Causal Behaviours and Nets

Joost-Pieter Katoen

Department of Computer Science, University of Twente
P.O. Box 217, 7500 AE Enschede, The Netherlands

"Abandonment of causality as a matter of principle should be permitted only in the most extreme emergency"
Albert Einstein, 1924[1]

Abstract. Specification formalisms in which causality and independence of actions can be explicitly expressed are beneficial from a design point of view. The explicit presence (or absence) of a causal dependency between actions can be used effectively during the design. We consider a specification formalism in which causal relations between actions play a central role and provide a semantics in terms of (an extension of) labelled place/transition nets. The behaviour of nets is defined by labelled partially ordered sets.

1 Introduction

A number of formal description techniques are known that assist the development of complex distributed systems. Although such techniques are a significant contribution to the treatment of complexity in, for instance, protocol design and verification, an engineer may wonder whether the abstractions made by these approaches are appropriate. From an engineer's view, the preferred scenario is to first investigate which design concepts are convenient and what their intended meaning should be [33]. Once this is investigated, an appropriate and intuitively appealing representation of these concepts can be chosen, eventually resulting in a "design" formalism, a formalism that can be used at different stages of the design trajectory. Such considerations have led to the introduction of a design model which comprehends design concepts, an overall design methodology, and a design formalism [32, 33]. A full treatment of this model is given in [8].

Needless to say, a (design) formalism should have a formal semantics. This gives a precise meaning to the introduced concepts, forms a basis for the definition of equivalence notions and implementation relations that are suitable to effectively support the design trajectory, and is indispensable for the development of tools. A formal treatment of the formalism in [8, 32, 33] is currently lacking. This paper presents a formal semantics of the kernel of the proposed formalism — causal relations and causal behaviours — in terms of place/transition nets [27].

The formalism is adherent to the true concurrency (or partial-order) approach. In a nutshell the main novelties of the formalism are the inclusion of labelled internal actions on which no synchronisations may take place but which are visible by the

[1] A. Pais, *'Subtle is the Lord' – The science and life of Albert Einstein.* Oxford Univ. Press, 1983.

environment (unlike the usual invisible τ or $\mathbf{i}$), a generalisation of sequential composition for composing behaviours, and the treatment of different types of causality at a syntactical level which enables expressing interleaving of actions syntactically.

The basic ingredient of the formalism are causal relations between actions. Causal relations can be of various types. In current process algebras [1, 14, 23] such relations are induced by means of parallel composition, sequential composition, and so on, and are restricted to be of the 'and'-type. For example, $a\,;\,b$ specifies that b is enabled once a has occurred and $(a \,|||\, b)\,;\,c$ specifies that c is enabled once both a and b have occurred (in either order). Using causal relations this is denoted as $a \rightarrow b$ and $a \;\wedge\; b \rightarrow c$, respectively. The latter construct is called 'and'-causality. The natural complementary construct, $a \;\vee\; b \rightarrow c$, called 'or'-causality, denotes that c is enabled once either a or b has occurred before. The fact that b is disabled by a is denoted by $\neg\, a \rightarrow b$ ('not'-causality), and means that once a occurs b becomes disabled.

Event-based models and Petri nets are well-suited for providing a semantics to the formalism at hand. Due to the presence of not-causalities a type of event structure, like prime and stable event structures [34], whose interpretation can be given in terms of families of configurations is insufficient as pointed out in [21, 28]. More sophisticated event-based models [15, 21, 25] support the notion of asymmetric conflict, a notion that corresponds to not-causality. The mapping of the formalism to such models is, however, not straightforward, and — in some cases — even impossible due to some (technical) restrictions. In this paper we take Petri nets [27, 29] as a basis for providing a semantics. Petri nets are based on causal relations (i.e., flows) between transitions, have an intuitive graphical representation, an extensive tool support is available, and a lot of theories on extensions are known — like timed, stochastic, and coloured nets — that may be useful when considering the full formalism [8]. A summary of net theory is given in Appendix A. The merits of giving an operational interpretation of a specification formalism in terms of nets is well recognized — a large number of attempts for other formalisms like CSP [10], ACP [31], and LOTOS [9] have been made.

The main contribution of this paper is the definition of a relation between the concepts of causal relation and behaviour, adopted from [8, 32, 33], and nets in a fully compositional way. This work can be used as a basis for defining an operational semantics of the full formalism, incorporating data and time, to determine the expressivity of the formalism, and to prove certain properties of specifications. The behaviour of nets is defined by families of labelled partially ordered sets [28]. Based on this partial-order semantics two expansion theorems are defined that enable composed behaviours to be rewritten into equivalent monolithic ones[2].

2 Related Work

The use of causalities for the description of distributed systems is not new and is, in fact, introduced by Lamport, with his so-called 'happened before' relation[3] between events [20]. The description of a distributed system by means of combining causal

[2] Limits on the available space prevent us from providing all proofs of our results; cf. [18].

[3] The term 'happened before' is somewhat misleading as it suggests a temporal ordering, not a causal ordering. A causal ordering implies a temporal one, but not vice versa.

relations — to our knowledge — originates from [19] by introducing and-causalities. A more formal treatment of causal relations (again only including and-causalities) is described in [5]. The incorporation of disabling (or not-causalities) between actions in a specification language was coined originally by Janicki [16].

An elaborated treatment of causal relations (excluding not-causalities) can be found in [11, 13]. The most extensive treatment of causality is given in [12] which incorporates all causality types. Gunawardena uses the notion of geometric automata for the operational characterization of his formalism. It turns out that the set of traces of a geometric automaton is a pure transitive trace set. (A trace set is transitive if the prefix ordering on traces up to permutation is a partial order. Pureness means that each action appears at most once in any trace.) However, not the entire class of pure transitive trace sets can be generated by the class of geometric automata. Although geometric automata are quite similar to the formalism we present in this paper they are not that expressive and do not support the notion of behaviour.

[4] introduces a (graphical) specification language for concurrent systems. In their notation, Bolognesi and Ciaccio use and- and or-causalities plus a symmetric not-causality to express the constraints on the temporal ordering of events, the so-called 'when' constraints. By using symmetric not-causality a standard choice is expressed. They lack, however, a means to express ordinary (asymmetric) not-causalities. In [4] a preliminary interleaving semantics for their notation is given. The true concurrency semantics in this paper can be used for the 'when' constraints of their notation.

We conclude by stating that various research activities on the fundamental notion of causality are currently ongoing, e.g. on how to express causalities [26], and the use of causalities for the design and synthesis of distributed algorithms [30].

3 Causal Relations and Behaviours

The behaviour of a distributed system consists of actions and causal relations between them. A causal relation describes the conditions for a single action to happen. The class **Pr** of these conditions, called *preconditions*, is defined as follows. Let $a \in \mathbf{Act}$ and $n \in \mathcal{I}$, where **Act** denotes the universe of action symbols and $\mathcal{I}, \mathcal{J}$ arbitrary finite index sets.

Definition 1. $E ::= \text{true} \mid \text{false} \mid a \mid \neg\, a \mid \mathit{entry}_n \mid (E \wedge E) \mid (E \vee E)$.

$a \in E$ is true iff a occurs in E. Preconditions differ from boolean expressions since $\neg$ is only applicable to actions, not to preconditions in general, e.g. $\neg\,(a \wedge b)$ is not an allowed precondition, whereas $\neg\, a \vee \neg\, b$ is. Unary operators bind stronger than binary ones and $\wedge$ and $\vee$ bind equally strong. Parentheses are omitted when this does not introduce ambiguities.

A causal relation consists of an action and a precondition stating the necessary and sufficient conditions for the enabling of that action.

Definition 2. (E, a) is a *causal relation* for $a \in \mathbf{Act} \cup \{\, \mathit{exit}_k \mid k \in \mathcal{J} \,\}$ and $E \in \mathbf{Pr}$.

A causal relation (E, a) is usually written as $E \rightarrow a$. It is assumed that an action is unique and occurs at most once in a system run[4]. This, in fact, implies that each

[4] In literature this is also often called an *event*.

causal relation $E \rightarrow a$ should be read as $E \ \wedge \ new_a \rightarrow a$ where new_a is a predicate which is true iff a has not yet appeared. For the sake of convenience we omit this syntactical construct and leave it implicit. In the semantics — of course — this interpretation will be made explicit.

Example 1. Some examples of causal relations and their intuitive meaning are as follows. $a \rightarrow b$ denotes that a is causal for the occurrence of b, that is, if b happens then a must have happened before, true $\rightarrow a$ expresses that a is always allowed to occur, $\neg\, a \rightarrow b$ expresses that if a and b both occur, b should occur before a, and $\neg\, a \ \vee \ b \rightarrow c$ expresses that if c happens then either a has not happened before or b has happened before.

The interpretation of $a \rightarrow b$ is that b is enabled — and thus may occur — once a appeared. This is called may-enabling and is the opposite of forcing the appearance of b once a has occurred (must-enabling).

$entry_n$ and $exit_k$ ($n \in \mathcal{I}$, $k \in \mathcal{J}$) denote so-called *entry* and *exit points*. An entry point defines a point from which actions in a behaviour can be enabled and an exit point defines the conditions that can be used to enable actions of other behaviours. This allows one to connect behaviours by connecting entries and exits. Entry and exit points are just placeholders and should be distinguished from actions.

Using causal relations, behaviours can be constructed. A behaviour consists of two types of actions: those that are internal, and those on which participation with other behaviours is possible. Elements of the latter category are called *interactions*. Opposed to process algebras where internal actions are turned into silent actions (e.g. denoted **i**), internal actions, or simply *actions*, are labelled.

Definition 3. A *simple behaviour* B is a triple $\langle I, A, S\rangle$ with I ($I \subseteq \mathbf{Act}$), a finite set of interactions, A ($A \subseteq \mathbf{Act}$), a finite set of actions A such that $I \cap A = \varnothing$, and a finite set of causal relations $S = \{\, E_1 \rightarrow a_1, \ldots, E_n \rightarrow a_n \,\}$ satisfying:

1. $\{\, a_1, \ldots, a_n \,\} \setminus \{\, a_i \mid \exists\, k : a_i = exit_k \,\} = I \cup A$
2. $\forall\, E_i \rightarrow a_i, E_j \rightarrow a_j \in S : i \neq j \ \Rightarrow \ a_i \neq a_j$
3. $\forall\, E \rightarrow a \in S, b \in E : b \in I \cup A \ \vee \ (\exists\, n : b = entry_n)$.

The first two conditions imply that each action, interaction, and exit of B appears precisely once in the right-hand side of one of the causal relations of B. (Here, it is assumed that each exit is uniquely numbered.) The last condition states that any action in the precondition of one of the causal relations of B must either be an action of B or an entry. $I \cup A$ is called the *alphabet* of B.

Example 2. The most elementary behaviour is $B_\varnothing = \langle \varnothing, \varnothing, \varnothing \rangle$. Other example behaviours are $\langle \{\, a, b \,\}, \varnothing, \{\, \neg\, a \rightarrow b, \neg\, b \rightarrow a \,\} \rangle$, $\langle \{\, a, b \,\}, \varnothing, \{\, a \rightarrow b, b \rightarrow a \,\} \rangle$, $\langle \{\, d \,\}, \{\, f, e \,\}, \{\, \text{true} \rightarrow d, d \rightarrow e, e \ \vee \ d \rightarrow f \,\} \rangle$, and $\langle \{\, a, b \,\}, \varnothing, \{\, entry_1 \rightarrow a, a \rightarrow b, b \rightarrow exit_1 \,\} \rangle$.

Let G be a set of interactions, $G \subseteq \mathbf{Act}$, and C a partial function, not necessarily injective, mapping entries of one behaviour to exits of others. The class of finite behaviours, **Beh**, is defined as:

Definition 4. $B \ ::= \ \langle I, A, S \rangle \ \big|\ (B \,||_G\, B) \ \big|\ (B \rightarrowtail_C B)$.

The simplest behaviour is the behaviour defined according to Definition 3. It does not consist of any sub-behaviours. There are two ways in which behaviours can be composed into more complex behaviours. In the first type of composition synchronisation on a set of interactions G is performed. This is quite similar to multi-way synchronisation. For $B_1 \,||_G\, B_2$ it is assumed that $G \subseteq I_1 \cup I_2$ and that B_1 and B_2 have disjoint sets of actions, entries, and exits.

In $B_1 \rightarrowtail_C B_2$, conditions of exit points of B_1 are connected to entry points in B_2. C defines which exits of B_1 are connected to which entries of B_2. Recall that entry and exit points are just placeholders and should be distinguished from actions. Linking entries and exits is therefore essentially different from synchronisation on interactions. This form of composition can be considered as a generalization of sequential composition. It is assumed that two behaviours whose exits and entries are linked have disjoint sets of actions, interactions, entries, and exits.

$||_G$ and $\rightarrowtail_C$ bind equally strong and associate to the left. Parenthesis are omitted when not causing ambiguities.

Note 5. We assume that in a causal relation of the form $E \rightarrow exit_k$, E does not contain any $\neg\, a$ occurrences. It would be difficult to interpret such causal relations in connecting behaviours as the non-occurrence of an action in one behaviour becomes a precondition for an action in another behaviour. Although this assumption is not essential for our results, it allows for a smoother and more intuitive presentation.

4 Semantical Model

One of the major characteristics of Petri nets is that causal dependencies and independencies between actions may be represented explicitly [27]. Independent actions are not temporally ordered — as in interleaving semantics — but are treated independently as intended. In addition, Petri nets are simple and intuitive and allow for the representation of interleaving of actions. Independence and interleaving can thus be distinguished[5]. These aspects make Petri nets well suited for providing a formal, operational interpretation of causal relations and behaviours.

The basic idea is that transitions correspond to actions, and flows to causal relations. The concept of may-enabling is modeled conveniently in nets as enabled transitions may fire but are not forced to do so. The net corresponding to $a \rightarrow b$ is depicted in Fig. 1(a). The causal dependence between a and b is explicitly reflected in the net — transition b is only enabled after a has fired — and the fact that each action is unique is represented by equipping each transition with a single place initially containing one token and having no input transitions. Fig. 1(b) and (c) show the nets corresponding to $a \ \wedge \ b \rightarrow c$ and $a \ \vee \ b \rightarrow c$, respectively. In (b) transitions a and b must have fired in order to enable c, and in (c) either a or b must have fired to enable c.

Not-causalities seem to require an extension of nets, known as *inhibitor arcs* [24]. An inhibitor arc connects a place s to a transition t such that t is enabled iff

[5] This does not hold when an interpretation of nets is given in terms of sets of step sequences where each step consists of a single transition. In this interpretation independent transitions can only fire in an interleaved order (see also section 6).

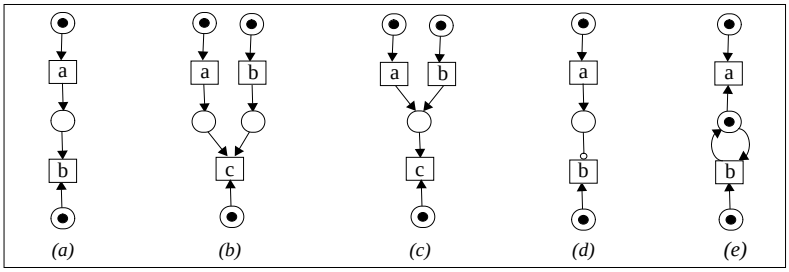

Fig. 1. Elementary causal relations in nets.

s contains no tokens (and, of course, its normal places contain sufficient tokens). The net corresponding to $\neg\, a \rightarrow b$ is depicted in Fig. 1(d) where an inhibitor arc is represented as a line with a small circle at its end[6]. As inhibitor arcs complicate the interpretation of nets considerably [6] and hinder compositionality we stick to traditional nets (see Fig. 1(e)).

Entries and exits are placeholders and are represented as places. A similar treatment of entries and exits is present in Petri Boxes [3].

4.1 Labelled Place/Transition Nets

As a semantical model for causal behaviours we consider a generalisation of place/transition nets (P/T-nets, see Appendix A) with unbounded capacity of places and arc weights equal to one. The generalisation is called labelled P/T-net, simply called net in the sequel, and is a P/T-net equipped with a labelling of places (with a set of entries and exits), and a labelling of transitions (with action symbols). Formally,

Definition 6. A labelled net is a 6-tuple $N \mathrel{\widehat{=}} (S, T, M^0, F, \lambda, \mu)$ where (S, T, M^0, F) is a P/T-net, λ a labelling of places $S \longrightarrow \mathcal{P}(\{\, e_k \mid k \in \mathcal{J} \,\} \cup \{\, x_n \mid n \in \mathcal{I} \,\})$ and μ a labelling of transitions $T \longrightarrow \mathbf{Act}$, satisfying

$$(\forall\, s \in S : |\, \{\, x_n \in \lambda(s) \,\}\, | \leq 1) \;\wedge\; (\forall\, t_1, t_2 \in T : t_1 \neq t_2 \;\Rightarrow\; \mu(t_1) \neq \mu(t_2)) \quad .$$

λ labels a place of N with a set of entries and exits. (e_k is used as abbreviation for $entry_k$ and x_n for $exit_n$.) A place is labelled with at most one exit point. We consider nets upto isomorphism and duplication equivalence [3]. So, two nets which only have different place and/or transition identities and/or duplicated places[7] are considered to be equivalent. Nets are represented with a minimal number of places.

Some notational remarks are in order. Places, transitions, flows, and markings are represented in the usual way. Identities of transitions are indicated at the right or left side of transitions, and transition labels are indicated in the center of a transition. Place labels are indicated in the middle of a place, or at the left or right side of it.

[6] Although it seems that both transitions a and b are initially enabled, this is not the case according to the formal semantics in [6].

[7] Two places are duplicated when they have the same initial marking, label, presets and postsets.

Place labels equal to $\varnothing$ are omitted. When there exists a flow between s and t and vice versa this is indicated by a bidirectional arrow.

A distinction with other types of nets is that we use transition sets (see Definition 43), instead of transition bags [6] as generalisation of a single transition. This simplification is possible since any transition in our nets can fire at most once.

4.2 Operations on Labelled Nets

In this section we define two operations — *compose* and *join* — on nets which will be used in the next section for defining a compositional semantics of causal behaviours. The basic idea behind composition is to combine transitions, while for joining places are combined. The formal definition of the two operations is technically involved. The intuition of the operators will be illustrated by examples (see also next section). Let $N_i = (S_i, T_i, M_i^0, F_i, \lambda_i, \mu_i)$ for i=1, 2 be labelled P/T-nets.

Composition At the composition of nets N_1 and N_2 equally labelled transitions whose label is in some given set of actions G are combined. Transitions whose label is not in G are paired with the auxiliary symbol '$*$' such that all transitions of the resulting net are pairs. Transitions in N_1 (N_2) labelled with an label in G, but for which there is no accompanying transition in N_2 (N_1) disappear in the resulting net. As a prerequisite to the full definition of composition we define the pairing of sets of transitions with respect to set G. For set of transitions T and action set G, let $T^G = \{\, t \in T \mid \mu(t) \in G \,\}$ and $T^{\bar{G}} = T \setminus T^G$.

Definition 7. $T_1 \circledast_G T_2 \mathrel{\widehat{=}} (T_1^{\bar{G}} \times \{*\}) \cup (\{*\} \times T_2^{\bar{G}}) \cup \{\, (t_1, t_2) \in T_1^G \times T_2^G \mid \mu_1(t_1) = \mu_2(t_2) \,\}$.

Proposition 8. $G = rng(\mu_1) \cap rng(\mu_2) \;\Rightarrow\; \forall\, t_1 \in T_1^G : (\exists\, t_2 \in T_2^G : (t_1, t_2) \in T_1 \circledast_G T_2)$.

Definition 9. $S_1 +_G S_2 \mathrel{\widehat{=}} \{\, s \in S_1 \mid {}^\bullet s \cup s^\bullet \neq \varnothing \wedge (({}^\bullet s \cup s^\bullet) \times (T_2 \cup \{*\})) \cap T_1 \circledast_G T_2 = \varnothing \,\} \cup \{\, s \in S_2 \mid {}^\bullet s \cup s^\bullet \neq \varnothing \wedge ((T_1 \cup \{*\}) \times ({}^\bullet s \cup s^\bullet)) \cap T_1 \circledast_G T_2 = \varnothing \,\}$.

For composition, the incoming (outgoing) places of a paired transition are formed by the union of the incoming (outgoing) places of its components. The places of the combined net are the places of N_1 and N_2 where the places which have become disconnected are eliminated. The marking of places is left unchanged.

Definition 10. Given action set G, the composition of N_1 and N_2 satisfying $S_1 \cap S_2 = \varnothing$, denoted by $N_1 \oplus_G N_2$, is net $(S, T, M^0, F, \lambda, \mu)$ with

1. $S = (S_1 \cup S_2) \setminus (S_1 +_G S_2)$
2. $T = T_1 \circledast_G T_2$
3. $M^0 = (M_1^0 \cup M_2^0) \restriction S$
4. ${}^\bullet(t_1, t_2) = {}^\bullet t_1 \cup {}^\bullet t_2$ and $(t_1, t_2)^\bullet = t_1^\bullet \cup t_2^\bullet$ with ${}^\bullet * = *^\bullet = \varnothing$
5. $\lambda = (\lambda_1 \cup \lambda_2) \restriction S$
6. $(t_1 = * \;\Rightarrow\; \mu((t_1, t_2)) = \mu_2(t_2))$ and $(t_1 \neq * \;\Rightarrow\; \mu((t_1, t_2)) = \mu_1(t_1))$.

$\restriction$ means restriction. If G equals the set of all common action symbols of two nets (that is, $G = rng(\mu_1) \cap rng(\mu_2)$) then $\oplus_G$ is abbreviated as $\oplus$. Note that in this case no transitions are eliminated.

Example 3. The composition operator is exemplified in Fig. 2. In the first example (a) the two transitions labelled c, transitions z and 3, are combined and the rest of the transitions remain. For examples (b) and (c) the argument nets are identical and only the set G differs. In case (b) there is no transition in the right net labelled b and transition x disappears from the resulting net. Transitions labelled a and c are combined. Note that in the resulting net transition c can never appear. In case (c) no combination of transitions labelled b is required ($b \notin G$) and only transitions labelled a and c are combined.

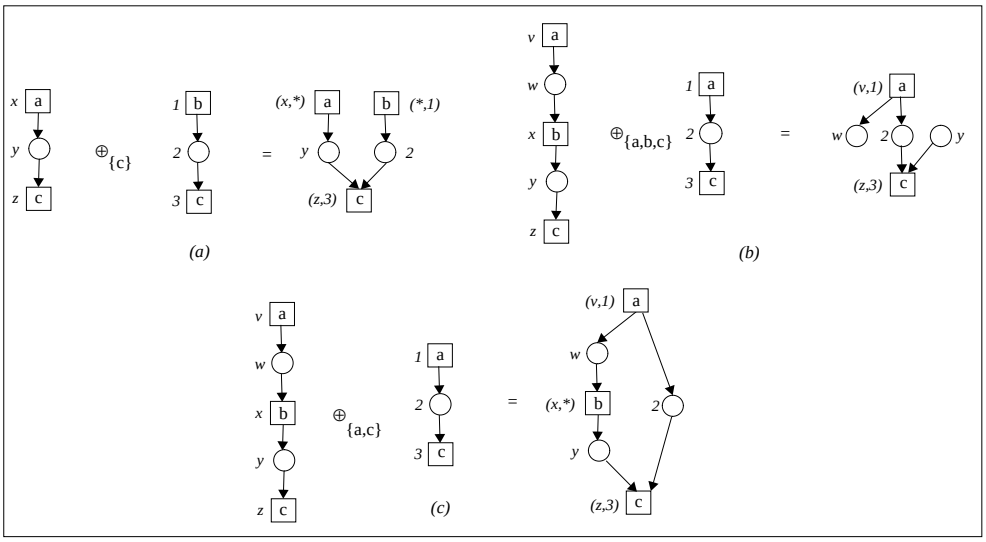

Fig. 2. Some example applications of $\oplus_G$.

Proposition 11. *$N_1 \oplus_G N_2$ is a labelled P/T-net.*

Joining The idea of joining is combining places of nets. The places of N_1 and N_2 that are to be combined are determined by some predicate P. Places that are not paired remain unchanged (technically this is achieved by combining them with a special symbol '$*$' in a similar way as transitions are paired before). Thus, all places of the resulting net are pairs. For predicate P and sets of places S_1, S_2, pairing of places is defined by:

Definition 12. $S_1 \star_P S_2 \mathrel{\widehat{=}} \{\, (s_1, s_2) \mid P((s_1, s_2)) \,\} \cup (\{\, s \in S_1 \mid \forall\, s' \in S_2 : \neg\, P((s, s')) \,\} \times \{\, * \,\}) \cup (\{\, * \,\} \times \{\, s \in S_2 \mid \forall\, s' \in S_1 : \neg\, P((s', s)) \,\})$.

The marking of a paired place is the sum of the markings of its components, and the labelling of such a place is the union of the labels of its components minus the labels in C. Equally labelled transitions are paired as well, and other transitions are left unchanged (by combining them with '$*$'). Labelling of transitions is analogous to labelling in composition. A flow between (t_1, t_2) and (s_1, s_2) is introduced iff a flow exists between t_1 and s_1 in N_1, or between t_2 and s_2 in N_2.

Definition 13. The join of nets N_1 and N_2, $N_1 \div_{C,P} N_2$, is $(S, T, M^0, F, \lambda, \mu)$ with

1. $S = S_1 \star_P S_2$
2. $T = T_1 \circledast_G T_2$ where $G = rng(\mu_1) \cap rng(\mu_2)$
3. $M^0((s_1, s_2)) = M_1^0(s_1) + M_2^0(s_2)$ where $M_i^0(*) = 0$.
4. $\quad {}^\bullet(t_1, t_2) = \{\, (s_1, s_2) \in S \mid s_1 \in {}^\bullet t_1 \ \vee \ s_2 \in {}^\bullet t_2 \,\}$
   and $(t_1, t_2)^\bullet = \{\, (s_1, s_2) \in S \mid s_1 \in t_1^\bullet \ \vee \ s_2 \in t_2^\bullet \,\}$ where ${}^\bullet * = *^\bullet = \varnothing$
5. $\lambda((s_1, s_2)) = (\lambda_1(s_1) \cup \lambda_2(s_2)) \setminus dom(C \cup C^{-1})$ where $\lambda_i(*) = \varnothing$
6. $(t_1 = * \ \Rightarrow \ \mu((t_1, t_2)) = \mu_2(t_2))$ and $(t_1 \neq * \ \Rightarrow \ \mu((t_1, t_2)) = \mu_1(t_1))$

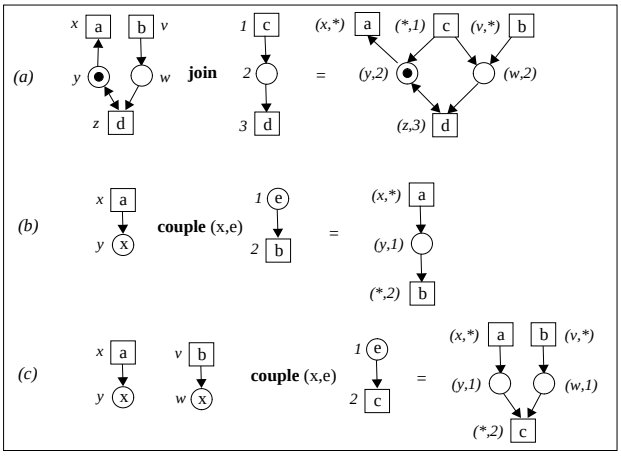

Fig. 3. Some example applications of $\div_{C,P}$.

In the next section we use $\div_{C,P}$ in two ways. In the first way, the intuitive idea is to combine places of two nets that form an input to an equally labelled transition in both nets and, in addition, pairing places that have a common exit label. This operation is denoted $\bowtie$ (join). For the second operation, denoted $\curvearrowright_C$ (couple), the idea is to pair places in N_1 labelled with exits in C to entries in N_2 which are combined according to C.

Definition 14. 1. For $C = \varnothing$ and $P((s_1, s_2)) = (\exists\, t \in s_1^\bullet \cap s_2^\bullet : \mu_1(t) = \mu_2(t)) \ \vee$ $(\exists\, x : x \in \lambda_1(s_1) \cap \lambda_2(s_2))$ let $N_1 \bowtie N_2 \mathrel{\widehat{=}} N_1 \div_{C,P} N_2$.
2. For C a partial function from entries of N_2 to exits of N_1 and $P((s_1, s_2)) =$ $(\exists\, x \in \lambda_1(s_1), e \in \lambda_2(s_2) : (x, e) \in C)$ let $N_1 \curvearrowright_C N_2 \mathrel{\widehat{=}} N_1 \div_{C,P} N_2$.

Example 4. An example of $\bowtie$ is given in Fig. 3(a). Places of the argument nets which are connected to the common transition d are paired, and the d transitions

are combined. Figs. 3(b,c) depict examples of $\curvearrowright_C$. Here, places labelled x in the left argument are combined with the place labelled e in the right argument.

Proposition 15. *$N_1 \curvearrowright_C N_2$ is a labelled P/T-net.*

5 A Compositional Semantics

A semantics is a mapping which assigns to every syntactical construct a meaning or interpretation, that is, an element of a semantical domain. In this section we define a net semantics which assigns to every behaviour $B \in \mathbf{Beh}$ a net $[\![B]\!]$. This is done in a compositional way, starting from a mapping of causal relations onto nets.

5.1 Simple Behaviours

The net representation of elementary causal relations is illustrated in Fig. 4. The following causal relations are considered: (a) true $\rightarrow a$, (b) false $\rightarrow a$, (c) $entry_n \rightarrow a$, (d) $a \rightarrow b$, (e) $\neg\, a \rightarrow b$, (f) true $\rightarrow exit_k$, (g) false $\rightarrow exit_k$, (h) $entry_n \rightarrow exit_k$, (i) $a \rightarrow exit_k$, (j) $a \rightarrow a$, and (k) $\neg\, a \rightarrow a$. For the sake of brevity, a complete definition is omitted. Remark that false $\rightarrow a$, $a \rightarrow a$, and $\neg\, a \rightarrow a$ all denote that a is permanently disabled.

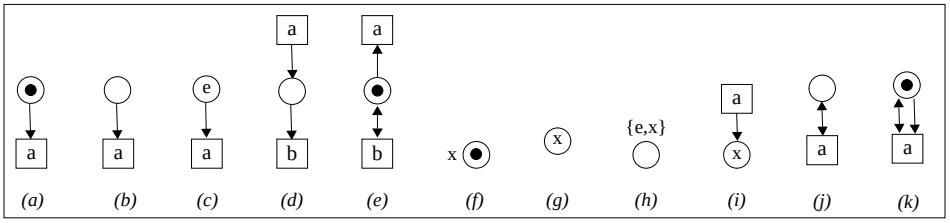

Fig. 4. Net semantics of elementary causal relations.

Definition 16. For $E_1, E_2 \in \mathbf{Pr}$ and $e \in \mathbf{Act} \cup \{\, exit_k \mid k \in \mathcal{J} \,\}$ we define:

- $[\![(E_1 \ \wedge \ E_2) \rightarrow e]\!] \mathrel{\widehat{=}} ([\![E_1 \rightarrow e]\!] \oplus [\![E_2 \rightarrow e]\!])$
- $[\![(E_1 \ \vee \ E_2) \rightarrow e]\!] \mathrel{\widehat{=}} ([\![E_1 \rightarrow e]\!] \bowtie [\![E_2 \rightarrow e]\!])$.

Proposition 17. *$[\![E \rightarrow e]\!]$ is a labelled P/T-net.*

The semantics of some elementary causal relations is illustrated in Fig. 5. The nets corresponding to the following causal relations are presented: (a) $a \ \wedge \ b \rightarrow c$, (b) $a \ \vee \ b \rightarrow c$, (c) $\neg\, a \ \wedge \ b \rightarrow c$, (d) $\neg\, a \ \vee \ b \rightarrow c$, and (e) $\neg\, a \ \vee \ \neg\, b \rightarrow c$.

Recall that each causal relation $E \rightarrow a$ should be read as $E \ \wedge \ new_a \rightarrow a$, where predicate new_a is true iff a has not occurred yet. The most obvious way to represent this matter is to equip each transition with an additional input state (without any incoming transitions) initially marked with a single token. For N this operation is denoted as $\wr N$ (a formal definition is straightforward and omitted). Then,

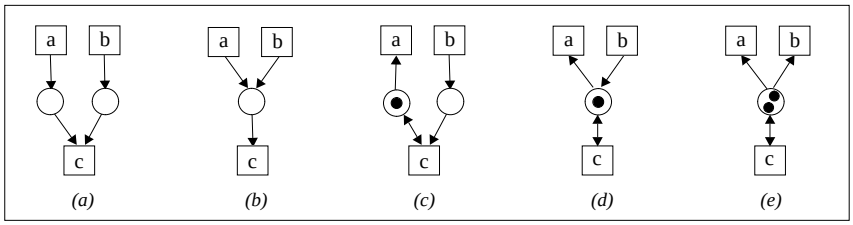

Fig. 5. Net semantics of some and, or causal relations.

Definition 18. For simple behaviour B, $[\![B]\!]$ is defined as follows ($n \geq 0$):

$[\![\langle I, A, \{ E_1 \rightarrow a_1, \ldots, E_n \rightarrow a_n \} \rangle]\!] \mathrel{\widehat{=}} \wr ([\![E_1 \rightarrow a_1]\!] \oplus \ldots \oplus [\![E_n \rightarrow a_n]\!])$.

Example 5. Consider B with alphabet $\{ a, b \}$ and causal relations $\{ \neg a \rightarrow b, \neg b \rightarrow a \}$. Then $[\![B]\!] = \wr([\![\neg a \rightarrow b]\!] \oplus [\![\neg b \rightarrow a]\!])$ is depicted in Fig. 6(a). This corresponds to the standard choice in process algebras. Thus, standard choice is not considered as a primitive construct, as in process algebras, but is expressed using causalities. As a second example consider alphabet $\{ a, b \}$ and set of causal relations

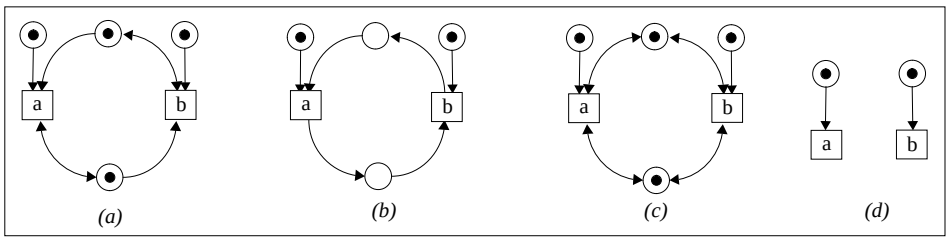

Fig. 6. Semantics of some simple behaviours.

$\{ a \rightarrow b, b \rightarrow a \}$. The resulting net is depicted in Fig. 6(b). Neither a nor b can fire, and thus, the set of reachable markings only consists of the initial marking, M^0.

Example 6. Consider behaviours B_1 and B_2 with alphabet $\{ a, b \}$ and the following sets of causal relations: $B_1 = \{ b \vee \neg b \rightarrow a, a \vee \neg a \rightarrow b \}$, and $B_2 = \{ \text{true} \rightarrow a, \text{true} \rightarrow b \}$. In $[\![B_1]\!]$ transitions a and b can execute one after the other, but not simultaneously, cf. Fig. 6(c). (Note that we could omit one of the joined states of a and b in $[\![B_1]\!]$, as they are duplicate states.) In $[\![B_2]\!]$ a and b can occur in any order, or even simultaneously, as they are independent, cf. Fig. 6(d).

Example 7. Consider B with $\{ a, b, c, d \}$ and $\{ \neg c \vee d \rightarrow a, a \rightarrow b, \neg a \vee b \rightarrow c, c \rightarrow d \}$. Initially there is a choice between a and c. If a (c) is chosen b (d) becomes enabled and c (a) becomes disabled. After the appearance of b (d), c (a) becomes enabled. The corresponding net is depicted in Fig. 7(a).

Example 8. Suppose we have three actions a, b, and c, say, with no conditions on the occurrence of a and b, and where c is enabled after either a or b has occurred, but not both. The set of causal relations is thus $\{ \text{true} \rightarrow a, \text{true} \rightarrow b, (a \vee b) \wedge (\neg a \vee \neg b) \rightarrow c \}$. The net corresponding to this behaviour is depicted in Fig. 7(b).

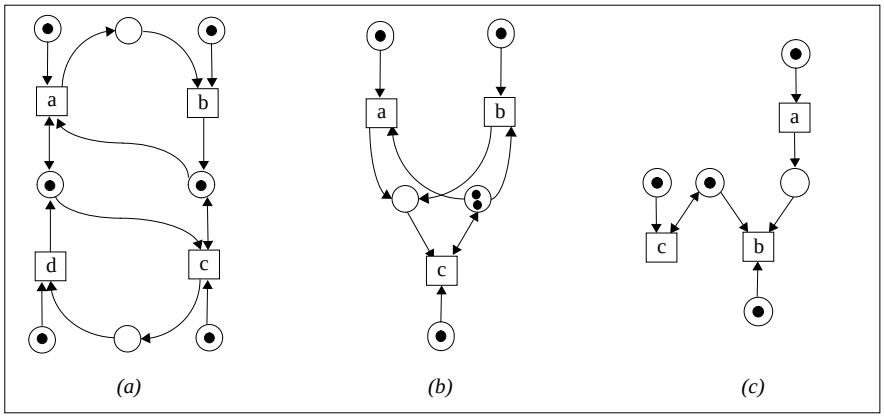

Fig. 7. Example nets corresponding to behaviours.

Example 9. As an example of a generated net that contains confusion (see Definition 50) consider behaviour $B = \langle \{ a, b, c \}, \{ \text{true} \rightarrow a, a \rightarrow b, \neg b \rightarrow c \} \rangle$. $[\![B]\!]$ is illustrated in Fig. 7(c). Let M be the marking obtained by executing transition a from the initial marking, that is, $M^0 \stackrel{a}{\Longrightarrow} M$. The conflict sets of c for markings M^0 and M are $\mathbf{cfl}(c, M^0) = \varnothing$ and $\mathbf{cfl}(c, M) = \{ b \}$. Then (M^0, c, a) is a confusion since

$$M^0 \stackrel{\{c,a\}}{\Longrightarrow} \ \wedge \ M^0 \stackrel{a}{\Longrightarrow} M \ \wedge \ \mathbf{cfl}(c, M^0) \neq \mathbf{cfl}(c, M).$$

(Note that (M^0, a, c) is not a confusion. Therefore, this type of confusion is sometimes called *asymmetric* confusion [29].) In words, when c fires in the initial marking we obtain a marking without conflicts, whereas the firing of a in the initial marking leads to a conflict between b and c.

5.2 Composed Behaviours

In order to define the mapping of composed behaviours onto nets in a compositional way, the operators on nets defined in section 4 can be used.

Definition 19. For $B_1, B_2 \in \mathbf{Beh}$, $G \subseteq I_1 \cup I_2$ and C a function from entries to exits define $[\![B_1 \,||_G\, B_2]\!] \mathrel{\widehat{=}} [\![B_1]\!] \oplus_G [\![B_2]\!]$ and $[\![B_1 \rightarrowtail_C B_2]\!] \mathrel{\widehat{=}} [\![B_1]\!] \curvearrowright_C [\![B_2]\!]$.

Proposition 20. *For all* $B \in \mathbf{Beh}$: $[\![B]\!]$ *is a labelled P/T-net.*

A net is called K-safe iff each place in any reachable marking from the initial marking the number of tokens is at most K, for some fixed natural K. A net is called *bounded safe* if there exists a K for which the net is K-safe.

Proposition 21. *For all* $B \in \mathbf{Beh}$: $[\![B]\!]$ *is bounded safe.*

5.3 Recursive Behaviours

This paper only deals with finite behaviours. For specifying realistic distributed systems recursion is indispensable. Recursive causal behaviours can be specified by relaxing the constraints on causality-oriented composition – e.g. by allowing multiple exits to be linked to a single entry (see [8]). As an example of a simple tail-recursive specification, let $B_1 = \langle \varnothing, \varnothing, \{ \text{true} \rightarrow exit_1 \} \rangle$ and $B_2 = \langle \{ a \}, \varnothing, \{ entry_1 \rightarrow a, a \rightarrow exit_2 \} \rangle$. For $C = \{ (entry_1, exit_1), (entry_1, exit_2) \}$ let $B = B_1 \rightarrowtail_C B_2$. B is intended to be a behaviour that can perform an a action infinitely often.

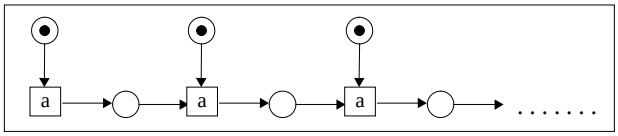

Fig. 8. An infinite net representation.

As each transition can fire at most once, recursive behaviours have to be represented by infinite nets. For example, a straightforward possibility for a net semantics of B is depicted in Fig. 8.[8] A semantics for recursive behaviours can be provided by defining a complete partial order (cpo) on labelled P/T-nets. Without going into all details, the basic idea is to consider each definition $B := X$ as an equation of the form $B = F_X(B)$, where F_X is a function that substitutes a net for each occurrence of B in X. $[\![B]\!]$ is now defined as the solution of the equation $B = F_X(B)$. When F_X is continuous with respect to the cpo then the solution of this equation is defined as the least upper bound of the chain $F_X^0(\bot), F_X^1(\bot), F_X^2(\bot), \ldots$. Thus, $[\![B]\!]$ can be computed by generating (finite) approximations.

Due to the flexibility of linking exits and entries recursive behaviours of various complexity can be defined [8]. For the regular types of recursion such as tail recursion the approach sketched above is directly applicable. It is for further study to investigate a feasible class of recursive behaviours.

6 Labelled Partially Ordered Sets

Different interpretations of nets can be defined. One of the simplest interpretations is in terms of sets of step sequences where each step consists of a single transition (traces or occurrence sequences). This boils down to giving an interleaving semantics to nets. For example, using such a semantics the behaviours of Example 6 are considered to be equivalent, as their nets have the same set of traces $\{ \varepsilon, a, b, ab, ba \}$, where ε denotes the empty trace. The fact that B_2 can perform $\{ a, b \}$ as a single step (and B_1 not) is abstracted from in this interpretation.

A natural next step is to consider nets to be equivalent when they have the same set of step sequences — with steps of arbitrary cardinality. This type of semantics is known as step semantics. In this interpretation the independence of transitions is taken into consideration, but the causal dependencies between transitions are not.

[8] The fact that this net can also be finitely represented is not important for this discussion.

For instance, $B_1 = \langle \{ a, b \}, \varnothing, \{ \text{true} \rightarrow a, \text{true} \rightarrow b \} \rangle$ and $B_2 = \langle \{ a, b \}, \varnothing, \{ \text{true} \rightarrow a, \text{true} \vee a \rightarrow b \} \rangle$ have the same set of step sequences. The fact that B_2 allows b to be causally dependent on a (and B_1 not) is abstracted from in this interpretation.

In [28] it is argued convincingly that a semantics in terms of families (i.e., sets) of labelled partially ordered sets (lposets) forms the right basis for a lot of concurrency models. Interleaving and step semantics can be defined as abstractions of such a semantics. Using lposets the above behaviours B_1 and B_2 are distinguished — $\langle \{ e, e' \}, \{ e \prec e' \}, \{ (e, a), (e', b) \} \rangle$ is an lposet of B_2 but not of B_1. In this section we give an lposet semantics for the nets generated by $[\![\;]\!]$.

For Petri nets a partial-order semantics is traditionally defined by means of associating a set of processes to N [2]. A process is a specific type of occurrence net — an acyclic net where each place has at most one output transition and at most one input transition — and represents one possible behaviour of N. From a (labelled) occurrence net it is straightforward to obtain an lposet. We belief that the recipe described below to obtain lposets is simpler and more intuitive.

6.1 Decorated Traces

The basic idea is to label each token with the identity of the transition that has generated this token most recently. In this way, tokens are no longer anonymous objects moving around the net according to some firing rules, but they carry information that facilitates determining the causal dependencies between transitions (see below). Tokens present in the initial marking are not generated by some transition and are uniquely indicated by a natural. Let $T^* \mathrel{\widehat{=}} T \cup \mathbb{N}$ denote the set of token labels.

Definition 22. For net N, $M : S \rightarrow \mathcal{P}(T^*)$ is a *labelled marking* of N.

In the sequel of this section we assume a net to have a labelled marking. A transition is now enabled with respect to a certain set V of labelled tokens. As we consider nets with arc weights one such a selection can be considered as a function assigning to each input place of a transition to fire a token label (i.e., an element of T^*).

Definition 23. Transition (t, V) is *enabled* in labelled marking M, denoted $M \stackrel{(t,V)}{\Longrightarrow}$, iff $V : {}^\bullet t \rightarrow T^*$ such that $\forall\, s \in {}^\bullet t : V(s) \in M(s)$.

Notice that firings of transitions by consuming different sets of tokens are now explicitly distinguished. For instance, in case of a single transition t with a single input place s containing two tokens, labelled t_a and t_b, respectively, $(t, (s, t_a))$ and $(t, (s, t_b))$ denote different enablings of t. Firing of (t, V) is defined as follows.

Definition 24. For M, M' labelled markings of N and $t \in T$, $M \stackrel{(t,V)}{\Longrightarrow} M'$ iff

1. $M \stackrel{(t,V)}{\Longrightarrow} \;\wedge\; (\forall\, s, s' \in {}^\bullet t : V(s) \cap M(s') \neq \varnothing \;\Rightarrow\; V(s) = V(s'))$
2. $M'(s) = \begin{cases} M(s) - V(s) \cup \{ t \} & \text{if } s \in {}^\bullet t \cap t^\bullet \\ M(s) - V(s) & \text{if } s \in {}^\bullet t \;\wedge\; s \notin t^\bullet \\ M(s) \cup \{ t \} & \text{if } s \notin {}^\bullet t \;\wedge\; s \in t^\bullet \\ M(s) & \text{otherwise} \end{cases}$

Tokens are not labelled uniquely — when a transition t fires it produces a token labelled t for each of its output places. Labelling tokens enables the reconstruction of causality information from traces. To this end it suffices to equip each individual element of a trace with a set of transitions, identifying the causal dependencies.

Definition 25. $\sigma = (t_0, rng(V_0)) \ldots (t_n, rng(V_n))$ is a *decorated trace* of N iff $\exists\, M^1, \ldots, M^n : (\forall\, i : 0 \leq i < n : M^i \stackrel{(t_i, V_i)}{\Longrightarrow} M^{i+1})$.

The set of decorated traces of N is denoted $DT(N)$.

6.2 Extracting Lposets from Decorated Traces

Definition 26. A labelled partially ordered set (*lposet*) is a triple $\langle E, \leq, \ell \rangle$ where E is a finite set of events, $\leq$ a partial order on E, and $\ell : E \to \mathbf{Act}$ a labelling function.

Constructing lposets out of decorated traces is quite straightforward.

Definition 27. For $\sigma = (t_0, T_0) \ldots (t_n, T_n)$ a decorated trace let $T_\sigma \mathrel{\widehat{=}} \{\, t_0, \ldots, t_n \,\}$.

Proposition 28. *For* $\sigma = (t_0, T_0) \ldots (t_n, T_n) : T_i \cap T \subseteq T_\sigma$ *for all* $0 \leq i \leq n$.

Definition 29. Let $\sigma = (t_0, T_0) \ldots (t_n, T_n)$ a decorated trace of N. The precedence relation $\prec_\sigma \subseteq T_\sigma \times T_\sigma$ is defined by $t_i \prec_\sigma t_j$ iff $t_i \in T_j \cap T$ for $0 \leq i, j \leq n$.

$\leq_\sigma$ is defined as $\leq_\sigma \;\mathrel{\widehat{=}}\; \prec_\sigma^*$, i.e., the reflexive and transitive closure of $\prec_\sigma$.

Proposition 30. *For* $\sigma = (t_0, T_0) \ldots (t_n, T_n) \in DT(N) : t_i \leq_\sigma t_j \;\Rightarrow\; i \leq j$.

Theorem 31. $\leq_\sigma$ *is a partial order on* T_σ.

The family of lposets of net N is defined as the set of lposets associated with some trace of N. Here it suffices to take traces as a basis rather than step sequences — in the case of finite steps (as we consider) each step of finitely many transitions can be simulated by a trace. Taking traces thus does not restrict the possible behaviours.

Definition 32. For net N, $Lpos(N) \mathrel{\widehat{=}} \{\, \langle T_\sigma, \leq_\sigma, \mu \restriction T_\sigma \rangle \mid \sigma \in DT(N) \,\}$.

Example 10. Consider B with $\{\, \mathrm{true} \to a, \mathrm{true} \to b, a \vee b \to c \,\}$. The traces, decorated traces, and lposets of $[\![\, B \,]\!]$ are presented in Table 1. [9] For trace abc two decorated traces are obtained, representing that c causally depends on either a or b.

For families of lposets the actual event names are irrelevant; they are just unique identities. This means that in fact we are interested in families of lposets modulo a renaming morphism. For example, let L_1 and L_2 be lposets with sets of events E_1 and E_2, respectively. L_1 and L_2 are isomorphic iff there is a bijection $\phi : E_1 \to E_2$ such that $\phi(e) = e' \;\Rightarrow\; \ell_1(e) = \ell_2(e')$, and $e \leq_1 e' \Leftrightarrow \phi(e) \leq_2 \phi(e')$.

[9] Strictly speaking, traces should be sequences of transitions rather than of labels of transitions. For convenience, we use labels here. In addition, singleton sets are denoted by their single element and '$*$' denotes an initial token.

Trace	Decorated Trace σ	$\langle T_\sigma, \leq_\sigma, \mu \restriction T_\sigma \rangle$
ε	$()$	$\langle \varnothing, \varnothing, \varnothing \rangle$
a	$(a,*)$	$\langle t_a, \varnothing, (t_a, a) \rangle$
b	$(b,*)$	$\langle t_b, \varnothing, (t_b, b) \rangle$
ab	$(a,*)\,(b,*)$	$\langle \{ t_a, t_b \}, \varnothing, \{ (t_a, a), (t_b, b) \} \rangle$
ba	$(b,*)\,(a,*)$	$\langle \{ t_a, t_b \}, \varnothing, \{ (t_a, a), (t_b, b) \} \rangle$
ac	$(a,*)\,(c,a)$	$\langle \{ t_a, t_c \}, t_a \leq t_c, \{ (t_a, a), (t_c, c) \} \rangle$
bc	$(b,*)\,(c,b)$	$\langle \{ t_b, t_c \}, t_b \leq t_c, \{ (t_b, b), (t_c, c) \} \rangle$
acb	$(a,*)\,(c,a)\,(b,*)$	$\langle \{ t_a, t_b, t_c \}, t_a \leq t_c, \{ (t_a, a), (t_b, b), (t_c, c) \} \rangle$
bca	$(b,*)\,(c,b)\,(a,*)$	$\langle \{ t_a, t_b, t_c \}, t_b \leq t_c, \{ (t_a, a), (t_b, b), (t_c, c) \} \rangle$
abc	$(a,*)\,(b,*)\,(c,a)$	$\langle \{ t_a, t_b, t_c \}, t_a \leq t_c, \{ (t_a, a), (t_b, b), (t_c, c) \} \rangle$
abc	$(a,*)\,(b,*)\,(c,b)$	$\langle \{ t_a, t_b, t_c \}, t_b \leq t_c, \{ (t_a, a), (t_b, b), (t_c, c) \} \rangle$

Table 1. Decorated traces and family of lposets for $a \ \vee \ b \rightarrow c$.

For B with alphabet $\{ a, b \}$ and causal relations $\{ \text{true} \rightarrow a, \neg\, a \rightarrow b \}$ we obtain for trace ba the lposet with $t_b \leq t_a$. The intuitive interpretation is that if both t_a and t_b occur in a system run, then t_b causally precedes t_a (as in $b \rightarrow a$). This interpretation is identical to the interpretation of asymmetric conflict in event-based models [21, 25]. An alternative would be to consider it a temporal precedence rather than a causal one. It is for further study to investigate this alternative.

There is, however, one problem with the generation of lposets from nets obtained from $[\![\ ]\!]$: for example, for $[\![B]\!]$ where B has alphabet $\{ a, b, c \}$ and causal relations $\{ \text{true} \rightarrow a, \text{true} \rightarrow b, \neg\, a \ \vee \ b \rightarrow c \}$, we obtain for trace ab two lposets — one in which there is no relation between t_a and t_b, and one in which there is such a relation. The first is correct, but the latter suggests a causal relation between b and a, which is absent in B. The problem stems from the net representation $[\![B]\!]$ as in this representation t_a is not prevented from consuming the token generated by t_b.

Thus we conclude that the procedure in this section is applicable to a (somewhat) restricted set of causal behaviours to obtain families of lposets. The class of behaviours to which it is applicable is denoted $\mathbf{Beh}^*$ and is determined by putting restrictions on the preconditions allowed. These restrictions prevent negative causalities in combination with positive causalities to occur in the context of an or-causality.

Definition 33. $\mathcal{Q}(E, F) \mathrel{\widehat{=}} (\exists\, a \in E, b \in F : (\neg\, a) \in E \ \wedge \ b \neq a \ \wedge \ (\neg\, b) \notin F)$.

Definition 34. $\mathcal{R}(E)$ is true iff each subexpression $(E_1 \ \vee \ E_2)$ of E satisfies $\neg\, \mathcal{Q}(E_1, E_2) \ \wedge \ \neg\, \mathcal{Q}(E_2, E_1)$.

For S a set of causal relations $\mathcal{R}(S)$ is true iff $\mathcal{R}(E_i)$ holds for all $E_i \rightarrow a_i$ in S.

Definition 35. $\mathbf{Beh}^*$ is the largest subset of $\mathbf{Beh}$ such that any subexpression B' of $B \in \mathbf{Beh}$ satisfies

1. $B' = \langle I, A, S \rangle \ \Rightarrow \ \mathcal{R}(S)$
2. $B' = B_1 \rightarrowtail_C B_2 \ \Rightarrow \ \mathcal{R}(S)$ with
   - $S = \{ ((E[\mathit{entry}_{i_1}/E_1]) \ldots)[\mathit{entry}_{j_n}/E_n] \rightarrow e \mid E \rightarrow e \in S_2 \}$

– $C = \{\,(entry_{i_1}, exit_1), \ldots, (entry_{j_n}, exit_n)\,\}, E_k \rightarrow exit_k \in S_1$.

The approach of decorating tokens with the label of the transition that generated this token most recently is inspired by [7]. There, an equivalence notion on nets is defined at the net level (i.e. not on some underlying model such as occurrence graphs or (l)posets) exploiting the preservation of causality information in traces. The main difference with our approach is that we keep track of the causalities between transitions rather than between their labels. This implies that in the approach of [7] the branching information is lost (in fact, they consider pomsets rather than lposets; pomsets are isomorphism classes of lposets rather than sets of lposets [28]).

7 Expansion Laws

An equivalence notion on causal behaviours is useful to decide whether two specifications can be considered to be the same. Two behaviours are *causally equivalent* when they have the same set of lposets.

Definition 36. $\forall\, B_1, B_2 \in \mathbf{Beh}^* : B_1 \approx_{lpos} B_2$ iff $Lpos([\![B_1]\!]) = Lpos([\![B_2]\!])$.

Complex (distributed) systems can hardly be described as a monolithic specification. Such specifications need to be structured in order to keep them comprehensible and will be composed of sub-specifications. The operations $||_G$ and $\rightarrowtail_C$ are means to combine specifications, that is, causal behaviours. For verification and analysis purposes it is beneficial to be able to calculate from a composed specification the equivalent monolithic one. Below two expansion theorems are presented that allow for transforming causal behaviours of the form $B_1 \,||_G\, B_2$ and $B_1 \rightarrowtail_C B_2$, respectively, into equivalent monolithic behaviours. Let B_i be behaviour $\langle I_i, A_i, S_i\rangle$ $(i \in 1, 2)$.

$E[b/\text{false}]$ denotes E where all b's are replaced by false $((\neg\, b)[b/\text{false}] \mathrel{\widehat{=}} \text{true})$.

Theorem 37. $\forall\, B_1, B_2 \in \mathbf{Beh}^*, G \subseteq I_1 \cup I_2,\ B_1 \,||_G\, B_2 \approx_{lpos} \langle I, A_1 \cup A_2, S\rangle$ *with*

1. $I = (I_1 \cup I_2) \setminus (G \setminus (I_1 \cap I_2))$
2. S *is the smallest set such that*
   – $\forall\, a \in G \cap I_1 \cap I_2 : (E_1 \rightarrow a \in R_1 \ \wedge\ E_2 \rightarrow a \in R_2) \ \Rightarrow\ (E_1 \wedge E_2) \rightarrow a \in S$
   – $\forall\, a \notin G : (E \rightarrow a \in R_1 \cup R_2) \ \Rightarrow\ E \rightarrow a \in S$

   where for $i = 1, 2$ *and* $G \setminus (I_1 \cap I_2) = \{\, b_1, \ldots, b_n \,\}$

$$R_i = \{\, ((E[b_1/\mathit{false}]) \ldots)[b_n/\mathit{false}] \rightarrow a \mid E \rightarrow a \in S_i \,\} \quad .$$

Example 11. Let $B_1 = \langle \{\, a, b, c \,\}, \varnothing, \{\, \text{true} \rightarrow a, a \rightarrow b, b \rightarrow c \,\}\rangle$ and
$B_2 = \langle \{\, a, c, d \,\}, \varnothing, \{\, \text{true} \rightarrow a, a \rightarrow d, a \ \wedge\ d \rightarrow c \,\}\rangle$. Then

$$B_1 \,||_{\{\, a,c \,\}}\, B_2 \approx_{lpos} \langle \{\, a, b, c, d \,\}, \varnothing, \{\, \text{true} \wedge \text{true} \rightarrow a, a \rightarrow b, b \wedge a \wedge d \rightarrow c, a \rightarrow d \,\}\rangle$$
$$B_1 \,||_{\{\, a,b,c \,\}}\, B_2 \approx_{lpos} \langle \{\, a, c, d \,\}, \varnothing, \{\, \text{true} \rightarrow a, \text{false} \ \wedge\ a \ \wedge\ d \rightarrow c, a \rightarrow d \,\}\rangle$$

Let $B_1 = \langle \{\, b, c \,\}, \{\, a \,\}, \{\, \text{true} \rightarrow a, a \rightarrow b, b \rightarrow c \,\}\rangle$ and
$B_2 = \langle \{\, b, e \,\}, \{\, d \,\}, \{\, \text{true} \rightarrow d, d \rightarrow e, e \ \vee\ d \rightarrow b \,\}\rangle$. Then

$$B_1 \,||_b\, B_2 \approx_{lpos} \langle \{\, b, c, e \,\}, \{\, a, d \,\}, \{\, \text{true} \rightarrow a, a \wedge (e \vee d) \rightarrow b, b \rightarrow c, \text{true} \rightarrow d, d \rightarrow e \,\}\rangle.$$

Theorem 38. $\forall B_1, B_2 \in \mathbf{Beh}^*$ *and* $C = \{ (entry_{i_1}, exit_1), \ldots, (entry_{j_n}, exit_n) \}$ *a partial function from entries of* B_2 *to exits of* B_1, $B_1 \rightarrowtail_C B_2 \approx_{lpos} \langle I_1 \cup I_2, A_1 \cup A_2, S \rangle$ *where for* $E_k \rightarrow exit_k \in S_1$ $(0 < k \leq n)$

$$S = \{ E \rightarrow e \in S_1 \mid \forall k : 0 < k \leq n : e \neq exit_k \} \cup$$
$$\{ ((E[entry_{i_1}/E_1]) \ldots)[entry_{j_n}/E_n] \rightarrow e \mid E \rightarrow e \in S_2 \}.$$

Example 12. Consider $B_1 = \langle \{ a, b \}, \varnothing, \{ \text{true} \rightarrow a, \text{true} \rightarrow b, a \wedge b \rightarrow exit_1 \} \rangle$ and $B_2 = \langle \{ c \}, \varnothing, \{ entry_1 \rightarrow c \} \rangle$. Define $B = B_1 \rightarrowtail_C B_2$ with $C = \{ (entry_1, exit_1) \}$. Then, by Theorem 38, $B \approx_{lpos} \langle \{ a, b, c \}, \varnothing, \{ \text{true} \rightarrow a, \text{true} \rightarrow b, a \wedge b \rightarrow c \} \rangle$.

8 Conclusions and Future Work

In this paper we defined an operational, compositional semantics of the basic ingredients of the formalism in [8, 32, 33]. The notions of causal relation and causal behaviour were mapped on an extension of labelled place/transition nets, a generalisation of the notion of Petri Boxes [3]. We believe that the net representation helps in getting a better intuitive understanding of the formalism and is useful as an operational model.

An underlying semantics in terms of labelled partially ordered sets (lposets) was given, using decoration of traces. Based on this semantics two expansion theorems were defined that allow for the transformation of composed behaviours into equivalent monolithic ones. These expansion theorems are helpful for verification and analysis purposes and turned out to be reasonably simple and intuitive.

The number of extensions to our work are numerous. To mention a few:

- Extension of our work to deal with recursive behaviours.
- Generation of families of lposets out of nets in a compositional way.
- Incorporation of data, time, and probabilities. Extensions of nets such as coloured nets [17] and timed Petri nets [22] are expected to be useful here.

Acknowledgements: Lex Heerink is acknowledged for comments on an early draft. Chris Vissers, Rom Langerak, Robert Huis in 't Veld, Arend Rensink and the members of the Architecture Group are thanked for valuable discussions and their useful comments.

A A Summary of Net Theory

Definition 39. (S, T, F) is a *net* with S a finite set of *places*, T a finite set of *transitions* with $S \cap T = \varnothing$, and $F \subseteq (S \times T) \cup (T \times S)$, the *flow relation* with $T \subseteq dom(F \cup F^{-1})$.

Note that in the above definition, self-loops are not prohibited as for arbitrary $s \in S$ and $t \in T$ $(s, t) \in F$ and $(t, s) \in F$ is allowed. Let $N = (S, T, F)$.

Definition 40. For transition $t \in T$, ${}^\bullet t \mathrel{\widehat{=}} \{ s \in S \mid (s, t) \in F \}$ is the *preset* of t, and $t^\bullet \mathrel{\widehat{=}} \{ s \in S \mid (t, s) \in F \}$ is the *postset* of t.

In a similar way ${}^\bullet s$ and $s^\bullet$ are defined for $s \in S$.

Definition 41. For net N, $M : S \longrightarrow \mathbb{N}$ is a *marking* of N.

Definition 42. A *place/transition (P/T) net* is a 4-tuple (S, T, M^0, F) with (S, T, F) a finite net, and M^0 an initial marking.

In the sequel we often omit the prefix P/T, and write simply net instead of P/T-net. According to [27] we consider *marked nets*, as all arc weights are equal to one and each place has an unbounded capacity. Let $N = (S, T, M^0, F)$.

Definition 43. A *transition set* U is a non-empty set of transitions, i.e. $U \subseteq T \ \wedge \ U \neq \varnothing$.

For $\{ t \}$ we often simply write t. $\#(S)$ denotes the cardinality of arbitrary set S. $\#_a(S)$ is the number of occurrences of a in S. That is, $\#_a(S)$ is one if $a \in S$ and zero otherwise.

Definition 44. U is *enabled* in M, denoted $M \stackrel{U}{\Longrightarrow}$, iff $\forall\, s \in S : M(s) \geq \sum_{t \in U} \#_s({}^\bullet t)$.

Definition 45. For M^1, M^2 markings of N and U a transition set of N, $M^1 \stackrel{U}{\Longrightarrow} M^2$ iff

$$M^1 \stackrel{U}{\Longrightarrow} \ \wedge \ \forall\, s \in S : M^2(s) = M^1(s) + (\sum_{t \in U} \#_s(t^\bullet) - \sum_{t \in U} \#_s({}^\bullet t)).$$

Definition 46. The set $[M]$ of *reachable markings* of M is the smallest set of markings of N such that $M \in [M] \ \wedge \ \forall\, M^1 \in [M], U$ a transition set of $N : M^1 \stackrel{U}{\Longrightarrow} M^2 \ \Rightarrow \ M^2 \in [M]$.

Definition 47. N is *K-safe* iff for some fixed natural K $\forall\, s \in S, M \in [M^0] : M(s) \leq K$.

Definition 48. $t_1, t_2 \in T$ are in *conflict* in M iff $M \stackrel{t_1}{\Longrightarrow} \ \wedge \ M \stackrel{t_2}{\Longrightarrow} \ \wedge \ \neg\, M \stackrel{\{ t_1, t_2 \}}{\Longrightarrow}$.

Definition 49. For transition $t \in T$ and marking M such that $M \stackrel{t}{\Longrightarrow}$, the *conflict set* of t in M is $\mathbf{cfl}(t, M) \mathrel{\widehat{=}} \{ t' \in T \mid M \stackrel{t'}{\Longrightarrow} \ \wedge \ \neg\, M \stackrel{\{ t, t' \}}{\Longrightarrow} \}$.

Definition 50. (M, t_1, t_2) is a *confusion* iff $M \stackrel{\{ t_1, t_2 \}}{\Longrightarrow} \wedge M \stackrel{t_2}{\Longrightarrow} M' \wedge \mathbf{cfl}(t_1, M) \neq \mathbf{cfl}(t_1, M')$.

References

1. J. Baeten and W. Weijland. *Process Algebra*. Cambridge University Press, 1990.
2. E. Best and R. Devillers. Sequential and concurrent behaviour in Petri net theory. *Th. Comp. Sci.*, **55**:87–136, 1987.
3. E. Best, R. Devillers, and J. Hall. The Box calculus: a new causal algebra with multi-label communication. In G. Rozenberg (ed), *Advances in Petri Nets*, LNCS **609**:21–69. 1992.
4. T. Bolognesi and G. Ciaccio. Cumulating constraints on the 'when' and the 'what'. In R. Tenney *et al* (eds), *Formal Description Techniques VI*, pp. 433–448. 1994.
5. M. Broy. Formalization of distributed, concurrent, reactive systems. In E. Neuhold and M. Paul (eds), *Formal Description of Programming Concepts*, pp. 319–361. 1991.
6. G. Chiola, M. Ajmone Marsan, G. Balbo, and G. Conte. Generalized stochastic Petri nets: A definition at the net level and its implications. *IEEE Trans. on Software Eng.*, **19**(2):89–106, 1993.
7. R. Coelho da Costa and J.-P. Courtiat. Using Petri Nets as a model for Petri Nets. *Proc. IEEE Workshop on Future Trends of Distr. Comp. Syst.*, pp. 41–47. 1992.
8. L. Ferreira Pires. *Architectural Notes: a Framework for Distributed Systems Development*. PhD thesis, Univ. of Twente, 1994.

9. H. Garavel and J. Sifakis. Compilation and verification of LOTOS specifications. In L. Logrippo *et al* (eds), *PSTV X*, pp. 359–376. 1990.
10. U. Goltz and W. Reisig. CSP-programs as nets with individual tokens. In G. Rozenberg *et al* (eds), *Advances in Petri Nets*, LNCS **188**:169–196. 1984.
11. J. Gunawardena. Causal automata I: Confluence ≡ {AND, OR}-causality. In M. Kwiatkowska *et al* (eds), *Semantics for Concurrency*, pp. 137–156. 1990.
12. J. Gunawardena. Geometric logic, causality and event structures. In J. Baeten and J. Groote (eds), *Concur'91*, LNCS **527**:266–280. 1991.
13. J. Gunawardena. Causal automata. *Th. Comp. Sci.*, **101**:265–288, 1992.
14. C. Hoare. *Communicating Sequential Processes*. Prentice-Hall, 1985.
15. P. Hoogers, H. Kleijn, and P. Thiagarajan. Local event structures and Petri nets. In E. Best (ed), *Concur'93*, LNCS **715**:462–476. 1993.
16. R. Janicki. A formal semantics for concurrent systems with a priority relation. *Acta Informatica*, **24**:33–55, 1987.
17. K. Jensen. Coloured Petri nets and the invariant-method. *Th. Comp. Sci.*, **14**:317–336, 1981.
18. J.-P. Katoen Causal behaviours and nets. Tech. rep. 94-70, Univ. of Twente. 1994.
19. C. Koomen. Algebraic specification and verification of communication protocols. *Science of Computer Programming*, **5**(1):1–37, 1985.
20. L. Lamport. Time, clocks and the ordering of events. *CACM*, **21**(7):558–565, 1978.
21. R. Langerak. *Transformation and Semantics for LOTOS*. PhD thesis, Univ. Twente, 1992.
22. P. Merlin and D. Farber. Recoverability of communication protocols – implications of a theoretical study. *IEEE Trans. on Communications*, **24**:1036–1043, 1976.
23. R. Milner. *Communication and Concurrency*. Prentice-Hall, 1989.
24. J. Peterson. *Petri net theory and modeling of systems*. Prentice-Hall, 1981.
25. G. Pinna and A. Poigné. On the nature of events. In I. Havel and V. Koubek (eds), *Mathematical Foundations of Computer Science'92*, LNCS **629**:430–441. 1992.
26. G. Pinna and A. Poigné. On the specification of elementary reactive behaviour. In S. Brookes *et al* (eds), *Mathematical Foundations of Programming Semantics'93*, LNCS **802**:271–292. 1994.
27. W. Reisig. *Petri Nets – An Introduction*. Springer-Verlag, 1985.
28. A. Rensink. Posets for configurations! In W. Cleaveland (ed), *Concur'92*, LNCS **630**:269–285. 1992.
29. G. Rozenberg and P. Thiagarajan. Petri nets: Basic notions, structure, behaviour. In J. de Bakker *et al* (eds), *Current Trends in Concurrency*, LNCS **224**:585–668. 1986.
30. R. Schwarz and F. Mattern. Detecting causal relationships in distributed computations: in search of the holy grail. *Distributed Computing*, **7**:149–174, 1994.
31. R. van Glabbeek and F. Vaandrager. Petri net models of algebraic theories of concurrency. In J. de Bakker *et al* (eds), *PARLE'87*, LNCS **259**:224–242. 1987.
32. M. van Sinderen, L. Ferreira Pires, C. Vissers, and J.-P. Katoen. A design model for open distributed processing systems. *Comp. Netw. & ISDN Syst.*, 1995.
33. C. Vissers, M. van Sinderen, and L. Ferreira Pires. What makes industries believe in formal methods. In A. Danthine *et al* (eds), *PSTV XIII*, pp. 3–26. 1993.
34. G. Winskel. An introduction to event structures. In J. de Bakker *et al* (eds), *Linear Time, Branching Time and Partial Order in Logics and Models for Concurrency*, LNCS **354**:364–397. 1989.

This article was processed using the LaTeX macro package with LLNCS style