf{n}], c' : [\mathbf{n}] \to [\mathbf{p}]}{c; c' = c \odot c'}$$

Table 4. *The model of observations* $\mathscr{H}$

lelism). Notice that applying the nondeterministic choice operator gives a *global state*, that is, a state with just one autonomous component. This assumption corresponds to having a centralized mechanism to deal with non-deterministic choices (Degano *et al.* 1990).

The interpretation of the operator $[\mu, u\rangle$ expresses the fact that after the action μ a *fork* operation is performed, making explicit the distributed structure of state u. Moreover, the non-deterministic operation first requires an action of choice between the two alternatives, and then the execution of the chosen alternative. Finally, in the interpretation of the

https://doi.org/10.1017/S0960129597002284 Published online by Cambridge University Press

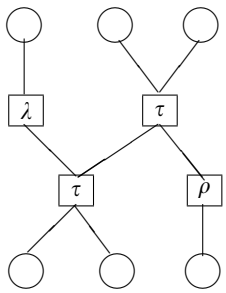


Fig. 3. The graphical representation of the term $(h_\lambda \otimes (\vee_2; h_\tau; \wedge_2)); ((\vee_2; h_\tau; \wedge_2) \otimes h_\rho)$

synchronization operator, we have adopted a normal form of computations like the one introduced in Gorrieri and Montanari (1990).

In the following, instead of writing $h_1 \odot h_2$ for the sequential composition of histories we will write $h_1; h_2$. Using similar arguments as in the proof of Theorem 12, we can prove that $\mathscr{H}$ is a model of observations.

Theorem 21. (Soundness of $\mathscr{H}$)

(a) $\mathscr{H}$ is an object of the category $\mathscr{STS}[\mathscr{CCS}]$.
(b) $\mathscr{H}$ is a model of observations.

By applying the labelling construction, we obtain the category $\mathscr{STS}[\mathscr{CCS}] \gg \mathscr{H}$ of structured transition systems satisfying the STS presentation of CCS with observations in $\mathscr{H}$. The pair $(\mathscr{I}_{\mathscr{CCS}}, H : \mathscr{I}_{\mathscr{CCS}} \to \mathscr{H})$ is the initial object of this category.

The observation in $\mathscr{H}$ of the transition $[\alpha, nil\rangle \rfloor \beta.nil$ is the history $h_\alpha \otimes idle_1$; the observation of the computation $[\alpha, nil\rangle \rfloor \beta.nil; nil \lfloor [\beta, nil\rangle$ is the history $h_\alpha \otimes h_\beta$. Finally, the observation of the computation of the Example 15 is the history

$$(h_\lambda \otimes (\vee_2; h_\tau; \wedge_2)); ((\vee_2; h_\tau; \wedge_2) \otimes h_\rho).$$

This history is represented pictorially in Figure 3. The labelled indexed ordering functions are the obvious ones, so we do not represent the numbering on processes.

Example 22. Consider the CCS agent $\mathrm{E} = (\alpha.nil \mid \beta.nil) \mid ((\overline{\alpha}.nil \mid \delta.nil) + \gamma)$. The synchronization between α and $\overline{\alpha}$ is represented in $\mathscr{H}$ by

$$(\bowtie; (idle_1 \otimes h_\alpha); \bowtie) \mid (\wedge_2; (h_{\overline{\alpha}}; idle_1)) = (\bowtie \otimes \wedge_2); (idle_1 \otimes (\vee_2; h_\tau; \wedge_2) \otimes idle_1); (\bowtie \otimes idle_2)$$

where $\bowtie$ has two origins p_1 and p_2, two destinations q_1 and q_2, and no events, with $p_1 \leqslant q_2$, $p_2 \leqslant q_1$, $\beta(p_i) = \gamma(q_i)$, $i = 1, 2$. See Figure 4 for a pictorial representation.

3.3. *Internal moves*

The models of observations we have defined in the previous section treat all actions as observable. A further abstraction making some actions invisible to the external observer is possible. The standard example of abstraction from internal moves is provided by Milner's τ action. The idea is that τ-actions represent an internal activity of the system that does not affect the observable behaviour.

https://doi.org/10.1017/S0960129597002284 Published online by Cambridge University Press

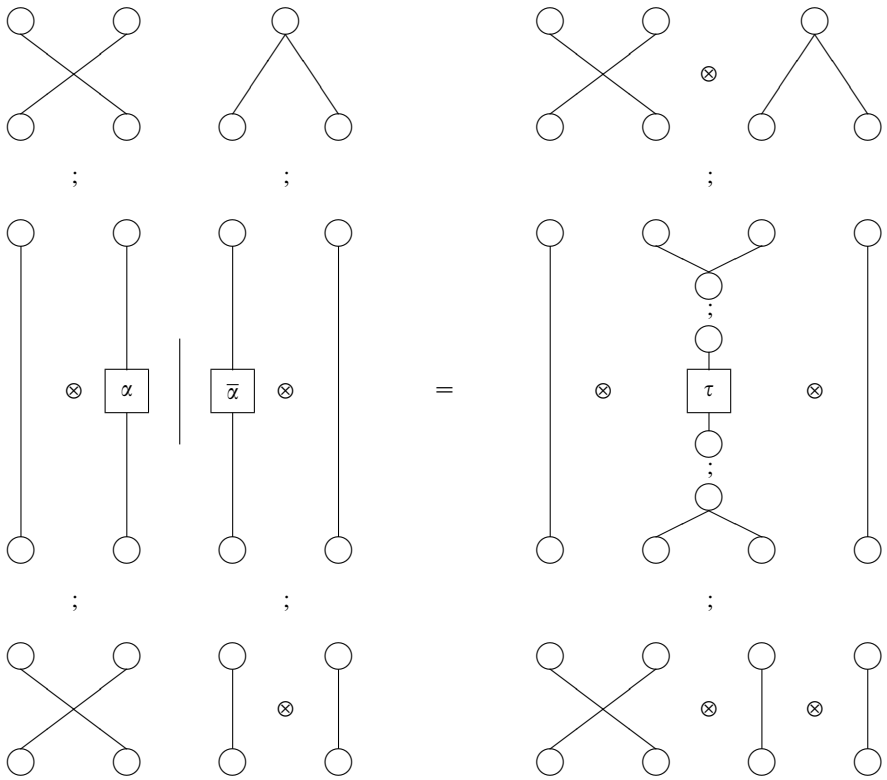


Fig. 4. A graphical representation of the observation for the synchronization of transitions in the CCS agent $(\alpha.nil \mid \beta.nil) \mid ((\bar{\alpha}.nil \mid \delta.nil) + \gamma)$

In our framework the abstraction that makes some moves internal is obtained by imposing axioms that express the *invisibility* of some transitions. Such transitions can be removed from terms describing behaviour.

Definition 23. The model of observations $\mathscr{M}_\tau$ is obtained by adding to the definition of $\mathscr{M}$ the equation $\tau = \epsilon$, where ϵ is the empty string.

Definition 24. The model of observations $\mathscr{H}_\tau$ is obtained by adding to the definition of $\mathscr{H}$ with the equation $h_\tau = idle_1$.

The extra equations imposed on the models ensure that in $\mathscr{M}_\tau$ each computation not in $*$ is equal to a unique string over Λ, and that in $\mathscr{H}_\tau$ each computation not in $*$ is equal to a unique Concatenable Concurrent History for which all the events are labelled in Λ.

Theorem 25. (Soundness)

(a) $\mathscr{M}_\tau$ is an object of the category $\mathscr{STS}[\mathscr{CCS}]$, and is a model of observations.
(b) $\mathscr{H}_\tau$ is an object of the category $\mathscr{STS}[\mathscr{CCS}]$, and is a model of observations.

https://doi.org/10.1017/S0960129597002284 Published online by Cambridge University Press

Notice that in each of these models of observations, the element $[\alpha, \tau.nil\rangle ; [\tau, nil\rangle$ is equal to the element $[\alpha, nil\rangle$ (that is, they are equal to the string α or the history h_α), and therefore has type `trans`, whereas in the initial algebra $[\alpha, (\tau.nil)\rangle ; [\tau, nil\rangle$ does not have type `trans`.

3.4. *A range of observational semantics*

We have already remarked that, depending on the application, different aspects of behaviour may be important. For instance, at the user level one is typically interested in seeing that the system performs certain operations, but not in whether these are performed sequentially or in parallel. So interleaving observations provide enough information. In contrast, an observational semantics based on the algebra of concatenable concurrent histories can be used profitably at the design level. For instance, if we imagine deriving the design of a distributed system from the system specification, we could be interested in having some measure of the degree of parallelism of the specification in order to define the architecture of the system. Information about the distributed structure of the specification is available in the case of the observation mechanism using concatenable concurrent histories.

In the algebraic framework we propose here, the observable behaviour of computations is described by defining a suitable model of observations. We change the notion of observability just by changing the model of observations, *i.e.*, by suitably interpreting the operations to embody the notion of observability that is needed for the application at hand. Therefore, the full power of the framework can be exploited to deal with a wide range of possible observations.

In our running example, the model $\mathscr{M}$ gives the standard interleaving semantics of CCS. However, this proximity is not forced by the presentation. For instance, we can easily introduce a CSP like synchronization mechanism just by modifying the judgment concerning the operation of synchronization on transitions other than $*$ as follows:

$$\lambda_1 \mid \lambda_2 = \text{ if } \lambda_1 = \lambda_2 \text{ then } \lambda_1 \text{ else } * .$$

Hence, the observation of $[\alpha, nil\rangle \mid [\alpha, nil\rangle$ will be the action α, and the observation of $[\alpha, nil\rangle \mid [\bar{\alpha}, nil\rangle$ will be $*$.

Furthermore, we can consider a model of observations that has a Petri net structure along the lines of Gorrieri and Montanari (1990). Of course other observational semantics can be defined.

The definition of a range of observational semantics for the same category of structured transition systems can be completely formulated in the language of category theory by resorting to the general *comma category* construction (Mac Lane 1971).

Definition 26. Let $F : \mathbf{A} \to \mathbf{C}$ and $G : \mathbf{B} \to \mathbf{C}$ be functors. The comma category $(F \downarrow G)$ is the category whose objects are triples

$$(a, f : F(a) \to G(b), b)$$

where a is an object of $\mathbf{A}$, b is an object of $\mathbf{B}$ and f is a morphism of $\mathbf{C}$.

https://doi.org/10.1017/S0960129597002284 Published online by Cambridge University Press

The pair (h,k), with $h : a \to a'$ is a morphism of **A** and $k : b \to b'$ is a morphism of **B**, is an arrow from $(a, f : F(a) \to G(b), b)$ to $(a', f' : F(a') \to G(b'), b')$ provided that $f \circ G(k) = F(h) \circ f'$.

The machinery of category theory can be used to create a uniform framework in which different observational semantics of the same process language can be compared. For example, let $\mathscr{STS}[\Gamma]$ be a category of structured transition systems and let $\mathscr{O}_i$, with $i \in \{1,\ldots,n\}$, be a finite family of models of observations such that for each j, $j \in \{1, n-1\}$ there exists a morphism $\eta_j : \mathscr{O}_j \to \mathscr{O}_{j+1}$. We now consider the comma category $(Id \downarrow N)$ where Id is the identity functor on $\mathscr{STS}[\Gamma]$ and N is the functor from the discrete category **n** of the first n natural numbers to $\mathscr{STS}[\Gamma]$ such that $N(i) = \mathscr{O}_i$. The objects of this category are pairs of the form $(\ell_i : \mathscr{T} \to \mathscr{O}_i)$: each such pair describes the i^{th} observational semantics of the STS presentation $\mathscr{T}$. A morphism from the i^{th} to the j^{th} observational semantics of $\mathscr{T}$ is just an arrow $\eta_{i,j} : \mathscr{O}_i \to \mathscr{O}_j$ such that $\ell_i \circ \eta_{i,j} = \ell_j$. It is easy to see that this equation states the compatibility between the two observational semantics.

4. Parametric bisimulation semantics

In this section, we introduce a notion of bisimulation parameterized with respect to the chosen notion of observability.

Let $\mathscr{STS}[\Gamma] \gg \mathscr{O}$ be a category of structured transition systems with observations in $\mathscr{O}$, and let $(\mathscr{I}, O)$ be the initial object of this category.

Definition 27. (Parametrized Bisimulation) A binary symmetric relation R over the elements of type `state` of $\mathscr{I}$ is called a bisimulation if it satisfies:

(i) whenever $u_1 R u_2$, then $O(u_1) = O(u_2)$;
(ii) whenever $u_1 R u_2$ and $e_1 : u_1 \Rightarrow v_1$, $O(e_1) : \texttt{trans}$, $O(e_1) \notin *$, there exists $e_2 : u_2 \Rightarrow v_2$ such that $O(e_1) = O(e_2)$ and $v_1 R v_2$

Two elements u and v of type `state` are bisimilar with respect to $\mathscr{O}$, notation $u \sim_{\mathscr{O}} v$, if there exists a bisimulation containing the pair (u,v).

The following properties of the relation $\sim_{\mathscr{O}}$ can be proved using standard arguments.

Proposition 28. The relation $\sim_{\mathscr{O}}$ is a bisimulation.

Proposition 29. The relation $\sim_{\mathscr{O}}$ is an equivalence relation.

Parameterized bisimulation is defined only on the elements of type `state` of the initial STS. However, parameterized bisimulation can be defined over arbitrary models of a given STS presentation. Basically, this leads to a bisimulation semantics that takes into account the *structural congruence* induced by the unique morphism from the initial STS.

Remark 30. The general principle of bisimulation is that every experiment carried on by one process must be matched by an equivalent experiment of the other process: *the transfer property*. In our case, the transfer property is defined in terms of the observability of elements of type `comp` in the model of the observation $\mathscr{O}$ so that an experiment observed as a transition can be matched by a computation. It also requires that equivalent states have the same structure, *i.e.*, they are labelled with the same state in $\mathscr{O}$. This allows us

https://doi.org/10.1017/S0960129597002284 Published online by Cambridge University Press

to handle in a uniform way both strong and weak bisimulations without introducing a special definition for the weak case. Moreover, as we will prove later, a 'bisimulation as a congruence theorem' holds, namely parameterized bisimulation is always a congruence. This would not be the case if we had considered the definition of 'weak' bisimulation as given in Milner (1989).

We now consider bisimulation equivalences for our running example $\mathscr{STS}[\mathscr{CCS}]$. Let us consider the category $\mathscr{STS}[\mathscr{CCS}] \gg \mathscr{M}$. It is easy to see that the elements $\alpha.nil + \alpha.nil$ and $\alpha.nil$ are related by $\sim_{\mathscr{M}}$. Indeed, for any element u of type `state` we have that $u + u \sim_{\mathscr{M}} u$. It is easily shown that relation $\sim_{\mathscr{M}}$ coincides with Milner's Strong Observational Equivalence (see Ferrari (1990) and Ferrari and Montanari (1990)).

We now consider the category $\mathscr{STS}[\mathscr{CCS}] \gg \mathscr{M}_\tau$. We have that $\alpha.nil + \tau.\alpha.nil$ and $\tau.\alpha.nil$ are related by $\sim_{\mathscr{M}_\tau}$. In fact the transition $[\alpha, nil\rangle$ can be simulated by the computation $[\tau, \alpha.nil\rangle ; [\alpha, nil\rangle$: in the model of the observation $\mathscr{M}_\tau$ they are both observed as the element α of type transition. Using a similar argument, it is easy to see that $\lambda.(u + \tau.v) + \lambda.v$ and $\lambda.(u + \tau.v)$ are related by $\sim_{\mathscr{M}_\tau}$. Notice that these are the second and the third Milner's τ-laws. On the other hand, the first τ-law is not satisfied. For instance, consider the elements $\alpha.\tau.\beta.nil$ and $\alpha.\beta.nil$. After the execution of the transition $[\alpha, \tau.\beta.nil\rangle$ the first element reaches a state where the transition $[\tau, \beta.nil\rangle$ cannot be simulated by the second element. Relation $\sim_{\mathscr{M}_\tau}$ coincides with Dynamic Bisimulation Congruence (Montanari and Sassone 1992).

In the case of the category $\mathscr{STS}[\mathscr{CCS}] \gg \mathscr{H}$ we have that $\alpha.nil + \alpha.nil \sim_{\mathscr{H}} \alpha.nil$. However, the two elements $(\alpha.nil \mid \beta.nil) + (\alpha.nil \mid \beta.nil)$ and $(\alpha.nil \mid \beta.nil)$ are not identified: because of the global control mechanism for the non- deterministic choice, the first element is labelled in $\mathscr{H}$ with [**1**], while the second is labelled with [**2**].

The elements $\alpha.nil \mid nil$ and $nil \mid \alpha.nil$ are not identified, and therefore $\mid$ is not commutative with respect to $\sim_{\mathscr{H}}$. However, it is easy to convince oneself that this must be the case. In fact, assume the two states above are the intermediate states of computations starting from $\beta.\alpha.nil \mid \gamma.nil$ and $\beta.nil \mid \gamma.\alpha.nil$. Identification of these intermediate states (after the parallel execution of the actions β and γ) would make it impossible to detect the correct cause of the action α.

The two CCS agents

$$E_1 = \alpha.(\beta.nil + \gamma.nil) + \alpha.nil \mid \beta.nil$$

$$E_2 = \alpha.(\beta.nil + \gamma.nil) + \alpha.nil \mid \beta.nil + \alpha.\beta.nil$$

are indistinguishable by Pomset Bisimulation Equivalence (Boudol and Castellani 1988). However, they are distinguished by $\sim_{\mathscr{H}}$. This is because the agent E_2 can perform a transition with observation h_α ending in a state from which a transition with observation h_γ is impossible. The agent E_1 cannot do this because there is no possible transition with observation h_α from the state $\alpha.nil \mid \beta.nil$, although there is a transition with observation $h_\alpha \otimes idle_1$.

Relation $\sim_{\mathscr{H}_\tau}$ is coarser than $\sim_{\mathscr{H}}$ and satisfies the second and the third τ-laws. Another law that holds is that $(\tau.u_1) \mid u_2$ and $u_1 \mid (\tau.u_2)$ (with both u_1 and u_2 observed as [**1**]) are identified. The underlying idea is that the observer is able to detect that the internal move

https://doi.org/10.1017/S0960129597002284 Published online by Cambridge University Press

— $u \models TRUE$ for all elements u of type `state`
— $u \models S_\sigma$ if and only if $\ell_{\mathscr{I}}(u) = \sigma$
— $u \models \wedge_{i \in J} \psi_j$ if and only if $u \models \psi_i$ for all $i \in J$
— $u \models \neg\psi$ if and only if $u \not\models \psi$
— $u \models \langle t \rangle \psi$ if and only if there is some c in $\mathscr{I}$ such that $c : u \Rightarrow v$, $\ell_{\mathscr{I}}(c) = t$, and $v \models \psi$.

Table 5. *The satisfaction relation*

has taken place, but is not able to detect the location of such a move. The distribution of initial states is still observed, so that, for instance, $\tau.(\alpha.nil \mid \beta.nil)$ and $(\tau.\alpha.nil \mid \beta.nil)$ are not identified.

4.1. *Logics and bisimulations: a parametrization result*

In this subsection we give a characterization of the bisimulation relation in terms of a modal logic, along the lines of Hennessy–Milner logic (Hennessy and Milner 1985).

Definition 31. Let $\mathscr{O}$ be a model of the observations in the category $\mathscr{SIS}[\Gamma]$, define the modal logic HML($\mathscr{O}$) to be the logic with syntax

$$\psi ::= TRUE \mid S_\sigma \mid \wedge_{i \in J} \psi_i \mid \neg\psi \mid \langle t \rangle \psi$$

where J is a (possibly infinite) indexing set, σ ranges over the elements of type `state` in $\mathscr{O}$, and t ranges over the elements of type `trans` in $\mathscr{O}$ that are not in the set $*$.
The satisfaction relation $\models$ between the set of elements of $\mathscr{I}$ of type `state` and the set of formulae of the logic is defined in Table 5.

The family of atomic formulae S_σ is introduced to describe properties of the structure of states. A state satisfies the atomic formula S_σ if and only if it is labelled by the state σ in $\mathscr{O}$. States that have discernibly different structure will be labelled by different elements. In the standard formulation of HML this kind of atomic formula does not provide any discriminating power since a global state is assumed in the model of observations, *i.e.*, the formula $S_{\mathbf{1}}$ is always satisfied. In contrast, this is not the case for the truly concurrent semantics based on the algebra of Concatenable Concurrent Histories we have proposed.

Theorem 32. If e_1, e_2 are elements of $\mathscr{I}$ of type `state`, then $e_1 \sim_{\mathscr{O}} e_2$ if and only if e_1 and e_2 satisfy the same formulae in HML($\mathscr{O}$).

The proof of the proposition above is standard, so we do not give details here. For one direction of the implication it is enough to prove that bisimilar states satisfy the same set of formulae: this is done by induction on the structure of the formulae. For the other direction, it suffices to show that the equivalence induced by the logic is a bisimulation. The proof of this step uses the fact that states that satisfy the same formulae have the same observation (thanks to the family of atomic formulae S_σ).

https://doi.org/10.1017/S0960129597002284 Published online by Cambridge University Press

4.2. *Parameterized bisimulation as a congruence*

In the last few years there have been several attempts to find out which formats of the SOS inference rules ensure that bisimulation is automatically a congruence (Bloom *et al.* 1995; De Simone 1985; Baeten and Verhoef 1993; Groote 1990; Groote and Vaandrager 1992). The aim of this section is to extend such results to cover our algebraic framework. In other words, we want to find conditions on the algebraic structure of an STS presentation under which parameterized bisimulation turns out to be a congruence.

We will use Σ_∂ to indicate the signature of an STS presentation without the operators ∂_0 and ∂_1. If Γ is an STS presentation with ∂ format (the definition of which we will give below), it follows from the fact that Γ is in ∂ format that the initial model does not have any identifications on elements of type `state`. In other words, the only identifications between such elements will be provided by the bisimulation semantics. Furthermore, whenever Γ has ∂ format it follows that the binary operation $-;-$ is the only operation yielding well-typed computations.

As a first stage in defining ∂ format for STS presentations, we define a particular format for judgments, ∂_{dS} format, which will be the format of some (but not all) of the judgments in a presentation in ∂ format.

Definition 33. (∂_{dS}-Format) Let (Σ, Γ, V) be an STS presentation. We say that a judgment is in ∂_{dS} format if and only if it has the form

$$\frac{\{z_i : x_i \to y_i \mid i \in I\}}{\sigma(z_1, \ldots z_r) : f(x_1, \ldots x_r) \to u}$$

where

— $\sigma, f \in \Sigma_\partial$,
— $I \subseteq \{1, \ldots, r\}$,
— $x_1, \ldots, x_n$, y_i, $(i \in \{1, \ldots, r\})$ and $z_i (i \in I)$ are all distinct variables. Moreover, $z_i = x_i$ when $i \notin I$.
— u is a term of $T_{\Sigma_\partial}(X')$ where each variable in X' occurs at most once, $X' = \{x'_i \mid 1 \leqslant i \leqslant r\}$ and $x'_i = y_i$ if $i \in I$, otherwise $x'_i = x_i$.

It is not difficult to see that ∂_{dS} format for judgments corresponds to De Simone format for inference rules.

Definition 34. (STS presentations in ∂ Format) An STS presentation (Σ, Γ, V) is in ∂ format if whenever J is a judgment in Γ, one of the following three conditions holds:

(*a*) J is in ∂_{dS} format, and no other judgment with ∂_{dS} format in Γ has the same operator σ, or

(*b*) J is one of the judgments

$$\frac{e : \texttt{trans}}{e : \texttt{comp}}$$

$$\frac{e_1 : u \Rightarrow v, e_2 : v \to w}{e_1; e_2 : u \Rightarrow w}$$

or

https://doi.org/10.1017/S0960129597002284 Published online by Cambridge University Press

(*c*) J is a typing judgment of the form

$$\frac{\varphi_1(e_1), \ldots \varphi_n(e_n)}{\sigma(e_1, \ldots e_n) : e}$$

such that if $C[\overline{e}]$ is an expression (including the term $\overline{e}$) that can be obtained by a substitution of variables in a subexpression of J, and $\Gamma \vdash \overline{e} : \texttt{comp}$, then $C[\overline{e}] = C_1[\partial_i(C_2[\overline{e}])]$ for some contexts C_1, C_2, and $i \in \{1, 2\}$.

For instance, the judgment below does not satisfy point (c) of the definition of ∂ format

$$\frac{u : \texttt{state}, c : \texttt{comp}}{\gamma(u, c) : \texttt{state}}.$$

In fact, this constraint ensures that the only way in which an element of type `comp` can influence the construction of an element of type `state` is via its source and target states. Furthermore, these rules ensure that any element of the initial model with type `comp` is of the form $t_1; \ldots; t_k$ for some elements $t_1, \ldots, t_k$ of type `trans` ($k \geqslant 1$).

By interpreting the recursive terms as solutions to fixed point equations, the STS presentation of CCS given in the previous sections are in ∂ format.

We are now able to prove the main theorem of this section.

Theorem 35. (Parameterized bisimulation as a congruence) Let (Σ, Γ, V) be an STS presentation and let $\mathcal{O}$ be a model of observations for such presentation. If the presentation is in ∂ format, $f(u_1, \ldots, u_r) \sim_{\mathcal{O}} f(v_1, \ldots, v_r)$ whenever $u_i \sim_{\mathcal{O}} v_i$ for all $1 \leqslant i \leqslant r$, f is an r-ary function symbol in Σ and $f(u_1, \ldots, u_r)$, $f(v_1, \ldots v_r)$ have type `state`.

Before going into the details of the proof, let us first try to explain the main ideas underlying it. In the case of transition system specifications satisfying a certain format, the proof that bisimulation is a congruence usually uses induction on the structure of the proof trees of derivations of transitions. Here, instead, we have replaced inference rules by operations, so we measure the complexity of terms of type `comp` by considering just the operations yielding elements of type `comp`. The proof will be by induction on this measure.

Let $c = \sigma(e_1, \ldots, e_n)$ be an element of the initial object with type `comp` and define the natural number $degree(c)$ to be

$$degree(c) = 1 \text{ if for each } i\ e_i \text{ has not type } \texttt{comp}$$

$$degree(c) = 1 + max\{degree(e_i) \mid e_i : \texttt{comp}, 1 \leqslant i \leqslant n\}, \text{ otherwise.}$$

Intuitively, the *degree* of an element of type `comp` gives the maximum number of operations yielding computations that occur in the element. The assumptions on the structure of the STS presentation ensure that this number is defined for each element of type `comp` in the initial object, and that, moreover, it is well defined. As an example, in the initial model of the STS presentation of CCS we have

$$degree([\mu, u\rangle) = 1$$

for all elements u of type `state`, and

$$degree(([\alpha, \beta.nil\rangle \lhd nil); [\beta, nil\rangle) = 3.$$

The proof that bisimulation is a congruence will use induction on this integer.

https://doi.org/10.1017/S0960129597002284 Published online by Cambridge University Press

Proof of Theorem 35.

Proof. Let $\mathscr{R}$ be the least equivalence relation over $\mathscr{I}$ satisfying

$$\sim_{\mathscr{O}} \subseteq \mathscr{R}$$

$$\frac{u_i \mathscr{R} v_i \text{ for all } i \in \{1,\dots n\}}{f(u_1,\dots,u_n)\mathscr{R} f(v_1,\dots,v_n)} \text{ for each n-ary operator } f \in \Sigma_\partial$$

$$\frac{\partial_0(e_1) \sim_{\mathscr{O}} \partial_0(e_2), \partial_1(e_1) \sim_{\mathscr{O}} \partial_1(e_2), \ell_{\mathscr{I}}(e_1) = \ell_{\mathscr{I}}(e_2)}{e_1 \mathscr{R} e_2}.$$

Since the labelling map $\ell_{\mathscr{I}}$ from $\mathscr{I}$ to $\mathscr{O}$ is a morphism and $u \sim_{\mathscr{O}} v$ implies $\ell_{\mathscr{I}}(u) = \ell_{\mathscr{I}}(v)$, it follows that $u \mathscr{R} v$ implies $\ell_{\mathscr{I}}(u) = \ell_{\mathscr{I}}(v)$.

By construction, $\mathscr{R}$ is certainly preserved by all operators in Σ_∂. It is straightforward to check (by properties of the ∂ format) that it is also preserved by the operators ∂_0, ∂_1, so it is a congruence. Furthermore, the restriction of the relation $\mathscr{R}$ to elements of type `state` contains $\sim_{\mathscr{O}}$. Therefore, to prove the theorem, it is enough to show that this restriction is a parametrized bisimulation.

Let $e : u \Rightarrow u'$ be an element of the initial model such that $\ell_{\mathscr{I}}(e) :$ `trans` and $\ell_{\mathscr{I}}(e) \notin *$. Assume that $u \mathscr{R} v$ for some $v :$ `state`. In order to prove that $\mathscr{R}$ is a bisimulation, we have to show that there is an element $e' : v \Rightarrow v'$ such that, $\ell_{\mathscr{I}}(e) = \ell_{\mathscr{I}}(e')$ and $u' \mathscr{R} v'$.

Let us consider the ways in which the equivalence $u \mathscr{R} v$ may arise. If $u \sim_{\mathscr{O}} v$, the result holds, since $\sim_{\mathscr{O}}$ is a bisimulation.

Therefore without loss of generality we can assume that there is some operator $f \in \Sigma_\partial$ and elements $u_1,\dots,u_r$, $v_1,\dots,v_r$ such that $u = f(u_1,\dots,u_r)$, $v = f(v_1,\dots,v_r)$ and $u_i \mathscr{R} v_i$ for $1 \leqslant i \leqslant r$. The ∂ format ensures that either $e = t$, $t :$ `trans` or $e = t_1; t_2; \dots; t_k$ for some $t_1,\dots t_k :$ `trans`, $k \geqslant 2$.

Induction Base. Assume $degree(e) = 1$. In this case, e must arise from the substitution of variables in an expression $\sigma(x_1,\dots x_r)$, whose typing judgment is of the form

$$\frac{}{\sigma(x_1,\dots x_r) : f(x_1,\dots x_r) \to w}$$

where $x_1,\dots,x_r$ are variables standing for elements of type `state`.

Hence, $e = \theta(\sigma(x_1,\dots x_r)$, for a suitable substitution of variables θ such that $\theta(x_i) = u_i$. Define θ' to be the substitution such that $\theta'(x_i) = v_i$ and $\theta'(x) = \theta(x)$ for $x \neq x_i$. Put $e' = \theta'(\sigma(x_1,\dots x_r)$. The required properties for e' follow from the fact that the STS presentation is in ∂ format.

Induction Step. Assume that whenever $e_i : u_i \Rightarrow u_i'$, $degree(e_i) < degree(e)$, $\ell_{\mathscr{I}}(e_i) :$ `trans`, $\ell_{\mathscr{I}}(e_i) \notin *$, there is some element $e_i' : v_i \Rightarrow v_i'$ such that $\ell_{\mathscr{I}}(e_i') = \ell_{\mathscr{I}}(e_i)$ and $u_i' \mathscr{R} v_i'$.

We have two cases. The first case is that e is an element of type `trans`. Because the STS presentation is in ∂ format, we have that the judgment that assigns the type `trans` to e is in ∂_{dS} and is of the form

$$\frac{\{z_i : x_i \to y_i \mid i \in I\}}{\sigma(z_1,\dots z_r) : f(x_1,\dots x_r) \to w}.$$

https://doi.org/10.1017/S0960129597002284 Published online by Cambridge University Press

Hence, there is some substitution of variables θ satisfying the assumptions of the judgment such that $\theta(x_i) = u_i$ for $1 \leqslant i \leqslant r$ and $e = \sigma(\theta(z_1), \ldots \theta(z_r))$. For each $i \in I$ we have that $\theta(z_i) : \theta(x_i) \to \theta(y_i)$, and $degree(\theta(z_i)) < degree(e)$. Since $*$ is absorbent and $\ell_{\mathscr{I}}(e) \notin *$, it follows that $\ell_{\mathscr{I}}(\theta(z_i)) \notin *$. Therefore, by the induction hypothesis, there is some $e'_i : v_i \to v'_i$ such that $\theta(x_i)\mathscr{R}v'_i$ and $\ell_{\mathscr{I}}(e'_i) = \ell_{\mathscr{I}}(\theta(z_i))$. Let θ' be the substitution sending z_i to e'_i for $i \in I$ and x_i to v_i for $1 \leqslant i \leqslant r$. Put $e' = \sigma(\theta'(z_1), \ldots, \theta'(z_r))$. The fact that the judgment is in ∂_{dS} format ensures that this element has the required properties.

The remaining case is that $e = t_1; \ldots ; t_k$ for some elements $t_1, \ldots t_k$ of type `trans`, $k \geqslant 2$. Now for each $1 \leqslant i \leqslant k$ we have $degree(t_i) < degree(e)$ and $\ell_{\mathscr{I}}(t_i) : \texttt{trans}$ since $t_i : \texttt{trans}$ and $\ell_{\mathscr{I}}(t_i) \notin *$ because $*$ is absorbent and $\ell_{\mathscr{I}}(e) = \ell_{\mathscr{I}}(t_1); \ldots \ell_{\mathscr{I}}(t_k) \notin *$. So, by the induction hypothesis, there are elements $t'_1, \ldots, t'_k$ such that $\ell_{\mathscr{I}}(t'_i) = \ell_{\mathscr{I}}(t_i)$ for all i, $\partial_0(t'_1) = f(v_1, \ldots, v_n)$, $\partial_1(t'_i) = \partial_0(t'_{i+1})$ for each $1 \leqslant i \leqslant k-1$, and $\partial_1(t'_k)\mathscr{R}\partial_1(t_k)$. Put $e' = t'_1; \ldots ; t'_k$. This has the properties required, so the induction step is proved. □

The main consequence of this theorem is that we can prove that bisimulation is a congruence just by proving that the STS presentation is in ∂ format. As simple corollaries of the previous theorem we have the following results.

Theorem 36. (CCS Observational Congruences)

1 Relation $\sim_{\mathscr{M}}$ is a congruence for CCS.
2 Relation $\sim_{\mathscr{H}}$ is a congruence for CCS.
3 Relation $\sim_{\mathscr{M}_\tau}$ is a congruence for CCS.
4 Relation $\sim_{\mathscr{H}_\tau}$ is a congruence for CCS.

Remark 37. We have shown that it is possible to extend De Simone format to cover our algebraic framework. A similar result can be obtained by considering the $tyft/tyxt$ format of Groote and Vaandrager (Groote and Vaandrager 1992). The format obtained this way gives constraints on the equational rules defining the source and target operations, just as in the case of the De Simone format. However, in the case of the $tyft/tyxt$ format we have to modify the definition of STS presentation, since Groote and Vaandrager do not require the presence of certain relations between the premises and the conclusion on an inference rule. Instead, we explicitly require that the type of an operation depends on the types of the arguments.
Finally, the special feature of the GSOS format of Bloom, Istrail and Mayer (Bloom *et al.* 1995) is the possibility of having negative premises. Negative premises cannot be expressed in the ET logic as it is. However, negative premises can be added to the ET logic by adopting the logic programming technique of stratification (Apt 1990) as pursued in (Groote 1990). This leads to an extension of the definition of an STS presentation that allows negative premises.

To conclude this section, we show that semantics process description languages in which predicates over the structure of agents may occur can be naturally described in terms of STS. The idea is that the evaluation of predicates over the structure of agents is internalized in the definition of the labelling map. The limitation to predicates on the structure of processes is the main difference with respect to the approach of Baeten and Verhoef (1993), where arbitrary predicates are allowed.

https://doi.org/10.1017/S0960129597002284 Published online by Cambridge University Press

$$\mu : \mu :\longrightarrow nil$$

$$\frac{t : u \longrightarrow v}{t \bullet w : u \cdot w \longrightarrow v \cdot w} \qquad \frac{t : u \longrightarrow v}{w \bullet t : w \cdot u \longrightarrow v}$$

$$[\![\mu]\!] = \mu$$

$$[\![t \bullet u]\!] = [\![t]\!]$$

$$[\![u \bullet t]\!] = \texttt{if } [\![u]\!] = \surd \texttt{ then } [\![t]\!] \texttt{ else } *$$

$$[\![nil]\!] = \surd \qquad [\![\mu]\!] = \neg\surd$$

$$\frac{[\![u]\!] = \surd, [\![v]\!] = \surd}{[\![u \cdot v]\!] = \surd}$$

Table 6. *Process termination*

The example concerns the treatment of successful termination in the presence of an operator of sequential composition. This is taken from the operational semantics of Aceto and Hennessy (1992). The principle of Aceto and Hennessy (1992) is to define a termination predicate, denoted by $\surd$. Table 6 presents both the STS presentation and the specific observation algebra that encapsulates the termination predicate as an operation on the element of type `state`. To avoid cumbersome notation, we shall use $[\![e]\!]$ to indicate the meaning of the term e in the model of observations. It is immediate to see that the 'bisimulation as a congruence theorem' holds also for this class of STS specification, where predicates over the structure on processes may occur.

5. Simplification morphisms and minimal realizations

The fact that the bisimulation semantics for an STS presentation in ∂ format is a congruence brings us to an interesting question: is bisimulation semantics characterized by an abstract structured transition system? In other words, is there a structured transition system whose elements of type `state` are congruence classes of bisimilar states? If such a structured transition system exists, it plays the role of a minimal realization with respect to the bisimulation semantics. This issue will be the main concern of this section.

Definition 38. (Transition Preserving Homomorphisms) Let (T_1, ℓ_1), (T_2, ℓ_2) be objects of the category $\mathscr{STS}[\Gamma] \gg \mathscr{O}$. A morphism $h : (T_1, \ell_1) \to (T_2, \ell_2)$ is called a transition preserving homomorphism if and only if

— h is surjective;
— if $c_2 : h(u_1) \Rightarrow v_2$, $u_1 :$ `state`, $c_2 \in T_2$, $\ell_2(c_2) \notin *$, there exists an element $c_1 \in T_1$, $c_1 : u_1 \Rightarrow v_1$ such that $h(c_1) = c_2$

Transition preserving homomorphisms (tp-morphisms for short) are morphisms that, as well as the algebraic structure, the typing and the observations, preserve the *correct*

https://doi.org/10.1017/S0960129597002284 Published online by Cambridge University Press

computations starting from any given state. A tp-morphism can be understood as a transformation that takes a system and yields a structurally simpler but equivalent system. In other words, let $h : (T_1, \ell_1) \to (T_2, \ell_2)$ be a tp-morphism, and let u : `state`, $t : u \to v$, $c : u \Rightarrow w$ be elements of T_1. Then, $h(u)$ is the transformation in T_2 of the program u, $h(t)$ is the elementary instruction of T_2 that corresponds to t, and $h(c)$ is the result of running the program u on T_2.

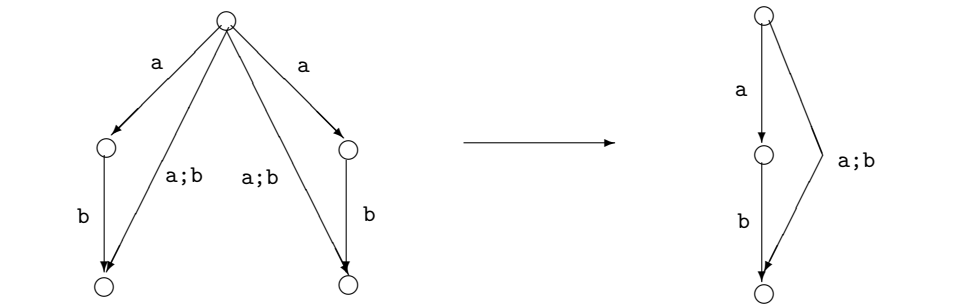


Fig. 5. A transition preserving homomorphism

We provide some examples to illustrate the properties of tp-morphisms. For the presentation, we represent an object of $\mathscr{SIS}[\Gamma] \gg \mathscr{O}$ as a labelled graph. For simplicity we do not draw arrows labelled with elements in $*$. Figure 5 illustrates a mapping that is a tp-morphism, while Figure 6 shows a mapping that is not a tp-morphism. This is because if we take the arrow labelled c from $h(u) = h(u')$ to $h(v')$, we cannot find any arrow labelled c starting from u that is mapped onto it.

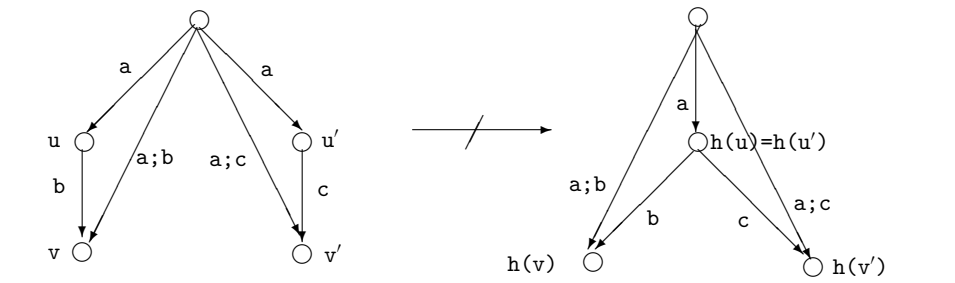


Fig. 6. A mapping that is not a transition preserving homomorphism

Transition preserving homomorphisms have been independently introduced by Degano, De Nicola and Montanari (Degano *et al.* 1988), and by Arnold and Dicky (Arnold and Dicky 1989), who call them *transition system homomorphisms*. Tp-morphisms can

https://doi.org/10.1017/S0960129597002284 Published online by Cambridge University Press

be considered as a generalization of the pseudo-morphisms introduced by Segerberg (Segerberg 1968), and called *zig-zag* morphisms in van Benthem (1984). Zig-zag morphisms are adopted in modal logic theory to characterize the equivalence of logical models with respect to the satisfiability of logical formulae.

Proposition 39. The identity morphism Id is a tp-morphism.

Proposition 40. The composition of tp-morphisms is a tp-morphism.

Definition 41. (The Category of Admissible Realizations) The category of *admissible realizations*, $\mathscr{A}[\Gamma, \mathscr{O}]$ is the subcategory of $\mathscr{SFS}[\Gamma] \gg \mathscr{O}$ whose objects are all the objects that are reachable from the initial object via tp-morphisms, and whose morphisms are tp-morphisms.

We now study the structure of the category of admissible realizations. We will use $\circ$ to indicate composition of morphisms, that is, $(f \circ g)(x)$ means $g(f(x))$.

Definition 42. (The Congruence Induced by Tp-morphisms) Let $h : (T_1, \ell_1) \to (T_2, \ell_2)$ be a tp-morphism. $R(h)$ is the relation on T_1 defined as $\{(e_1, e_2) \mid e_1, e_2 \in T_1 \text{ and } h(e_1) = h(e_2)\}$

Proposition 43. $R(h)$ is a congruence on T_1.

Definition 44. Let $f : (\mathscr{I}, \ell_{\mathscr{I}}) \to (T_1, \ell_1)$ and $g : (\mathscr{I}, \ell_{\mathscr{I}}) \to (T_2, \ell_2)$ be tp-morphisms. Then we say that $f \sqsubseteq g$ if and only if there exists a tp-morphism $h : (T_1, \ell_1) \to (T_2, \ell_2)$ such that $f \circ h = g$.

Notice that since all tp-morphisms are surjective, if such a morphism h exists, it is unique.

Proposition 45. Relation $\sqsubseteq$ is a preorder on the set of tp-morphisms.

Theorem 46. Let $f : (\mathscr{I}, \ell_{\mathscr{I}}) \to (T_1, \ell_1)$ and $g : (\mathscr{I}, \ell_{\mathscr{I}}) \to (T_2, \ell_2)$ be tp-morphisms. If $f \sqsubseteq g$ and $g \sqsubseteq f$, then T_1 is isomorphic to T_2.

Proof. By definition, $f \sqsubseteq g$ implies that $f \circ h = g$ for some tp-morphism h. Similarly, we have that $f = g \circ k$, for some tp-morphism k. Using these equalities, we have

$$f \circ h = g \circ k \circ h = g = g \circ Id.$$

Now, because tp-morphisms are right-cancellative, we can conclude that $k \circ h = Id$. Using similar reasoning, we can prove that $h \circ k = Id$. From this the thesis follows. □

Theorem 47. (Tp-morphisms and Congruences) $f \sqsubseteq g$ if and only if $R(f) \subseteq R(g)$.

Proof. Assume that $f \sqsubseteq g$ and $e_1 R(f) e_2$. We have to show that $e_1 R(g) e_2$. By definition of $\sqsubseteq$, we have that $f \circ h = g$ for some tp-morphism h. Now

$$g(e_1) = (f \circ h)(e_1) = h(f(e_1)) = h(f(e_2)) = g(e_2).$$

Conversely, assume that $R(f) \subseteq R(g)$. Let $h : (T_1, \ell_1) \to (T_2, \ell_2)$ be the mapping given by $h(e') = g(e)$ for $e' = f(e)$.

The mapping is well defined because f is surjective and $R(f) \subseteq R(g)$. It is easy to check that h is a tp-morphism and that $f \circ h = g$. □

https://doi.org/10.1017/S0960129597002284 Published online by Cambridge University Press

Because of the previous theorems, in order to study the properties of the preorder of tp-morphisms it is enough to study the partially ordered set $(C_{tp}, \subseteq)$ of congruences induced on the initial object by tp-morphisms and ordered by inclusion.

Definition 48. (The Maximal Congruence) Let Θ be a relation over the elements of the initial object defined as $(\bigcup_{i\in J} R(f_i))^*$ where $\{f_i : (\mathscr{I}, \ell_{\mathscr{I}}) \to (T_i, \ell_i) \mid i \in J\}$ is the set of all tp-morphisms starting at $(\mathscr{I}, \ell_{\mathscr{I}})$.

Clearly, Θ contains each relation in C_{tp}. It is easy to prove that it is a congruence since every $R(f_i)$ is a congruence relation. Furthermore, $e_1 \Theta e_2$ implies that $\ell_{\mathscr{I}}(e_1) = \ell_{\mathscr{I}}(e_2)$.

Definition 49. (The Quotient Construction) Let $\mathscr{Q}$ be the quotient of the initial object $\mathscr{I}$ with respect to the congruence Θ. $\mathscr{Q}$ is an equational type algebra whose elements are the congruences classes $[e]_\Theta$. The typing structure on $\mathscr{Q}$ is given by: $[e]_\Theta : a$ provided that there exists some $e' \in [e]_\Theta$ such that $e' : a$.

Notice that because $\mathscr{O}$ has no confusion on types, if a is a type in $\mathscr{I}$ and $\ell_{\mathscr{I}}(a) = \ell_{\mathscr{I}}(e)$, then $e = a$. Therefore, $[a]_\Theta$ contains just the single element a, and by abuse of notation we call this congruence class a as well. The action of each operator σ in the signature on the quotient $\mathscr{Q}$ is given by $\sigma([e_i]_\Theta, \ldots, [e_n]_\Theta) = [\sigma(e_1, \ldots, e_n)]_\Theta$. This action is well defined, because Θ is a congruence.

Lemma 50. The quotient $\mathscr{Q}$ is an element of the category $\mathscr{STS}[\Gamma]$.

Let $\ell_{\mathscr{Q}} : \mathscr{Q} \to \mathscr{O}$ be the mapping given by $\ell_{\mathscr{Q}}([e]_\Theta) = \ell_{\mathscr{I}}(e)$, and let q_Θ be the quotient mapping, that is, $q_\Theta(e) = [e]_\Theta$.

Lemma 51. The mapping $\ell_{\mathscr{Q}}$ is a morphism of the category $\mathscr{STS}[\Gamma]$.

Proof. The mapping $\ell_{\mathscr{Q}}$ is well defined since $e_1 \Theta e_2$ implies that $\ell_{\mathscr{I}}(e_1) = \ell_{\mathscr{I}}(e_2)$. It preserves the typing relation, since $\ell_{\mathscr{I}}$ preserves the typing relation. Moreover, $\ell_{\mathscr{Q}}$ is a morphism of the category, since $\ell_{\mathscr{I}}$ is a morphism of $\mathscr{STS}[\Gamma]$. □

Because of the previous Lemmas, we have the following result.

Proposition 52. $(\mathscr{Q}, \ell_{\mathscr{Q}})$ is an object of the category $\mathscr{STS}[\Gamma] \gg \mathscr{O}$.

Theorem 53. (The Quotient Mapping is a Tp-morphism) $q_\Theta : (\mathscr{I}, \ell_{\mathscr{I}}) \to (\mathscr{Q}, \ell_{\mathscr{Q}})$ is a tp-morphism.

Proof. By construction, q_Θ is a surjective morphism. It remains to prove that it is transition preserving.

Let $c_2 : q_\Theta(u) \Rightarrow v, \ell_{\mathscr{Q}}(c_2) \notin *$, be a computation in the quotient. Because q_Θ is surjective, there exists an element $c_1 \in \mathscr{I}$ such that $q_\Theta(c_1) = c_2$. We have that $\partial_0(c_1) \Theta u$, since

$$q_\Theta(\partial_0(c_1)) = \partial_0(q_\Theta(c_1)) = \partial_0(c_2) = u.$$

Now, by definition of the congruence Θ, there exists a sequence $f_1, \ldots, f_n$ of tp-morphisms starting from the initial object such that

$$\partial_0(c_1) R(f_1) u_1 R(f_2) u_2, \ldots, u_{n-1} R(f_n) u_n = u.$$

Since each f_i is a tp-morphism, the thesis follows. □

https://doi.org/10.1017/S0960129597002284 Published online by Cambridge University Press

Corollary 54. The congruence Θ is the greatest element of the partially ordered set $(C_{tp}, \subseteq)$.

Theorem 55. (Minimal Realization Theorem) $(\mathscr{Q}, \ell_{\mathscr{Q}})$ is the final object of the category of admissible realizations.

Proof. The previous corollary ensures that the congruence Θ is the maximal congruence induced on the initial object by tp-morphisms. Furthermore, $(\mathscr{Q}, \ell_{\mathscr{Q}})$ is an object in the category of admissible realizations. We have to show that given any object (T, ℓ) in such a category there always exists a unique tp-morphism from this object to $(\mathscr{Q}, \ell_{\mathscr{Q}})$. By initiality, there exists a unique tp-morphism f from $(\mathscr{I}, \ell_{\mathscr{I}})$ to (T, ℓ). Clearly, $R(f) \subseteq R(q_\Theta)$. From this we can derive that $f \sqsubseteq q_\Theta$, hence, there exists a unique tp-morphism h such that $q_\Theta = f \circ h$. □

Lemma 56. (Minimal Realization as Bisimulation) The restriction of the relation Θ to elements of $\mathscr{I}$ with type `state` is a bisimulation.

Proof. If $u_1 \Theta u_2$, then $\ell_{\mathscr{I}}(u_1) = \ell_{\mathscr{I}}(u_2)$. It remains to show that the restriction of the relation Θ has the transfer property of bisimulation. Suppose that $u_1 \Theta u_2$, u_1, u_2 : `state` and $\partial_0(e_1) = u_1$, $\ell_{\mathscr{I}}(e_1)$: `trans` and $\ell_{\mathscr{I}}(e_1) \notin *$. Then $\partial_0(q_\Theta(e_1)) = q_\Theta(\partial_0(e_1)) = q_\Theta(u_1) = q_\Theta(u_2)$. The label of e_1 is equal to $\ell_Q(q_\Theta(e_1))$ and is not in $*$, so, since q_Θ is a tp-morphism, there is some $e_2 : u_2 \Rightarrow v_2$ such that $q_\Theta(e_2) = q_\Theta(e_1)$. It is straightforward to check that $\ell_{\mathscr{I}}(e_1) = \ell_{\mathscr{I}}(e_2)$ and $\partial_1(e_1) \Theta \partial_1(e_2)$. □

Similarly, one can prove that for any tp-morphism f from the initial object, congruence $R(f)$ is a bisimulation. In other words, any tp-morphism from the initial object induces a structural congruence on the elements of type `state` that respects bisimulation.

The last result of this section establishes a connection between the minimal realization and the maximal bisimulation when the presentation has ∂ format.

Theorem 57. (Minimal Realization and Bisimulation as a Congruence) Let (Σ, Γ, V) be a STS presentation in ∂ format. Two elements of $\mathscr{I}$ of type `state` are mapped together under the morphism from the initial to the final element of the category of admissible realizations $\mathscr{A}[\Gamma, \mathscr{O}]$ if and only if they are equivalent under $\sim_{\mathscr{O}}$.

Proof. The proof of Theorem 35 constructs a congruence $\mathscr{R}$ whose restriction to elements of type `state` is $\sim_{\mathscr{O}}$, and, moreover, whenever $\partial_0(e_1) \sim_{\mathscr{O}} \partial_0(e_2)$, $\partial_1(e_1) \sim_{\mathscr{O}} \partial_1(e_2)$, and $\ell_{\mathscr{I}}(e_1) = \ell_{\mathscr{I}}(e_2)$, the elements e_1, e_2 are related under $\mathscr{R}$.

To prove the theorem, we show that relation $\mathscr{R}$ is characterized by a tp-morphism. The idea is to make a quotient of the initial model with respect to relation $\mathscr{R}$, and prove that the canonical quotient mapping is a tp-morphism. The quotient construction and the canonical quotient morphism $q_{\mathscr{R}}$ can be constructed just as above using the congruence $\mathscr{R}$ instead of the congruence Θ.

Now we show that $q_{\mathscr{R}}$ is a tp-morphism. It is certainly surjective. Suppose that c_2 : $q_{\mathscr{R}}(u) \Rightarrow v$ is a computation in the quotient u : `state`, and the label of c_2 under the unique morphism to $\mathscr{O}$ is not in $*$. Since c_2 : `comp`, the typing relation on the quotient implies that there is an element c_1 : `comp` $\in \mathscr{I}$ satisfying $c_2 = q_{\mathscr{R}}(c_1)$.

Since Γ has ∂ format, there are elements $t_1, \ldots, t_k$ in $\mathscr{I}$ such that $c_1 = t_1; \ldots; t_k$ and t_i : `trans` for all i. Because $*$ is absorbent and $\ell_{\mathscr{I}}$ is a morphism, $\ell_{\mathscr{I}}(t_i) \notin *$ for each i. Now $q_{\mathscr{R}}(\partial_0(c_1)) = \partial_0(c_2) = q_{\mathscr{R}}(u)$, so $\partial_0(t_1) = \partial_0(c_1) \mathscr{R} u$. Both $\partial_0(t_1)$ and u have type `state`,

https://doi.org/10.1017/S0960129597002284 Published online by Cambridge University Press

so $\partial_0(t_1) \sim_{\mathscr{O}} u$. Therefore there is some element t'_1 such that $\partial_0(t'_1) = u$, $\ell_{\mathscr{I}}(t'_1) = \ell_{\mathscr{I}}(t_1)$ and $\partial_1(t'_1) \sim_{\mathscr{O}} \partial_1(t_1) = \partial_0(t_2)$. By a similar process, we can find elements $t'_2, \ldots, t'_k$ such that $c'_1 = t'_1; \ldots t'_k$ has type `comp`, $\ell_{\mathscr{I}}(t'_i) = \ell_{\mathscr{I}}(t_i)$ for all i, so $\ell_{\mathscr{I}}(c'_1) = \ell_{\mathscr{I}}(c_1)$ and $\partial_0(c'_1) \sim_{\mathscr{O}} \partial_0(c_1)$, $\partial_1(c'_1) \sim_{\mathscr{O}} \partial_1(c_1)$. The properties of $\mathscr{R}$ ensure that $c'_1 \mathscr{R} c_1$, and hence $c_2 = q_{\mathscr{R}}(c'_1)$. Since $\partial_0(c'_1) = u$, it follows that $q_{\mathscr{R}}$ is a tp-morphism, as required.

The definition of Θ now implies that $\mathscr{R} \subseteq \Theta$. Moreover, by Lemma 56 the restriction of Θ to elements of type `state` is a parametrized bisimulation, so is contained in $\sim_{\mathscr{O}}$, which is the restriction of $\mathscr{R}$ to elements of type `state`. The result follows from Theorem 55. □

The terminal object of the category of *admissible realizations* $\mathscr{A}[\mathscr{CCS}, \mathscr{M}_\tau]$ does not provide the minimal realization of Milner's weak bisimulation (Milner 1989) (sometimes called *Observation Congruence*). The reason is that observation congruence is not a bisimulation, and the definition of tp-morphisms implies that, from congruent states, matching transitions lead to congruent states. In other words, the mimimal realization theorem holds for bisimulations that are also congruences. Indeed, the terminal object of the category $\mathscr{A}[\mathscr{CCS}, \mathscr{M}_\tau]$ is the minimal realization of CCS dynamic bisimulation congruence (Montanari and Sassone 1992). However, observation equivalence can be characterized by a universal property of finality in a suitable 'ad hoc' category that is constructed from the category $\mathscr{STS}[\mathscr{CCS}] \gg \mathscr{M}_\tau$. The details of this construction can be found in Ferrari (1990).

5.1. *Discussion*

An algebraic characterization of bisimulation semantics in terms of simplification morphisms was presented for the first time in Castellani *et al.* (1983). The tight relation between bisimulation and simplification morphisms has been discussed in Arnold and Dicky (1989), Benson and Ben-Shachar (1988), and Castellani (1987). The last of these proves the existence of a minimal canonical representative for the equivalence classes of (interleaving) bisimulation (this result has been extended in Montanari and Sgamma (1989) to partial ordering bisimulation semantics). Similar properties have been generalized to *co-algebras* in Aczel and Mendler (1989) (see also Rutten and Turi (1994)). In the present paper, the main property of tp-morphisms is the *parameterization* with respect to the model of observations. This shows up in the uniform characterization of several notions of bisimulation semantics as minimal realizations.

Recently, an abstract categorical formulation of bisimulation has been presented by Joyal, Nielsen and Winskel (Joyal *et al.* 1994). They proved that bisimulation is expressed through a span of open maps (in the suitable category). By forgetting the algebraic structure on states and transitions, it can be proved that tp-morphisms are open maps: it suffices to consider the fibre of transition systems over the set underlying the model of the observation $\mathscr{O}$ and then apply the same construction of Joyal *et al.* (1994, Proposition 1). Moreover, the *pure*-morphisms of Benson and Ben-Shachar (1988), the abstraction homomorphisms of Castellani (1987) and Montanari and Sgamma (1989), and the *saturating quasi-homomorphism* of Arnold and Dicky (1989) are special instances of open maps.

This paper contributes to the foundations of the semantics of process description languages by showing that parametric bisimulation semantics give a minimal realization

https://doi.org/10.1017/S0960129597002284 Published online by Cambridge University Press

(in the sense of classical minimal state realization of automata theory). This idea is captured by the construction of the terminal structured transition system (*i.e.*, an algebra without redundant states) in the suitable category of equational type algebras.

Hence, all the constructions we have presented have a 'universal algebra' flavour. This is what makes the difference between the work of Joyal *et al.* (1994) and what has been presented here.

6. Concluding remarks

In this paper we have introduced a general algebraic framework for analysing observational semantics of process description languages by providing some general techniques that are completely parameterized with respect to the notion of observation. This has allowed us to integrate some results for the observational semantics of process description languages within our algebraic setting. However, many other questions are left open.

For instance, bisimulations can be classified, roughly, into two main groups depending on whether the bisimulation step is independent or not from the history of the computation. A bisimulation is *history independent* when no information relative to some part of the ongoing (and possibly not completed) computation executed so far is taken into account in the bisimulation step. History independent bisimulations gather knowledge about process behaviours *step-by-step* (that is, *incrementally*). All the bisimulation semantics considered in this paper are incremental bisimulations.

The class of *history dependent* bisimulations includes, *partial ordering* (Degano *et al.* 1987), *history preserving* (van Glabbeek and Goltz 1989; Rabinovich and Trakhtenbrot 1988), *causal* (Darondeau and Degano 1989) and *location* (Boudol *et al.* 1993). The common feature of all these semantical notions is that the bisimulation step depends on the effects of the past moves of the ongoing computation. Hence, the characterizing feature of history dependent bisimulations is that bisimulation steps do 'remember' part of the ongoing computation.

To handle history dependent bisimulation semantics, we consider an *unfolded* transition system, namely a transition system where states are pairs consisting of an agent together with the sequence of transitions that have led to its occurrence, *i.e.*, an abstraction on what happened in the past. This can be expressed in terms of STS by the introduction of a *state operator* (Baeten and Bergstra 1988) together with a suitable notion of type for histories.

Assuming that ξ is an element (of a certain state space) of type `history`, if u has type `state`, $\xi :: u$ has type `state`. The typing judgment for the state operator is:

$$\frac{t : u \longrightarrow v}{\delta(\xi, t) : \xi :: u \longrightarrow \eta(\xi, t) :: v}$$

The idea is that a step in the execution of an agent (the premise of the rule) has a side effect over the history ξ. The binary function η takes care of updating the history ξ. The function δ yields the result of the state transformation. In the formulation of Baeten and Bergstra (1988) the function δ (called the action function) always returns an action,

https://doi.org/10.1017/S0960129597002284 Published online by Cambridge University Press

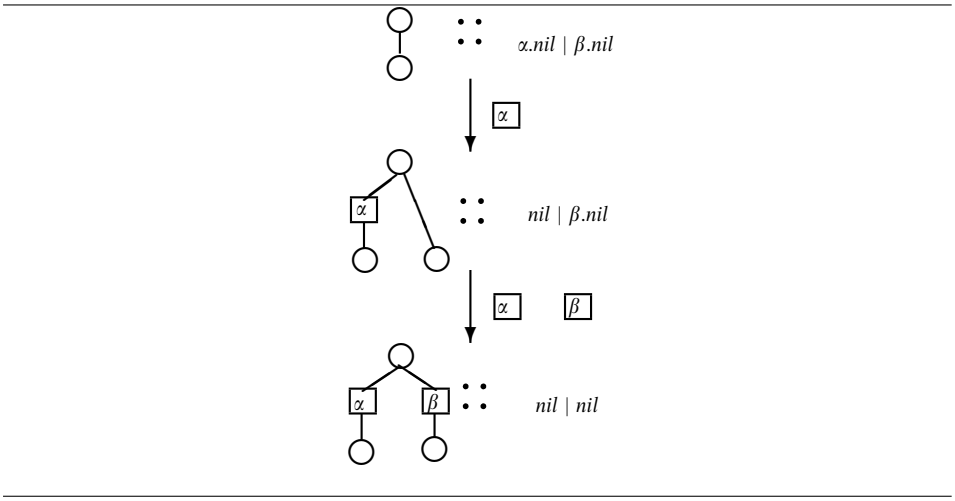


Fig. 7. The state operator: an example

whereas in this formulation the function δ is simply an operator symbol of the STS presentation (and hence it does not necessarily yield an action).

The presence of the algebraic structure makes function δ become an abstraction function that reveals only a suitable abstraction of the history dependency of process behaviours. For instance, this formulation of the state operator is at the basis of the alternative formulation of the π-calculus (Milner *et al.* 1992) presented in Ferrari *et al.* (1994). Figure 7 illustrates an example of computation of the state operator. In the example, histories ξ are viewed as concatenable concurrent histories, and, through the labelling construction, transitions are labelled with partial orderings (*i.e.*, the function δ abstracts from the detailed structure of origins and destinations).

Another interesting research theme is the use of our algebraic framework to give general techniques for comparing observational semantics, thus revealing the weaker and stronger points of the different approaches. To this end, a useful machinery is provided by the general comma category construction outlined in Subsection 3.4. One challenging application is to approach a formal discussion of the interleaving versus true concurrency controversy. In fact, consider the models of observations $\mathscr{H}$ and $\mathscr{M}$. It is easy to see that there is a morphism $\eta : \mathscr{H} \to \mathscr{M}$ that sends the history h_μ to the the action μ. Hence, the construction of the comma category of $\mathscr{STS}[\mathscr{CCS}]$ over the models of observations $\mathscr{H}$ and $\mathscr{M}$ automatically introduces a framework in which an interleaving and a truly concurrent observational semantics of CCS can be studied and compared.

Acknowledgements

The authors are grateful to the anonymous referees for their constructive comments.

https://doi.org/10.1017/S0960129597002284 Published online by Cambridge University Press

References

Aceto, L. and Hennessy, M. (1992) Termination, Deadlock and Divergence. *Journal of ACM* **39** (1) 147–187.

Aczel, P. and Mendler, P. (1989) A Final Coalgebra Theorem. In: Proc. Category Theory and Computer Science. *Springer-Verlag Lecture Notes in Computer Science* **389** 356–365.

Apt, K. (1990) Logic Programming. In: van Leeuwen (ed.) *Handbook of Theoretical Computer Science*, *Formal Methods and Semantics*, North-Holland 495–574.

Arnold, A. and Dicky, A. (1989) An Algebraic Characterization of Transition System Equivalences. *Information and Computation* **82** 198–229.

Baduel, E. and Darondeau, P. (1992) Structural Operational Specifications and Trace Automata. In: Proc. CONCUR 92. *Springer-Verlag Lecture Notes in Computer Science.*

Baeten, J. and Bergstra, J. (1988) Global Renaming Operators in Concrete Process Algebra. *Information and Computation* **78** (3) 205–245.

Baeten, J. and Verhoef, C. (1993) A Congruence Theorem for Structured Operational Semantics with Predicates. In: Proc. CONCUR 93. *Springer-Verlag Lecture Notes in Computer Science* **715** 477–492.

Bednarczyk, M. (1988) *Categories of Asynchronous Systems*, Ph.D. Thesis, University of Sussex, Technical Report 1/88.

Benson, D. and Ben-Shachar, O. (1988) Bisimulations of Automata. *Information and Computation* **79** 60–83.

van Benthem, J. (1984) Correspondence Theory. In: Gabbay and Guenthner (eds.) *Handbook of Philosophical Logics* **Vol II**, Reidel.

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

Best, E. and Devillers, R. (1987) Sequential and Concurrent Behaviour in Petri Net Theory. *Theoretical Computer Science* **55** 87–136.

Bloom, B., Istrail, S. and Meyer, A. (1995) Bisimulation Can't be Traced. *Journal of the ACM* **42** 232–268.

Boudol, G. and Castellani, I. (1988) Concurrency and Atomicity. *Theoretical Computer Science* **59** (1-2) 25–84.

Boudol, G. and Castellani, I. (1989) Permutations of Transitions: An Event Structure Semantics for CCS. In: Proc. REX School, Workshop on Linear Time Branching Time and Partial Orders in Logics and Models for Concurrency. *Springer-Verlag Lecture Notes in Computer Science* **354** 411–437.

Boudol, G., Castellani, I., Hennessy, M. and Kiehn, A. (1993) Observing Localities. *Theoretical Computer Science* **114** 31–61.

Castellani, I. (1987) Bisimulations and Abstraction Homomorphisms. *Journal of Computer and System Sciences* **34** 210–235.

Castellani, I., Franceschi, P. and Montanari, U. (1993) Labelled Event Structures: A Model for Observable Concurrency. In: *Proc. IFIP TC2 Working Conference on Formal Description of Programming Concepts – II*, North-Holland, 383–400.

Castellani, I. and Hennessy, M. (1989) Distributed Bisimulation. *Journal of ACM* **36** 887–911.

Corradini, A., Ferrari, G. and Montanari, U. (1990) Transition Systems with Algebraic Structure as Models of Computations. In: Semantics of Systems of Concurrent Processes. *Springer-Verlag Lecture Notes in Computer Science* **469** 185–222.

Darondeau, P. and Degano, P. (1989) Causal Trees. In: Proc. ICALP 89. *Springer-Verlag Lecture Notes in Computer Science* **372** 234–248.

https://doi.org/10.1017/S0960129597002284 Published online by Cambridge University Press

Degano, P., De Nicola, R. and Montanari, U. (1987) Observational Equivalences for Concurrency Models. In: Wirsing, M. (ed.) *Formal Description of Programming Languages III*, North-Holland 105–132.

Degano, P., De Nicola, R. and Montanari, U. (1988) A Distributed Operational Semantics for CCS Based on Condition/Event Systems. *Acta Informatica* **26** 59–91.

Degano, P., De Nicola, R. and Montanari, U. (1990) A Partial Ordering Semantics for CCS. *Theoretical Computer Science* **75** 223–262.

Degano, P., Meseguer, J. and Montanari, U. (1989) Axiomatizing Net Computations and Processes. In: *Fourth Symposium on Logic in Computer Science (LICS)* 175–185.

Degano, P. and Montanari, U. (1987) Concurrent Histories: A Basis for Observing Distributed Systems. *Journal of Computer and System Sciences* **34** 422–461.

De Simone, R. (1985) Higher Level Synchronising Devices in MEIJE-SCCS. *Theoretical Computer Science* **37** (3) 245–267.

Ferrari, G. (1990) *Unifying Models of Concurrency*, Ph.D. Thesis TD - 4/90, Dipartimento di Informatica, Univ. Pisa.

Ferrari, G. and Montanari, U. (1990) Towards the Unification of Models for Concurrency. In: Proc. CAAP'90. *Springer-Verlag Lecture Notes in Computer Science* **431** 162–176.

Ferrari, G., Montanari, U. and Mowbray, M. (1991) On Causality Observed Incrementally, Finally. In: Proc. TAPSOFT'91. *Springer-Verlag Lecture Notes in Computer Science* **493** 26–41.

Ferrari, G., Montanari, U. and Quaglia, P. (1994) A π-calculus with Explicit Substitutions: the Late Semantics. In: MFCS '94. *Springer-Verlag Lecture Notes in Computer Science* **841**.

Gischer, J. (1988) The Equational Theory of Pomsets. *Theoretical Computer Science* **61** 199–224.

van Glabbeek, R. and Goltz, U. (1989) Equivalence Notions for Concurrent Systems and Refinement of Actions. In: Proc. MFCS '89. *Springer-Verlag Lecture Notes in Computer Science* **397**.

Gorrieri, R. and Montanari, U. (1990) SCONE: A Simple Calculus of Nets. In: Proc. CONCUR'90. *Springer-Verlag Lecture Notes in Computer Science* **458** 2–30.

Groote J. F. (1990) Transition System Specifications with Negative Premises. In: Proc. CONCUR'90. *Springer-Verlag Lecture Notes in Computer Science* **458** 332–341.

Groote, J. and Vaandrager, F. (1992) Structured Operational Semantics and Bisimulation as a Congruence. *Information and Computation* **100** 202–260.

Hennessy, M. (1988) *An Algebraic Theory of Processes*, MIT Press.

Hennessy, M. and Milner, R., (1985) Algebraic Laws for Nondeterminism and Concurrency. *Journal of ACM* **32** (1).

Hoare, C. A. R. (1985) *Communicating Sequential Processes*, Prentice Hall.

Joyal, A., Nielsen, M. and Winskel, G., (1994) Bisimulation from Open Maps, BRICS Technical Report RS-94-7, Aarhus University. (Preliminary version in *Proc, LICS'93*).

Keller, R. (1976) Formal Verification of Parallel Programs. *Com. ACM* **7** 371–384.

Lloyd, J. W. (1987) *Foundations of Logic Programming*, Springer-Verlag (Second Edition).

Mac Lane, S. (1971) *Categories for the Working Mathematician*, Springer-Verlag.

Manca, V., Salibra, A. and Scollo, G. (1990) Equational Type Logic. *Theoretical Computer Science* **77** 131–159.

Mazurkiewicz, A. (1989) Basic Notions of Trace Theory. In: Proc. REX School, Workshop on Linear Time Branching Time and Partial Orders in Logics and Models for Concurrency. *Springer-Verlag Lecture Notes in Computer Science* **354** 285–363.

Meseguer, J. and Montanari, U. (1990) Petri Nets are Monoids. *Information and Computation* **88** (2) 105–155.

Milner, R. (1989) *Communication and Concurrency*, Prentice Hall.

https://doi.org/10.1017/S0960129597002284 Published online by Cambridge University Press

Milner, R., Parrow, J. and Walker, D. (1992) A Calculus of Mobile Processes, (Parts I and II). *Information and Computation* **100** 1–77.

Montanari, U. and Sassone, V. (1992) Dynamic congruence vs. progressing bisimulation for CCS. *Fundamenta Informaticae* **16** 171–199.

Montanari, U. and Sgamma, M. (1989) Canonical Representatives for Observational Equivalence Classes. In: *Proc. Colloquium on the Resolution of Equations in Algebraic Structures*, Academic Press 292–319.

Montanari, U. and Yankelevich, D. (1994) Combining CCS and Petri Nets Via Structural Axioms. *Fundamenta Informaticae* **20** 193-229.

Nielsen, M., Plotkin, G. and Winskel, G. (1981) Petri Nets, Event Structures and Domains (Part I). *Theoretical Computer Science* **13** (1) 85–108.

Nielsen, M. and Winskel, G. (1982) Models for Concurrency. DAIMI PB 429 (to appear as a chapter in the *Handbook of Logic in Computer Science*, Oxford University Press).

Olderog, E.-R.m (1987) Operational Petri Nets Semantics for CCSP. In: Advances in Petri Nets 87. *Springer-Verlag Lecture Notes in Computer Science* **266** 196–223.

Park, D. (1981) Concurrency and Automata on Infinite Sequences. In: Proc. GI. *Springer-Verlag Lecture Notes in Computer Science* **104** 167–183.

Plotkin, G. (1981) A Structured Approach to Operational Semantics, DAIMI FN-19, Computer Science Dept. University of Aarhus.

Pratt, V. (1986) Modelling Concurrency with Partial Orders. *Int. Journal of Parallel Programming* **15** (1) 381–400.

Rabinovich, A. and Trakhtenbrot, B. (1988) Behaviour Structures and Nets. *Fundamenta Informaticae* **16** 357–404.

Reisig, W. (1985) *Petri Nets: An Introduction*, EATCS Monograph, Springer-Verlag.

Rutten, J. and Turi, D. (1993) Initial and Final Semantics for Concurrency. In: Proc. A Decade of Concurrency. *Springer-Verlag Lecture Notes in Computer Science* **803**.

Segerberg, K. (1968) Decidability of S 4.1. *Theoria* **34**.

Stark, E. W. (1989) Concurrent Transition Systems. *Theoretical Computer Science* **64** 221–269.

https://doi.org/10.1017/S0960129597002284 Published online by Cambridge University Press