The 4C Spectrum of Fundamental Behavioral Relations for Concurrent Systems

Artem Polyvyanyy[1], Matthias Weidlich[2], Raffaele Conforti[1], Marcello La Rosa[1,3], and Arthur H.M. ter Hofstede[1,4]

[1] Queensland University of Technology, Brisbane, Australia
[2] Imperial College London, London, United Kingdom
[3] NICTA Queensland Lab, Brisbane, Australia
[4] Eindhoven University of Technology, Eindhoven, The Netherlands
{artem.polyvyanyy;raffaele.conforti;m.larosa;a.terhofstede}@qut.edu.au
m.weidlich@imperial.ac.uk

Abstract. The design of concurrent software systems, in particular process-aware information systems, involves behavioral modeling at various stages. Recently, approaches to behavioral analysis of such systems have been based on declarative abstractions defined as sets of behavioral relations. However, these relations are typically defined in an ad-hoc manner. In this paper, we address the lack of a systematic exploration of the fundamental relations that can be used to capture the behavior of concurrent systems, i.e., co-occurrence, conflict, causality, and concurrency. Besides the definition of the spectrum of behavioral relations, which we refer to as the 4C spectrum, we also show that our relations give rise to implication lattices. We further provide operationalizations of the proposed relations, starting by proposing techniques for computing relations in unlabeled systems, which are then lifted to become applicable in the context of labeled systems, i.e., systems in which state transitions have semantic annotations. Finally, we report on experimental results on efficiency of the proposed computations.

1 Introduction

Process models play a key role in the development of concurrent software systems as they describe the functionality of a system by means of actions and their interdependencies for the coordination of action execution. On the one hand, such models are used to document system requirements, thereby guiding implementation efforts [1]. On the other hand, process-aware information systems rely on process models as implementation artifacts that are deployed in an execution environment, e.g., a workflow engine or a service orchestration framework [2,3].

Process models are *system models* according to the classification of models of concurrency presented by Sassone et al. [4], i.e., they feature an explicit representation of states and define how actions lead to state changes. This stands in contrast to *behavior models* that define occurrences of actions over time while abstracting from states. As such, the interpretation of a process model, e.g., a Labeled Transition System (LTS) [5], a Petri net [6], or a UML activity diagram [7], under a certain semantics defines a behavior model, e.g., a language over actions [8] or an event structure [9]. Semantics, in turn, are broadly classified in two dimensions [4]. First, a semantics is *concurrent* or *interleaving*, depending on whether the difference between concurrency and non-determinism is

considered to be important. Second, a semantics is *linear time* or *branching time*, which ignores or captures the moments of choice between the occurrences of different actions.

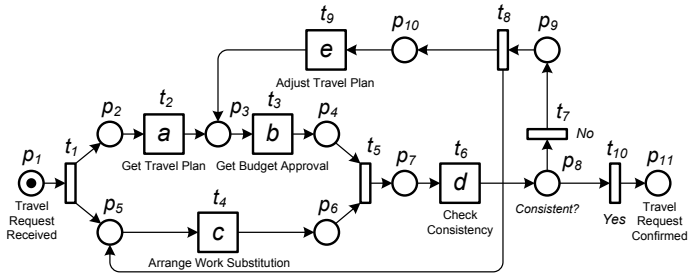
Given that process models are system models, analysis techniques are often grounded on actions (system model level) and not on action occurrences (behavior model level). However, recent work advocated to ground such analysis not directly in the procedural description that is inherent to every system model, but rather to rely on a declarative characterization of the behavior model that is induced by the process model under a certain semantics, cf. [10] for details. Then, different sets of behavioral relations are defined over pairs of actions (system model level), but capture *characteristics*, or *features* in data mining terminology [11], of the occurrences of these actions (behavior model level). Examples of these relations include different notions of exclusiveness, concurrency, successorship, co-occurrence, or precedence. A specific example would be a binary exclusiveness relation defined over actions represented by transition labels in an LTS that holds if the language of the LTS does not comprise a word that contains both labels. Such behavioral relations may be used to judge the consistency of process models, conduct similarity search and querying in a process model repository, or to realize change propagation between process models, cf. [12].

The benefit of relying on a declarative approach for the analysis of process models is clearly witnessed by existing work. However, there is a plethora of specific definitions of behavioral relations. Even for a single semantics and, thus, for a single behavior model that acts as the interpretation of the process model, we observe a lack of understanding of subtle differences in the definition of behavioral relations and their interplay. In this paper, therefore, we provide a rigorous analysis of the spectrum of possible behavioral relations. We base our analysis on two well-established formalisms for representing system models and behavior models, i.e., Petri nets [6] and their interpretation in terms of concurrent runs (or *processes* for short) [13]. We focus on four fundamental properties that are of particular relevance for behavioral analysis: co-occurrence, conflict, causality, and concurrency, jointly referred to in this paper as the 4C relations. For each of these relations, we define a spectrum of relations over actions (transitions of Petri net systems or transition labels) that are based on action occurrences (events of processes of Petri net systems). As such, our work provides the foundation for selecting specific behavioral relations for the analysis of a process model with a declarative characterization of the behavior model induced under a certain semantics.

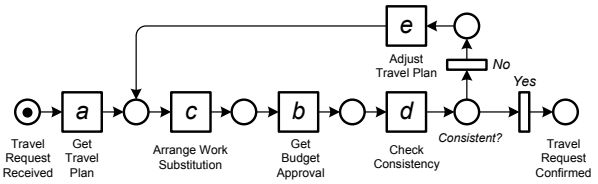
More specifically, this paper makes the following contributions:

- It defines sets of co-occurrence, conflict, causality, and concurrency relations for Petri nets and shows that these relations give rise to implication lattices.
- It provides operationalizations of the proposed behavioral relations, i.e., it shows how these relations can be computed.
- It lifts the proposed behavioral relations, as well as the proposed computation techniques, to labeled systems.
- It presents an implementation and evaluation of the proposed computations.

The remainder of the paper is structured as follows. The next section illustrates the application of behavioral relations with an example. Section 3 presents preliminaries for our work. Section 4 defines a spectrum of fundamental behavioral relations. The computation of these relations is addressed in Section 5. Subsequently, we extend the class of considered models by lifting the relations, as well as their operationalizations, to



(a)



(b)

Fig. 1: Two behavioral specifications given as Petri net systems

labeled systems. We evaluate the proposed computation methods in Section 7. Finally, Section 8 reviews related work before Section 9 concludes the paper.

2 Motivating Example

We illustrate the application of fundamental behavioral relations for the analysis of process models with two use cases, process model search and querying. Fig. 1 shows two Petri net systems. Both models describe operations as they may be implemented in a process-aware information system to realize the handling of a travel request. While we define the formal semantics for Petri net systems later, here it suffices to see that both models describe similar behaviors. For instance, in both models, action a is executed at most once and always before action b, which may be repeated. However, the model in Fig. 1(a) defines that actions a and b may be executed concurrently to action c, whereas these actions are executed in a sequential order in Fig. 1(b).

Management of large process model collections requires effective search and querying techniques [12]. Since process models specify behaviors, search and querying should not be limited to the syntax of process models but should also consider semantics [14,15]. Behavioral relations have been used as the formal grounding for many search and querying techniques. For instance, behavioral similarity of two models can be defined as the Jaccard coefficient, i.e., the size of the intersection divided by the size of the union of exclusiveness, order, and interleaving relations, defined for trace semantics [16]. Then, the pair (b,c) would be in the intersection and union of order relations of both models in Fig. 1 and, thus, account for similar behavior. However, the relations defined in [16] do not distinguish concurrent and interleaved occurrences, so that the pair (b,c) would also be in the intersection of the relations representing interleaving. Similar approaches have been defined on other behavioral relations, cf. [17,18], raising the question of the underlying spectrum of relations for similarity assessment.

Turning to querying of process models, APQL [14] has been presented as a rich query language that supports 16 different forms of causal relations between actions, grounded

in trace semantics. As an example, a relation may require that occurrences of b are succeeded by occurrences of c *immediately* in some occurrence sequence, which is the case in Fig. 1(a), but not in Fig. 1(b). Another relation captures that occurrences of b may be succeeded by occurrences of c *eventually*, which holds true in Figs. 1(a) and 1(b).

These examples illustrate that the benefit of using a declarative characterization of the behavior model induced by the process model under a certain semantics is clearly recognized. However, the choice of behavioral relations to use in a certain setting is taken in an ad-hoc manner. By precisely defining and exploring the spectrum of relations that may qualify as a formal grounding, our work, thus, lays the foundation for a rigorous application of behavioral relations in various use cases.

3 Preliminaries

This section introduces *Petri nets*, *net systems*, and *processes* of net systems.

3.1 Petri Nets and Net Systems

Petri nets are a well-established formalism for describing process models, i.e., system models [6]. They allow for the rigorous definition of semantics of systems and reuse of the well-developed mathematical theory for analysis of systems.

Definition 3.1 (Petri net)
A *Petri net*, or a *net*, is an ordered triple $N := (P, T, F)$, where P and T are finite disjoint sets of *places* and *transitions*, respectively, and $F \subseteq (P \times T) \cup (T \times P)$ is a *flow* relation. ⌟

An element $x \in P \cup T$ is a *node* of N. A *node* $x \in P \cup T$ is an *input* (an *output*) node of a node $y \in P \cup T$ iff $(x, y) \in F$ ($(y, x) \in F$). By $\bullet x$ ($x\bullet$), $x \in P \cup T$, we denote the *preset* (the *postset*) of x, i.e., the set of all input (output) nodes of x. For a set of nodes $X \subseteq P \cup T$, $\bullet X := \bigcup_{x \in X} \bullet x$ and $X\bullet := \bigcup_{x \in X} x\bullet$. A node $x \in P \cup T$ is a *source* (a *sink*) node of N iff $\bullet x = \varnothing$ ($x\bullet = \varnothing$). Given a net $N := (P, T, F)$, by $Min(N)$ we denote the set of all source nodes of N. For convenience considerations, we require all nets to be T-restricted. A net N is *T-restricted* iff the preset and postset of every transition is not empty, i.e., $\forall\, t \in T : \bullet t \neq \varnothing \neq t\bullet$.

The execution semantics of Petri nets is proposed in terms of states and state transitions and can be regarded as a 'token game'. A state of a net is captured by the concept of a *marking*, which specifies a distribution of *tokens* on the net's places.

Definition 3.2 (Marking of a net) A *marking*, or a *state*, of a net $N := (P, T, F)$ is a function $M : P \to \mathbb{N}_0$ that assigns to each place $p \in P$ a number $M(p)$ of *tokens* at p.[1] ⌟

For M and M' being two markings of $N := (P, T, F)$, it holds that M is *covered* by M', denoted by $M \leq M'$, iff $M(p) \leq M'(p)$, for every $p \in P$. We shall often refer to a marking M as to the multiset of places that contains $M(p)$ copies of place p for every $p \in P$. Additionally, we shall use the symbol $\uplus$ to denote the union of multisets.

A *net system* is a Petri net at a certain state/marking.

Definition 3.3 (Net system) A *net system*, or a *system*, is an ordered pair $S := (N, M)$, where N is a net and M is a marking of N. ⌟

[1] $\mathbb{N}_0$ denotes the set of all natural numbers including zero.

In the graphical notation, places are usually visualized as circles, transitions are drawn as rectangles, the flow relation is given as directed edges, and tokens are depicted as black dots inside assigned places; refer to Fig. 1(a) for a visualization of a net system with transitions $t_1 \dots t_{10}$, places $p_1 \dots p_{11}$, and a single token at place p_1.

A transition can be *enabled* at a given marking of a net. An enabled transition can *occur*. An occurrence of a transition leads to a new marking of the net.

Definition 3.4 (Semantics of a system) Let $S := (N, M)$, $N := (P, T, F)$, be a net system.

- A transition $t \in T$ is *enabled* in S, denoted by $S[t\rangle$, iff every input place of t contains at least one token, i.e., $\forall\, p \in {\bullet}t : M(p) > 0$.
- If a transition $t \in T$ is enabled in S, then t can *occur*, which leads to a *step* from S to $S' := (N, M')$ via t, where M' is a fresh marking such that $M'(p) := M(p) - \mathbf{1}_F((p,t)) + \mathbf{1}_F((t,p))$, for every $p \in P$, i.e., transition t 'consumes' one token from every input place of t and 'produces' one token for every output place of t.[2] ⌟

The fact that there exists a step from S to S' via a transition t is denoted by $S[t\rangle S'$. A marking M of a net N is a *terminal* marking iff there are no enabled transitions in (N, M). A net system induces a set of its *occurrence sequences/executions*.

Definition 3.5 (Occurrence sequence, Execution) Let $S_0 := (N, M_0)$ be a net system.

- A sequence of transitions $\sigma := t_1 \dots t_n$, $n \in \mathbb{N}_0$, of N is an *occurrence sequence* in S_0 iff there exists a sequence of net systems $S_0, S_1 \dots S_n$, such that for every position i in σ, $1 \leq i \leq n$, it holds that $S_{i-1}[t_i\rangle S_i$; we say that σ *leads* from S_0 to S_n.
- An occurrence sequence σ in S_0 is an *execution* iff σ leads from S_0 to (N, M), where M is a terminal marking in (N, M). ⌟

A marking M' is a *reachable* marking in a net system $S := (N, M)$, $N := (P, T, F)$, iff there exists an occurrence sequence σ in S that leads from S to (N, M'). By $[S\rangle$, we denote the set of all reachable markings in S. A marking M' is a *home* marking in S iff it is reachable from every reachable marking in S, i.e., $\forall\, M'' \in [S\rangle : M' \in [(N, M'')\rangle$. S is *n-bounded*, or *bounded*, iff there exists a number $n \in \mathbb{N}_0$ such that for every reachable marking M' in S and for every place $p \in P$ it holds that the number of tokens at p is at most n, i.e., $\forall\, M' \in [S\rangle\, \forall\, p \in P : M'(p) \leq n$. It is easy to see that the set of all reachable markings in a bounded net system is finite. Finally, a transition $t \in T$ is *dead* in S iff there does not exist a reachable marking in S that enables t.

3.2 Processes of Net Systems

Occurrence sequences and executions suit well when it comes to capturing *orderings* of transition occurrences. This section presents *processes* of net systems [19]. One can use processes to adequately represent *causality* and *concurrency* relations on transition occurrences. A process of a net system is a net of a particular kind, called a *causal net*, together with a mapping from nodes of the causal net to nodes of the net system.

Definition 3.6 (Causal net) A net $N := (B, E, G)$ is a *causal* net iff: (i) for every $b \in B$ it holds that $|{\bullet}b| \leq 1$ and $|b{\bullet}| \leq 1$, and (ii) N is acyclic, i.e., G^+ is irreflexive.[3] ⌟

[2] $\mathbf{1}_F$ denotes the characteristic function of F on the set $(P \times T) \cup (T \times P)$.

[3] R^+ denotes the transitive closure of binary relation R.

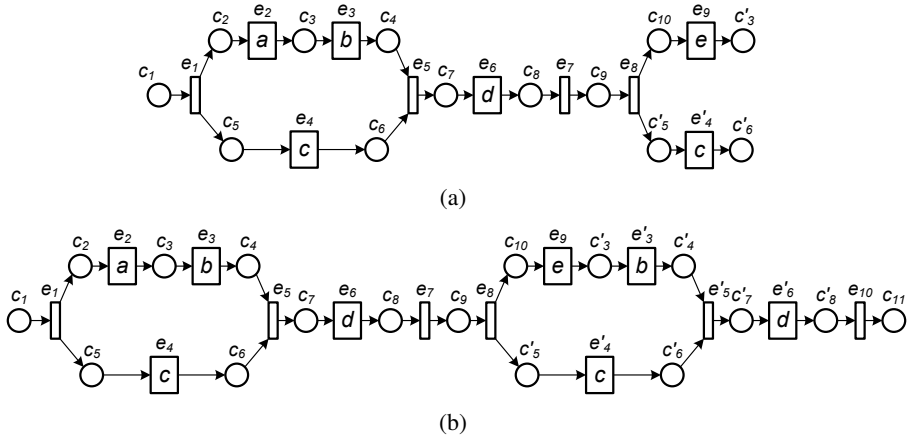


Fig. 2: Two processes of the net system in Fig. 1(a)

Elements of B and E are called *conditions* and *events* of N, respectively. Two nodes x and y of a causal net $N := (B, E, G)$ are *causal*, or x is a *cause* for y, iff $(x, y) \in G^+$. They are *concurrent*, iff $x \neq y$, $(x, y) \notin G^+$, and $(y, x) \notin G^+$. A *cut* of a causal net is a maximal (with respect to set inclusion) set of its pairwise concurrent conditions.

Events of causal nets can be used to describe transition occurrences.

Definition 3.7 (Process, adapted from [19])
A *process* of a system $S := (N, M)$, $N := (P, T, F)$, is an ordered pair $\pi := (N_\pi, \rho)$, where $N_\pi := (B, E, G)$ is a causal net and $\rho : B \cup E \to P \cup T$ is such that:

- $\rho(B) \subseteq P$ and $\rho(E) \subseteq T$, i.e., ρ preserves the nature of nodes,
- $Min(N_\pi)$ is a cut and $\forall p \in P : M(p) = |\rho^{-1}(p) \cap Min(N_\pi)|$, i.e., π starts at M, and
- for every event $e \in E$ and for every place $p \in P$ it holds that $\mathbf{1}_F((p, \rho(e))) = |\rho^{-1}(p) \cap {\bullet e}|$ and $\mathbf{1}_F((\rho(e), p)) = |\rho^{-1}(p) \cap e{\bullet}|$, i.e., ρ respects the environment of transitions.[4]

Given a process π, we shall often write E_π and ρ_π to denote the set of events E and the mapping function ρ of π, respectively. We lift the aforementioned relations to processes by defining $\|_\pi$ and $\nrightarrow_\pi$ as the concurrency relation and the causality relation on nodes of the causal net of π, respectively. We shall omit the subscripts where the context is clear.

Figs. 2(a) and 2(b) show processes π_1 and π_2 of the net system in Fig. 1(a), respectively. When visualizing processes, conditions $c_i, c_i' \ldots$ refer to place p_i, i.e., $\rho(c_i) = \rho(c_i') = p_i$. Similarly, we assume events $e_j, e_j' \ldots$ to refer to transition t_j, i.e., $\rho(e_j) = \rho(e_j') = t_j$. In Fig. 2(a), for example, it holds that $e_2 \| e_4$, $e_4 \nrightarrow e_4'$, and $e_4' \| e_9$. Intuitively, the fact that one event is a cause for another event in a process π of a system S tells us that transitions which these events refer to can occur in order in occurrence sequences in S. The fact that two events are concurrent in π signals that the respective transitions can be both enabled at a reachable marking M, and can occur in any order starting from M.

Every event $e \in E_\pi$ describes an occurrence of transition $\rho_\pi(e)$. In [20], Jörg Desel suggests how a process of a net system S relates to occurrence sequences in S. In particular, every occurrence sequence in (N_π, M), where N_π is the causal net of a process (N_π, ρ) of S and M is a marking that puts one token at every source condition of N_π and

[4] $\rho(X) := \{\rho(x) \mid x \in X\}$ and $\rho^{-1}(z) := \{y \in Y \mid \rho(y) = z\}$, where X is a subset of ρ's domain Y.

no tokens elsewhere, *describes* via mapping ρ an occurrence sequence in S. For instance, process π_1 describes, among others, occurrence sequences $t_1 t_4 t_2$ and $t_1 t_2 t_3 t_4 t_5 t_6 t_7 t_8 t_9 t_4$.

Given a net system S, by Π_S we denote the set of all processes (up to isomorphism) that describe all executions in S. It is easy to see that the set Π_S, where S is the system in Fig. 1(a), is *infinite*, where $\pi_1 \notin \Pi_S$ and $\pi_2 \in \Pi_S$. Indeed, process π_2 in Fig. 2(b) describes, among nine executions, execution $t_1 t_2 t_3 t_4 t_5 t_6 t_7 t_8 t_9 t_3 t_4 t_5 t_6 t_{10}$ in the system in Fig. 1(a).

4 Definition of the 4C Spectrum

This section explores the definition of fundamental behavioral relations along two dimensions: occurrences of actions and their ordering. The first aspect is covered by different notions of conflict and co-occurrence relations (Section 4.1). Given two actions of a process model, these two relations relate to the presence or absence of joint occurrences of the two actions in the behavior model under a certain semantics. The second aspect is addressed by causality and concurrency relations (Section 4.2). Whenever there are joint occurrences of two actions, their ordering can be described by means of these relations.

4.1 Conflict & Co-occurrence

We start our study of behavioral relations on transitions of systems with a close look at the phenomena of conflict and co-occurrence. We noticed that every relation which characterizes how transition occurrences correlate across executions in systems can be captured in terms of two basic relations; one that checks if two transitions can both occur in some execution in the system, and the other that explores whether a transition can occur without some other transition.

Definition 4.1 (Basic conflict and co-occurrence)
Let $S := (N, M)$, $N := (P, T, F)$, be a system and let $x, y \in T$ be transitions of N.

- x *can conflict with* y in S, denoted by $x \dashrightarrow_S y$, iff there exists an execution σ of S such that $x \in \sigma$ and $y \notin \sigma$.
- x and y *can co-occur* in S, denoted by $x \rightharpoonup_S y$, iff there exists an execution σ of S such that $x \in \sigma$ and $y \in \sigma$.[5]

We shall often omit subscripts where the context is clear (for all the subsequently proposed relations). For each transition x it holds that $\overline{x \dashrightarrow x}$. If a transition x is dead, then it holds that $\overline{x \rightharpoonup x}$; otherwise $x \rightharpoonup x$.

To give some examples we consider the net system in Fig. 3. Here, among other relations, it holds that $t_1 \dashrightarrow t_2$, $t_2 \dashrightarrow t_5$, $t_2 \dashrightarrow t_6$, $\overline{t_3 \dashrightarrow t_4}$, $\overline{t_1 \rightharpoonup t_2}$, $t_2 \rightharpoonup t_6$, and $t_3 \rightharpoonup t_4$.

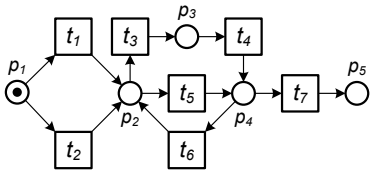


Fig. 3: A net system

Given a pair of transitions, one can use basic conflict and basic co-occurrence checks (refer to Definition 4.1) as atomic propositions, or *atoms*, in propositional logic formulas. These formulas can express 'rich' behavioral relations between pairs of transitions of a system. For instance, given two transitions x and y for which it holds that x can conflict with y, y can conflict with x, and x and y cannot co-occur,

[5] Given a sequence σ, $x \in \sigma$ denotes the fact that x is an element of σ.

i.e., $x \dashrightarrow y \wedge y \dashrightarrow x \wedge \overline{x \rightharpoonup y}$ evaluates to true, one can conclude that occurrences of x and y are *mutually exclusive*, or in *conflict*, in all executions in the system, e.g., transitions t_1 and t_2 are mutually exclusive in all executions in the system in Fig. 3.

Every well-formed propositional formula has an equivalent formula that is in disjunctive normal form (DNF), i.e., it is a disjunction of *conjunctive clauses*, where a conjunctive clause is a conjunction of one or more *literals*[6]. We are interested in *satisfiable* formulas that are not *tautologies*, as formulas that are either false or true in all interpretations are useless when it comes to characterizations of systems. A formula in DNF is satisfiable iff at least one of its conjunctive clauses is satisfiable.[7] Clearly, a conjunctive clause cannot be satisfied if it contains both the atoms a and $\overline{a}$. Thus, every satisfiable conjunctive clause that is composed of checks from Definition 4.1 (for a fixed pair of transitions) can never contain more than three literals.

A propositional formula is in *perfect* DNF if it is in DNF and each of its conjunctive clauses contains exactly one occurrence of each of the atoms. Every well-formed propositional formula has a *unique* equivalent formula that is in perfect DNF. Indeed, every elementary truth function over a set of atoms can be expressed as a formula in DNF where each conjunctive clause corresponds to one of the rows in the truth table for which the function is true. For each atom a that is made true in the row, the conjunctive clause should contain a and for atom a that is made false in that row, the conjunctive clause should contain $\overline{a}$. For example, formula $(x \dashrightarrow y \wedge y \dashrightarrow x \wedge \overline{x \rightharpoonup y}) \vee (x \dashrightarrow y \wedge y \dashrightarrow x \wedge x \rightharpoonup y)$ is an equivalent perfect DNF of formula $x \dashrightarrow y \wedge y \dashrightarrow x$.

Since every formula that exploits basic conflict and basic co-occurrence checks between a fixed pair of transitions operates with at most three literals, it has a unique equivalent formula in perfect 3DNF, i.e., a formula that is in DNF with each of its conjunctive clauses composed of exactly three different literals. These formulas in 3DNF can be seen as *canonical* forms of all propositional formulas over atoms induced by checks specified in Definition 4.1. Next, we take a closer look at all possible conjunctive clauses that can appear as parts of canonical formulas. These conjunctive clauses constitute fundamental conflict and co-occurrence checks between a fixed pair of transitions of a system, while disjunctions of these clauses allow one to obtain the 'relaxed' forms.

Definition 4.2 (Conflict and co-occurrence)
Let $S := (N, M)$, $N := (P, T, F)$, be a system and let $x, y \in T$ be transitions of N.

- x and y *co-occur* in S, denoted by $x \leftrightarrow_S y$, iff $\overline{x \dashrightarrow_S y} \wedge \overline{y \dashrightarrow_S x} \wedge x \rightharpoonup_S y$.
- x and y are in *conflict* in S, denoted by $x \#_S y$, iff $x \dashrightarrow_S y \wedge y \dashrightarrow_S x \wedge \overline{x \rightharpoonup_S y}$.
- x *requires* y in S, denoted by $x \rightharpoonup_S y$, iff $\overline{x \dashrightarrow_S y} \wedge y \dashrightarrow_S x \wedge x \rightharpoonup_S y$.
- x and y are *independent* in S, denoted by $x \rightleftharpoons_S y$, iff $x \dashrightarrow_S y \wedge y \dashrightarrow_S x \wedge x \rightharpoonup_S y$. ⌟

It is easy to check that transitions t_3 and t_4 co-occur in the system in Fig. 3, i.e., it holds that $t_3 \leftrightarrow t_4$, t_1 and t_2 are in conflict, i.e., $t_1 \# t_2$, t_5 requires t_7, i.e., $t_5 \rightharpoonup t_7$, and t_3 and t_5 are independent, i.e., $t_3 \rightleftharpoons t_5$. Note that $\rightharpoonup_S$ is asymmetric, whereas all other relations in Definition 4.2 are symmetric. In a system that is free from dead transitions every pair of transitions is in exactly one of the relations from Definition 4.2. Clearly, the relations in Definition 4.2 are disjoint. The fact that in the absence of dead transitions

[6] A *literal* is an atomic proposition or its negation.

[7] A formula is *satisfiable* if it is possible to find an interpretation that makes the formula true.

other conjunctive clauses (those not put into use in Definition 4.2) over the basic conflict and co-occurrence relations never evaluate to true is justified by the next proposition.

Proposition 4.3 (Basic conflict and co-occurrence) *If $S := (N, M)$, $N := (P, T, F)$, is a system without dead transitions, and $x, y \in T$ are transitions of N, then it holds that: (a) $\overline{x \dashrightarrow_S y}$ implies $x \rightharpoonup_S y$, and (b) $\overline{x \rightharpoonup_S y}$ implies $x \dashrightarrow_S y$ and $y \dashrightarrow_S x$.* ⌟

The proof of Proposition 4.3 is immediate. If x cannot conflict with y, i.e., $\overline{x \dashrightarrow_S y}$, then x and y can co-occur; otherwise x is a dead transition. Similarly, if x and y cannot co-occur, i.e., $\overline{x \rightharpoonup_S y}$, then there exist executions in S in which occurrences of x and y are mutually exclusive; otherwise either x, or y, or both x and y are dead transitions.

Next, we explore the remaining conjunctive clauses. These can relate dead transitions.

Definition 4.4 (Never (co-)occur)
Let $S := (N, M)$, $N := (P, T, F)$, be a system and let $x, y \in T$ be transitions of N.

- x and y *never occur* in S, denoted by $x \mid_S y$, iff $\overline{x \dashrightarrow_S y} \wedge \overline{y \dashrightarrow_S x} \wedge \overline{x \rightharpoonup_S y}$.
- x *but not* y *occurs* in S, denoted by $x \vdash_S y$, iff $x \dashrightarrow_S y \wedge \overline{y \dashrightarrow_S x} \wedge \overline{x \rightharpoonup_S y}$. ⌟

Indeed, in the general case, it is possible that either both transitions x and y are dead, in which case it holds that $x \mid y$, or one of the two transitions is dead, in which case it holds that $x \vdash y$ or $y \vdash x$. Definitions 4.2 and 4.4 propose 'strong' relations, i.e., every pair of transitions of a system is exactly in one of the proposed relations (they partition the Cartesian product of transitions). If one is interested in 'relaxed' forms of conflicts and co-occurrences, one can rely on the full spectrum of those, see the next definition.

Definition 4.5 (Spectrum of conflict and co-occurrence relations)
Let $S := (N, M)$, $N := (P, T, F)$, be a system, let $x, y \in T$ be transitions of N, and let Ω be a propositional formula on atomic propositions $x \dashrightarrow_S y$, $y \dashrightarrow_S x$, and $x \rightharpoonup_S y$.

- Ω specifies a *conflict* relation between x and y iff each conjunctive clause of the perfect 3DNF of Ω contains either $x \dashrightarrow_S y$ or $y \dashrightarrow_S x$ among its literals.
- Ω specifies a *co-occurrence* relation between x and y iff each conjunctive clause of the perfect 3DNF of Ω contains $x \rightharpoonup_S y$ as a literal. ⌟

Given a system S, by $\otimes_S$ and $\oplus_S$, we shall denote the sets of all, as per Definition 4.5, conflict and co-occurrence relations of S, respectively. For example, $x \leftrightarrow y \vee x \rightharpoonup y$ specifies a co-occurrence check between transitions x and y. Given two transitions x and y, there exist four conjunctive clauses (of the above discussed form) that contain literal $x \rightharpoonup y$; these are (i) $x \dashrightarrow y \wedge y \dashrightarrow x \wedge x \rightharpoonup y$, (ii) $\overline{x \dashrightarrow y} \wedge y \dashrightarrow x \wedge x \rightharpoonup y$, (iii) $x \dashrightarrow y \wedge \overline{y \dashrightarrow x} \wedge x \rightharpoonup y$, and (iv) $\overline{x \dashrightarrow y} \wedge \overline{y \dashrightarrow x} \wedge x \rightharpoonup y$. Thus, there are in total $2^4 - 1$, i.e., fifteen, distinct co-occurrence relations; each co-occurrence relation is either a single, or a disjunction of several, above proposed conjunctive clauses. By applying the same rationale, one can conclude that there are 63 distinct conflict relations.

4.2 Causality & Concurrency

This section looks into causality and concurrency phenomena. Similarly to the co-occurrence relations from Section 4.1, which report whether or not two actions can both occur in some execution in a system, causality and concurrency study situations

when two actions occur in the same execution. However, causality and concurrency additionally enforce orderings on action occurrences.

Causality and concurrency are well-studied concepts in the context of behavior models [13,9,19]. These models focus on patterns of occurrences of actions, i.e., every occurrence of an action within a behavior model is supported with a dedicated modeling construct. Thus, there can exist several events in a process of a system which describe different occurrences of the same transition of the system, see the process in Fig. 2(b) and the corresponding net system in Fig. 1(a). This modeling approach has a simple characterization in terms of causality and concurrency relations on events, i.e., every two distinct events from a causal net that underpins the process are either causal or concurrent. Two concurrent events can be enabled at the same time, i.e., the corresponding actions can be executed simultaneously, while two causal events indicate the presence of an order, i.e., one action is a prerequisite of the other. In fact, both representations, i.e., (i) a process, and (ii) its causality and concurrency relations, are equivalent; given a causal net one can construct its causality and concurrency relations, and vice versa [9].

In what follows, we systematically discover causality and concurrency relations (of different 'strengths') by projecting the corresponding phenomena from action occurrences of behavior models into actions of system models, i.e., from events of processes into transitions of net systems. This way a comprehensive classification of causality and concurrency relations for system models is obtained. Given two transitions x and y of a system, the classification is founded on three 'dimensions'. Intuitively, these dimensions correspond to: (i) whether a relation holds in some or all processes of a net system, (ii) whether a relation holds for one or all occurrences of x, and (iii) whether a relation holds for one or all occurrences of y. Before proceeding, for convenience considerations, we define a filter that selects those processes that describe occurrences of x and y, i.e., $\Delta_S(x,y) := \{\pi \in \Pi_S \mid \exists\, e_1 \in E_\pi\ \exists\, e_2 \in E_\pi : e_1 \neq e_2 \wedge \rho_\pi(e_1) = x \wedge \rho_\pi(e_2) = y\}$. The constraint $e_1 \neq e_2$ is introduced to allow excluding those processes from $\Delta_S(x,x)$ that contain only one event that describes an occurrence of x. Thus, $\Delta_S(x,y)$ contains processes that are of interest when checking causality and concurrency. Note that the causality and concurrency relations on events of a causal net are irreflexive. Next, we propose concurrency checks between transitions x and y that imply checks in all processes that describe occurrences of x and y to obtain *total* relations.

Definition 4.6 (Total concurrency)
Let $S := (N,M)$, $N := (P,T,F)$, be a system and let $x,y \in T$ be transitions of N.

- x and y are *total (mutual) concurrent* in S, denoted by $x \,||_S^{\forall\forall}\, y$, iff:
  $\forall\, \pi \in \Delta_S(x,y)\ \forall\, e_1 \in E_\pi\ \forall\, e_2 \in E_\pi : (e_1 \neq e_2 \wedge \rho_\pi(e_1) = x \wedge \rho_\pi(e_2) = y) \Rightarrow e_1 \,||_\pi\, e_2$.
- x is *total functional concurrent* for y in S, denoted by $x \,||_S^{\forall\exists}\, y$, iff:
  $\forall\, \pi \in \Delta_S(x,y)\ \forall\, e_1 \in E_\pi\ \exists\, e_2 \in E_\pi : \rho_\pi(e_1) = x \Rightarrow (\rho_\pi(e_2) = y \wedge e_1 \,||_\pi\, e_2)$.
- x is *total dominant concurrent* for y in S, denoted by $x \,||_S^{\exists\forall}\, y$, iff:
  $\forall\, \pi \in \Delta_S(x,y)\ \exists\, e_1 \in E_\pi\ \forall\, e_2 \in E_\pi : \rho_\pi(e_1) = x \wedge ((\rho_\pi(e_2) = y \wedge e_1 \neq e_2) \Rightarrow e_1 \,||_\pi\, e_2)$.
- x and y are *total existential concurrent* in S, denoted by $x \,||_S^{\exists\exists}\, y$, iff:
  $\forall\, \pi \in \Delta_S(x,y)\ \exists\, e_1 \in E_\pi\ \exists\, e_2 \in E_\pi : \rho_\pi(e_1) = x \wedge \rho_\pi(e_2) = y \wedge e_1 \,||_\pi\, e_2$. ⌟

Instead of checking concurrency patterns in all processes where two transitions co-occur, one can ask if these patterns hold in at least one process. Such an intent results in the next definition of existential concurrent relations.

Definition 4.7 (Existential concurrency)
Let $S := (N, M)$, $N := (P, T, F)$, be a system and let $x, y \in T$ be transitions of N.

- x and y are *existential total concurrent* in S, denoted by $x \,||_S^{\exists\forall\forall}\, y$, iff:
  $\exists\, \pi \in \Delta_S(x,y)\ \forall\, e_1 \in E_\pi\ \forall\, e_2 \in E_\pi : (e_1 \neq e_2 \wedge \rho_\pi(e_1) = x \wedge \rho_\pi(e_2) = y) \Rightarrow e_1 \,||_\pi\, e_2$.
- x is *existential functional concurrent* for y in S, denoted by $x \,||_S^{\exists\forall\exists}\, y$, iff:
  $\exists\, \pi \in \Delta_S(x,y)\ \forall\, e_1 \in E_\pi\ \exists\, e_2 \in E_\pi : \rho_\pi(e_1) = x \Rightarrow (\rho_\pi(e_2) = y \wedge e_1 \,||_\pi\, e_2)$.
- x is *existential dominant concurrent* for y in S, denoted by $x \,||_S^{\exists\exists\forall}\, y$, iff:
  $\exists\, \pi \in \Delta_S(x,y)\ \exists\, e_1 \in E_\pi\ \forall\, e_2 \in E_\pi : \rho_\pi(e_1) = x \wedge ((\rho_\pi(e_2) = y \wedge e_1 \neq e_2) \Rightarrow e_1 \,||_\pi\, e_2)$.
- x and y are *existential (mutual) concurrent* in S, denoted by $x \,||_S^{\exists\exists\exists}\, y$, iff:
  $\exists\, \pi \in \Delta_S(x,y)\ \exists\, e_1 \in E_\pi\ \exists\, e_2 \in E_\pi : \rho_\pi(e_1) = x \wedge \rho_\pi(e_2) = y \wedge e_1 \,||_\pi\, e_2$. ⌟

Similarly, one can talk about causality relations of different strengths. These relations can be trivially obtained by replacing all the concurrent checks $e_1 \,||_\pi\, e_2$ in Definitions 4.6 and 4.7 with the causal checks $e_1 \rightarrowtail_\pi e_2$.

For instance, the *total (mutual) causal* check between transitions x and y verifies if for every process π of the respective system and for every two distinct events e_1 and e_2 of π it holds that if e_1 describes an occurrence of x and e_2 describes an occurrence of y, then e_1 is a cause for e_2, i.e., it holds that $e_1 \rightarrowtail_\pi e_2$. The concurrency relations from Definitions 4.6 and 4.7 give rise to a lattice induced by logical implications; the same holds for the corresponding causality relations. It is easy to see that $x \,||^{\forall\forall\forall}\, y$ implies $x \,||^{\forall\forall\exists}\, y$. Clearly, if in every process that describes occurrences of x and y it holds that every occurrence of x is concurrent with some occurrence of y, and vice versa, then in each of these processes it also trivially holds that for every occurrence of x one can find a corresponding concurrent occurrence of y. Fig. 4 summarizes all the implications; transitive implications are not visualized. Here, every arrow specifies a logical implication between relations encoded in its endpoints, e.g., the implication from the above example is highlighted with a thicker arrow in the figure.

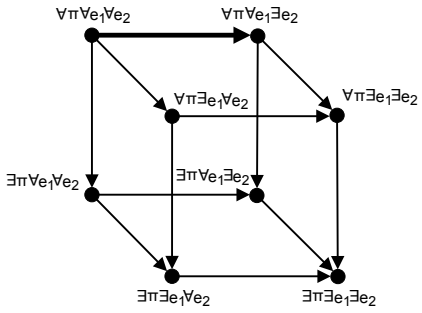


Fig. 4: The lattice of causality/concurrency

Finally, we define the spectrum of all causal and concurrent relations where each relation stems from a particular combination of co-occurrence and causal/concurrent patterns of action occurrences. Let $\Phi := \{\forall\forall\forall, \forall\forall\exists, \forall\exists\forall, \forall\exists\exists, \exists\forall\forall, \exists\forall\exists, \exists\exists\forall, \exists\exists\exists\}$.

Definition 4.8 (Spectrum of causality and concurrency relations)
Let $S := (N, M)$, $N := (P, T, F)$, be a system and let $x, y \in T$ be transitions of N.

- x and y are *causal*, denoted by $x \rightarrowtail_{S,\diamond}^{\phi} y$, iff it holds that
  $(x \diamond_S y) \wedge (x \rightarrowtail_S^{\phi} y)$, where $\diamond_S \in \oplus_S$ and $\phi \in \Phi$.
- x and y are *concurrent*, denoted by $x \,||_{S,\diamond}^{\phi}\, y$, iff it holds that
  $(x \diamond_S y) \wedge (x \,||_S^{\phi}\, y)$, where $\diamond_S \in \oplus_S$ and $\phi \in \Phi$. ⌟

It is easy to see that Definition 4.8 can be used to induce $15 \times 8 = 120$ distinct causal relations and the same number of concurrent relations on transitions of net systems. Definitions 4.5 and 4.8 jointly specify the 4C spectrum of behavioral relations.

Coming back to our motivating example in Section 2, one can clearly differentiate systems in Fig. 1 by using the above proposed behavioral relations on transitions. For instance, in Fig. 1(a), one can verify that the only transition with label *b* and the only transition with label *c* are existential causal, existential total concurrent, as well as total functional concurrent. In the system in Fig. 1(b), it also holds that the only transition with label *b* and the only transition with label *c* are existential causal. However, these transitions are not concurrent as per Definitions 4.6 and 4.7.

	a	*b*	*c*	*d*	*e*
a	$\rightleftharpoons$	$\forall\forall\forall\nearrow$	$\exists\forall\exists \,\|\|\, \exists\forall\exists\nearrow$	$\forall\forall\forall\nearrow$	$\forall\forall\forall\nearrow$
b	$\rightleftharpoons$	$\exists\exists\forall\nearrow$	$\exists\forall\exists \,\|\|\, \exists\exists\exists\nearrow$	$\forall\exists\forall\exists\nearrow$	$\forall\exists\forall\nearrow$
c	$\forall\exists\forall \,\|\|$	$\exists\forall\exists \,\|\|\, \exists\exists\exists\nearrow$	$\exists\exists\forall\nearrow$	$\forall\exists\forall\exists\nearrow$	$\exists\exists\forall \,\|\|\, \forall\exists\forall\nearrow$
d	$\rightleftharpoons$	$\exists\exists\exists\nearrow$	$\exists\exists\exists\nearrow$	$\exists\exists\forall\nearrow$	$\forall\exists\forall\nearrow$
e	$\rightleftharpoons$	$\exists\forall\exists\nearrow$	$\exists\forall\exists \,\|\|\, \exists\exists\exists\nearrow$	$\exists\forall\exists\nearrow$	$\exists\exists\forall\nearrow$

Table 1: Behavioral relations in Fig. 1(a)

	a	*b*	*c*	*d*	*e*
a	$\rightleftharpoons$	$\forall\forall\forall\nearrow$	$\forall\forall\forall\nearrow$	$\forall\forall\forall\nearrow$	$\forall\forall\forall\nearrow$
b	$\rightleftharpoons$	$\exists\exists\forall\nearrow$	$\exists\exists\exists\nearrow$	$\forall\exists\forall\exists\nearrow$	$\forall\exists\forall\nearrow$
c	$\rightleftharpoons$	$\forall\exists\forall\exists\nearrow$	$\exists\exists\forall\nearrow$	$\forall\exists\forall\exists\nearrow$	$\forall\exists\forall\nearrow$
d	$\rightleftharpoons$	$\exists\exists\exists\nearrow$	$\exists\exists\exists\nearrow$	$\exists\exists\forall\nearrow$	$\forall\exists\forall\nearrow$
e	$\rightleftharpoons$	$\exists\forall\exists\nearrow$	$\exists\forall\exists\nearrow$	$\exists\forall\exists\nearrow$	$\exists\exists\forall\nearrow$

Table 2: Beh. relations in Fig. 1(b)

Tables 1 and 2 summarize all the fundamental relations for the systems in Fig. 1. Note that only *strong* relations are proposed, where a relation is classified as strong iff it is not implied by any other relation. Tables 1 and 2 do not suggest *can conflict* relations. For both systems in Fig. 1 it trivially holds that *a*, *b*, *c*, and *d* can conflict with *e*.

5 Computation of Behavioral Relations

This section addresses the problem of computing behavioral relations.

5.1 Conflict & Co-occurrence

Section 4.1 introduced a spectrum of conflict and co-occurrence relations. Despite the large number of relations that aim at recognizing small differences between patterns of action occurrences, all the relations are founded on two basic checks from Definition 4.1. In what follows, we show how one can reduce these basic checks to the *reachability problem* [8]. To this end, we rely on the next transformation of net systems.

Definition 5.1 (Transition guarding)
Let $S := (N, M)$, $N := (P, T, F)$, be a system and let $x \in T$ be one of its transitions. An *injection of a guard* for x in S results in a system $S' := (N', M')$, $N' := (P', T', F')$, where:

- $P' := P \cup \{p, p'\}$, $T' := T \cup \{x'\}$, where p, p', x', are fresh nodes in N, $M' := M \uplus \{p\}$,
- $F' := F \cup \{(y, x') \mid y \in \bullet x\} \cup \{(x', y) \mid y \in x\bullet\} \cup \{(p, x'), (x', p'), (p', x), (x, p')\}$. ⌟

We say that fresh transition x' is the *guard* for x, whereas fresh places p and p' are the *guard* and the *control* place for x, respectively. It is easy to see that different orderings of injections of guards for all transitions from some set of transitions lead to the same system. Finally, computations of conflicts and co-occurrences are due to the next result.

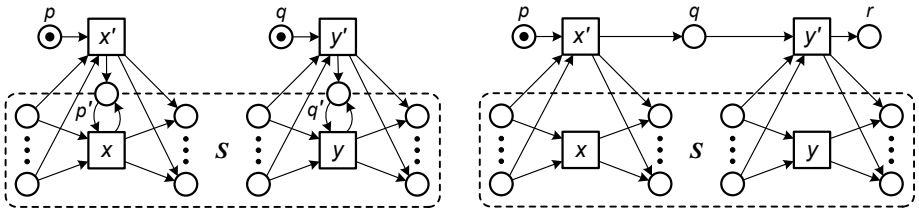


Fig. 5: Visualizations of: (a) the proof of Proposition 5.2, and (b) the proof of Proposition 5.6

Proposition 5.2 (Basic conflict and co-occurrence check)
Let $S := (N, M)$, $N := (P, T, F)$, be a net system with a unique reachable terminal marking M_t. Let $x, y \in T$, $x \neq y$, be transitions of N. Let $S' := (N', M')$, $N' := (P', T', F')$, be a net system obtained via injections of guards for x and y, where $p' \in P'$ and $q' \in P'$ are the control places for x and y, respectively, and $q \in P'$ is the guard place for y.

1. *x can conflict with y in S, i.e., $x \dashrightarrow_S y$, iff $M_t \uplus \{p', q\}$ is a reachable marking in S'.*
2. *x and y can co-occur in S, i.e., $x \rightharpoonup_S y$, iff $M_t \uplus \{p', q'\}$ is a reachable marking in S'.* ⌟

Fig. 5(a) visualizes the construction of the proof of Proposition 5.2. In the figure, transitions x' and y' are the guards for transitions x and y, respectively, whereas p, q and p', q' are the corresponding guard and control places. The box with the dashed borderline demarcates boundaries of the original (not augmented) net system S. It holds that in every occurrence sequence in the augmented system S' transition x' can occur at most once (instead of the first occurrence of transition x in S). Indeed, the preset of x' consists of the preset of x and one additional place p, which is marked in S'. An occurrence of x' implements the effect of the occurrence of x, i.e., the postset of x' contains that of x. Additionally, an occurrence of x' marks place p' that allows subsequent occurrences of x. The same principles apply to occurrences of transitions y and y' in S'. Thus, it appears that every execution σ in S can be trivially transformed into an execution σ' in S' by replacing the first occurrences of x and y with those of x' and y', respectively. Moreover, the presence and absence of transitions x and y in σ is clearly witnessed by the presence and absence of tokens at places p' and q', respectively, in a marking that is reachable via σ' in S'. The above proposed rationale justifies both statements in Proposition 5.2.

Note that the *reachability problem*, which given a net system S and a marking M consists of deciding if M is a reachable marking in S, is decidable [21,22].

5.2 Causality & Concurrency

This section proposes computations of causality and concurrency relations. To demonstrate feasibility, this section addresses the computation of two extreme cases (as per Section 4). We start by showing how to compute existential concurrency.

Proposition 5.3 (Existential concurrency check)
Let $S := (N, M)$, $N := (P, T, F)$, be a net system with a unique reachable terminal marking M_t, such that M_t is a home marking in S. Let $x, y \in T$ be transitions of N. Transitions x and y are existential concurrent in S, i.e., $x \parallel_S^{\exists\exists\exists} y$, iff there exists a reachable marking M' in S such that $\bullet x \uplus \bullet y$ is covered by M', i.e., $\exists\, M' \in [S\rangle : \bullet x \uplus \bullet y \leq M'$. ⌟

The statement in Proposition 5.3 follows immediately from the process construction principles described in [19,20]; refer to the proof of Theorem 3.6 in [19]. Note that the fact that M_t is a home marking ensures that every process of S that contains two concurrent events that refer to transitions x and y can indeed be extended to a process that describes an execution in S. Next, we propose a solution for checking total causality.

Proposition 5.4 (Total causality check) *Let $S := (N, M)$, $N := (P, T, F)$, be a net system with a unique reachable terminal marking M_t, such that M_t is a home marking in S. Let $x, y \in T$ be transitions of N. Transitions x and y are total causal in S, i.e., $x \rightarrowtail_S^{\forall\forall\forall} y$, iff for every occurrence sequence $\sigma := t_1 \ldots t_n$, $n \in \mathbb{N}_0$, in S and for every two positions i and j in σ, $1 \leq i, j \leq n$, $i \neq j$, such that $t_i = x$ and $t_j = y$ it holds that $i < j$.* ⌟

Again, the existence of a reachable terminal marking M_t, which is a home marking, ensures that every occurrence sequence of interest can be extended to an execution. It is immediate that if x and y are total causal then in every occurrence sequence y can never be observed before x. As for the converse statement, assume that there exists an occurrence sequence σ in which y precedes x and it holds that $x \rightarrowtail_S^{\forall\forall\forall} y$. Then, it also holds that $x \|_\pi y$ or $y \rightarrowtail_\pi x$, where π is a process induced by σ, refer to [20] for details, which leads to a contradiction. Thus, one can verify total causality between x and y by testing for the absence of an occurrence sequence in which y precedes x. To this end, we suggest to rely on the next transformation of net systems.

Definition 5.5 (Precedence test) Let $S := (N, M)$, $N := (P, T, F)$, be a system and let $x, y \in T$, be its transitions. An *injection of a precedence test* for x before y in S results in a system $S' := (N', M')$, $N' := (P', T', F')$, where:

- $P' := P \cup \{p, q, r\}$, $T' := T \cup \{x', y'\}$, where p, q, r and x', y' are fresh nodes in N,
- $F' := F \cup \{(z, x') \mid z \in {\bullet x}\} \cup \{(x', z) \mid z \in x^\bullet\} \cup \{(z, y') \mid z \in {\bullet y}\} \cup \{(y', z) \mid z \in y^\bullet\} \cup \{(p, x'), (x', q), (q, y'), (y', r)\}$, and $M' := M \uplus \{p\}$. ⌟

We say that the fresh place r is the *precedence control* place for x before y. Given transitions x and y of a net system, the injection of a precedence test for x before y can be used to check if x can precede y in some occurrence sequence in the net system.

Proposition 5.6 (Precedence test) *Let $S := (N, M)$, $N := (P, T, F)$, be a net system and let $x, y \in T$ be transitions of N. Let $S' := (N', M')$, $N' := (P', T', F')$, be a net system obtained via injection of a precedence test for x before y, where $r \in P'$ is the precedence control place for x before y. There exists an occurrence sequence $\sigma := t_1 \ldots t_n$, $n > 1$, where $t_i \in T$, $i \in 1..n$, in S such that an occurrence of x precedes an occurrence of y in σ, i.e., there exist positions i and j in σ, $1 \leq i < j \leq n$, such that $t_i = x$ and $t_j = y$, iff there exists a reachable marking M' in S' such that $\{r\}$ is covered by M', i.e., $\exists\, M' \in [S'\rangle : \{r\} \leq M'$.* ⌟

An injection of a precedence test is visualized in Fig. 5(b). Clearly, for every occurrence sequence in the augmented net system which leads to a marking that covers the precedence control place r it holds that transition x' precedes transition y'. As transition x and y can always occur instead of x' and y', both in the original and in the augmented net system, it holds that there exists an occurrence sequence in the original net system in which x precedes y, refer to the discussion in Section 5.1 for the intuition.

Note that the *covering problem*, which given a net system S and a marking M consists of deciding if there exists a reachable marking M' in S such that $M \leq M'$, i.e., M is

covered by M', is decidable [23]. Moreover, the *home state problem*, which given a net system S and a reachable marking M in S consists of deciding if M is a home marking in S, is decidable [24]. Finally, one can often ensure the existence of a unique reachable terminal marking in a given net system, e.g., if the system is bounded.

6 The 4C Spectrum for Labeled Net Systems

So far we discussed behavioral relations for systems in which actions are represented by transitions. We now turn to labeled systems that represent actions by (transition) labels, for which we first recall basic notions. We then lift the relations of the 4C spectrum to labels and show how their operationalizations are traced back to the unlabeled case.

6.1 Labeled Net Systems

In the context of process models, the motivation for labeled net systems is twofold. First, it is often convenient to distinguish between *observable* and *silent* transitions, separating actions that carry semantics in the application domain from those that have no domain interpretation. Second, one may wish to assign a certain semantics in the application domain to different transitions. Note that the models in Fig. 1 are, in fact, labeled systems, although so far the labels have not been considered in our investigations. Fig. 6 depicts another labeled system, in which two transitions are assigned the same label.

We briefly recall notions and notations for labeled net systems as follows.

Definition 6.1 (Labeled net) A *labeled net* is an ordered tuple $N := (P, T, F, \mathcal{T}, \lambda)$, where (P, T, F) is a net, $\mathcal{T}$ is a set of labels, where $\tau \in \mathcal{T}$ is a special label, and $\lambda : T \to \mathcal{T}$ is a function that assigns to each transition in T a *label* in $\mathcal{T}$. ⌟

If $\lambda(t) \neq \tau$, $t \in T$, then t is *observable*; otherwise, t is *silent*. Labeling directly extends to systems, i.e., a *labeled net system*, or a *labeled system*, is an ordered pair $S := (N, M)$, where N is a labeled net and M is a marking of N.

In the graphical notation, a label of a transition is shown next to the transition rectangle with its short form shown inside of the rectangle, whereas silent transitions are visualized as empty rectangles. For instance, two transitions in the system in Fig. 6 'wear' label "*Get Budget Approval*", which we also refer to via its short form *b*. Note that Fig. 3 visualizes a regular, i.e., unlabeled, net system.

For a labeled system $S := (N, M)$, $N := (P, T, F, \mathcal{T}, \lambda)$, any occurrence sequence $\sigma := t_1 \dots t_n$, $n \in \mathbb{N}_0$, of N can be interpreted as a sequence of observable labels. Let $\Theta(\sigma, k) := |\{t_i \in \sigma \mid i \leq k \wedge \lambda(t_i) = \tau\}|$ be the number of silent transitions of σ up to and including position k. Then, the *observable sequence* induced by σ, denoted by $\lambda(\sigma) := \lambda_1 \dots \lambda_m$, $m \leq n$, is such that $\lambda_i := \lambda(t_j)$, $j = i + \Theta(\sigma, j)$, for $0 \leq i \leq m$, and $n = m + \Theta(\sigma, n)$.

In the same vein, processes of a labeled system can be interpreted in terms of transition labels. That is, given a process $\pi := (N_\pi, \rho)$, $N_\pi := (B, E, G)$, of a labeled net system $S := (N, M)$, $N := (P, T, F, \mathcal{T}, \lambda)$, each event $e \in E$ is not only related to a transition $\rho(e) \in T$, but also to a label $(\lambda \circ \rho)(e) \in \mathcal{T}$.

6.2 Relations over Transition Labels

Conflict & Co-occurrence. All the proposed conflict and co-occurrence relations from Section 4.1 are based on the binary *can conflict* and *can co-occur* relations that are

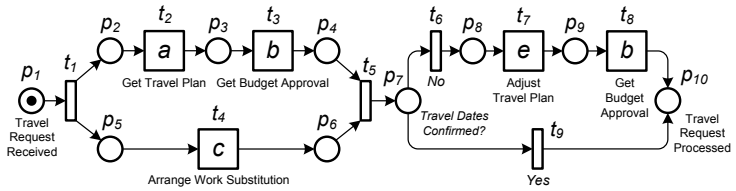


Fig. 6: A behavioral specification given as a labeled Petri net system

defined over transitions, see Definition 4.1. Once any occurrence sequence and, thus, any execution in a labeled system $S := (N, M)$, $N := (P, T, F, \mathcal{T}, \lambda)$, is interpreted as a sequence of observable labels, both relations can be naturally lifted to labels. Label λ_x *can conflict with* label λ_y in S, $\lambda_x, \lambda_y \in \mathcal{T} \setminus \{\tau\}$, denoted by $\lambda_x \dashrightarrow_{\lambda,S} \lambda_y$, iff there exists an execution σ of S such that $\lambda_x \in \lambda(\sigma)$ and $\lambda_y \notin \lambda(\sigma)$. Labels λ_x and λ_y *can co-occur* in S, denoted by $\lambda_x \rightharpoonup_{\lambda,S} \lambda_y$, iff there exists an execution σ in S such that $\lambda_x \in \lambda(\sigma)$ and $\lambda_y \in \lambda(\sigma)$. E.g., for the net system in Fig. 6 it holds that $c \dashrightarrow_\lambda e$, $\overline{e \dashrightarrow_\lambda c}$, and $c \rightharpoonup_\lambda e$.

Causality & Concurrency. The causality and concurrency relations from Section 4.2 test for processes that contain events related to certain transitions. For the case of labeled systems, these tests can refer to events that relate to certain observable labels. We exemplify how the relations are lifted with the total (mutual) concurrency relation. For a labeled system $S := (N, M)$, $N := (P, T, F, \mathcal{T}, \lambda)$ and two labels $\lambda_x, \lambda_y \in \mathcal{T} \setminus \{\tau\}$, let $\Delta_{\lambda,S}(\lambda_x, \lambda_y) := \{\pi \in \Pi_S \mid \exists\, e_1 \in E_\pi\, \exists\, e_2 \in E_\pi : e_1 \neq e_2 \wedge (\lambda \circ \rho_\pi)(e_1) = \lambda_x \wedge (\lambda \circ \rho_\pi)(e_2) = \lambda_y\}$ be the set of processes containing events of interest. Then, two labels λ_x and λ_y are total (mutual) concurrent in S, denoted by $\lambda_x \,||^{\forall\forall\forall}_{\lambda,S}\, \lambda_y$, iff $\forall \pi \in \Delta_{\lambda,S}(\lambda_x, \lambda_y)\, \forall e_1 \in E_\pi \forall e_2 \in E_\pi : (e_1 \neq e_2 \wedge (\lambda \circ \rho_\pi)(e_1) = \lambda_x \wedge (\lambda \circ \rho_\pi)(e_2) = \lambda_y) \Rightarrow e_1 \,||_\pi\, e_2$. For example, for the net system in Fig. 6 it holds that $c \,||^{\forall\forall\exists}_{\lambda}\, b$ and $c \rightarrowtail^{\exists\forall\exists}_{\lambda} b$.

6.3 Computation

Computation of the relations defined over labels can be traced back to the computation of the relations defined over transitions, as introduced in Section 5. The idea behind our approach is to implement *label unification*, such that every occurrence of the same label is manifested in the occurrence of a specific (possibly newly introduced) transition. To this end, we perform structural transformations of labeled net systems.

Definition 6.2 (Label unification) Let $S := (N, M)$, $N := (P, T, F, \mathcal{T}, \lambda)$, be a labeled net system and let $\lambda_x \in \mathcal{T}$ be a label. Let $T_{\lambda_x} := \{t \in T \mid \lambda(t) = \lambda_x\}$ be the set of all transitions of N labeled with λ_x. A *label unification* of λ_x in S results in a labeled net system S', such that: (i) if $\lambda_x = \tau$ or $|T_{\lambda_x}| \leq 1$ then $S' := S$, i.e., the net system is not changed, and (ii) if $\lambda_x \neq \tau$ and $|T_{\lambda_x}| > 1$ then $S' := (N', M')$, $N' := (P', T', F', \mathcal{T}, \lambda')$, where:

- $P' := P \cup \{*p, p* \} \cup \bigcup_{t \in T_{\lambda_x}} \{p_t, \hat{p}_t\}$, $T' := T \cup \{\hat{t}\} \cup \bigcup_{t \in T_{\lambda_x}} \{*t, t*\}$, such that $(\{*p, p*, \hat{t}\} \cup \bigcup_{t \in T_{\lambda_x}} \{p_t, \hat{p}_t, *t, t*\}) \cap (P \cup T) = \emptyset$,
- $F' := F \cup \{(*p, \hat{t}), (\hat{t}, p*)\} \cup \bigcup_{t \in T_{\lambda_x}} \{(p, *t) \mid p \in {\bullet t}\} \cup \bigcup_{t \in T_{\lambda_x}} \{(t*, p) \mid p \in t\bullet\} \cup \bigcup_{t \in T_{\lambda_x}} \{(\hat{p}_t, t), (*t, p_t), (p_t, t*), (*t, *p), (p*, t*)\}$,
- $\lambda'(t) := \lambda(t)$, for every $t \in T$, $\lambda'(\hat{t}) := \lambda_x$, $\lambda'(*t) := \lambda'(t*) := \tau$, for every $t \in T_{\lambda_x}$, and
- $M'(p) := M(p)$, for every $p \in P$, $M'(*p) := M'(p*) := 0$, $M'(p_t) := M'(\hat{p}_t) := 0$, for every $t \in T_{\lambda_x}$. ⌟

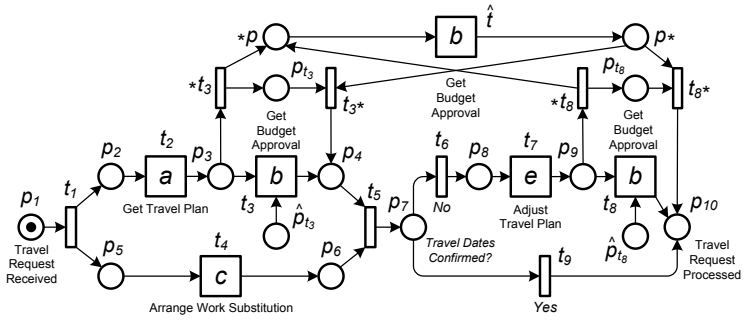


Fig. 7: The system of Fig. 6 after applying unification of label b

If $\lambda_x \neq \tau$ and $|T_{\lambda_x}| > 1$, we say that transition $\hat{t}$ is the *solitary* transition for label λ_x in S'. Additionally, we say that transition t is the *solitary* transition for label λ_x in S' if $\lambda_x \neq \tau$ and $T_{\lambda_x} = \{t\}$. If $\lambda_x \neq \tau$ and $|T_{\lambda_x}| > 1$, all transitions in T_{λ_x} are dead in S'; since places $\hat{p}_{t_1}$ to $\hat{p}_{t_n}$ have empty presets and are not marked in M'. Thus, given a label λ_x, the solitary transition for λ_x (if such a transition exists) is the only transition with label λ_x that may appear in an occurrence sequence in S'. If $\lambda_x \neq \tau$, $|T_{\lambda_x}| > 1$, and $t \in T$ is such that $\lambda(t) = \lambda_x$, we say that $*t$ and $t*$ are the *presolitary* and the *postsolitary* transition of t for λ_x, respectively. For the example net system shown in Fig. 6, the result of applying unification of label b is illustrated in Fig. 7. It is easy to see that all applications of the same set of unification operations (in different orders) result in the same system.

In what follows, we argue that an original net system and its augmented version that is obtained via unification of every label in a given set of labels induce – from the point of view of an external observer – equivalent behaviors. In order to facilitate subsequent discussions, we rely on the next auxiliary definition.

Definition 6.3 (Labeled elementary event structure)
A *labeled elementary event structure* is an ordered tuple $\mathcal{E} := (E, \prec, \mathcal{T}, \lambda)$, where E is a set of *events*, $\prec$ is a partial order over E, called the *causality relation*, $\mathcal{T}$ is a set of labels, and $\lambda : E \to \mathcal{T}$ is a function that assigns to each event in E a *label* in $\mathcal{T}$. ⌟

The ordered pair $(E, \prec)$ is a partially ordered set of events, also known as an *elementary event structure*. In [9], the authors show that elementary event structures are alternative representations of causal nets and, thus, of processes of net systems [19]. Consequently, every labeled elementary event structure is a partially ordered set of labeled events that can be used to encode essential information about observable transition occurrences.

Let $\pi := (N_\pi, \rho)$, $N_\pi := (B, E, G)$, be a process of a labeled system $S := (N, M)$, $N := (P, T, F, \mathcal{T}, \lambda)$, and let $E' := \{e \in E \mid (\lambda \circ \rho)(e) \neq \tau\}$ be the set of events of N_π that correspond to observable transitions of N. Then, $\mathcal{E}[\pi] := (E, \nrightarrow_\pi |_E, \mathcal{T}, \lambda \circ (\rho|_E))$ is the labeled elementary event structure induced by π, and $\mathcal{O}[\mathcal{E}[\pi]] := (E', \nrightarrow_\pi |_{E'}, \mathcal{T} \setminus \{\tau\}, \lambda \circ (\rho|_{E'}))$ is the *observable* elementary event structure induced by $\mathcal{E}[\pi]$. Note that similar transformations are proposed in [9,25]; for instance, in [25], the authors propose the λ*-abstraction* of π. Given a process π, the observable elementary event structure $\mathcal{O}[\mathcal{E}[\pi]]$ describes all occurrences of observable transitions that are also captured in π, as well as the causality and the concurrency relations on the corresponding events.

Intuitively, two event structures describe the same observable behavior if they are *order-isomorphic*, refer to [25] for details.

Definition 6.4 (Order-isomorphism) Let $\mathcal{O}_1 := (E_1, \prec_1, \mathcal{T}, \lambda_1)$ and $\mathcal{O}_2 := (E_2, \prec_2, \mathcal{T}, \lambda_2)$ be two observable elementary event structures (both with labels in $\mathcal{T}$). Then, $\mathcal{O}_1$ and $\mathcal{O}_2$ are *order-isomorphic*, denoted by $\mathcal{O}_1 \cong \mathcal{O}_2$, iff there exists a bijection $\beta : E_1 \to E_2$, such that : (i) $\forall e \in E_1 : \lambda_1(e) = (\lambda_2 \circ \beta)(e)$ and (ii) $\forall e, e' \in E_1 : e \prec_1 e' \Leftrightarrow \beta(e) \prec_2 \beta(e')$. ⌟

Using the notion of observable elementary event structures, we show that, indeed, label unification does not change the behavior of a net system.

Lemma 6.5 (Equivalence of observable behaviors)
Let $S := (N, M)$, $N := (P, T, F, \mathcal{T}, \lambda)$, and S' be labeled net systems, where S' is obtained from S via label unification of every label in $\mathcal{T}' \subseteq \mathcal{T}$. For every process π of S there exists a process π' of S', such that $\mathcal{O}[\mathcal{E}[\pi]] \cong \mathcal{O}[\mathcal{E}[\pi']]$, and vice versa. ⌟

Proof. (Sketch) Let $S' := (N', M')$, $N' := (P', T', F', \mathcal{T}, \lambda')$. Let $\pi := (N_\pi, \rho)$, $N_\pi := (B, E, G)$, be a process of S, and let $\mathcal{E}[\pi] := (E, \prec, \mathcal{T}, \eta)$ be the labeled elementary event structure induced by π. We prove the statement by induction on the structure of $\mathcal{E}[\pi]$. It is easy to see that the empty processes of S and S', i.e., processes without a single event, induce order-isomorphic observable elementary event structures. By the induction hypothesis, there exists a process π' of S' such that $\mathcal{O}[\mathcal{E}[\pi']]$ is order-isomorphic with $\mathcal{O}[(E', \prec|_{E'}, \mathcal{T}, \eta|_{E'})]$, where E' is a *left-closed* subset of E, i.e., $e \in E'$ and $e' \prec e$, $e' \in E$, implies $e' \in E'$. Let $d \in E$ be an event such that $d \notin E'$ and $E'' := E' \cup \{d\}$ is a left-closed subset of E. Next, we construct a process π^* from π' by appending to π' a fresh event, refer to [19] for details. When appending a fresh event that refers to transition t, we also append conditions that map to places in the postset of t. Additionally, the flow relation of the causal net of the process under construction is completed to respect the environment of t. We distinguish the following two cases:

(i) $(\lambda \circ \rho)(d) = \tau$ or $|\{t \in T \mid \lambda(t) = (\lambda \circ \rho)(d)\}| = 1$ or $(\lambda \circ \rho)(d) \notin \mathcal{T}'$.
We construct π^* by appending a fresh event f to π' that maps to transition $\rho(d)$.

(ii) Otherwise, we construct π^* by appending three fresh events f, f', and f'' to π' such that f maps to the solitary transition for label $(\lambda \circ \rho)(d)$, whereas f' and f'' map to the presolitary and the postsolitary transition of $\rho(d)$ for $(\lambda \circ \rho)(d)$, respectively. Event f' must be appended first. Then, f can be appended before appending f''.

It is easy to see that, because of Definition 6.2, both proposed constructions can be implemented and in both cases it holds that $\mathcal{O}[\mathcal{E}[\pi^*]] \cong \mathcal{O}[(E'', \prec|_{E''}, \mathcal{T}, \eta|_{E''})]$.

The proof of the converse statement proceeds similarly to the one proposed above. ∎

Given a label λ_x, $\lambda_x \neq \tau$, and a labeled system S, unification of λ_x in S results in a system with at most one transition that carries label λ_x and is not dead. Thus, it is enforced that all occurrences of λ_x result from occurrences of a single transition. This allows to trace back the relations of the 4C spectrum over labels to those over transitions. To this end, it suffices to consider basic conflict, basic co-occurrence, and the eight instantiations ($\Phi := \{\forall\forall\forall, \forall\forall\exists, \forall\exists\forall, \forall\exists\exists, \exists\forall\forall, \exists\forall\exists, \exists\exists\forall, \exists\exists\exists\}$) of causality and concurrency.

Proposition 6.6 (Relations on labels) *Let $S := (N, M)$, $N := (P, T, F, \mathcal{T}, \lambda)$, and $S' := (N', M')$, $N' := (P', T', F', \mathcal{T}, \lambda')$, be labeled systems, where S' is obtained from S via label unification of every label in $\mathcal{T}' \subseteq range(\lambda')$.[8] Let $\lambda_x, \lambda_y \in \mathcal{T}' \setminus \{\tau\}$ be two labels. Let $t_x \in T'$ and $t_y \in T'$ be the solitary transitions for λ_x and λ_y, respectively.*

[8] *range*(f) denotes the range of function f, i.e., the image of f's domain under f.

- t_x *can conflict with* t_y *in* S'*, i.e.,* $t_x \dashrightarrow_{S'} t_y$*, iff* λ_x *can conflict with* λ_y *in* S*,* $\lambda_x \dashrightarrow_{\lambda,S} \lambda_y$*.*
- t_x *and* t_y *can co-occur in* S'*, i.e.,* $t_x \rightharpoonup_{S'} t_y$*, iff* λ_x *and* λ_y *can co-occur in* S*,* $\lambda_x \rightharpoonup_{\lambda,S} \lambda_y$*.*
- t_x *and* t_y *are in the causality relation* $\rightarrowtail^{\phi}_{S'}$ *in* S' *iff* λ_x *and* λ_y *are in the causal relation* $\rightarrowtail^{\phi}_{\lambda,S}$ *in* S*, where* $\phi \in \Phi$*.*
- t_x *and* t_y *are in the concurrency relation* $||^{\phi}_{S'}$ *in* S' *iff* λ_x *and* λ_y *are in the concurrency relation* $||^{\phi}_{\lambda,S}$ *in* S*, where* $\phi \in \Phi$*.* ⌟

Proposition 6.6 is a consequence of Lemma 6.5. Recalling our earlier examples in Figs. 6 and 7, for instance, we observe that $c \ ||^{\forall\forall\exists}_{\lambda} \ b$ and $c \rightarrowtail^{\exists\forall\exists}_{\lambda} b$ in the system in Fig. 6 since it holds that $t_4 \ ||^{\forall\forall\exists}_{\lambda} \hat{t}$ and $t_4 \rightarrowtail^{\exists\forall\exists}_{\lambda} \hat{t}$ in the system in Fig. 7, respectively.

Finally, note that the transformation presented in this section has value beyond the operationalization of behavioral relations on labels. Since each label occurrence in the resulting system relates to the occurrence of a unique transition while preserving the observable behaviors of the original system (Lemma 6.5), this result allows for the grounding of any computation over processes of labeled systems in the unlabeled case.

7 Evaluation

The approach for computing behavioral relations from Section 5 has been implemented and is publicly available as part of the `jBPT` initiative [26].[9] Using this implementation, we conducted an experiment to assess the performance of the technique. The experiment was carried out on a laptop with a dual core Intel CPU with 2.26 GHz, 4GB of memory, running Windows 7 and SUN JVM 1.7 (with standard allocation of memory). To eliminate load time, each check of a behavioral relation between a pair of transitions was executed six times, and we recorded average times of the second to sixth execution.

The study was conducted on a set of 367 systems that model financial services and processes from the telecommunication domain. The systems were selected from a collection of 735 models [27]. To ensure that each system has a unique terminal marking, unsound systems [28] were filtered out. In the experiment, the reachability and the covering problem (for bounded systems) were tackled using the LoLA tool ver. 1.14.[10]

Table 3 reports average times (in milliseconds) for checking behavioral relations. The first two columns report on the characteristics of the collection by providing information on the number 'n' of systems within a given 'Size' range (measured as the number of nodes). In order to obtain average values, checks of behavioral relations between random pairs of different transitions were performed. Each value was measured as the

Size	n	$x \dashrightarrow y$	$x \rightharpoonup y$	$x \ ||^{\exists\exists\exists} \ y$	$x \rightarrowtail^{\forall\forall\forall} y$
1–50	211	45	45	44	44
51–100	106	50	49	50	50
101–150	39	95	97	351	492
151–200	5	182	187	1 626	1 544
201–250	3	66	70	90	82
251–548	3	65	128	145	132
1–548	367	54	55	101	115

Table 3: Average computation times (in ms)

[9] `http://code.google.com/p/jbpt/`

[10] `http://service-technology.org/lola/`

average time of checking 1000 random transition pairs and is recorded in the last four columns. The last row in the table shows average computation times over all systems in the collection. Note that the significant increase in computation times for $x \parallel^{\exists\exists\exists} y$ (1 626 ms) and $x \nrightarrow^{\forall\forall\forall} y$ (1 544 ms) in the fourth row of the table is due to a single system for which checks of each behavioral relation took approximately seven seconds.

8 Related Work

We already discussed how our work is embedded in existing classifications of models of concurrency, i.e., fundamental behavioral relations provide a declarative characterization of the behavior model induced by a process model under a certain semantics, cf. [4]. Below, we focus on other declarative behavioral formalisms and their applications.

For interleaving semantics, declarative characterizations of behavior models can be formalized using temporal logic, most prominently Linear Temporal Logic (LTL) for linear time semantics and Computational Tree Logic (CTL) for branching time, see [5] for an overview. Then, behavioral relations are directly grounded in logic formulas over actions as realized, for instance, in the DecSerFlow language for process modeling [29]. For concurrent semantics, in turn, rewrite logics have been used as a formal grounding [30]. While all these approaches provide the basis for declarative characterizations of behavior models, they do not specify which characteristics shall be captured, which is the question addressed in this work. Also, only very few works target completeness of such declarative characterizations under a certain semantics. For process models given as net systems of a particular class (free-choice, live, and bounded), completeness of a certain declarative characterization of interleaving, linear time semantics was demonstrated in [10]. However, we lack more general results for larger classes of process models under concurrent or branching time semantics.

Applications of basic behavioral relations for process model search and querying have been discussed in Section 2 already. Many more applications have been explored in the literature. Basic behavioral relations are at the core of many techniques in process mining, which connects process models with events that represent action occurrences [31]. These techniques include discovery algorithms, such as the α-algorithm [32], as well as methods for checking conformance between models and events [33]. Also, basic behavioral relations are the foundation for techniques for model synchronization [34]. Here, a change that is located using behavioral relations in one model helps to locate a region for applying the change in another model.

9 Conclusion

In this paper, we addressed the lack of a systematic exploration of fundamental behavioral relations for the analysis of concurrent systems. To this end, we defined a set of relations that capture co-occurrence, conflict, causality, and concurrency, proved that the proposed relations give rise to implication lattices of relations, and suggested operationalizations of these relations. Further, we lifted the relations as well as the operationalizations to labeled net systems. The computation of behavioral relations was evaluated with real-world process models. As such, our work builds a rigorous formal foundation for analysis techniques that are based on behavioral relations.

Some immediate directions for future work include: (i) development of techniques for computing the remaining causality and concurrency relations of the 4C spectrum (beyond the proposed extreme cases), (ii) generalization, and subsequent operationalization, of the proposed behavioral relations, e.g., one may be willing to generalize results to weighted nets, as well as to lift the proposed behavioral relations from executions to fair runs [35], and (iii) application of the relations as behavioral abstractions for modeling of, reasoning over, and analysis of systems.

Acknowledgments This work is partly funded by the ARC Linkage Project LP110100252. NICTA is funded by the Australian Government (Department of Broadband, Communications and the Digital Economy) and the Australian Research Council through the ICT Centre of Excellence program.

References

1. Winkler, S.: Information flow between requirement artifacts. Results of an empirical study. In: REFSQ. Volume 4542 of LNCS., Springer (2007) 232–246
2. Dumas, M., van der Aalst, W.M.P., ter Hofstede, A.H.M.: Process-Aware Information Systems: Bridging People and Software Through Process Technology. Wiley (2005)
3. ter Hofstede, A.H.M., van der Aalst, W.M.P., Adams, M., Russell, N., eds.: Modern Business Process Automation — YAWL and its Support Environment. Springer (2010)
4. Sassone, V., Nielsen, M., Winskel, G.: Models for concurrency: Towards a classification. TCS **170**(1–2) (1996) 297–348
5. Baier, C., Katoen, J.P.: Principles of Model Checking. MIT Press (2008)
6. Reisig, W.: Elements of Distributed Algorithms: Modeling and Analysis with Petri Nets. Springer (1998)
7. Object Management Group (OMG): Unified Modeling Language: Superstructure. Version 2.1.2. Technical report (November 2007)
8. Hack, M.: Decidability Questions for Petri Nets. Outstanding Dissertations in the Computer Sciences. Garland Publishing, New York (1975)
9. Nielsen, M., Plotkin, G.D., Winskel, G.: Petri nets, event structures and domains, Part I. TCS **13** (1981) 85–108
10. Weidlich, M., van der Werf, J.M.E.M.: On profiles and footprints — relational semantics for Petri nets. In: Petri Nets. Volume 7347 of LNCS., Springer (2012) 148–167
11. Dunham, M.H.: Data Mining: Introductory and Advanced Topics. Prentice-Hall (2002)
12. Dijkman, R.M., La Rosa, M., Reijers, H.A.: Managing large collections of business process models — current techniques and challenges. Computers in Industry **63**(2) (2012) 91–97
13. Petri, C.A.: Non-Sequential Processes. ISF. GMD (1977)
14. ter Hofstede, A.H.M., Ouyang, C., La Rosa, M., Song, L., Wang, J., Polyvyanyy, A.: APQL: A process-model query language. In: AP-BPM. Volume 159 of LNBIP. (2013) 23–38
15. Polyvyanyy, A., La Rosa, M., ter Hofstede, A.H.M.: Indexing and efficient instance-based retrieval of process models using untanglings. In: CAiSE. (2014) To appear.
16. Kunze, M., Weidlich, M., Weske, M.: Behavioral similarity — a proper metric. In: BPM. Volume 6896 of LNCS., Springer (2011) 166–181
17. van Dongen, B.F., Dijkman, R.M., Mendling, J.: Measuring similarity between business process models. In: CAiSE. Volume 5074 of LNCS., Springer (2008) 450–464
18. Zha, H., Wang, J., Wen, L., Wang, C., Sun, J.: A workflow net similarity measure based on transition adjacency relations. Computers in Industry **61**(5) (2010) 463–471
19. Goltz, U., Reisig, W.: The non-sequential behavior of Petri nets. IANDC **57**(2/3) (1983)
20. Desel, J.: Validation of process models by construction of process nets. In: BPM. Volume 1806 of LNCS., Springer (2000) 110–128

21. Mayr, E.W.: Persistence of vector replacement systems is decidable. Acta Inf. **15** (1981) 309–318
22. Kosaraju, S.R.: Decidability of reachability in vector addition systems (preliminary version). In: STOC, ACM (1982) 267–281
23. Rackoff, C.: The covering and boundedness problems for vector addition systems. TCS **6** (1978) 223–231
24. Escrig, D.F.: Decidability of home states in place transition systems (1986) Internal Report. Dpto. Informatica y Automatica. Univ. Complutense de Madrid.
25. Best, E., Devillers, R.R., Kiehn, A., Pomello, L.: Concurrent bisimulations in Petri nets. Acta Inf. **28**(3) (1991) 231–264
26. Polyvyanyy, A., Weidlich, M.: Towards a compendium of process technologies: The jBPT library for process model analysis. In: CAiSE Forum. Volume 998 of CEUR. (2013)
27. Fahland, D., Favre, C., Koehler, J., Lohmann, N., Völzer, H., Wolf, K.: Analysis on demand: Instantaneous soundness checking of industrial business process models. DKE (5) (2011) 448–466
28. van der Aalst, W.M.P.: Verification of workflow nets. In: ICATPN. Volume 1248 of LNCS., Springer (1997) 407–426
29. van der Aalst, W.M.P., Pesic, M.: DecSerFlow: Towards a truly declarative service flow language. In: WS-FM. Volume 4184 of LNCS., Springer (2006) 1–23
30. Meseguer, J., Talcott, C.L.: A partial order event model for concurrent objects. In: CONCUR. Volume 1664 of LNCS., Springer (1999) 415–430
31. van der Aalst, W.M.P.: Process Mining — Discovery, Conformance and Enhancement of Business Processes. Springer (2011)
32. van der Aalst, W.M.P., Weijters, T., Maruster, L.: Workflow mining: Discovering process models from event logs. TKDE **16**(9) (2004) 1128–1142
33. Weidlich, M., Polyvyanyy, A., Desai, N., Mendling, J., Weske, M.: Process compliance analysis based on behavioural profiles. IS **36**(7) (2011) 1009–1025
34. Weidlich, M., Mendling, J., Weske, M.: Propagating changes between aligned process models. JSS **85**(8) (2012) 1885–1898
35. Kindler, E., van der Aalst, W.M.P.: Liveness, fairness, and recurrence in Petri nets. IPL **70**(6) (1999) 269–27