ELSEVIER Theoretical Computer Science 149 (1995) 299–332

Theoretical Computer Science

# Location equivalence in a parametric setting[☆]

Ugo Montanari[a,*], Daniel Yankelevich[b,1]

[a] *Dipartimento di Informatica, Universita di Pisa, Corso Italia 40, 56100 Pisa, Italy*
[b] *Departamento de Computación, Universidad de Buenos Aires, Buenos Aires, Argentina*

Received February 1993; revised February 1994
Communicated by V. Kotov

**Abstract**

Location equivalence has been presented in [5] as a bisimulation-based equivalence able to take into account the spatial distribution of processes.

In this work, the parametric approach of [12] is applied to location equivalence. An observation domain for localities is identified and the associated equivalence is shown to coincide with the equivalence introducted in [6, 16]. The observation of a computation is a forest (defined up to isomorphism) whose nodes are the events (labeled by observable actions) and where the arcs describe the sublocation relation.

We show in the paper that our approach is really parametric. By performing minor changes in the definitions, many equivalences are captured: partial and mixed ordering causal semantics, interleaving, and a variation of location equivalence where the generation ordering is not evidenced. It seems difficult to modify the definitions of [6, 16] to obtain the last observation. The equivalence induced by this observation corresponds to the very intuitive assumption that different locations cannot share a common clock, and hence the ordering between events occurring in different places cannot be determined.

Thanks to the general results proved in [12] for the parametric approach, all the observation equivalences described in this paper come equipped with sound and complete axiomatizations.

## 1. Introduction

There are many models and theories of concurrency, which differ notationally or on formal basis or in the assumptions about practical issues. Besides the advantages that this development can offer, the proliferation of models produces confusion. Even their classification is difficult, because of the fact that each one uses a different notation, introduces ad-hoc definitions, and there is no common framework to compare them. In addition, it is quite likely that there is no best model, since each one emphasizes different aspects of systems, and hence has advantages in proving some kind of

* Corresponding author. Email addresses: ugo@di.unipi.it and dany@se.uba.ar

☆ Research partially supported by Hewlett-Packard, Pisa Science Center, within the MDC project.

properties or in specifying a particular type of application. The choice of a model depends on one's preference in a particular situation or when specific premises apply.

We restrict ourselves to models equipped with an operational semantics in Plotkin's structural operational semantics (SOS) style [23], and have some variation of a bisimulation relation [22]. Also, within this class of models, a variety of approaches have been proposed in the literature. Often a lot of work is redone and several definitions are rewritten in an ad hoc manner, changing minor points. This redundancy can be avoided by using a general, parametric approach, capable of describing various models. A parametric approach to the definition of concurrent systems has been proposed in [12] which is the evolution of ideas already present in [10]. In its general outline, this approach consists of four steps:

(1) Define a transition system that captures the desired operational behavior.
(2) Build the possible computations as paths in the transition system, and structure them as an *observation tree* (ordering them by prefix).
(3) Define the observations of computations. These appear as labels on the nodes of the observation tree.
(4) Define an equivalence between observation trees based on the observations defined in step (3).

The third step characterizes different semantics according to the observations they utilize. Once the transition system of step (1) has been fixed, and even choosing as the equivalence of step (4) the same observational equivalence, different observations yield very different semantic models. Each observation corresponds to distinct assumptions about what an observer of the system should see. In this way, just by changing the observation of the computations, many theories are captured. Moreover, this point of view helps in comparing different semantics, showing how to obtain one from the other, and gives a common context to classify them (showing that, for example, they coincide in some of the four steps and differ on others, or showing how the different observations are related). In particular, a general axiomatization of observational congruence for (finite) observation trees is given in [12], which is parametric with respect to the observation. Hence, by defining the semantics within this setting one automatically gets an axiomatization just as an instance of the general result.

In the present paper, we test this approach with location equivalence [5] which has been presented as a bisimulation-based equivalence that takes into account the spatial distribution of processes. Intuitively, the idea is that each action occurs at a location and locations may have sublocations (for example, a fork operation at some place may create two sublocations of that place). For CCS, the assumption is that in $E|E'$ the two subprocesses $E$ and $E'$ are at different locations.

These semantics, which detect the spatial distribution of events, are very useful. For example, in a context of faulty processors, or when information about the assignment process–processor has to be taken into account. For an extended motivation of such an approach we refer the reader to [5].

The framework for the equivalence proposed in [5, 6] is given by an observer who can find out where an event takes place, i.e., the location where an action occurs. More precisely, a dynamic mechanism is given to determine the relative location of an event with respect to the locations of previous events. Hence, it is possible to discern if a number of events occur in the same place or if they are distributed over many locations.

This proposal takes spatial information into account, but not *causal information.* A causal relation between events describes the necessary conditions to enable an event. There are some examples which are distinguished using a causality-sensitive equivalence [11] and not using location equivalence, but there are also examples for which the location approach is more discriminating.

For example, consider the two CCS processes:

$$\alpha.\beta.\gamma.NIL \quad \text{and} \quad (\alpha.\delta.NIL|\bar{\delta}.\beta.\varepsilon.NIL|\bar{\varepsilon}.\gamma.NIL)\backslash\delta\backslash\varepsilon$$

both of which can perform a sequence of three actions, with the causal relation between action occurrences forming a total ordering on both processes. However, the former executes the actions in one location and the latter executes each action at a different location. From a causal point of view both processes are equivalent, but when examining their locations they are different. In this work we show examples of processes which are equivalent with respect to the spatial distribution of events but not with respect to the causal relationship between events.

The idea of defining a process algebra's semantics which are sensitive to the spatial distribution of the bisimulation testing methodology goes back to the work of [7] on distributed bisimulation. However, recent developments [5, 6, 16] have made it easier to embed this approach in the parametric setting outlined above.

In this paper, we fully separate the different conceptual levels of the theory presented in [6], and give an alternative characterization of the proposed semantics. We identify an observation domain for localities and we show that the resulting equivalence coincides with the equivalence introduced in [6, 16]. It turns out that these observations constitute a very simple domain: the observation of a computation is a forest (defined up to isomorphism) whose nodes are the events (labeled with observable actions) and where the arcs describe the sublocation relation. We statically characterize the sublocation relation for CCS by very simple rules. For example, the forests corresponding to the maximal computations of the terms $\alpha.\beta.\gamma.NIL$ and $(\alpha.\delta.NIL|\bar{\delta}.\beta.\varepsilon.NIL|\bar{\varepsilon}.\gamma.NIL)\backslash\varepsilon\backslash\delta$ are shown in Fig. 1(a) and (b), respectively. The natural numbers in the events denote the (total) generation ordering, i.e., the ordering in which events occur.

Reducing an ad hoc definition to a particular instance of a parametric approach is worthwhile, because

- it is convenient to use a general theory as it lets us reuse general theorems, axiomatizations or parametric tools; and
- the resulting characterization clarifies details and highlights points, which could be hidden in the ad hoc definition, helping in the development of new semantics or in testing new ideas.

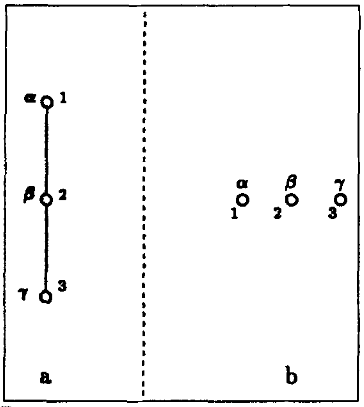

Fig. 1. Observations for localities.

In [7] there is no clear cut separation between the operational semantics of the language, the description of observation (i.e., the assumptions on what an observation of the system is) and the mechanism of the equivalence (i.e., the assumptions on how computations are organized and how the bisimulation is calculated).

In [5, 6, 16] for localities and also in [8, 16] for causal mixed orderings, the transition system that defines the operational semantics is given specifically for the observation. In particular, the states are terms of an enriched language which captures some of the structure of the observations (for example, including causes explicitly in the syntax of the terms). This means that some kind of *unfolding* of the original transition system is performed, where states have information about the computations that can reach them and their observations. For example, in [6], the transition system corresponding to $rec\,x.\,\alpha.x$ has no cycle, since in each transition, a prefix of the form $l$:: is introduced. Hence, the right-hand side of transitions always grow. This definition also implies that the transition systems of [5, 6, 16] are infinitely branching, since in each transition a name for a location can be arbitrarily chosen from an infinite set of location names.

We show in this paper that our approach is really parametric. By slightly altering definitions, many equivalences are obtained, such as partial and mixed ordering causal semantics, interleaving, and a variation of location equivalence where the generation ordering is not evidenced. It is difficult to modify the definitions of [6] to obtain the last observation. The equivalence induced by this observation corresponds to the intuitive assumption that different locations may not share a common clock; hence, the ordering between events occurring in different places cannot be determined.

Thanks to the general results proved in [12] for the parametric approach, all the observation equivalences described in the paper come equipped with sound and complete axiomatizations.

In Section 2 we introduce spatial terms, which provide the notation for the spatial distribution of events, along with the observation domain for localities. In Section 3, we recall the notion of an observation tree from [12] and we enunciate the main result of the work: the equivalence defined in this paper coincides with the location equivalence of [6, 16]. Section 4 presents the other observations and Section 5 discusses some related work.

This paper is the revised and full version of [21], and contains the proof of the coincidence of the parametric location equivalence as defined here and the location equivalence of [6]. The definition of spatial information for CCS and of the operational semantics is done here in an algebraic way, by means of a *structured transition system* following [14, 25], instead of an ad hoc transition system as in [21].

## 2. Observing spatial distribution

### 2.1. Spatial information for CCS

We develop the first step of the methodology skechted out in the introduction for CCS [19]. The first step of our approach concerns the definition of the operational semantics associated with a language.

This leads to the question of what a programming/specification language is. We do not address here this problem with much detail, but we need to answer this question at least for CCS.

It is unsatisfactory to consider a language from a syntactic point of view only. On the other hand, to fix a syntax and a semantics for a language is too rigid. For instance, expressions such as “a truly concurrent semantics for CCS”, “three semantics for CCS”, etc., which have appeared in the literature would not make sense in that case, since a language would only have one semantics. Hence, programming and specification languages have often been considered to be the syntax together with some “basic” meaning. This meaning restricts the range of interpretations that can be given to the operations, but does not completely fix the abstraction level in which a language has to be described (for example, it says that the | operator of CCS is intended to be the parallel combinator but not whether parallelism can be reduced or not to nondeterminism).

This can be formalized by specifying a class of admissible meanings, leaving open which particular model of the class is chosen in each case. One way of specifying a class of models is to constrain the possible meanings by presenting the syntax with a concrete semantics for the language. The admissible models are those which are obtained from this semantics by abstracting some details.

In the context of *bisimulation semantics* it is well known what such concrete semantics are. Usually, a language is equipped with an operational semantics, given by SOS rules. These rules form a (natural) deductive system. A transition between two states is allowed if and only if it can be proved in the formal system. We thus assume that all the information about the intended operational behavior of a language is given by its SOS rules. Hence, all the information about a particular transition from one state to another is given by the *proof* of the transition in the natural deduction system defined by the SOS rules.

The idea of extracting information from proofs of transitions has been proposed in [4]. Here, we associate an $n$-ary operator with each rule with $n$ premises, obtaining an algebra whose terms are proofs of the SOS deduction system. Instead of considering transitions labeled by proofs, as in [4], here the transitions *are* terms of this algebra, whose operations are proof constructors. In this way, the concrete semantics of a language is given in a similar way as the abstract syntax: as terms of an algebra.

A language, then, is defined by a two-sorted algebra, with one sort for the states which are the terms of the language (whose operations are the constructors of the language) and a sort for transitions, whose terms are proofs and whose operations are proof constructors. For CCS this algebra has been presented in [20, 14]. However, instead of many-sorted algebras, we use here *typed algebras* [18].

Typed algebras are not formally introduced here. For the present paper it is enough to say that typed algebras are one-sorted algebras equipped with a binary *typing* relation on the carrier. The signature and set of axioms that define an algebra (or a class of algebras) is called *presentation*. Typed algebras presentations consist of conditional formulas of the form $\Gamma$ *implies* $\Gamma'$, where $\Gamma, \Gamma'$ are sets of atomic formulas. An atomic formula can be either an axiom of the form $t = t'$ or a typing rule of the form $t \div t'$, which are interpreted as "$t$ and $t'$ denotes the same element" and "$t'$ is the type of $t$", respectively. Sets of formulas are interpreted as conjunctions. There is a full calculus (similar to equational calculus) for typed algebras presentations, called *equational typed logic*. Any typed algebra presentation has an associated class of models, the class of algebras satisfying the presentation, which has an initial algebra [18]. In what follows, we introduce the language CCS in this style.

**Definition 1** (*CCS agents*). Let $\Delta$ be a denumberable set of action names, ranged over by $\alpha$, $\beta$. Let $\bar{\Delta}$ *be the set of action complements* (i.e. $\bar{\alpha}$) for all $\alpha \in \Delta$, and $\tau$ an action not appearing in $\Delta, \bar{\Delta}$. Let $\Lambda = \Delta \cup \bar{\Delta} \cup \{\tau\}$, ranged over by $\mu$. Function is extended to all of $\Delta \cup \bar{\Delta}$ in such a way that $\bar{\bar{\alpha}} = \alpha$. Let $X$ be a set of process variables, ranged over by $x$. A *relabelling* function $\Phi$ is a function from $\Lambda$ to $\Lambda$ which respects $\tau$ and the $^-$ operation.

The syntax of CCS is given by the following grammar;

$$e \ ::= \ nil,\ \mu.e, e + e', e|e', e[\Phi], e\backslash\alpha, rec\,x.\,e$$

and CCS terms respect the following axiom:

$$e[rec\,x.\,e/x] = rec\,x.\,e$$

where $e[e'/x]$ denotes the substitution in $e$ of $x$ by $e'$.

CCS agents (denoted $E$, $E_1$, $E_2$, etc.) are those terms which are closed (without free variables) and guarded (each variable appears within a $\mu$. context).

The definition above can be viewed as a typed algebra presentation. For instance, a production $e ::= \mu.e$ of the grammar is interpreted as the conditional formula *$e \div$ term, $\mu \div$ action implies $\mu.e \div$ term.* As usual, the type of a variable can be inferred from the symbol used to denote it. Recursion is handled with the *unfolding* axiom $e[rec\, x.\ e/x] = rec\, x.\ e$.

We assume that CCS agents have the type CCS. A type system assigning type CCS only to closed and guarded CCS terms can be expressed within typed algebras, but it is not particularly simple or suggestive [25]. The substitution operation can also be formalized in this context by means of axioms and typing rules. As usual, we skip the final *nil* in the agents, i.e., $\alpha.nil$ is abbreviated $\alpha$.

**Definition 2** (*CCS transition rules*)

(act) $[\mu, E\rangle : \mu.E \xrightarrow{\mu} E$

(res) $$\frac{t : E_1 \xrightarrow{\mu} E_2, \qquad \mu \notin \{\beta, \bar{\beta}\}}{t \backslash \beta : E_1 \backslash \beta \xrightarrow{\mu} E_2 \backslash \beta}$$

(rel) $$\frac{t : E_1 \xrightarrow{\mu} E_2}{t[\Phi] : E_1[\Phi] \xrightarrow{\Phi(\mu)} E_2[\Phi]}$$

(sum) $$\frac{t : E_1 \xrightarrow{\mu} E_2}{t < + E : E_1 + E \xrightarrow{\mu} E_2} \qquad \frac{t : E_1 \xrightarrow{\mu} E_2}{E + > t : E + E_1 \xrightarrow{\mu} E_2}$$

(par) $$\frac{t : E_1 \xrightarrow{\mu} E_2}{t \rfloor E : E_1 | E \xrightarrow{\mu} E_2 | E} \qquad \frac{t : E_1 \xrightarrow{\mu} E_2}{E \lfloor t : E | E_1 \xrightarrow{\mu} E | E_2}$$

(syn) $$\frac{t : E_1 \xrightarrow{\alpha} E_2,\ t' : E_1' \xrightarrow{\bar{\alpha}} E_2'}{t | t' : E_1 | E_1' \xrightarrow{\tau} E_2 | E_2'}$$

In terms of type algebras, the rule

$$\frac{t : E_1 \xrightarrow{\mu} E_2}{t \rfloor E : E_1 | E \xrightarrow{\mu} E_2 | E}$$

has the following formal meaning. First, the notation $t : E_1 \xrightarrow{\mu} E_2$ represents the formula $t \div$ *transition*, $E_1, E_2 \div$ *CCS*, *source*$(t) = E_1$, *target*$(t) = E_2$, $l(t) = \mu$, where *source*, *target* and $l$ are functions intended to denote the source, the target and the labeling of a transition. Thus the conditional formula associated with the above rule is *$t \div$ transition, source$(t) = E_1$, target$(t) = E_2$, $l(t) = \mu$, $E \div$ CCS implies $t \rfloor E \div$ transition, source$(t \rfloor E) = E_1 | E$, target$(t \rfloor E) = E_2 | E$, $l(t \rfloor E) = \mu$.*

For example, given the rules in Definition 2, using equational typed logic we can prove that

$$(([\alpha, nil\rangle \rfloor \beta)\rfloor \gamma)\backslash\gamma : ((\alpha|\beta)|\gamma)\backslash\gamma \xrightarrow{\alpha} ((nil|\beta)|\gamma)\backslash\gamma$$

where the term $(([\alpha, nil\rangle \rfloor \beta)\rfloor \gamma)\backslash\gamma$ describes the proof of the transition $((\alpha|\beta)|\gamma)\backslash\gamma \xrightarrow{\alpha} ((nil|\beta)|\gamma)\backslash\gamma$ in the SOS deductive system.

Our aim is to represent and to collect the spatial information which is present in CCS agents. Each CCS agent represents many sequential components assembled by the | constructor. That is, in the agent $E|E'$ we assume that agents $E$ and $E'$ are in two different places. The spatial information associated with CCS transitions can be extracted from proofs. Before doing this, however, we have to show how to represent explicitly spatial information.

**Definition 3** (*Spatial terms*). Spatial terms are defined by the following (abstract) syntax: $S ::= \bullet,\ \emptyset,\ S|S$, and respect the following *axiom*: $\emptyset|\emptyset = \emptyset$.

If we accept that the structure given by | in a CCS agent determines the distributed nature of a number of sequential components, spatial terms reflect this structure. A spatial term gives the place of a component or of a group of components.

Intuitively, spatial terms are used to describe information about the physical distribution of events. For instance, the term $\bullet|\emptyset$ describes a place having an idle component at the right, and $\bullet|\bullet$ describes the locality of an event which is performed simultaneously on both sides of the system (a synchronization). The locality of the occurrence of the action $\alpha$ in $((\alpha|\beta)|\gamma)\backslash\gamma$ is described by the term $(\bullet|\emptyset)|\emptyset$.

Notice that spatial terms with one bullet describe the occurrence of actions, while spatial terms with two bullets correspond to communications. Spatial terms with more than two bullets cannot be obtained in CCS, but can be obtained in process algebras whose mechanisms of communication allow synchronizations of more than two processes.

The following grammar defines the type $ST_U$ of spatial terms with exactly one bullet.

**Definition 4** (*Type* $ST_U$). Spatial terms with one bullet are defined by

$$ST_U ::= \bullet,\ ST_U|\emptyset,\ \emptyset|ST_U$$

We define a relation $\leqslant$ between spatial terms which represents sublocality: a spatial term $s$ of type $ST_U$ is a sublocality of another $ST_U$ term $s'$ if and only if $s \leqslant s'$. For instance, $\bullet|\emptyset \leqslant (\bullet|\emptyset)|\emptyset$.

**Definition 5** (*Sublocalities*). Let $s, s_1, s_2$ be $ST_U$ terms. Then, $\leqslant$ is defined inductively by the following rules:

$$\bullet \leqslant s$$

$$s_1 \leqslant s_2 \text{ implies } s_1|\emptyset \leqslant s_2|\emptyset$$

$$s_1 \leqslant s_2 \text{ implies } \emptyset|s_1 \leqslant \emptyset|s_2$$

The above relation defines a tree-like partial ordering between spatial terms. The root is $\bullet$ and a spatial term $s$ is greater than $s'$ (written $s' \leqslant s$) if $s$ is obtained from $s'$ by replacing the bullet with a spatial term. This partial ordering captures the intuitive idea that $\bullet$ represents the whole system and terms of the form $s|\emptyset$ or $\emptyset|s$ are parts of the system, or sublocalities.

We can now show how to obtain spatial terms from transitions. They are defined by structural induction on transition terms.

**Definition 6** (*From transitions to spatial terms*)

$$sp([\mu, E\rangle) = \bullet$$

$$sp(t \langle\!+ E) = sp(t)$$

$$sp(E +\!\rangle t) = sp(t)$$

$$sp(t \rfloor E) = sp(t)|\emptyset$$

$$sp(E \lfloor t) = \emptyset|sp(t)$$

$$sp(t|t') = sp(t)|sp(t')$$

$$sp(t[\Phi]) = sp(t)$$

$$sp(t\backslash\alpha) = sp(t)$$

The algebraic approach can be easily extended: for instance, in [14], CCS is defined as a two-sorted algebra. The labeling operation and the interleaving bisimulation relation are defined via algebraic operations in a category of models. It is also possible to apply this approach to other process description languages provided that their semantics are defined in terms of bisimulation.

**Example 7.** Extend CCS with an encapsulation operation [_], as in [9]. Intuitively, this operator says that the term inside the [_] has to be seen as a black box, and it is not possible to see inside it if the actions are distributed or not. All the actions it performs are causally (and temporally) ordered. Notice that in the interleaving semantics this operator is useless. The agent $[E]$ is seen as a sequential process. Hence, for example, the equation $[\alpha|\beta] = \alpha.\beta + \beta.\alpha$ is valid also in truly concurrent models.

In this algebraic context, we have to extend the signature of the algebra with a new operation (denoted by [_]) and to add one rule:

$$\frac{t: E \xrightarrow{\mu} E'}{[t]:[E] \xrightarrow{\mu} [E']}$$

Since the interpretation of [_] is an encapsulation operation, the intended meaning of [_] is that any action inside it is seen as occurring in a common site and there is no distribution. This fact is reflected in how the spatial term of a transition $[t]$ is defined:

$$sp([t]) = \bullet$$

A CCS computation from $E$ to $E'$ is, as usual, a sequence of transitions such that the target of a transition coincides with the source of the following one.

Computations are introduced in the algebra by means of the following axiom and rules.

**Definition 8** (*Computations*).

$$\frac{t: E \xrightarrow{\mu} E'}{t: E \Rightarrow E'} \quad \frac{t: E \Rightarrow E',\ t: E' \Rightarrow E''}{t;t': E \Rightarrow E''} \qquad t;(t';t'') = (t;t');t''$$

If $t: E \Rightarrow E'$, $E$ is the source of $t$ and $E'$ is the target of $t$ as before. The set of computations is called $\mathscr{C}_{CCS}$.

The presentation we have given so far for CCS has an initial model, which contains CCS agents, (proofs of) transitions, and computations. As usual, the elements of the initial algebra are terms modulo the axioms. Besides the initial one, the set of formulas has many models: some of these models are transition systems, which may be of interest. For instance, in one of these models all transitions with the same label between a pair of agents are identified. Hence, a transition is uniquely determined by its source, target and label. Such a model is equivalent to Milner's original transition system, where transitions are triples.

Since we are only concerned with causality-based, location-based, and interleaving semantics for CCS, some of the information given in the proofs of the transitions is useless. Thus, we can choose another model (another transition system) by abstracting away the useless information from the proofs. In our case, it is enough to add the following rule:

$$\frac{t: p_1 \xrightarrow{\mu} p_2, \qquad t': p_1 \xrightarrow{\mu} p_2, \qquad sp(t) = sp(t')}{t = t'}$$

Transitions are now equivalence classes of terms. In what follows, a transition is defined by its source, target, label and spatial term. We will use the notation $p \xrightarrow{\mu @ s} p'$ (read $\mu_i$ *at* $s_i$) to denote it. Similarly, a computation of the form $t_1;t_2;\cdots;t_n$ is represented by a sequence of transitions $p_0 \xrightarrow{\mu_1 @ s_1} p_1 \xrightarrow{\mu_2 @ s_2} \cdots \xrightarrow{\mu_n @ s_n} p_n$.

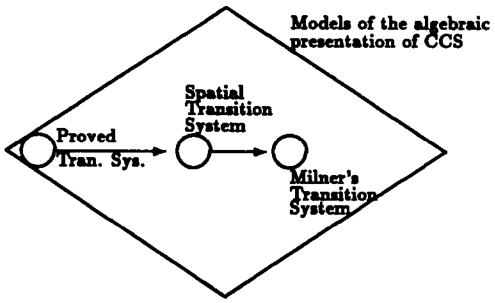
Models of the algebraic presentation of CCS
Spatial Transition System
Proved Tran. Sys.
Milner's Transition System

Fig. 2. The class of transition systems for CCS.

Fig. 2 shows the class of models of the axiomatization for CCS. The initial model is the proved transition system. There is one model where transitions are triples ⟨agent, label, agent⟩, the original Milner's one. The transition system that we consider (called in the figure *spatial transition system*) is in between these models. Notice that a transition of the spatial transition system is uniquely determined by the triple ⟨agent, label@spatial term, agent⟩.

### 2.2. *Observing computations*

The next step in the parametric approach is to determine what is observed out of a single computation. Given a description of a computation of the system, what is seen by an external observer is established by means of an *observation function*, which maps computations of the system into their corresponding observations.

**Definition 9.** An observation **obs** is an observation domain $\langle D, \sqsubset \rangle$, where $D$ is a set of observations and $\sqsubset$ is a partial order on $D$, together with a monotonic function $o$ from computations ordered by prefix to the observation domain.

We say that $O \in D$ is the observation of a computation c if $o(c) = O$.

Monotonicity is the only requirement on observation functions: if one observes a computation and obtains some information, observing a "longer" computation (i.e. a greater computation in the prefix ordering) one obtains more information. Formally, observation domains are ordered, and the observation function has to be monotonic on computations when ordered by prefix. We call an *observation* the observation domain together with the function from the computations to the domain.

In the following definition we introduce the observation for location equivalence, called **loc**.

**Definition 10** (*Observation* **loc**). Given a computation $p_0 \xrightarrow{\mu_1 @ s_1} p_1 \xrightarrow{\mu_2 @ s_2} \cdots \xrightarrow{\mu_n @ s_n} p_n$, we associate to it a structure $\langle Ev, <, \prec, \ell \rangle$, where $\langle Ev, \prec \rangle$ is a forest of events and $<$ is a total *generation* ordering on events, defined in the following way:

- $Ev = \{e_i \mid i = 1 \dots n\}$ is a set of $n$ different events,
- $e_i < e_j$ iff $i \leqslant j$,
- $e_i \prec e_j$ iff$(s_i \leqslant s_j \wedge i \leqslant j)$,
- $\ell(e_i) = \mu_i$.

The observation **loc** is obtained from the structure $\langle Ev, <, \prec, \ell \rangle$ by forgetting all elements having a $\tau$ label, and by restricting the orderings and labeling to the remaining events.

In the labeled forest $\langle Ev, <, \prec, \ell \rangle$ an event $e'$ is a successor of $e$ if $e'$ occurs in the same locality as $e$, or if $e'$ occurs in a sublocality of $e$. Intuitively, a branch in the forest corresponds to a "fork" operation, in which two (or more) sublocalities are created. Only visible actions are taken into account to construct the structure; in fact, this observation is "weak", in the sense that it does not observe $\tau$'s. Moreover, $\tau$'s have no influence in the construction of the observation. This fact corresponds to the idea that communications have no locality, and that locality (unlike causality) cannot be "transmitted" by synchronizations.

The fact that location observations are forests relies strongly on the form of the relation of sublocalities in CCS, which has the form of a tree. In another language, the structure could not verify the property of being a forest, for example, if the language contains a "join" operation, which can put together many localities. In general, one could get an acyclic graph instead of a forest.

We consider observations modulo isomorphism. This means that the names of the events, for example, have no relevance (but of course their labels do); two isomorphic location observations are considered to be the same. Observations are regarded as abstract entities.

**Example 11.** Let $c$ be the computation $(\alpha.\gamma.\delta | \beta.\bar{\gamma}.\varepsilon)\backslash\gamma \xrightarrow{\alpha @ \bullet | \emptyset} (\gamma.\delta | \beta.\bar{\gamma}.\varepsilon)\backslash\gamma \xrightarrow{\beta @ \emptyset | \bullet} (\gamma.\delta | \bar{\gamma}.\varepsilon)\backslash\gamma \xrightarrow{\tau @ \bullet | \bullet} (\delta | \varepsilon)\backslash\gamma \xrightarrow{\delta @ \bullet | \emptyset} (NIL | \varepsilon)\backslash\gamma \xrightarrow{\varepsilon @ \emptyset | \bullet} (NIL | NIL)\backslash\gamma$. The forest $\langle E, <, \prec, \ell \rangle$ corresponding to the **loc** observation of $c$ is shown in Fig. 3. The set $E$ has four events, one for each no-tau transition, labelled with $\alpha$, $\beta$, $\delta$ and $\varepsilon$, respectively. The total ordering $<$ is indicated in the figure via the numbers labeling the events and it is obtained directly from the occurrence ordering in the computation $c$.

The observation domain of locations is ordered in the following way: we have $O \sqsubset O'$ if $O$ is isomorphic to a left-closed subset of $O'$ (i.e. $O'$ is an extension of $O$ with more events). It is easy to verify that this relation defines a partial ordering on location observations, and that the observation function is monotonic with respect on observations.

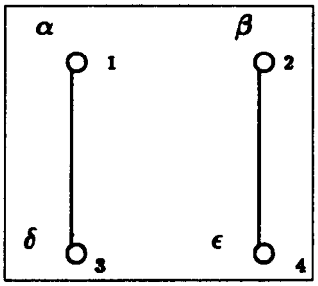

Fig. 3. Observation of $c$ from Example 11.

## 3. Observational equivalences

### 3.1. Observation trees: equivalences and axiomatization

Up to now, the observations of computations have been defined. In this section, we define observational equivalences with respect to such observations. To this aim, the computations an agent can perform are ordered by prefix, generating a tree-like structure. Notice that this structure is fixed, independently of the observation. These structures have been presented in [10] with the name of nondeterministic measurement systems (NMS) and have also been studied, with the name of *observation trees*, in [12] where a parametric approach has been developed. In particular, there is a complete axiomatization of observational congruence on observation trees (for the finite case), parametric with respect to the observation.

**Definition 12** (*Observation tree*). The observation tree corresponding to a CCS agent $E$ with observation **obs** is the tree of all the computations from $E$ ordered by prefix, where each node is labeled by the observation **obs** of the corresponding computation.

**Example 13.** Fig. 4 shows the observation tree corresponding to the agent $((\alpha + \bar{\beta})|(\alpha.\beta.\gamma))\backslash\beta$ with **loc** observations.

We will denote an observation tree as $\langle N, \leqslant, o\rangle$, where $N$ is the set of nodes, $\leqslant$ the ordering relation and $o$ the observation function. A node $n'$ is a *successor* of a node $n$ in $\langle N, \leqslant, o\rangle$ if $n \leqslant n'$; and it is an *immediate* successor if $n < n'$ (i.e. $n \leqslant n'$ and $n \neq n'$) and there is no $n''$ such that $n < n'' < n'$.

Several notions of bisimulation have been defined in [12] on node-labeled structures. In particular, a notion of weak bisimulation has been introduced which is the natural extension of the same notion on arc-labeled structures. Instead, the definition and axiomatization of the *jumping bisimulation* of [12] are simpler. However, it is easy to show that for observations without *atomic actions*, and this is the case for all the observations presented in this work, jumping and weak bismulations coincide. Thus,

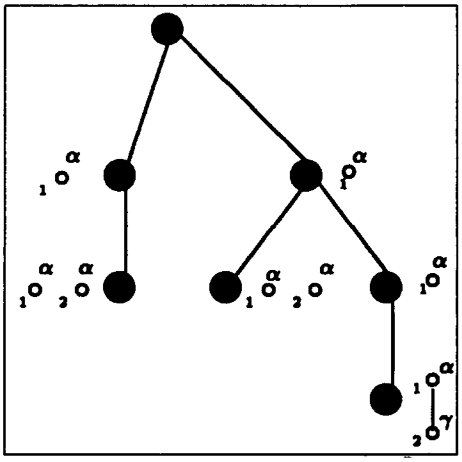

Fig. 4. The observation tree of $((\alpha + \bar{\beta})|(\alpha.\beta.\gamma))\backslash\beta$.

we can use the simpler definition and still be consistent with the natural extension of bisimulation proposed in [12].

**Definition 14** (*Weak bisimulation*). Let $t = \langle N, \leqslant, o\rangle$ and $t' = \langle N', \leqslant', o'\rangle$ be two observation trees (over the same observation domain), and let $R$ be a symmetric binary relation on $N \cup N'$. We will say that $R$ is a weak bisimulation iff $n_1 R n_2$ implies that

- $(n_1) = o(n_2)$, and
- for every immediate successor $n'_1$ of $n_1$ there exists a successor $n'_2$ of $n_2$ such that $n'_1 R n'_2$, and for all $n''_2$ such that $n_2 \leqslant n''_2 \leqslant n'_2$, the observation of $n''_2$ coincides with the observation of $n_2$ or with the observation of $n'_2$

**Definition 15** ((*Jumping*) *bisimulation*). Let $t = \langle N, \leqslant, o\rangle$ and $t' = \langle N', \leqslant', o'\rangle$ be two observation trees (over the same observation domain), and let $R$ be a symmetric binary relation on $N \cup N'$. We will say that $R$ is a bisimulation iff $n_1 R n_2$ implies that

- $o(n_1) = o(n_2)$, and
- for every successor $n'_1$ of $n_1$ there exists a successor $n'_2$ of $n_2$ such that $n'_1 R n'_2$.

In the sequel, we will call just *bisimulation* the *jumping bisimulation.*

**Definition 16** (*Nonatomic observations*). Let **obs** be an observation, with observation domain $\langle D, \sqsubset\rangle$ and observation function $o$ from $\mathscr{C}_{CCS}$ ordered by prefix to $\langle D, \sqsubset\rangle$. Then, **obs** satisfies the *nonatomic property* if and only if for each $n_1, n_2$ such that $n_2$ is an immediate successor of $n_1$ in the observation tree of some $p \in CCS$ the following property holds:

$\forall p' \in CCS$ if $n'_2$ is a successor of $n'_1$ and $o(n'_i) = o(n_i)$ for $i = 1, 2$, then $n'_2$ is an immediate successor of $n'_1$.

A nonatomic observation grows step by step, and it does not allow long observations (which correspond to computations of more than one step) all of a sudden.

**Lemma 17** (Bisimulation and weak bisimulation coincide over nonatomic domains). Let **obs** *be an observation satisfying the nonatomic property. Let* $p_1, p_2 \in CCS$. *Let* $t_i$ *be the observation tree associated with* $p_i$ *for* $i = 1, 2$. *Then, there is a bisimulation between* $t$ *and* $t'$ *if and only if there is a weak bisimulation between them.*

**Proof.** The *if* part is always true, since weak bisimulation is stronger than jumping bisimulation. For the *only if* part, suppose that two trees $t$ and $t'$ are jumping bisimilar but not weakly bisimilar. Hence, there exists a jumping bisimulation $R$ between the nodes of $t$ and those of $t'$, and there exist nodes $n_1, n_2$ of $t$ and $n'_1, n'_2$ of $t'$ such that $n_2$ is an immediate successor of $n_1$, $n'_2$ is a successor of $n'_1$ and $n_i R n'_i$ for $i = 1, 2$ but for some $n''$ which verifies $n'_1 \leqslant n'' \leqslant n'_2$, the observation of $n''$ is different from the observation of $n'_1$ and from the observation of $n'_2$. Since $n_i R n'_i$, the observation of $n_i$ and $n'_i$ coincide, for $i = 1, 2$, and hence, by definition of nonatomicity (since $n_2$ is an immediate successor of $n_1$), $o(n'') = o(n'_1)$ or $o(n'_1)$ or $o(n'') = n'_2$, which is an absurd. □

**Definition 18** (*Observational equivalence and congruence*). Two trees $t$ and $t'$ are *observationally equivalent* if there exists a bisimulation $R$ such that the roots are in the relation $R$, and they are *observationally congruent* if there exists a bisimulation $R$ rooted at the roots of $t$ and $t'$, i.e. such that the roots are in the relation $R$ and no other node is in the relation with them.

We will say that two CCS agents $E$ and $E'$ are observationally equivalent with respect to an observation **obs** ($E \approx_{obs} E'$) if their corresponding **obs** observation trees are observationally equivalent.

An interesting feature of observation trees is that a complete axiomatization has been given for observational congruence (parametric with respect to the observation) in the finite case.

In [12] a particular syntax for representing (finite) observation trees as terms of an algebra is introduced. In fact, given a domain of observation $D$ (in this case a domain of labeled forests), an *observation term* over $D$ is defined in the following way: $O_t$ := NIL, $A(O_t)$, $O_t \oplus O_t$, where $A \in D$. In addition, the operations are partial, and this restriction is expressed via a type system. Typed observation terms are denoted by $P$, $Q$, $R$. The interpretation of the operations on trees is as follows. Constant NIL denotes the empty tree. The operation $A(P)$ prefixes the tree $P$ with a node labeled with $A$, and it is defined only if the root of $P$ (if any) is labeled with an observation equal or greater than $A$ (because the observation function has to be monotonic on computations ordered by prefix). The operation $\oplus$ between observation trees merges two nonempty trees by collapsing their roots, hence it is defined only if both trees have the same observation associated to the root.

Table 1
Axioms for observational congruence

| | |
|---|---|
| (A1) | $P \oplus Q = Q \oplus P$ |
| (A2) | $(P \oplus Q) \oplus R = P \oplus (Q \oplus R)$ |
| (A3) | $\textsc{a}(NIL) \oplus P = P$ |
| (A4) | $P \oplus P = P$ |
| (A5) | $\textsc{a}(\textsc{b}(\textsc{b}(P) \oplus Q)) \oplus \textsc{a}(P) = \textsc{a}(\textsc{b}(P) \oplus Q)$ |

Observation terms, modulo axioms (A1)–(A3) of Table 1, are isomorphic to finite observation trees.

In [12] it is proved that the set of axioms (A1)–(A5), shown in Table 1, are consistent and complete with respect to observational congruence over observation trees. That is, two trees represented by observation terms $P$ and $Q$ are observationally congruent iff (A1)–(A5) $\vdash P = Q$.

**Example 19.** Let $p = (\alpha.\gamma | \bar{\gamma}.\beta)\backslash\gamma + (\gamma.\alpha | \beta.\bar{\gamma})\backslash\gamma$ and $q = \alpha | \beta$. Then, $p \approx_{loc} q$.

Let $\varepsilon$ denote the empty location observation, $\alpha$ denote the location observation with just one element labeled with $\alpha$, and let $\beta$ denote the location observation with one element labeled $\beta$.

Let $\alpha\&\beta$ denote the location observation with two elements, one labeled with $\alpha$ and the other with $\beta$, which are not related in the partial ordering and in the total ordering the element labeled with $\alpha$ is minor than the element labeled $\beta$. The observation term corresponding to $p$ is $\varepsilon(\alpha(\alpha(\alpha\&\beta(NIL)))) \oplus \varepsilon(\beta(\beta(\beta\&\alpha(NIL))))$ and the observation term corresponding to $q$ is $\varepsilon(\alpha(\alpha\&\beta(NIL))) \oplus \varepsilon(\beta(\beta\&\alpha(NIL)))$.

(1) By axiom (A5), letting $P = NIL$, $Q = \textsc{b}(R)$, we deduce

$$\textsc{a}(\textsc{b}(\textsc{b}(NIL) \oplus \textsc{b}(R))) \oplus \textsc{a}(NIL) = \textsc{a}(\textsc{b}(NIL) \oplus \textsc{b}(R))$$

(2) From (1), by applying axiom (A3), we prove the lemma:

$$\textsc{a}(\textsc{b}(\textsc{b}(R))) = \quad \textsc{a}(\textsc{b}(R))$$

(3) As instances of the Lemma in (2), by substitution, we obtain

$$\varepsilon(\alpha(\alpha(\alpha\&\beta(NIL)))) = \varepsilon(\alpha(\alpha\&\beta(NIL))) \quad \text{and}$$

$$\varepsilon(\beta(\beta(\beta\&\alpha)(NIL)))) = \varepsilon(\beta(\beta\&\alpha(NIL)))$$

(4) By congruence,

$$\varepsilon(\alpha(\alpha(\alpha\&\beta(NIL)))) \oplus \varepsilon(\beta(\beta(\beta\&\alpha(NIL)))) = \varepsilon(\alpha(\alpha\&\beta(NIL))) \oplus \varepsilon(\beta(\beta\&\alpha(NIL))).$$

## *3.2. Location equivalence*

In this section the correspondence between the equivalence defined so far and the location equivalence of [6, 16] is established.

In [5, 6, 16] the grammar of CCS is extended with an operator of local cause prefixing $l::E$ where $l$ is a location and $E$ a process. Locations form a fixed, countable set. The idea is that each (sequential) process is assigned to a different location where it executes, and an observer of the system perceives the visible actions of the system and also the location where they take place. More precisely, location transitions are of the form $E \xrightarrow{\alpha;l_1 l_2 \ldots l_n} E'$ where $\alpha$ is the action and $l_1\ l_2 \ldots l_n$ is a sequence of location names. The prefix ordering in strings of location names represents the concept of sublocality. For example in the process $\alpha.(\beta|\gamma)$, action $\alpha$ may occur at any location name, for example, $l$ (names of locations are arbitrarily chosen at each step), and $\beta$ and $\gamma$ both are performed at different sublocations of $l$, for example, $l\ l'$ and $ll''$.

The operational semantics of the language is defined by means of two transition relations: one giving *location transitions* discussed above, and another giving *standard transitions*, where no location is assigned to transitions. The second transition relation defines the $\tau$-moves, corresponding to both $\tau$ prefixing and communication. A bisimulation-like equivalence is defined on these transition systems, and the resulting equivalence (called location equivalence and denoted with $\approx_l$) is shown to be consistent with the original interleaving semantics for CCS agents.

**Definition 20.** Location terms are defined by the following abstract syntax:

$$p = nil,\ \mu.p,\ p+p,\ p|p,\ p[\Phi],\ p\backslash\alpha,\ l::p,\ x,\ rec\,x.\ p$$

where $l \in Loc$ and $x$ is a variable.

Location agents are guarded, closed location terms. We call $P_{loc}$ the set of all location agents.

If $p \in P_{loc}$, $pure(p)$ is the agent obtained by removing all locations from $p$ and $loc(p)$ denotes the set of location names that occur in $p$.

In [6] the syntax of the language is extended also with variables for localities. We consider here the version without location variables, more similar to the syntax of [5, 16].

**Definition 21.** *Location transitions*:

$$\alpha.p \xrightarrow{\alpha;u} l::p \qquad l \in Loc$$

$$p \xrightarrow{\alpha;u} p' \quad \text{implies} \quad k::p\ p \xrightarrow{\alpha;ku} k::p$$

$$p \xrightarrow{\alpha;u} p' \quad \text{implies} \quad p + p'' \xrightarrow{\alpha;u} p' \ \text{ and } \ p'' + p \xrightarrow{\alpha;u} p'$$

$$p \xrightarrow{\alpha;u} p' \quad \text{implies} \quad p|p'' \xrightarrow{\alpha;u} p'|p'' \ \text{ and } \ p''|p \xrightarrow{\alpha;u} p''|p'$$

$$p \xrightarrow{\alpha;u} p' \quad \text{implies} \quad p[\Phi] \xrightarrow{\Phi(\alpha);u} p'[\Phi]$$

$$p \xrightarrow{\beta;u} p' \quad \text{implies} \quad p\backslash\alpha \xrightarrow{\beta;u} p'|\alpha \quad \beta \notin \{\alpha, \bar{\alpha}\}$$

$$p[rec\,x.\,p/x] \xrightarrow{\alpha;u} p' \quad \text{implies} \quad rec\,x.\ p \xrightarrow{\alpha;u} p'$$

**Definition 22.** *Standard transitions*

| | |
|---|---|
| $\mu.p \xrightarrow{\mu} p$ | |
| $p \xrightarrow{\mu} p'$ | implies $k::p \xrightarrow{\mu} k::p'$ |
| $p \xrightarrow{\mu} p'$ | implies $p + p'' \xrightarrow{\mu} p'$ and $p'' + p \xrightarrow{\mu} p'$ |
| $p \xrightarrow{\mu} p'$ | implies $p\|p'' \xrightarrow{\mu} p'\|p''$and $p''\|p \xrightarrow{\mu} p''\|p'$ |
| $p \xrightarrow{\mu} p'$ | implies $p[\Phi] \xrightarrow{\Phi(\mu)} p'[\Phi]$ |
| $p \xrightarrow{\mu} p'$ | implies $p\backslash\alpha \xrightarrow{\mu} p'\|\alpha \quad \mu \notin \{\alpha, \bar{\alpha}\}$ |
| $p[rec\,x.\,p/x] \xrightarrow{\mu} p'$ | implies $rec\,x.\,p \xrightarrow{\mu} p'$ |
| $p \xrightarrow{\alpha;u} p'$ | and $q \xrightarrow{\bar{\alpha};u'} q'$ implies $p\|q \xrightarrow{\tau} p'\|q'$ |

Some definitions (as weak transitions, etc.) are given for transition systems containing an internal $\tau$ action. In this way, some of the theorems of this section are also valid for a general class of systems, not only for the one considered here. This is an advantage of the parametric approach: some general results can be established which are then applied to many particular cases.

**Notation 23** (*Weak transitions*). Let $TS = (S, T, A)$ be a transition system such that $A$ includes a distinguished invisible action $\tau$. Then, for $p, p' \in S$,

- $p \overset{\varepsilon}{\Rightarrow} p'$ iff $p \xrightarrow{\tau} p_1 \xrightarrow{\tau} \cdots \xrightarrow{\tau} p_r = p'$ for some $p_1, \ldots, p_r \in S$ or $p = p'$,
- $p \overset{\alpha}{\Rightarrow} p'$ iff $p \overset{\varepsilon}{\Rightarrow} p_1 \xrightarrow{\alpha} p_2 \overset{\varepsilon}{\Rightarrow} p'$ for some $p_1, p_2 \in S$,
- $c : p \Rightarrow p'$ if $c$ is a computation (i.e. sequence of transitions) from $p$ to $p'$,
- if $c : p \Rightarrow p'$ and $c' : p' \Rightarrow p''$, with $p'' \in S$, $c; c' : p \Rightarrow p''$ is the concatenation of $c$ and $c'$,
- $\varepsilon_p$ is the empty computation from $p$ to $p$.

**Definition 24** (*Location equivalence*). A symmetric relation $R \subseteq P_{loc} \times P_{loc}$ is called a location bisimulation iff $(p, q) \in R$ implies

- $p \overset{\varepsilon}{\Rightarrow} p'$ implies $q \overset{\varepsilon}{\Rightarrow} q'$ for some $q'$ such that $(p', q') \in R$,
- $p \overset{\alpha;u}{\Rightarrow} p'$ implies $q \overset{\alpha;u}{\Rightarrow} q'$ for some $q'$ such that $(p', q') \in R$.

Two agents $p$ and $q$ are said to be location equivalent, $p \approx_l q$, iff there is a location bisimulation $R$ such that $(p, q) \in R$.

The transition system given above, following [6] is infinitely branching, because on each step a location name is chosen from an (infinite) set of location names. Each place has an explicit name, and hence observations are not considered up to isomorphism, that is, the isomorphism has to be explicitly constructed.

As has been pointed out by Kiehn [16], an important point about the name of locations is that, when one chooses a new name to be used in a computation, one has to assure that the name does not appear in the agent. In some sense, the name of a location is similar to the name of an event (the name determines the event in a unique way). Choosing twice the same location name in a computation, no information is added. In fact, the most discriminating computations are those in which a different name is chosen each time. The other computations can be constructed by renaming some locations, and it has been proved [16, Proposition 2.8] that this operation of renaming does not affect the equivalences, i.e., it is irrelevant.

Following these considerations, the proof of coincidence of the semantics proceeds in three steps:

(1) The formal definitions and results showing that infinite branching is not needed are introduced. If one is interested only in CCS terms (and this is our case), it is possible to systematically use natural numbers as location names, introducing them in order and using them only once in each execution sequence. With this aim, a transition system called *numbered localities* is defined.

(2) For the transition system introduced in (1), bisimulation over the transition system and bisimulation over the associated observation trees (considering as observation of a computation just the sequence of its labels), is shown to coincide for CCS terms. This result is proved in a rather general theorem, that may be applied to many transition systems.

(3) The observation tree that we associate with a CCS term and the observation tree associated in step (2) are observational equivalent. That is, it is possible to construct a bisimulation that relates observation trees and the trees of computations introduced in the second step. For this, it is necessary to show how to recover the labels of the trees from location observations and vice versa.

Since the notion of bisimulation of observation trees considered for (2) and (3) is the same, and bisimulation is a transitive relation, the correspondence of the semantics follows from the steps above.

In the sequel, $Loc = \mathbb{N}$, that is, location names are natural numbers.

**Notation 25.** The set of all the computations of the location transition system is called $\mathscr{C}_l$. If $u$ is a sequence of location names and $l$ is a location name, $l \in u$ iff $l$ appears in the sequence $u$. A sequence of elements $e_1, \ldots, e_n$ is denoted $e_1 \cdot e_2 \cdots e_n$, and the concatenation of two sequences[1] $s$ and $s'$ is denoted $s \cdot s'$.

**Definition 26.** Let $c \in \mathscr{C}_l$. We define the norm of $c$, written $\|c\|$, as the number of events in $c$ labeled with an observable action.

The norm of $c$ is less than or equal to the length of $c$.

[1] Note that a different notation is used for concatenation of labels and for concatenation of computations.

**Definition 27.** We type location terms with the following type system:

$$nil : \emptyset \qquad X : \emptyset$$

$$\frac{p : \emptyset}{\mu.p : \emptyset} \qquad \frac{p : M}{p\backslash\alpha : M} \qquad \frac{p : M}{p[\Phi] : M}$$

$$\frac{p : \emptyset, \quad p' : \emptyset}{p + p' : \emptyset} \qquad \frac{p : M, \quad p' : M' \quad M \cap M' = \emptyset}{p|p' : M \cup M'} \qquad \frac{p : M}{rec\, x.\ p : M}$$

$$\frac{p : M, \quad l < min(M)}{(l :: p) : (M \cup \{l\})} \qquad \frac{p : \{1 \ldots n\}}{p : comp(n)} \qquad \frac{p : \emptyset}{p : CCS}$$

where $min(M)$ returns the minimum number in the set $M$. By convention, $min(\emptyset) > n$ for any number $n$.

Notice that not all terms in $P_{loc}$ have a type and that some have more than one type. Each type gives a different kind of information about the term.

**Proposition 28.** *Let $p \in P_{loc}$. Then, $p \in CCS$ iff $p : CCS$.*

If $p : M$ where $M$ is a set, the location names appearing in $p$ are exactly those of $M$.

**Proposition 29.** *Let $p : M$. Then, $loc(p) = M$.*

**Lemma 30.** *Let $p : comp(n)$. Then,*

- *if $p \xrightarrow{\alpha; u \cdot n+1} p'$ then $p'$: $comp(n+1)$,*
- *if $p \xrightarrow{\tau} p'$ then $p' : comp(n)$,*
- *$max(loc(p)) = n$.*

**Proof.** The first point is proved by induction on the rules of location transitions. The second is immediate, since the transition does not change the location names appearing in the term. The third is implied by the previous proposition. □

**Definition 31.** The transition system for numbered localities (NL) has as set of states $S$, where $S = \{p \in P_{loc} \mid \exists n\ p\text{: } comp(n)\}$, and the transitions are defined as follows:

$$p \xrightarrow{\alpha; u \cdot n} p' \text{ iff } p \xrightarrow{\alpha; u \cdot l} p'' \text{ is a location transition and } l \text{ does not appear in } p,$$

where $p' = p''[l \leftarrow n]$ and $n = max(loc(p)) + 1$, and

$$p \xrightarrow{\tau} p' \text{ iff } p \xrightarrow{\tau} p' \text{ is a standard transition}$$

The transition system for numbered localities is finitely branching. In this transition system location names are introduced in order, and each location name is used only once.

For CCS terms, the bisimulation equivalence that results from the use of the NL transition system coincides with location equivalence.

The next definition recalls the weak bisimulation equivalence for states of a transition system.

**Definition 32.** Let $TS = (S, T, A)$ be a transition system such that $A$ includes a distinguished invisible action $\tau$, and let $s_0, s'_0 \in S$. The states $s_0$ and $s'_0$ are observational equivalent (written $s_0 \approx_{TS} s'_0$) iff there exists a symmetric relation $R \subseteq S \times S$ such that $(s_0, s'_0) \in R$, and $(s_1, s'_1) \in R$ implies that

- whenever $s_1 \overset{\varepsilon}{\Rightarrow} s_2$ there exists $s'_2$ such that $s_2 \overset{\varepsilon}{\Rightarrow} s'_2$ and $(s_2, s'_2) \in R$, and
- whenever $s_1 \overset{\varepsilon}{\Rightarrow} s_2$ there exists $s'_2$ such that $s_2 \overset{\varepsilon}{\Rightarrow} s'_2$ and $(s_2, s'_2) \in R$.

**Lemma 33.** *Let $p, p' \in CCS$.* Then *$p \approx_l p'$ iff $p \approx_{NL} p'$.*

**Proof.** (Only if) Since one can choose any location name at each step in the bisimulation, in particular one can choose $n$ as location name in the $n$th step of the bisimulation, and in this way one obtains a bisimulation of NL.

(If) Suppose that $p \approx_{NL} p'$. Then, there exists a bisimulation $R$ of $NL$. Let $R' = \{(p_1, p_2) | \exists p'_1, p'_2$ such that $(p'_1, p'_2) \in R$ and $\rho(p'_i) = p_i, i = 1, 2\}$ where $\rho$ is a function from location names to location names, i.e., a renaming. To check that $R'$ is a location bisimulation is straightforward. □

The set of computations of NL is called $\mathscr{C}_n$. Notice that with $Loc = \mathbb{N}$, we have that $\mathscr{C}_n \subseteq \mathscr{C}_l$.

**Lemma 34.** *Let $p' \in P_{loc}$ such that $p' : comp(n)$. Then, there exist $p \in CCS$ and $c \in \mathscr{C}_n$ such that $c : p \Rightarrow p'$ and $\|c\| = n$.*

**Proof.** We have that $loc(p') = \{1 \ldots n\}$. Suppose that $\alpha_1 \ldots \alpha_n$ do not appear in $p$. Let $p_i = p[i \leftarrow \alpha_i][i+1 \leftarrow \alpha_{i+1}] \ldots [n \leftarrow \alpha_n]$. It is easy to see that $p_i \xrightarrow{\alpha'_i; u \cdot i} p_{i+1}$ $\forall i \in \{1 \ldots n-1\}$ (where $\alpha'_i$ is possibly a relabeling of $\alpha_i$). Then, there exists a computation from $p_1$ to $p'$, and $p_1 \in CCS$. □

**Definition 35.** Let $TS = (S, T, A)$ be a transition system such that $A$ includes a distinguished invisible action $\tau$, and let $s_0 \in S$. Then, $obtree_{TS}(s_0)$ is the observation tree of $s_0$ in $TS$ such that the labels of the nodes are the sequences of observable labels of the computation (i.e. different from $\tau$).

Given a computation $c$, its label is denoted by label($c$).

**Theorem 36.** *Let $TS = (S, T, A)$ be a transition system such that $A$ includes a distinguished invisible action $\tau$, and let $s_0, s_0' \in S$. Then, $s_0 \approx_{TS} s_0'$ iff $obtree_{TS}(s_0)$ is observationally equivalent to $obtree_{TS}(s_0')$.*

**Proof.** (Only if) Suppose that $s_0 \approx_{TS} s_0'$. Then, there exists a bisimulation $R$ such that $(s_0, s_0') \in R$. Let $R' = \{(c_1, c_2) | c_1, c_2$ are computations of $TS$, $label(c_1) = label(c_2)$ and $(target(c_1), target(c_2)) \in R\}$. Then, $R'$ is a bisimulation of observation trees. It is easy to see that $(\varepsilon_{s_0}, \varepsilon_{s_1}) \in R'$. Now, let $(c_1, c_2) \in R'$ and $c_1'$ be a successor of $c_1$. Then, there are two cases:

1. $target(c_1) \overset{\varepsilon}{\Rightarrow} s_1$ and $label(c_1') = label(c_1)$,
2. $target(c_1 \overset{\alpha}{\Rightarrow} s_1$ and $label(c_1') = label(c_1); \alpha$.

Since $(target(c_1), target(c_2)) \in R$, we have that

1. $target(c_2) \overset{\varepsilon}{\Rightarrow} s_2$ and $(s_1, s_2) \in R$ or,
2. $target(c_2) \overset{\alpha}{\Rightarrow} s_2$ and $(s_1, s_2) \in R$,

respectively. Let $c$ be the computation from $target(c_2)$ to $s_2$, and let $c_2' = c_2; c$. The computation $c_2'$ has target $s_2$. Hence, since the labels of $c_1'$ and $c_2'$ coincide and their targets are in $R$, $(c_1', c_2')$ is in $R'$.

(If) Let $R$ be a bisimulation such that $(\varepsilon_{s_0}, \varepsilon_{s_1}) \in R$. Define $R' = \{(target(c_1), target(c_2)) | (c_1, c_2) \in R\}$. It is immediate that $(s_0, s_0') \in R'$. Now, suppose that $(s_1, s_2) \in R'$ and $s_1 \overset{\alpha}{\Rightarrow} s_1'$. Since $(s_1, s_2) \in R'$, there exist computations $c_1$ and $c_2$ such that $(c_1, c_2) \in R'$, $label(c_i) = seq$ and $target(c_i) = s_i$ for $i = 1, 2$. Let $c_1'$ be the result of concatenating $c_1$ and the computation from $s_1$ to $s_1'$. Then, the label of $c_1'$ is $seq$; $\alpha$ and there exists $c_2'$, successor of $c_2$ with label $seq$; $\alpha$. Hence, there exists computation from $target(c_2)$ to $target(c_2')$ with label $\alpha$. Thus, $s_2 \overset{\alpha}{\Rightarrow} s_2'$ (where $s_2' = target(c_2')$) and by definition $(s_1', s_2') \in R'$.

**Corollary 37.** *Let $p, p' \in CCS$. Then, $p \approx_l p'$ iff $obtree_{NL}(p)$ is observationally equivalent to $obtree_{NL}(p')$.*

**Proof.** From the previous lemma, $obtree_{NL}(p)$ is observationally equivalent to $obtree_{NL}(p')$ iff $p \approx_{NL}! \, p'$, and by Lemma 33, $p \approx_{NL} p'$ iff $p \approx_l p'$. □

The function *where* defined below finds out where a location name appears in a term. This place is described with a spatial term, and $\emptyset$ is used to represent that the location name does not appear. For example, $where(\alpha.\beta | 1 :: 2 :: \beta, 2) = \emptyset | \bullet$, and $where(\alpha.\beta | 1 :: 2 :: \beta, 3) = \emptyset | \emptyset = \emptyset$.

**Definition 38** (*Function where*). Given a term $p$ of $P_{loc}$ and a location $i$, the place where location $i$ occurs in $p$ is described by the spatial term given by the following function:

$where(NIL, i) = \emptyset$,

$where(u.p, i) = \emptyset$,

$$where(p|p', i) = where(p, i)|where(p', i),$$

$$where(p + p', i) = \emptyset$$

$$where(p[\Phi], i) = where(p, i),$$

$$where\ (p\backslash\alpha, i) = where(p, i),$$

$$where(i :: p, i) = \bullet,$$

$$where(i :: p, j) = where(p, j) \quad \text{if } i \neq j,$$

$$where(rec\, x.\, p, i) = where(p[rec\, x.\, p/x], i).$$

We now need a function that given a computation in $\mathscr{C}_C CS$ abstract away the irrelevant information and returns a computation of $\mathscr{C}_n$. Evidently, each computation of $\mathscr{C}_n$ may have many corresponding computations in $\mathscr{C}_C CS$, since the spatial information given in the spatial transition system is more concrete than that given in $NL$: each event has an address describing the exact place in the term where it occurs, whether in $NL$ this address is relative to previous events. However, it can be shown that some computations of $\mathscr{C}_{CCS}$ may have more than one $\mathscr{C}_n$ corresponding ones, since the latter do not contain any information about locations. Thus a relation, rather than a function, is defined.

The relation $ch$ defines a correspondence between the computations from an agent in the CCS transition system and in $NL$. The intuition of $c\, ch\, c'$ is that the computations $c$ and $c'$ describe the same behavior, this implies, in particular, that they have the same observation.

**Definition 39** (*Relation ch*). The relation $ch \subseteq \mathscr{C}_n \times \mathscr{C}_{CCS}$, given a computation $p_0 \xrightarrow{\mu_1; u_1} p_1 \cdots p_{n-1} \xrightarrow{\mu_n; u_n} p_n$ of $\mathscr{C}_n$ such that $p_0 : CCS$, is defined as follows:

$$p_0\, ch\, pure(p_0)$$

$$\frac{c\, ch\, c'}{c;\, p_n \xrightarrow{\alpha;\, u_{n+1} \cdot m} p_{n+1}\, ch\, c';\, pure(p_n) \xrightarrow{\alpha\, @\, where(p_{n+1}, m)} pure(p_{n+1})}$$

$$\frac{c\, ch\, c'}{c;\, p_n \xrightarrow{\tau} p_{n+1}\, ch\, c';\, pure(p_n) \xrightarrow{\tau} pure(p_{n+1})}$$

**Lemma 40** (From NL transitions to CCS transitions).

- *Let $p \xrightarrow{\alpha;\, u \cdot m} p'$ be a transition of NL. Then, $pure(p) \xrightarrow{\alpha;\, @\, s} pure(p')$ is a CCS transition, with $s = where(p', m)$.*
- *Let $p \xrightarrow{\tau} p'$ be a transition of NL. Then, there exists s such that $pure(p) \xrightarrow{\tau @ s} pure(p')$ is a CCS transition.*

**Proof.** In order to prove the first item, we actually prove a stronger result, namely: let $p \xrightarrow{\alpha; u \cdot m} p'$ be a location transition, such that $m \notin loc(p)$ and each location name occurs at most once in $p$. Then, $pure(p) \xrightarrow{\alpha; @s} pure(p')$ is a CCS transition, with $s = where(p', m)$.

This stronger result is proved by induction on transitions. For the axiom, $\alpha.q \xrightarrow{\alpha; l} l :: q$, we have that $pure(q)$ is a CCS agent (since $q$ is in $P_{loc}$) and hence $\alpha.pure(q) \xrightarrow{\alpha @ \bullet} pure(q)$ is a CCS transition, and $where(l :: q, l) = \bullet$ by definition.

The inductive step is similar for all the rules. Let us show the case of the left-parallel, which is very representative. Suppose that $p | p'' \xrightarrow{\alpha; u \cdot m} p' | p''$ is a location transition, with $m \notin loc(p | p'')$ and location names occurring only once in $p | p''$. By inductive hypothesis, $pure(p) \xrightarrow{\alpha @ s} pure(p')$ is a CCS transition, with $s = where(p', m)$. By the rules for CCS transitions, $pure(p) | pure(p'') \xrightarrow{\alpha @ s | \emptyset} pure(p') | pure(p'')$ is a CCS transition. By definition, $where(p' | p'', m) = where(p', m) | where(p'', m)$. Since $m \notin loc(p | p'')$, we have that $m \notin loc(p'')$ and it is immediate to see in that case $where(p'', m) = \emptyset$. Hence, $where(p' | p'', m) = where(p', m) | \emptyset = s | \emptyset$. □

We now show some properties of the relation *ch*.

**Proposition 41** (Correctness of *ch*). *If* $c \, ch \, c'$ *then* $c \in \mathscr{C}_n$ *and* $c' \in \mathscr{C}_{CCS}$.

**Proof.** The first argument is clearly in $\mathscr{C}_n$. By induction on the length of the computation, using the previous lemma, the second argument is in $\mathscr{C}_{CCS}$. □

**Proposition 42.** *Let* $c \in \mathscr{C}_n$ *and* $c' \in \mathscr{C}_{CCS}$. *Then,* $c \, ch \, c'$ *implies that* $target(c') = pure(target(c))$.

**Proposition 43.** *Let* $c_1, c_2 \in \mathscr{C}_n$ *and* $c'_1, c'_2 \in \mathscr{C}_{CCS}$ *such that* $c_i \, ch \, c'_i$, $i = 1, 2$ *and* $c_1; c_2$ *and* $c'_1; c'_2$ *are defined. Then,* $c_1; c_2 \, ch \, c'_1; c'_2$.

The information about locations is syntactically included in location agents. When a transition occurs, this information is used to find out the location of the event. The function *site* uses this information to describe locations of events with an observable label. Hence, it takes as arguments two agents and a spatial term and returns the location, described as a string of location names, of an event which occurs in the location described by the spatial term.

**Definition 44** (*Function site*). The (partial) function site: $P_{loc} \times ST \times P_{loc} \to Loc^*$ is defined as follows:

$$site(\alpha.p, \bullet, m :: p) = \varepsilon,$$

$$site(p_1 | p_2, s | \emptyset, p'_1 | p'_2) = site(p_1, s, p'_1),$$

$$site(p_1 | p_2, \emptyset | s, p'_1 | p'_2) = site(p_2, s, p'_2),$$

$site(p_1 + p_2, \bullet, p') = \varepsilon,$

$site(p[\Phi], s, p'[\Phi]) = site(p, s, p'),$

$site(p\backslash\alpha, s, p'\backslash\alpha) = site(p, s, p'),$

$site(m :: p, s, m :: p') = m \cdot site(p, s, p'),$

$site(rec\ x.\ p, s, p') = site(p[rec\ x.\ p/x], s, p'),$

$site(p, s, rec\ x.\ p') = site(p, s, p'[rec\ x.\ p'/x]).$

**Lemma 45** (From CCS transitions to NL transitions).

- *Let* $p \xrightarrow{\alpha @ s} p'$ *be a CCS transition. Then,* $\forall q \in P_{loc}$ *such that* $pure(q) = p$ *and* $q : M$, *there exists* $q'$ *such that* $q \xrightarrow{\alpha; u \cdot n+1} q'$ *is a location transition, and* $pure(q') = p'$ *and* $u = site(q, s, q')$ *for any* $n \geqslant max(M)$.
- *Let* $p \xrightarrow{\tau @ s} p'$ *be a CCS transition. Then,* $\forall q \in P_{loc}$ *such that* $pure(q) = p$ *and* $q : M$, *there exists* $q'$ *with* $pure(q') = p'$ *such that* $q \xrightarrow{\tau} q'$ *is a standard transition.*

**Proof.** By induction on transitions. For the axiom, let $\alpha.p \xrightarrow{\alpha @ \bullet} p$. Let $q : M$ such that $pure(q) = \alpha.p$. Then, $q = i :: \cdots :: n :: \alpha.p$ (since agents of type $M$ cannot contain location names inside any $\alpha.$ context). By the location transition rules, $i :: \cdots :: n :: \alpha.p \xrightarrow{\alpha; i \cdots n \cdot n+1} i :: \cdots :: n :: n+1 :: p$, and, by definition, $site(i :: \cdots :: n :: \alpha.p, \bullet, i :: \cdots :: n :: n+1 :: p) = i \cdots n \cdot site(\alpha.p, \bullet, n+1 :: p) = i \cdots n \cdot n+1$.

The inductive step is very similar for the other rules. Let us show the case of the left-parallel, which is representative.

Let $p | p'' \xrightarrow{\alpha @ s | \emptyset} p' | p''$ be a c CCS transition. Let $q : M$ such that $pure(q) = p | p''$. Then, $q = i :: \cdots :: m :: (\bar{p} | \bar{p}'')$ with $pure(\bar{p}) = p$ and $pure(\bar{p}'') = p''$. Moreover, $\bar{p} : M'$ and $\bar{p}'' : M''$ with $M', M'' \subseteq M$. Note that $\bar{p}$ may contain $j ::$ prefixings. By inductive hypothesis, we have that $\bar{p} \xrightarrow{\alpha; u \cdot n+1} \bar{p}'$ is a location transition and $pure(\bar{p}') = p'$ and $u = site(\bar{p}, s, \bar{p}')$ for any $n \geqslant max(M')$, and in particular for any $n \geqslant max(M) \geqslant max(M')$.

Then, by the location transition rules, $i :: \cdots :: m :: (\bar{p} | \bar{p}'') \xrightarrow{\alpha; i \cdots m \cdot u \cdot n+1} i :: \cdots :: m :: (\bar{p}' | p'')$ is a location transition. Moreover, $site(i :: \cdots :: m :: (\bar{p} | \bar{p}''), s | \emptyset, i :: \cdots :: m :: (\bar{p}' | \bar{p}'')) = i \cdots m \cdot site(\bar{p} | \bar{p}'', s | \emptyset, \bar{p}' | \bar{p}'')$ and by definition of *site* and inductive hypothesis $site\ (\bar{p} | \bar{p}'', s | \emptyset. \bar{p}' | \bar{p}'') = site(\bar{p}, s, \bar{p}') = u$ and thus $site(i :: \cdots :: m :: (\bar{p} | \bar{p}''), s | \emptyset, i :: \cdots :: m :: (\bar{p}' | \bar{p}'')) = i \cdot \cdots m \cdot u$. $\square$

**Lemma 46** (From CCS transitions to NL transitions).

- *Let* $p \xrightarrow{\alpha @ s} p'$ *be a CCS transition. Then,* $\forall q : comp(n)$ *such that* $pure(q) = p$, *there exists* $q'$ *such that* $q \xrightarrow{\alpha; u \cdot n+1} q'$ *in* $NL$ *and* $pure(q') = p'$ *and* $u = site(q, s, q')$.
- *Let* $p \xrightarrow{\tau @ s} p'$ *be a CCS transition. Then,* $\forall q : comp(n)$ *such that* $pure(q) = p$, *there exists* $q'$ *with* $pure(q') = p'$ *such that* $q \xrightarrow{\tau} q'$ *is a transition of* $NL$.

**Proof.** If $q:\mathrm{comp}(n)$ then $q:M$ for $M=\{1,\dots,n\}$. Hence, by the previous Lemma, $q\xrightarrow{\alpha;u\cdot n+1}q'$ is a location transition.

By Lemma 30, if $q:\mathrm{comp}(n)$ and $q\xrightarrow{\alpha;u\cdot n+1}q'$ then $q':\mathrm{comp}(n+1)$ and thus $q\xrightarrow{\alpha;u\cdot n+1}q'$ is in NL. The same reasoning applies to standard transitions. □

The function *where* shows how to describe locations as spatial terms, and the functions *site* recovers locations from agents and spatial terms. We now show the relation between these functions.

**Lemma 47.** *Let $p:comp(n)$ and $p\xrightarrow{\alpha;u\cdot n+1}p'$ be a transition of NL, with where $(p',n+1)=s$. Then, $site(p,s,p')=u$.*

**Fact 48.** *Let $c\in\mathscr{C}_{CCS}$, $c:p\Rightarrow p'$. Then, for all $q\in P_{loc}$ such that $pure(q)=p$ there exist $c'\in\mathscr{C}_n$, $q'\in P_{loc}$ such that $c':q\Rightarrow q'$ and $pure(q')=p'$ and $c'\,ch\,c$.*

**Fact 49.** *Let $c\in\mathscr{C}_n$, $c:p\Rightarrow p'$. Then, there exists $c'\in\mathscr{C}_{CCS}$, $c':pure(p)\Rightarrow pure(p')$ such that $c\,ch\,c'$.*

The label of a computation in $obtree_{NL}(\ )$ is just the sequence of the observable labels that occur in the computation. These sequences can be seen as representations of forests of events. The following definition shows how to build **loc** observations from labels of $obtree_{NL}(\ )$.

**Definition 50.** Let $V$ be the function from labels of $\mathrm{obtree}_{NL}(\ )$ to **loc** observations defined in the following way:

$$V((\alpha_1;u_1)\cdot(\alpha_2;u_2)\cdots(\alpha_n;u_n))=(Ev,\ \leqslant,\prec,lab),$$

where
- $Ev=\{1,\dots,n\}$,
- $j\prec i$ iff $j$ appears in $u_i$,
- $\leqslant$ is the usual order on natural numbers restricted to $Ev$,
- $lab(i)=\alpha_i$.

Actually, the information given in *loc* observations and the information given in the labels of $obtree_{NL}(\ )$ is the same, in the following sense: given any *loc* observation it is possible to construct an equivalent $obtree_{NL}(\ )$, label, and given a $obtree_{NL}(\ )$ label it is possible to construct a corresponding *loc* observation.

**Proposition 51.** *The function $V$ has an inverse $V^{-1}$ given by $V^{-1}(Ev,\leqslant,\prec,lab)=(\alpha_1;u_1)\cdot(\alpha_2;u_2)\cdots(\alpha_n;u_n)$, where, denoting by $e_i$ the* ith *element in the total*

*order* $\leqslant$,
- $\alpha_i = lab(e_i)$,
- $u_i = \{m \mid e_m \prec e_i\}$ *sorted with the usual order on natural numbers,*

*which verifies that* $V(V^{-1}(S)) \cong S$ *and* $V^{-1}(V(S)) \cong S$.

**Lemma 52** (Observations). *Given* $c \in \mathscr{C}_n$ *with* $label(c) = seq$, *let* $c' \in \mathscr{C}_{CCS}$ *such that* $c\, ch\, c'$. *Then,* $V(seq) = obs(c')$.

**Lemma 53.** *Let* $c_1, c_2 \in \mathscr{C}_n$ *and let* $c'_1, c'_2 \in \mathscr{C}_{CCS}$ *such that* $c_i\, ch\, c'_i$. *Then,* $label(c_1) = label(c_2)$ *if and only if* $obs(c'_1) = obs(c'_2)$.

**Definition 54.** Let $\mathcal{O}(p)$ be $obtree_{NL}(p)$ where each label seq is replaced by $V(seq)$.

**Lemma 55.** *Let* $p$: *CCS. Then,* $\mathcal{O}(p)$ *is observational equivalent to the observation tree of* $p$ *in the transition system of CCS with labels* **loc**.

**Proof.** Let $R = \{(c, c') \mid c \in \mathscr{C}_n, c' \in \mathscr{C}_{CCS}, source(c) = p, source(c') = p, c\, ch\, c'\}$. Then, $R$ is a bisimulation of observation trees. Suppose $(c_1, c'_1) \in R$ and let $c_2$ be a successor of $c_1$. Then, $c_2 = c_1; \Delta c$, where $\Delta c: target(c_1) \Rightarrow target(c_2)$. By Lemma 40, there exists $\Delta c': pure(target(c_1)) \Rightarrow pure(target(c_2))$ and $\Delta c\, ch\, \Delta c'$. Since $pure(target(c_1)) = target(c'_1)$ if is possible to define $c'_2 = c'_1; \Delta c'$. From Proposition 43, $c_2\, ch\, c'_2$, and hence there exists a successor of $c'_1$ with $obs(c_2) = obs(c'_2)$ (since they are in $ch$) and $(c_2, c'_2) \in R$.

If $c'_2$ is a successor of $c'_1$, $c'_2 = c'_1; \Delta c'$, and using Lemma 46 the proof proceeds as before. □

Now we are ready to prove the main theorem which shows that both equivalences coincide.

**Theorem 56.** *Let* $p, p' \in CCS$. *Then,* $p \approx_l p'$ *if and only if* $p \approx_{loc} p'$.

**Proof.** It follows immediately from Lemma 33, Lemma 55 and Corollary 37. □

## 4. Other observations

In this section we fix the transition system for CCS and the equivalence (observational equivalence), and we experiment with different observations. In this way, many different equivalences are defined for the same language, changing the assumptions on what an observer can see.

Some of these equivalences can be defined via bisimulation directly on a suitably defined transition system (without explicitly constructing the computations). For example, the equivalence induced by *mixed ordering causal observations*, as presented below, has been defined on a transition system for a language including CCS in [8]. However, it is not immediate to define the equivalence induced by other observations directly on a transition system.

For instance, the observation corresponding to location equivalence gives information on the forest induced by locations but also on the generation ordering. In the classical approach to causality [11] one can choose between a mixed and a partial ordering (in the former the generation ordering is observed, while in the latter it is not). Here, one can choose either to observe or not to observe the generation ordering. The model where the generation ordering is not observed is to some extent more intuitive: it assumes that it is not possible to determine the order between two events that occur in different places. It is not obvious how to modify the definitions of [6, 16] to obtain this equivalence. Instead, in our parametric approach, just by performing a simple change on the observations of computations, namely by forgetting the total ordering and by leaving the rest (transition system, bisimulation) unchanged, this equivalence is obtained.

For each observation in this chapter, an axiomatization of the observational congruence induced can be obtained as an instance of the axioms of Table 1.

## 4.1. Abstract localities

Given a computation, the observation **aloc** is defined as a labeled forest $\langle E, \prec, \ell \rangle$ where $E$, $\prec$ and $\ell$ are defined as in Definition 10 for location observations.

As it has been pointed out above, this observation captures the intutive idea that there is no global clock, and that it is not possible to establish the order of occurrence of events in different, unrelated places. An interesting point is whether this observation induces a different equivalence (and hence different semantics). In fact, the equivalence $\approx_{loc}$ is finer than $\approx_{aloc}$, and the example showing this fact is the same example used to show that mixed ordering causal observations induces a finer ordering than partial ordering causal observations [11].

**Example 57.** Let $\mathbf{p} = (((\alpha|\bar{\delta}) + \alpha.\beta)|\alpha.\delta.\beta)\backslash\delta$ and $\mathbf{q} = ((\alpha + \alpha.\delta.\beta)|\alpha.\delta.\beta|\bar{\delta})\backslash\delta$. Then, we have that $\mathbf{p} \approx_{aloc} \mathbf{q}$ *but* $\mathbf{p} \not\approx_{loc} \mathbf{q}$.

The example above, while being finite, is not trivial. It has been checked, together with other examples of this work, by the automatic parametric verification tool of [15] (which is based on the theory of observation trees).

Using **loc** observations, when **p** has performed two $\alpha$ actions in two different locations, it has decided if the $\beta$ will take place in the same location where the first or the second $\alpha$ has been performed. Instead, when **q** has executed two $\alpha$ actions in two different locations, it may still choose where to perform $\beta$, and thus the processes are different. This difference is detected by the total ordering, but leaves no trace in the tree.

Although the intuitive nature of localities is to assign locations to events up to isomorphism, it is sensitive to a global clock, distinguishing two computations where an action occurs in a location which has been "the first" in executing an $\alpha$, or in a location which has been "the second" in performing an $\alpha$.

## 4.2. *Observing causality*

A different, well-known notion of observation is suggested by the causal approach. In these models, the observer of a system is supposed to be able to determine the causal dependencies between event occurrences.

Here we first define mixed ordering causal observations. This observation has been proposed previously in [11], and the equivalence that it induces coincides with the *causal bisimulation* of [8] and with the *history preserving bisimulation* of [24].

The main difference between this observation and the one defined for localities is the treatment of $\tau$'s. While localities are not affected by events labeled with $\tau$'s, causes can be inherited by means of communications. Hence, we have to extend the sublocality relation to deal with events labeled with $\tau$'s, i.e., with spatial terms with more than one $\bullet$.

We will now define a relation between spatial terms which includes the idea of overlap: two spatial terms are overlapping if they both cover some part of the system.

**Definition 58** (*Overlapping localities*)

$$s \neq \emptyset \text{ implies } \bullet \frown s,$$

$$s_1 \frown s_2 \text{ implies } s_1|s' \frown s_2|s'',$$

$$s_1 \frown s_2 \text{ implies } s'|s_1 \frown s''|s_2.$$

A dot in a spatial term represents the position where an action takes place. The idea of overlapping is that some "active" position of one spatial term coincides, or is a refinement of, an active position of the other spatial term. Notice that $\frown$ is neither symmetric nor antisymmetric.

A particular case of overlapping localities is when $s_1$ and $s_2$ are of type $ST_U$ (i.e. have only one $\bullet$) and both $s'$ and $s''$ in the last two rules are $\emptyset$: in that case the sublocality ordering is obtained.

**Definition 59** (*Mixed and partial ordering causal observations*). *Given a computation* $p_0 \xrightarrow{\mu_1 @ s_1} p_1 \xrightarrow{\mu_2 @ s_2} \cdots \xrightarrow{\mu_n @ s_n} p_n$ *we associate with it a structure* $\langle Ev, <, \underset{\sim}{\sqsubset}, \ell \rangle$, *where*

- $Ev = \{e_i \,|\, i = 1 \ldots n\}$ is a set of different events,
- $e_i < e_j$ iff $i \leqslant j$,
- $s_i \frown s_j$ and $i \leqslant j$ implies $e_i \underset{\sim}{\sqsubset} e_j$,
- $e_i \underset{\sim}{\sqsubset} e_k$ and $e_k \underset{\sim}{\sqsubset} e_j \; e_i \underset{\sim}{\sqsubset} e_j$,
- $\ell(e_i) = \mu_i$.

The **mo** observation of the given computation is given by the structure obtained from $\langle Ev, <, \underset{\sim}{\sqsubset}, \ell\rangle$ by forgetting all elements having a $\tau$ label, and by restricting the orderings and labeling to the remaining events.

The observation for causal partial orderings is obtained from causal mixed orderings as abstract localities from localities, i.e., by forgetting the total ordering $\leqslant$.

The construction of the causal partial ordering given above is straightforward. The only difference between the approaches based on causality and on locality is whether to allow causality to be inherited from communications or not. Thus, $\prec \subseteq \underset{\sim}{\sqsubset}$. The causality relation is the least partial order which contains $< \cap \frown$.

The coincidence of the mixed ordering equivalence of [11] with the observational equivalence with **mo** observations defined above is immediate. In fact, the definition of [11] employed NMS, but they are essentially the same as observation trees. A proof of the coincidence of mixed ordering bisimulation, causal bisimulation and history-preserving bisimulation for event structures can be found in [2].

## *4.3. Partial ordering localities*

Locality and mixed ordering observations give different information about the system. Combining them, we have a third kind of observations, namely a structure $\langle Ev, <, \prec, \underset{\sim}{\sqsubset}, \ell\rangle$ where $\langle Ev, <, \prec, \ell\rangle$ is the **loc** observation and $\langle Ev, <, \underset{\sim}{\sqsubset}, \ell\rangle$ is the **mo** observation. Since $\prec \subseteq \underset{\sim}{\sqsubset}$, we can give a graphical representation of these observations with colored partial orderings, i.e. with partial ordering diagrams where arcs belonging to both relations are painted with a color different from the color used for $\underset{\sim}{\sqsubset}$ (here we use plain lines for arcs belonging to both relations and dotted lines for arcs which are only in the causal ordering). We will call this observation **pol**, for partial ordering localities. The equivalence induced by this observation coincides with the local/global cause equivalence of [16].

**Example 60.** Fig. 5 shows the colored partial orderings corresponding to computation $c$ of Example 11 and to the computation $(\alpha.\gamma.\varepsilon|\beta.\bar{\gamma}.\delta)\backslash\gamma$ $\xrightarrow{\alpha@\cdot|\emptyset}(\gamma.\varepsilon|\beta.\bar{\gamma}.\delta)\backslash\gamma \xrightarrow{\beta@\emptyset|\cdot}(\gamma.\varepsilon|\bar{\gamma}.\delta)\backslash\gamma \xrightarrow{\tau@\cdot|\cdot}(\varepsilon|.\delta)\backslash\gamma \xrightarrow{\delta@\cdot|\emptyset}(\varepsilon|NIL)\backslash\gamma \xrightarrow{\varepsilon@\cdot|\emptyset}(NIL|$ $NIL)\backslash\gamma$.

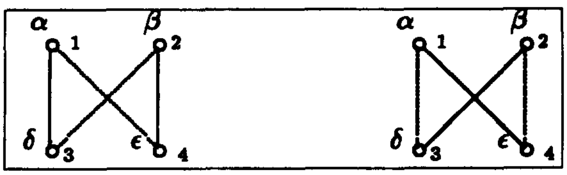

Fig. 5. Partial ordering localities observations for $(\alpha.\gamma.\delta|\beta.\bar{\gamma}.\varepsilon)\backslash\gamma$ and $(\alpha.\gamma.\varepsilon|\beta.\bar{\gamma}.\delta)\backslash\gamma$.

Notice that with location or colored partial (or mixed) orderings they are different. However, with partial (or mixed) causal orderings the two computations have the same observation.

The equivalence $\approx_{pol}$ is stronger than both the causality-based an the locality-based equivalences, i.e. $E \approx_{pol} E'$ implies $E \approx_{mo} E'$ and $E \approx_{loc} E'$, and it is stronger than the intersection of the equivalences $\approx_{mo}$ and $\approx_{loc}$, as shown by the following example.

**Example 61.** Let $r = (\alpha|\beta) + (\alpha.\gamma|\bar{\gamma}.\beta)\backslash\gamma + \alpha.\beta$ and $s = (\alpha|\beta) + \alpha.\beta$. Then, $r \approx_{mo} s$, since $(\alpha.\gamma|\bar{\gamma}.\beta)\backslash\gamma$ is causally absorbed by $\alpha.\beta$, and $r \approx_{loc} s$, since $(\alpha.\gamma|\bar{\gamma}.\beta)\backslash\gamma$ is locally absorbed by $(\alpha|\beta)$, but $r \not\approx_{pol} s$. The last inequality arises because $r$ has a computation whose observation is the structure $\langle\{1,2\}, <, \prec, \precsim, \ell\rangle$, with $1 < 2$, $\ell(1) = \alpha$, $\ell(2) = \beta$, $1 \precsim 2$ and $\prec = \emptyset$ while $s$ has not.

## 4.4. Weak and strong interleaving observations

Of course, as a particular case, interleaving observations can be defined in this context. The interesting point is that the difference between weak and strong equivalence is given here at the level of observations, and not at the level of equivalences.

**Definition 62** (*Interleaving observation*). Given a computation $p_0 \xrightarrow{\mu_1 @ s_1} p_1 \xrightarrow{\mu_2 @ s_2} \cdots \xrightarrow{\mu_n @ s_n} p_n$, we associate with it the sequence of labels $\mu_1; \mu_2; \cdots; \mu_n$.

The weak version of the interleaving observation is obtained by deleting $\tau$ labels from the sequence $\mu_1; \mu_2; \cdots; \mu_n$.

The correctness of this definition is an immediate corollary of Theorem 36.

**Corollary 63.** *Let $E, E' \in CCS$. Then, $E \approx_{int} E'$ iff $E \approx E'$, where $\approx$ denotes the interleaving weak observational equivalence* [19].

Fig. 6 shows all the weak observations presented in this work. An arrow from $x$ to $y$ means "$y$ is more abstract than $x$", and hence the equivalence induced by observation $x$ is finer than the equivalence induced by $y$. Observations which are not related by an arrow are incomparable, and counterexamples have been shown in the paper.

## 5. Related work

Location equivalence has been introduced in [5, 6, 16]. A static version of localities, where location names are associated with sites at "compile time" has been proposed in

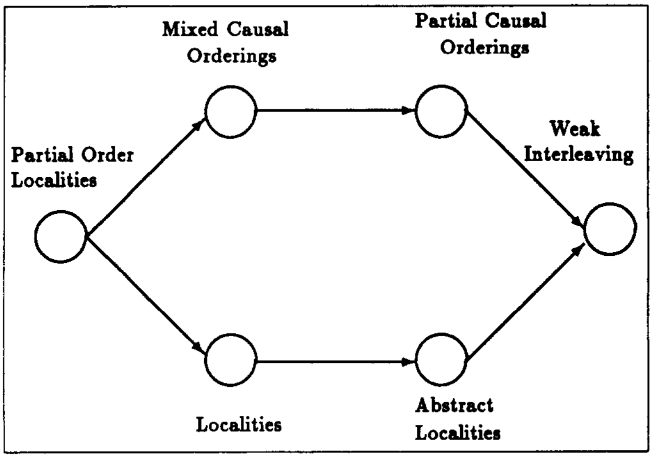

Fig. 6. The observations presented in the paper and their relationships.

[3] for a subset of CCS. This version may be useful for the development of algorithms for location equivalence checking. A variation of location equivalence has been introduced in [17] with many examples showing that this notion is useful for specifying protocols, in particular taking into account routing and distribution properties.

Kiehn studied the relationship between locality and causality-based equivalences [16] and proposed a model merging them. She extended the notion of *locations* to *local and global causes.* While local causes correspond to locations, global causes are used to represent the causal relation, in a similar way to the *backward pointers* of [8]. This approach can be viewed as dual to ours: the information needed to keep track of the history of a process is inserted into the process itself, by means of a constructor defined with this purpose. Dually, we here observe the computations, not the states.

However, given a generic observation, no systematic way has been given in [16, 8] for inserting the information in the processes, and the syntax of the processes cannot be chosen independently of the technical requirements, since certain constructors are used to represent part of the history of the process. Another general approach to observational equivalences for locality and causality, similar to the previous one, is discussed in [13]. This approach is more semantic than the one in [16], since the information about the history is not syntactically included in the processes. The main difference with our work is that the first and the second step (and part of the third step) of our approach are merged in [13] in the construction of a tree whose arcs are labeled by proofs. An advantage of the proposals in [16, 13] is that they are incremental, in the sense that the observation of each transition yields a single event. However, their proposals are likely to be less general than ours: the generation ordering is intrinsically included in their semantics and abstracting out from it may be difficult if at all possible. In addition, our notion of observation, being defined on computations, is more explicit and makes comparisons easier, since each observation has an associated domain.

## Acknowledgements

We wish to thank Ilaria Castellani for her helpful comments on an early version of this work, and Ed Harcourt for correcting many english flaws. Two anonymous referees also gave insightful comments.

## References

[1] *Proceedings 19th ICALP*, Vienna, Lecture Notes in Computer Science, Vol. 623 (Springer, Berlin, 1992).

[2] L. Aceto, History preserving, causal and mixed-ordering equivalence for stable event structures (note), Technical Memo HPL-PSC-91-28, Hewlett-Packard Laboratories, Pisa Science Center, Pisa, 1991; *Fundam. Inform.*, to appear.

[3] L. Aceto, A static view of localities, Rapport de Recherche 1483, INRIA, Sophia Antipolis, Valbonne, July 1991, *Formal Aspects Comput.*, to appear.

[4] G. Boudol and I. Castellani, A non-interleaving semantics for CCS based on proved transitions, *Fundam. Inform.* **11** (1988) 433–452.

[5] G. Boudol, I. Castellani, M. Hennessy and A. Kiehn, Observing localities, *Theoret. Comput. Sci.* **114** (1993) 31–61.

[6] G. Boudol, I. Castellani, M. Hennessy and A. Kiehn, A theory of processes with localities, Tech. Report 13/91, University of Sussex, December 1991, *Formal Aspects Comput.*, to appear.

[7] I. Castellani and M. Hennessy, Distributed bisimulations, *J. ACM*, **36** (1989) 887–911.

[8] P. Darondeau and P. Degano, Event structures, causal trees and refinements, *Theoret. Comput. Sci.* to appear.

[9] N. De Francesco, U. Montanari and D. Yankelevich, Axiomatizing different views of distributed systems, in: *Proc. ERCIM Workshop on Theory and Practice in Verification*, Pisa, 1992.

[10] P. Degano, R. De Nicola and U. Montanari, Observational equivalences for concurrency models, in: M. Wirsing, ed. *Formal Description of Programming Concepts – III, Proc. 3rd IFIP WG 2.2 Working Conf.*, Ebberup, 1986 (North-Holland, Amsterdam, 1987) 105–129.

[11] P. Degano, R. De Nicola and U. Montanari, Partial orderings descriptions and observations of nondeterministic concurrent processes, in: J. d. Bakker, W.-P. d. Roever and G. Rozenberg, eds., *REX School/Workshop on Linear Time, Branching Time and Partial Order in Logics and Models for Concurrency*, Noordwijkerhout, Lecture Notes in Computer Science, Vol. 354 (Springer, Berlin, 1989) 438–466.

[12] P. Degano, R. De Nicola and U. Montanari, Universal axioms for bisimulation, *Theoret. Comput. Sci.* **114** (1993) 63–91.

[13] P. Degano and C. Priami, Proved trees, in: *Proc. 19th ICALP*, Vienna, Lecture Notes in Computer Science, Vol. 623 (Springer, Berlin, 1992) 629–640.

[14] G. Ferrari, Unifying models of concurrency, Ph.D. Thesis, Dipartimento di Informatica, Universitȧ di Pisa, 1990. Available as report TD-4/90.

[15] P. Inverardi, C. Priami and D. Yankelevich, Verification of concurrent systems in SML, in: *Proc. ACM SIGPLAN Workshop on ML and its Applications* (1992) 169–175.

[16] A. Kiehn, Local and global causes, Tech. Report 342/23/91, Technische Universitat Munchen, 1991.

[17] P. Krishnan, Distributed CCS, in: *Proc. CONCUR'91*, Lecture Notes in Computer Science, Vol. 527 (Springer, Berlin, 1991) 393–407.

[18] V. Manca, A Salibra and G. Scollo, Equational type logic, *Theoret. Comput. Sci.*, **77** (1990) 131–159.

[19] R. Milner, *Communication and Concurrency* (Prentice-Hall, Englewood Cliffs, NJ, 1989).

[20] U. Montanari and D. Yankelevich, An algebraic view of interleaving and distributed operational semantics for CCS, in: *Proc. Category Theory and Computer Science '89*, Lecture Notes in Computer Science, Vol. 389 (Springer, Berlin 1991) 5–20.

[21] U. Montanari and D. Yankelevich, A parametric approach to localities, in: *Proc. 19th ICALP*, Vienna, Lecture Notes in Computer Science, Vol 623 (Springer, Berlin, 1992) 617–628.

[22] D. Park, Concurrency and automata on infinite sequences, in: P. Deussen, ed. *Proc. 5th GI Conf.* Lecture Notes in Computer Science, Vol. 104 (Springer, Berlin, 1981) 167–183.

[23] G. Plotkin, A structural approach to operational semantics, Report DAIMI FN-19, Computer Science Department, Aarhus University, 1981.

[24] A. Rabinovich and B. Trakhtenbrot, Behavior structures and nets, *Fundam. Inform.*, **11** (1988) 357–404.

[25] D. Yankelevich, *Parametric view of process description languages*, Ph.D. Thesis, Dipartimento di Informatica, Università di Pisa, 1993. Available as report TD-23/93.