On the Expressive Power of Behavioral Profiles

Artem Polyvyanyy[1]

Abel Armas-Cervantes[2,*]

Marlon Dumas[2]

Luciano García-Bañuelos[2]

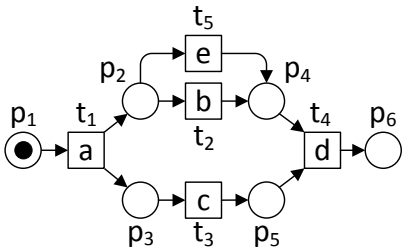
[1] Queensland University of Technology, Brisbane, Australia, artem.polyvyanyy@qut.edu.au
[2] Institute of Computer Science, University of Tartu, Estonia, {abel.armas,marlon.dumas,luciano.garcia}@ut.ee

Abstract. Behavioral profiles have been proposed as a behavioral abstraction of dynamic systems, specifically in the context of business process modeling. A behavioral profile can be seen as a complete graph over a set of task labels, where each edge is annotated with one relation from a given set of binary behavioral relations. Since their introduction, behavioral profiles were argued to provide a convenient way for comparing pairs of process models with respect to their behavior or computing behavioral similarity between process models. Still, as of today, there is little understanding of the expressive power of behavioral profiles. Via counter-examples, several authors have shown that behavioral profiles over various sets of behavioral relations cannot distinguish certain systems up to trace equivalence, even for restricted classes of systems represented as safe workflow nets. This paper studies the expressive power of behavioral profiles from two angles. Firstly, the paper investigates the expressive power of behavioral profiles and systems captured as acyclic workflow nets. It is shown that for unlabeled acyclic workflow net systems, behavioral profiles over a simple set of behavioral relations are expressive up to configuration equivalence. When systems are labeled, this result does not hold for any of several previously proposed sets of behavioral relations. Secondly, the paper compares the expressive power of behavioral profiles and regular languages. It is shown that for any set of behavioral relations, behavioral profiles are strictly less expressive than regular languages, entailing that behavioral profiles cannot be used to decide trace equivalence of finite automata and thus Petri nets.

1. Introduction

Behavioral profiles [20, 25] are behavioral abstractions of process models that have been widely applied in the context of behavioral comparison, similarity search, and compliance checking [22, 8, 20, 21, 23]. In a nutshell, a behavioral profile of a process model is a complete graph over the set of task labels of the model in which edges are labeled by binary behavioral relations, e.g., causality or conflict. Alternatively, a behavioral profile

Correspondence and offprint requests to: Artem Polyvyanyy, Queensland University of Technology, GPO Box 2434, Brisbane, QLD 4001, Australia. e-mail: artem.polyvyanyy@qut.edu.au;
* Since 2016, Abel Armas-Cervantes is with Queensland University of Technology.



(a) Net system S_1

	a	b	c	d	e
a	+	↦	↦	↦	↦
b	↤	+	⫴	↦	+
c	↤	⫴	+	↦	⫴
d	↤	↤	↤	+	↤
e	↤	+	⫴	↦	+

(b) Classic behavioral profile

Fig. 1. A net system (a) and its classic behavioral profile (b).

can be seen as a square matrix where rows and columns represent task labels and each cell is labeled by a binary behavioral relation.

For example, Fig. 1 shows a process model captured as a Petri net system along with the matrix-based representation of its behavioral profile defined over the three binary behavioral relations proposed in [20]: (i) *strict order* "↦" between two task labels denotes that one task label always occurs before the other task label in computations of the model, (ii) *exclusiveness* "+" signifies that two given task labels never occur together in computations of the model, and (iii) *interleaving* "⫴" indicates that a given pair of task labels is neither in the strict order, nor in the inverse strict order, nor in the exclusiveness relation (the task labels may occur in any order in computations of the model). In an extended version of behavioral profiles [24, 25], a fourth behavioral relation of *co-occurrence* "≫" is introduced, stressing the fact that the notion of behavioral profile is independent of a specific set of relations. Two task labels are co-occurring, if every computation of the model that contains the first label also contains the second label. The introduction of the co-occurrence relation in behavioral profiles allows capturing *optionality* and *causality* on task labels [25]. Hereinafter, behavioral profiles defined over the behavioral relations proposed in [20] and [25] are referred to as *classic* and *causal* behavioral profiles, respectively.

It has been suggested that the problem of behavioral comparison of process models can be reduced to comparison of their behavioral profiles [8]. Given that nodes of every behavioral profile have distinct task labels (no duplicates allowed), the comparison trivially reduces to matrix equality checking, provided that the rows and columns of both matrices follow the same order on task labels. This property gives a convenient basis for computing behavioral similarity between pairs of process models. However, as of today, the expressive power of behavioral profiles is not well understood [21]. Counter-examples have been put forward showing that two process models with identical behavioral profiles (defined over the three behavioral relations listed above) may be not trace equivalent [5], i.e., may define different sets of computations.

This paper studies the expressive power of behavioral profiles defined over different repertoires of behavioral relations. The paper shows that two systems captured as acyclic unlabeled workflow nets that have identical behavioral profiles over a repertoire of only two behavioral relations are configuration equivalent [18]. However, this result no longer holds for labeled systems, even for behavioral profiles defined over behavioral relations introduced in [20, 25] or a wider array of behavioral relations proposed in [11]. Having put into evidence the limitations of behavioral profiles for acyclic workflow nets under configuration equivalence, the paper then studies the expressive power of behavioral profiles over a weaker notion of equivalence namely *trace equivalence*, or *language equivalence*. The paper shows that behavioral profiles are "lossy" for the class of regular net systems, i.e., Petri net systems that recognize regular languages. In other words, two such systems may recognize different (regular) languages, yet have identical behavioral profiles. Importantly, this result holds regardless of over which finite repertoire of behavioral relations (and of which arity) these behavioral profiles are defined.

This paper is a revised and extended version of our previous work reported in [3]. The main additional result included in this paper is concerned with the comparison of the expressive powers of behavioral profiles and regular net systems. In addition, this paper proposes a new definition of a behavioral profile that conveniently abstracts from the repertoire of behavioral relations and their arity.

The rest of the paper is structured as follows. Section 2 states preliminary notions that are used in subsequent discussions. Section 3 proposes a new definition of a behavioral profile that generalizes all the existing similar definitions. Section 4 studies the expressive power of behavioral profiles for the class of

workflow net systems—a special class of Petri net systems that is widely used to encode business processes [16]; this section focuses on acyclic systems under the notion of *configuration equivalence*. Section 5 investigates the expressive power of behavioral profiles for a larger class of regular net systems under trace equivalence. Finally, Section 6 summarizes the contributions and outlines directions for future work.

2. Preliminaries

This section introduces Petri nets in general and two specific classes of nets, namely *workflow nets*, which is a class of nets that are traditionally used to model business processes, and *flow nets*, which is a class of nets that is used in this paper to analyze the expressive power of behavioral profiles. The section also introduces an alternative model of concurrency, called *flow event structure*, which is used in the subsequent discussions to establish a formal link between behavioral profiles and flow nets.

2.1. Petri nets

A Petri net is a mathematical formalism for the description of concurrent systems [9].

Definition 2.1 (Petri net).
A *(labeled) Petri net*, or a *net*, is a 5-tuple $N := (P, T, F, \Lambda, \lambda)$, where P is a finite set of *places*, T is a finite set of *transitions*, $F \subseteq (P \times T) \cup (T \times P)$ is the *flow relation*, Λ is a set of *labels* (P, T, and Λ are pairwise disjoint), and $\lambda : T \to \Lambda \cup \{\tau\}$ is a function that maps transitions of N to labels, where τ is a special label, $\tau \notin \Lambda$. ⌟

Places and transitions are conjointly referred to as *nodes* of the net. We write $\bullet y := \{x \in P \cup T \mid (x, y) \in F\}$ and $y\bullet := \{z \in P \cup T \mid (y, z) \in F\}$ to denote the *preset* and *postset* of a node $y \in P \cup T$, respectively.

Transitions of Petri nets are used to encode actions/tasks. It is often convenient to distinguish between *observable* and *silent* transitions to denote tasks that have a well-defined meaning in the problem domain and tasks that have no domain interpretation, respectively. If $\lambda(t) = \tau$, where $t \in T$, then t is said to be *silent*; otherwise t is said to be *observable*. In the special case when λ is injective and its image does not contain τ, i.e., all transitions are observable and 'wear' distinct labels, the net is referred to as *unlabeled*.

One can define the execution semantics of Petri nets in terms of markings.

Definition 2.2 (Marking).
A *marking* of a net $N := (P, T, F, \Lambda, \lambda)$ is a function $M : P \to \mathbb{N}_0$ that maps places of N to natural numbers. ⌟

For a place $p \in P$, we say that marking M puts $M(p)$ *tokens* at p. By abuse of notation, if a marking M puts one token at a place $p \in P$ and no tokens elsewhere then we denote the marking by $[p]$. A transition $t \in T$ is *enabled* in a marking M of N, denoted by $(N, M)[t\rangle$, 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 a marking M of N, then t can *occur*, which *leads* to a fresh marking M' of N, where $M'(p) = M(p) - 1$ if $p \in \bullet t \setminus t\bullet$, $M'(p) = M(p) + 1$ if $p \in t\bullet \setminus \bullet t$, and $M'(p) = M(p)$ otherwise. By $(N, M)[t\rangle(N, M')$, we denote the fact that an occurrence of transition t leads from marking M to marking M' of N.

A net system is a Petri net together with its initial marking.

Definition 2.3 (Net system).
A *net system* is a pair $S := (N, M)$, where N is a net and M is the *initial* marking of N. ⌟

By $\mathbb{S}$, we denote the universe of net systems. Net systems have a well-established graphical notation. In this notation, places and transitions are visualized as circles and rectangles, respectively. If a transition is silent, then the corresponding rectangle is drawn empty. If a transition $t \in T$ is observable, then the corresponding rectangle contains label $\lambda(t)$ within its boundaries. Every pair of nodes (x, y) in the flow relation is depicted as a directed arc that leads from x to y. Finally, the initial marking is visualized by drawing black dots inside places, i.e., for every place $p \in P$, $M(p)$ black dots (tokens) are placed inside the circle that represents p. For example, Fig. 1(a) shows a net system that can be formalized as a pair (N, M), $N := (P, T, F, \Lambda, \lambda)$, where $P := \{p_1, p_2, p_3, p_4, p_5, p_6\}$, $T := \{t_1, t_2, t_3, t_4, t_5\}$, $F := \{(p_1, t_1), (t_1, p_2), (t_1, p_3), (p_2, t_2), (p_2, t_5), (p_3, t_3), (t_2, p_4), (t_5, p_4), (t_3, p_5), (p_4, t_4), (p_5, t_4), (t_4, p_6)\}$, $\Lambda := \{\mathtt{a}, \mathtt{b}, \mathtt{c}, \mathtt{d}, \mathtt{e}\}$, $\lambda := \{(t_1, \mathtt{a}), (t_2, \mathtt{b}), (t_3, \mathtt{c}), (t_4, \mathtt{d}), (t_5, \mathtt{e})\}$, and $M := \{(p_1, 1), (p_2, 0), (p_3, 0), (p_4, 0), (p_5, 0), (p_6, 0)\}$. Note that the net in Fig. 1(a) can be referred to as unlabeled, as its transitions 'wear' distinct labels.

(a) WF-net system S_2 (b) WF-net system S_3

Fig. 2. WF-net systems.

Let A be a set of elements. Then, by $\sigma \coloneqq \langle a_1, a_2, \ldots, a_n \rangle \in A^*$, $n \in \mathbb{N}_0$, we denote a sequence of length n over A, where $a_i \in A$, $i \in [1..n]$. The empty sequence, i.e., the sequence without elements, is denoted by $\langle\rangle$.

Definition 2.4 (Occurrence sequence, execution).
Let $S \coloneqq (N, M_0)$, $N \coloneqq (P, T, F, \Lambda, \lambda)$, be a net system.

- A sequence of transitions $\sigma \coloneqq \langle t_1, t_2, \ldots, t_n \rangle \in T^*$ is an *occurrence sequence* in S iff σ is empty or there exists a sequence of markings $\langle M_0, M_1, \ldots, M_n \rangle$, such that for every position $i \in [1..n]$ in σ it holds that $(N, M_{i-1})[t_i\rangle(N, M_i)$; in the latter case we say that σ *leads* from M_0 to M_n and denote this fact by $(N, M_0)[\sigma\rangle(N, M_n)$.
- A marking M is *reachable* in S, denoted by $M \in [S\rangle$, iff $M = M_0$ or there exists an occurrence sequence σ in S that leads from M_0 to M.
- A marking M of N is *terminal* iff for every transition $t \in T$ it holds that t is not enabled in M.
- An occurrence sequence σ in S that leads to a terminal marking is called an *execution* of S. ⌟

For example, $\langle t_1, t_5 \rangle$ and $\langle t_1, t_3, t_2, t_4 \rangle$ are two occurrence sequences in net system S_1 from Fig. 1(a), whereas the latter sequence is also an execution of S_1. Given a net system S, by $\Omega(S)$ and $\Theta(S)$ we denote the set of all occurrence sequences in S and the set of all executions of S, respectively.

2.1.1. Workflow nets

Workflow nets, or WF-nets, is a class of Petri nets that is widely used in the context of business process modeling [15]. Every WF-net has a dedicated source place, a dedicated sink place, and every its transition is on a directed path from the source to the sink.

Definition 2.5 (WF-net, WF-net system).
A Petri net $N \coloneqq (P, T, F, \Lambda, \lambda)$ is a *workflow net*, or a *WF-net*, iff N has a dedicated *source* place $i \in P$, with $\bullet i = \varnothing$, N has a dedicated *sink* place $o \in P$, with $o\bullet = \varnothing$, and the *short-circuit* net $N^* \coloneqq (P, T \cup \{t^*\}, F \cup \{(o, t^*), (t^*, i)\})$ of N is strongly connected, where $t^* \notin T$ is a fresh transition. A *workflow system*, or a *WF-net system*, is a pair (N, M_i), where N is a WF-net with the source place i and $M_i = [i]$. ⌟

Commonly adopted criteria for correctness of WF-net systems are soundness and safeness [16]. A sound WF-net system guarantees that every execution ends with one token in the sink place and no tokens elsewhere.

Definition 2.6 (Liveness, boundedness, safeness, soundness).
Let $S \coloneqq (N, M)$, $N \coloneqq (P, T, F, \Lambda, \lambda)$, be a net system.

- S is *live* iff for every reachable marking $M' \in [S\rangle$ and for every transition $t \in T$ there exist a marking $M'' \in [(N, M')\rangle$ such that $(N, M'')[t\rangle$.
- A marking M' of N is *n-bounded*, $n \in \mathbb{N}_0$, iff for every place $p \in P$ it holds that $M'(p) \leq n$. S is said to be *bounded* iff there exists a number $n \in \mathbb{N}_0$, such that all reachable markings in S are n-bounded.
- S is *safe* if all its reachable markings are 1-*bounded*. Note that one can identify a 1-*safe* marking M' of N as the set of places $\{p \in P \mid M'(p) = 1\}$.

A WF-net system (N, M_i) is *sound* iff the net system (N^*, M_i), where N^* is the short-circuit net of N, is live and bounded. ⌟

Figs. 1(a), 2(a), and 2(b) show WF-net systems. For example, the system in Fig. 1(a) has one dedicated source place p_1, one dedicated sink place p_6, and every its node is on a directed path from p_1 to p_6. The systems in Figs. 1(a), 2(a) are safe and sound, whereas the system in Fig. 2(b) is 2-bounded and not sound.

2.1.2. Flow nets

Flow nets are a special class of Petri nets, which have the property that their partial order semantics can be given in terms of occurrence sequences [4]. Flow nets impose two semantic restrictions. Firstly, flow net systems are semantically acyclic. Intuitively, this means that a token cannot be put at a place which is in the preset of a transition that has already fired. Thus, every place cannot be marked more than once during an execution of a flow net system and all transitions in any occurrence sequence are distinct. For example, net system S_2 in Fig. 2(a) is not a flow net system since it violates the aforementioned restriction. Specifically, place p_4 is marked three times during execution $\langle t_1, t_2, t_3, t_4 \rangle$, as transitions t_2 and t_3 put a token back at p_4.

In the context of flow net systems, the notion of a causal dependency between transitions can be defined w.r.t. places. In particular, a transition t_j causally depends on a transition t_i if (i) there exists a place p between them, and (ii) whenever both t_i and t_j occur in an occurrence sequence, then t_i is the only transition that puts a token at p; p is said to be a *strong postcondition* of t_i. The notion of a causal dependency between transitions leads to the second semantic restriction of flow nets, which states that whenever there is a place between a pair of transitions, then there is also a strong postcondition between them. Fig. 2(b) shows an example of a net system that violates the second restriction since there is no strong postcondition, neither between t_2 and t_4, nor between t_3 and t_4. Hence, whenever both t_2 and t_3 occur, it is not possible to know which of them precedes the occurrence of transition t_4, thus the causal dependency between them cannot be determined. It is easy to check that net system S_1 in Fig. 1(a) is a flow net system, as it satisfies both stated requirements. Before presenting the definition of flow net systems, we formally define the notion of a strong postcondition to complement the above informal description.

Definition 2.7 (Strong postcondition, cf. [4]).
A place $p \in P$ of a net system $S := (N, M)$, $N := (P, T, F, \Lambda, \lambda)$, is a *strong postcondition* of a transition $t \in T$ iff $p \in t\bullet$, and for every occurrence sequence $\sigma := \langle t_1, \ldots, t_n \rangle \in \Omega(S)$, $n \in \mathbb{N}$, in which t occurs, i.e., $t = t_j$, for some $j \in [1..n]$, only t can mark p, i.e., $M(p) + \sum_{1 \leq j \leq n} |t_j\bullet \cap \{p\}| = 1$. ⌟

Finally, a flow net system is defined as follows.

Definition 2.8 (Flow net, flow net system, cf. [4]).
A net system $S := (N, M)$, $N := (P, T, F, \Lambda, \lambda)$, is a *flow net system* and N is a *flow net* iff for every occurrence sequence $\sigma := \langle t_1, \ldots, t_n \rangle \in \Omega(S)$, $n \in \mathbb{N}$, and for every $i, j \in [1..n]$, such that $i < j$, it holds that:
1. a place $p \in P$ is in the preset of at most one transition of σ, i.e., $\bullet t_i \cap \bullet t_j = \emptyset$, and
2. if $t_i\bullet \cap \bullet t_j \neq \emptyset$ then there exists $p \in t_i\bullet \cap \bullet t_j$, such that p is a strong postcondition of t_i. ⌟

An alternative way to define the execution semantics of a net system is using the notion of a *configuration*. The main difference between occurrence sequences and configurations is that the former are used to describe the interleaving semantics, whereas the latter capture the partial order semantics of the system. A configuration of a net system is a multiset of its transitions. A configuration induces the state during the execution of the system which is reached after occurrences of all the instances of transitions in the configuration; note that in general a transition can occur more than once in an execution of a net system. However, configurations of flow net systems are sets because of their semantically acyclic nature [4]. This fact is captured below.

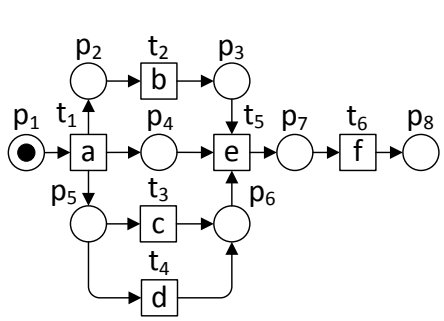
Definition 2.9 (Flow net configuration).
A *configuration* of a flow net system $S := (N, M)$, $N := (P, T, F, \Lambda, \lambda)$, is a set $C \subseteq T$ of transitions for which there exists an occurrence sequence $\sigma := \langle t_1, \ldots, t_n \rangle \in \Omega(S)$, $n \in \mathbb{N}_0$, $t_j \in T$, $j \in [1..n]$, that consists of the transitions in C, i.e., it holds that $C = \{t_1, \ldots, t_n\}$. ⌟

We will denote the set of all configurations of a flow net system S by $Conf(S)$.

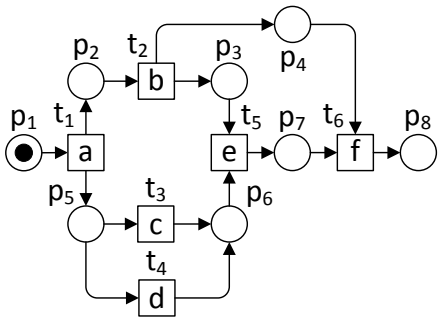
For example, Figs. 3(a) and 3(c) show a flow net system and its configurations ordered by set inclusion, respectively. Configurations can be used to define a well-known notion of behavioral equivalence in the spectrum of true concurrency, called *configuration equivalence* [18].[1] Intuitively, two flow net systems are configuration equivalent if they have the same configurations. Next, we give a formal form to this intuition.

Definition 2.10 (Configuration equivalence of flow net systems).
Let $S := (N, M)$, $N := (P, T, F, \Lambda, \lambda)$, and $S' := (N', M')$, $N' := (P', T', F', \Lambda', \lambda')$, be flow net systems. By $S \lesssim_{conf} S'$, we denote the fact that for every configuration C of S there exists a configuration C' of S' such

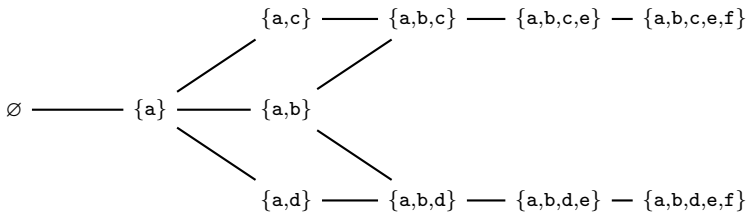
[1] The authors of [18] use the term *pomset-trace equivalence*



(a) Flow net system S_4



(b) Flow net system S_5



(c) Configurations of S_4 and S_5

Fig. 3. Flow net systems and their configurations ordered by set inclusion.

that there exists a bijection $\beta : C \to C'$ for which it holds that $\lambda(t) = \lambda'(\beta(t))$, $t \in C$. Flow net systems S and S' are *configuration equivalent*, denoted by $S \approx_{conf} S'$, iff $S \lesssim_{conf} S'$ and $S' \lesssim_{conf} S$. ⌟

Note that flow net systems S_4 and S_5 shown in Fig. 3(a) and Fig. 3(b), respectively, are configuration equivalent, as Fig. 3(c) also shows all the configurations of S_5.

2.2. Flow Event Structures

A Flow Event Structure (FES) describes the behavior of a concurrent system, e.g., a net system, by means of events (occurrences of tasks) and two binary behavioral relations on events.

Definition 2.11 (Labeled flow event structure).
A *(labeled) flow event structure* (FES) is a 5-tuple $\mathbb{F} := (E, \#, \lessdot, \Lambda, \lambda)$, where E is a set of *events*, $\# \subseteq E \times E$ is the *conflict* relation, $\lessdot \subseteq E \times E$ is the *flow* relation ($\#$ and $\lessdot$ are symmetric and irreflexive, respectively), Λ is a set of *labels*, and $\lambda : E \to \Lambda \cup \{\tau\}$ is a function that maps events of $\mathbb{F}$ to labels, where τ is a special label, $\tau \notin \Lambda$. ⌟

Again, if $\lambda(e) = \tau$, where $e \in E$, then e is said to be *silent*; otherwise e is *observable*.

Intuitively, the flow relation specifies possible immediate precedence of events, i.e., if two events e and e' are in the flow relation ($e \lessdot e'$), then event e can potentially occur before e'. The conflict relation represents mutual exclusion of events, i.e., two events e and e' in the conflict relation ($e \# e'$) cannot occur together.

Similar to configurations of flow net systems, one can talk about configurations of FESs.

Definition 2.12 (FES configuration).
A *configuration* of a FES $\mathbb{F} := (E, \#, \lessdot, \Lambda, \lambda)$ is a set of events $C \subseteq E$ that is conflict free, i.e., $\forall\, e, e' \in C$: $\neg(e \# e')$, has no flow cycles, i.e., the transitive closure of the restriction of $\lessdot$ to C is a partial order, and for all events $e' \in C$ and $e \notin C$ such that $e \lessdot e'$, it holds that there is an event $e'' \in C$ such that $e \# e''$ and $e'' \lessdot e'$. ⌟

We will denote the set of all configurations of a FES $\mathbb{F}$ by $Conf(\mathbb{F})$.

The notion of configuration equivalence, presented above for flow net systems, can be lifted to flow event structures as follows.

Definition 2.13 (Configuration equivalence of flow event structures).
Let $\mathbb{F} := (E, \#, \lessdot, \Lambda, \lambda)$ and $\mathbb{F}' := (E', \#', \lessdot', \Lambda', \lambda')$ be flow event structures. By $\mathbb{F} \lesssim_{conf} \mathbb{F}'$, we denote the

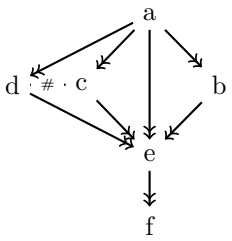


Fig. 4. FES $\mathcal{E}(S_4)$ of the WF-flow net system in Fig. 3(a)

fact that for every configuration C of $\mathbb{F}$ there exists a configuration C' of $\mathbb{F}'$ such that there exists a bijection $\beta : C \to C'$ for which it holds that $\lambda(e) = \lambda'(\beta(e))$, $e \in C$. Flow event structures $\mathbb{F}$ and $\mathbb{F}'$ are *configuration equivalent*, denoted by $\mathbb{F} \approx_{conf} \mathbb{F}'$, iff $\mathbb{F} \lesssim_{conf} \mathbb{F}'$ and $\mathbb{F}' \lesssim_{conf} \mathbb{F}$. ⌟

Fig. 4 shows the FES representing the same set of configurations as the net system S_4 in Fig. 3(a); the configurations are shown in Fig. 3(c). In this graphical notation, the letters represent events, the double-headed arrows denote the flow relation and the annotated dotted lines denote the conflict relation. Observe that for every transition in S_4 there is an event with the same label in the FES, and for every place between a pair of transitions there is a flow relation between the corresponding events.

3. Behavioral Profiles

This section contributes to the theory of behavioral profiles [20, 25]. A behavioral profile gives an alternative (declarative) characterization of a dynamic system. A behavioral profile of a system is a collection of behavioral constraints (relations) on its action/task labels that the system must obey in its computations. In the sequel, we propose a definition of a behavioral profile that can be seen as a generalization of all the existing (to the best of our knowledge) similar definitions. We also demonstrate how the notion of a classic behavioral profile [20] can be captured using the new formalism.

By $\mathbb{A}$, we denote the universe of labels. Let $S := (N, M)$, $N := (P, T, F, \Lambda, \lambda)$, be a net system. Then, $labels(S) := image(\lambda)$, where $image(\lambda)$ is the image of the entire domain T of λ.

Definition 3.1 (Behavioral predicates and relations).
An *(n-ary) behavioral predicate*, $n \in \mathbb{N}$, is a set of pairs $\pi \subseteq \{(S, R) \in \mathbb{S} \times (\mathbb{A}^n) \mid R \in (labels(S))^n\}$. An *(n-ary) behavioral relation*, $n \in \mathbb{N}$, of a net system $S \in \mathbb{S}$ induced by an n-ary behavioral predicate π, denoted by $\mathcal{R}[S, \pi]$, is the set $\{R \in \mathbb{A}^n \mid (S, R) \in \pi\}$. ⌟

An example of a behavioral predicate is a *weak order* predicate proposed in [20].

Definition 3.2 (Weak order [20]).
The *weak order* behavioral predicate is the set of pairs $\prec := \{(S, (\alpha, \beta)) \in \mathbb{S} \times (\mathbb{A}^2) \mid \alpha, \beta \in labels(S) \wedge (\exists\, \sigma \in \Omega(S)\ \exists\, i, j \in [1..|\sigma|] : i < j \wedge \lambda(\sigma_i) = \alpha \wedge \lambda(\sigma_j) = \beta)\}$. ⌟

Clearly, $\prec$ is a behavioral predicate. A *repertoire* of behavioral predicates Π is a finite non-empty set of behavioral predicates. Finally, a behavioral profile of a system induced by a repertoire of behavioral predicates is defined by behavioral relations of the system induced by these predicates.

Definition 3.3 (Behavioral profile).
A *behavioral profile* of a net system $S \in \mathbb{S}$ induced by a repertoire of behavioral predicates Π, denoted by $\mathbb{R}[S, \Pi]$, is a function that maps every behavioral predicate $\pi \in \Pi$ to its behavioral relation of S induced by π, i.e., $\mathbb{R}[S, \Pi] := \{(\pi, \mathcal{R}[S, \pi]) \mid \pi \in \Pi\}$. ⌟

In this light, the classic behavioral profile can be defined as follows.

Definition 3.4 (Classic behavioral profile [20]).
The *classic behavioral profile* of a net system $S \in \mathbb{S}$, denoted by $\mathbb{BP}^c(S)$, is the behavioral profile of S induced by the repertoire of behavioral predicates $\Pi := \{\nleftrightarrow, \mapsto, +, |||\}$, i.e., $\mathbb{BP}^c(S) := \mathbb{R}[S, \Pi]$, where:

- $\nleftrightarrow := \{(S, (\alpha, \beta)) \in \mathbb{S} \times (\mathbb{A}^2) \mid (S, (\alpha, \beta)) \in \prec \wedge (S, (\beta, \alpha)) \notin \prec\}$,

- $\mapsto := \{(S,(\alpha,\beta)) \in \mathbb{S}\times(\mathbb{A}^2) \mid (S,(\alpha,\beta)) \notin \prec \wedge (S,(\beta,\alpha)) \in \prec\}$,
- $+ := \{(S,(\alpha,\beta)) \in \mathbb{S}\times(\mathbb{A}^2) \mid (S,(\alpha,\beta)) \notin \prec \wedge (S,(\beta,\alpha)) \notin \prec\}$, and
- $\|\| := \{(S,(\alpha,\beta)) \in \mathbb{S}\times(\mathbb{A}^2) \mid (S,(\alpha,\beta)) \in \prec \wedge (S,(\beta,\alpha)) \in \prec\}$. ⌟

For example, the classic behavioral profile of the Petri net system S in Fig. 1(a) is given by $\mathbb{BP}^c(S) := \{(\nmapsto, \{(\mathtt{a},\mathtt{b}), (\mathtt{c},\mathtt{a}), (\mathtt{d},\mathtt{a}), (\mathtt{d},\mathtt{b}), (\mathtt{d},\mathtt{c}), (\mathtt{d},\mathtt{e}), (\mathtt{e},\mathtt{a})\}), (\mapsto, \{(\mathtt{a},\mathtt{b}), (\mathtt{a},\mathtt{c}), (\mathtt{a},\mathtt{d}), (\mathtt{a},\mathtt{e}), (\mathtt{b},\mathtt{d}), (\mathtt{c},\mathtt{d}), (\mathtt{e},\mathtt{d})\}), (+, \{(\mathtt{a},\mathtt{a}), (\mathtt{b},\mathtt{b}), (\mathtt{b},\mathtt{e}), (\mathtt{c},\mathtt{c}), (\mathtt{d},\mathtt{d}), (\mathtt{e},\mathtt{b}), (\mathtt{e},\mathtt{e})\}), (\|\|, \{(\mathtt{b},\mathtt{c}), (\mathtt{c},\mathtt{b}), (\mathtt{c},\mathtt{e}), (\mathtt{e},\mathtt{c})\})\}$.

Definition 3.5 (Basic repertoire of behavioral predicates).
A repertoire of n-ary behavioral predicates Π, $n \in \mathbb{N}$, is *basic* iff for every net system $S \in \mathbb{S}$ it holds that:
- for every two distinct behavioral predicates $\pi_1, \pi_2 \in \Pi$, $\mathcal{R}[S,\pi_1]$ and $\mathcal{R}[S,\pi_2]$ are disjoint, and
- $\bigcup_{\pi\in\Pi} \mathcal{R}[S,\pi] = (\mathit{labels}(S))^n$. ⌟

The repertoire of binary behavioral predicates $\Pi := \{\nmapsto, \mapsto, +, \|\|\}$, refer to Definition 3.4, is basic [20].

As pointed out in the Introduction, a behavioral profile of a net system induced by a basic repertoire of behavioral predicates can be visualized as a square matrix where rows and columns represent task labels and each cell is labeled by the respective behavioral relation. Recall that Fig. 1(b) encodes the classic behavioral profile of the net system in Fig. 1(a).

4. Behavioral Profiles and Equivalence Preservation

The section starts with the definition of the *equivalence preservation* property on repertoires of behavioral predicates. This property is used in Section 4.2 to study the expressive power of classic behavioral profiles w.r.t. configuration equivalence of WF-flow net systems. A *WF-flow net* is a flow net that can be used to define a sound WF-net system, e.g., the two WF-nets in Fig. 3 are WF-flow nets. Finally, Section 4.3 collects and discusses limitations of behavioral profiles.

4.1. Equivalence preserving repertoires of behavioral predicates

It is well-accepted that the expressive power of a modeling language for capturing dynamic systems should be defined with respect to a notion of (behavioral) equivalence. In the context of dynamic systems, there are several such notions, which are broadly classified into two categories: equivalences that are based on the interleaving semantics and those based on the true concurrency semantics, cf. [17, 18, 19, 12] for details.

Given that behavioral profiles do not have execution semantics per se, when studying their expressive power, one cannot directly reuse existing equivalences. To this end, we introduce the notion of *equivalence preservation*, which is a property on a repertoire of behavioral predicates.

Definition 4.1 (Equivalence preserving repertoire of behavioral predicates).
Let $\mathbb{S}' \subseteq \mathbb{S}$ be a class of net systems and let $\approx$ be an equivalence relation on $\mathbb{S}'$. A repertoire of behavioral predicates Π *preserves* $\approx$ on $\mathbb{S}'$ iff for all $S, S' \in \mathbb{S}'$ it holds that:

$$\mathbb{R}[S,\Pi] = \mathbb{R}[S',\Pi] \;\Leftrightarrow\; S \approx S'.$$

⌟

Intuitively, a repertoire of behavioral predicates preserves an equivalence on a class of net systems iff equivalent systems imply equal behavioral profiles of these systems induced by the behavioral predicates, and vice versa.

4.2. Behavioral profiles and WF-flow nets

In [4], the author shows that FESs correspond to the family of flow net systems. Specifically, it is always possible to compute a FES of a given flow net system, where configurations of the FES can be derived from occurrence sequences in the system. Next, we show how to construct a FES of a WF-flow net system.

Definition 4.2 (FES of a WF-flow net system).
The *FES of a WF-flow net system* $S := (N,M) \in \mathbb{S}$, $N := (P,T,F,\Lambda,\lambda)$, denoted by $\mathcal{E}(S)$, is the FES $(T,\#,\lessdot,\Lambda,\lambda)$, where for every $t,t' \in T$ it holds that:
- $t\#t' \Leftrightarrow_{def} \forall\, C \in \mathit{Conf}(S) : \{t,t'\} \nsubseteq C$, and
- $t \lessdot t' \Leftrightarrow_{def} \neg(t\#t') \wedge t\bullet \cap \bullet t' \neq \varnothing$. ⌟

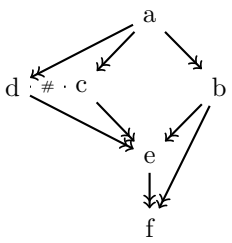


Fig. 5. FES $\mathcal{E}(S_5)$ of the WF-flow net system in Fig. 3(b).

Furthermore, occurrence sequences in a flow net system induce configurations of the corresponding FES, and vice versa. The following proposition restates one result from [4] for the set of unlabeled WF-flow net systems.

Proposition 4.3 (Configurations of flow net systems and FESs, cf. [4]).
If $\mathbb{F} := \mathcal{E}(S)$ is a FES of an unlabeled WF-flow net system S, then it holds that $\mathit{Conf}(S) = \mathit{Conf}(\mathbb{F})$. ⌟

A direct consequence of Proposition 4.3 is that two configuration equivalent WF-flow net systems induce configuration equivalent FESs. This is captured in the following corollary.

Corollary 4.4 (FES, WF-flow nets and configuration equivalence).
If $\mathbb{F} := \mathcal{E}(S)$ and $\mathbb{F}' := \mathcal{E}(S')$, where S and S' are unlabeled WF-flow net systems, then it holds that:

$$S \approx_{conf} S' \;\Leftrightarrow\; \mathbb{F} \approx_{conf} \mathbb{F}'.$$

⌟

The relation between the two formalisms, i.e., FESs and WF-flow net systems, demonstrates that the behavior of a WF-flow net system can be encoded using two binary relations. These relations can be employed to induce a new type of behavioral profiles.

Definition 4.5 (FES behavioral profile).
Let $\mathbb{S}^* \subset \mathbb{S}$ be the set of all unlabeled WF-flow net systems and let $S \in \mathbb{S}^*$, where $S := (N, M)$ and $N := (P, T, F, \Lambda, \lambda)$. The *FES behavioral profile* of S, denoted by $\mathbb{BP}^{fes}(S)$, is the behavioral profile induced by the repertoire of behavioral predicates $\Pi := \{\#, \lessdot\}$, i.e., $\mathbb{BP}^{fes}(S) := \mathbb{R}[S, \Pi]$, where:

- $\# := \{(S, (\alpha, \beta)) \in \mathbb{S}^* \times (\mathbb{A}^2) \mid \exists\, t_1, t_2 \in T\ \forall C \in \mathit{Conf}(S) : (\{t_1, t_2\} \nsubseteq C \wedge \lambda(t_1) = \alpha \wedge \lambda(t_2) = \beta)\}$,
- $\lessdot := \{(S, (\alpha, \beta)) \in \mathbb{S}^* \times (\mathbb{A}^2) \mid \exists\, t_1, t_2 \in T\ \exists C \in \mathit{Conf}(S) : (\{t_1, t_2\} \subseteq C \wedge t_1\bullet \cap \bullet t_2 \neq \emptyset \wedge \lambda(t_1) = \alpha \wedge \lambda(t_2) = \beta)\}$. ⌟

The notion of a configuration of a FES can be applied directly over the $\mathbb{BP}^{fes}$, such that any conclusion (w.r.t. behavior) derived from the $\mathbb{BP}^{fes}(S)$ holds in the corresponding unlabeled WF-flow net system S. Note that $\mathbb{BP}^{fes}$ is not a basic repertoire of behavioral predicates since there can be pairs of labels (events in the FES) neither in flow nor in conflict relation, i.e., $\bigcup_{\pi \in \{\#, \lessdot\}} \mathcal{R}[S, \pi] \neq (\mathit{labels}(S))^2$. Indeed, the flow relations of two configuration equivalent unlabeled WF-flow net systems can be different. This fact will be made obvious after the following proposition, which shows that all the places, except for the source and sink place, in a WF-flow net system are strong postconditions of some transitions and thus induce flow relations.

Proposition 4.6 (Strong postconditions and WF-flow net systems).
Let $S := (N, M)$, $N := (P, T, F, \Lambda, \lambda)$, be a WF-flow net system and let $\sigma := \langle t_1, \ldots, t_n \rangle \in \Theta(S)$, $n \in \mathbb{N}$, be an execution of S. A place $p \in t_i\bullet \cap \bullet t_j$ is a strong postcondition of t_i, where $1 \leq i, j \leq n$. ⌟

The proof of Proposition 4.6 follows immediately from Definition 2.8.

Consider unlabeled WF-flow net systems S_4 and S_5 in Fig. 3(a) and Fig. 3(b), respectively. Fig. 4 and Fig. 5 show FESs of S_4 and S_5, respectively. The two FESs are clearly not isomorphic since $\mathcal{E}(S_4)$ has a flow relation between a and e that is not present in $\mathcal{E}(S_5)$, while $\mathcal{E}(S_5)$ has a flow relation between b and f that is not present in $\mathcal{E}(S_4)$. Yet, they encode the same set of configurations (those shown in Fig. 3(c)). The differences in the flow relations between the FESs of S_4 and S_5 are translated into differences in the $\lessdot$ relations of $\mathbb{BP}^{fes}(S_4)$ and $\mathbb{BP}^{fes}(S_5)$. Hence, the repertoire of behavioral predicates $\{\lessdot, \#\}$ does not preserve configuration equivalence for the class of unlabeled WF-flow net systems.

We now turn our attention to the behavioral predicates of classic behavioral profiles and show that preserve configuration equivalence on unlabeled WF-flow net systems. We start by defining the *structural strict order* relation between a pair of transitions in an unlabeled WF-flow net system.

Definition 4.7 (Structural strict order).
Let $S := (N, M)$, $N := (P, T, F, \Lambda, \lambda)$, be a net system. The *structural strict order* behavioral predicate is the set of pairs $\twoheadrightarrow := \{(S, (\alpha, \beta)) \in \mapsto \mid \exists\, \sigma \in \Omega(S)\ \exists\, i, j \in [1..|\sigma|] : \lambda(\sigma_i) = \alpha \wedge \lambda(\sigma_j) = \beta \wedge \sigma_i \bullet \cap \bullet \sigma_j \neq \varnothing\}$, where $\mapsto$ is defined in Definition 3.4. ⌟

Intuitively, two task labels are in *structural strict order* relation if the transitions 'wearing' these labels have a place between them and one occurs before the other in all the occurrence sequences.

Definition 4.8 (Classic FES behavioral profile).
The *classic FES behavioral profile* of a net system $S \in \mathbb{S}$, denoted by $\mathbb{BP}^{c}_{fes}(S)$, is the behavioral profile of S induced by the repertoire of behavioral predicates $\Pi := \{\twoheadrightarrow, \oplus\}$, $\oplus := \{(S, (\alpha, \beta)) \in + \mid \alpha, \beta \in \mathit{labels}(S) \wedge \alpha \neq \beta\}$, where $\twoheadrightarrow$ and $+$ are defined in Definition 4.7 and Definition 3.4, respectively, i.e., $\mathbb{BP}^{c}_{fes}(S) := \mathbb{R}[S, \Pi]$. ⌟

The next proposition shows that given an unlabeled WF-flow net system S, it holds that $\mathbb{BP}^{fes}(S) = \mathbb{BP}^{c}_{fes}(S)$, if we assume the correspondence between $\oplus$ and $\#$ and between $\twoheadrightarrow$ and $\lessdot$.

Proposition 4.9 ($\mathbb{BP}^{fes}$ and $\mathbb{BP}^{c}_{fes}$).
Let S be an unlabeled WF-flow net system. Then, it holds that $\mathbb{BP}^{fes}(S) = \mathbb{BP}^{c}_{fes}(S)$, where $\mathbb{BP}^{fes}(S) := \mathbb{R}[S, \{\#, \lessdot\}]$ and $\mathbb{BP}^{c}_{fes}(S) := \mathbb{R}[S, \{\twoheadrightarrow, \oplus\}]$, i.e., for any two labels $\alpha, \beta \in \mathit{labels}(S)$ it holds that:
- $(S, (\alpha, \beta)) \in \#\ \Leftrightarrow\ (S, (\alpha, \beta)) \in \oplus$, and
- $(S, (\alpha, \beta)) \in \lessdot\ \Leftrightarrow\ (S, (\alpha, \beta)) \in \twoheadrightarrow$. ⌟

Proof. Without loss of generality, and since S is unlabeled, assume $t, t' \in T$ are the only transitions with labels α and β, respectively, i.e., $\lambda(t) = \alpha$ and $\lambda(t') = \beta$. Consider the following cases.

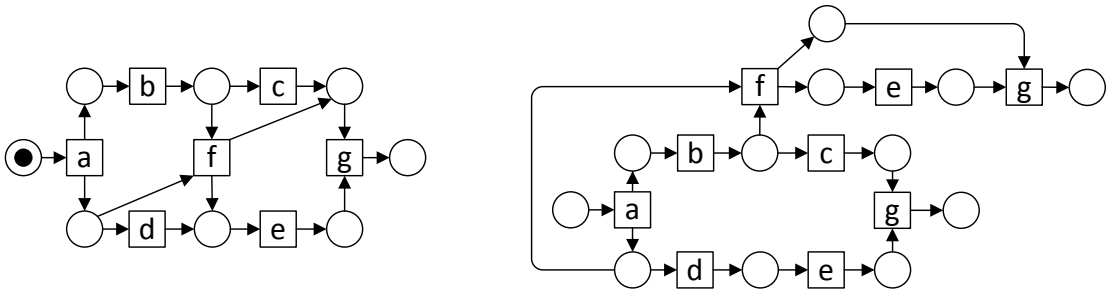
1. $(S, (\alpha, \beta)) \in \oplus\ \Leftrightarrow\ (S, (\alpha, \beta)) \in \#$.
   ($\Rightarrow$) By the definition of $\oplus$, we know that $\alpha \neq \beta$, which implies $t \neq t'$, and $(S, (\alpha, \beta)) \in +$. Additionally, by the definition of $+$ (Def. 3.4) in $\mathbb{BP}^{c}$, we have that $(S, (\alpha, \beta)) \notin \prec \wedge (S, (\beta, \alpha)) \notin \prec$. Given that α and β are not in weak order $\prec$, there is no sequence where both t and t' occur, i.e., $\nexists\, \sigma \in \Omega(S)\ \exists\, i, j \in [1..|\sigma|] : i < j \wedge ((\sigma_i = t \wedge \sigma_j = t') \vee (\sigma_i = t' \wedge \sigma_j = t)))$, see Definition 3.2. Therefore, there is no configuration C in the WF-flow net system (Def.2.9) with transitions $t_i, t_j \in C : t_i = t \wedge t_j = t'$, and so $t \# t'$, which leads to the desired results: $(S, (\lambda(t), \lambda(t'))) = (S, (\alpha, \beta)) \in \#$.
   ($\Leftarrow$) The inverse case clearly follows from Definition 4.2. I.e., $t, t' \in T : t \# t' \Leftrightarrow \forall\, C \in \mathit{Conf}(S) : \{t, t'\} \nsubseteq C$, thus there is no $\sigma \in \Omega(S) : \exists\, i, j \in [1..|\sigma|] : i < j \wedge ((\sigma_i = t \wedge \sigma_j = t') \vee (\sigma_i = t' \wedge \sigma_j = t))$. Hence, $(S, (\lambda(t), \lambda(t'))) = (S, (\alpha, \beta)) \in +$. Finally, given that $t \neq t'$, it also holds that $(S, (\lambda(t), \lambda(t'))) = (S, (\alpha, \beta)) \in \oplus$ as desired.
2. $(S, (\alpha, \beta)) \in \twoheadrightarrow\ \Leftrightarrow\ (S, (\alpha, \beta)) \in \lessdot$.
   ($\Rightarrow$) According to the definition of structural strict order $\twoheadrightarrow$, Definition 4.7, $\exists\, \sigma \in \Omega(S)\ \exists\, i, j \in [1..|\sigma|] : \sigma_i = t \wedge \sigma_j = t' \wedge t \bullet \cap \bullet t' \neq \varnothing$. Given that $t \bullet \cap \bullet t' \neq \varnothing$, by Proposition 4.6, $\exists p \in t \bullet \cap \bullet t'$ which is a strong postcondition of t. Thus, by Definition 4.2, we have that $t \lessdot t'$ in $\mathcal{E}(S)$ and $(S, (\lambda(t), \lambda(t'))) = (S, (\alpha, \beta)) \in \lessdot$ in $\mathbb{BP}^{fes}(S)$, as required.
   ($\Leftarrow$) Suppose $(S, (\alpha, \beta)) \in \lessdot$ in $\mathbb{BP}^{fes}(S)$, but $(S, (\alpha, \beta)) \notin \twoheadrightarrow$. By Definition 4.2, given that $(S, (\alpha, \beta)) \in \lessdot$ then $t \bullet \cap \bullet t' \neq \varnothing$ and $(S, (\alpha, \beta)) \notin \#$. Note that if $(S, (\alpha, \beta)) \notin \twoheadrightarrow$ then $(S, (\alpha, \beta)) \notin \mapsto$, see Definition 4.7, and since $(S, (\alpha, \beta)) \notin \#$ then $(S, (\alpha, \beta)) \notin \oplus$ – see previous case. Furthermore, given that the classic behavioral profile is basic, it holds that $(S, (\beta, \alpha)) \in \prec$, such that $(S, (\alpha, \beta)) \in \nmapsto$ or $(S, (\alpha, \beta)) \in \|\!|$. Hence, there is at least an occurrence sequence where t' occurs before t, i.e., $\sigma \in \Omega(S) \exists i, j \in [1 \ldots |\sigma|] : j < i \wedge \sigma_j = t' \wedge \sigma_i = t$. Moreover, there is a strong postcondition p of t (see Proposition 4.6) between t and t' since $(S, (\alpha, \beta)) \in \lessdot$. However, $p \in \bullet t'$ and p can be only marked by t, thus if $(S, (\beta, \alpha)) \in \prec$ then $\exists k \in [1 \ldots |\sigma|] : k < j\ \wedge\ \sigma_k = t$. The last contradicts the fact that S is a WF-flow net system, because σ would have 2 occurrences of t and, by Definition 2.7, p cannot be marked more than once by the transitions of any occurrence sequence $\sigma \in \Omega(S)$. Therefore, if $(S, (\alpha, \beta)) \in \lessdot$ then $(S, (\alpha, \beta)) \in \twoheadrightarrow$ and it concludes the proof. ∎

Thus, one can derive the configurations of S from $\mathbb{BP}^{c}_{fes}(S)$. In what follows, we prove that $\{\nmapsto, \mapsto, +, \|\!|\}$ preserves configuration equivalence on the class of unlabeled WF-flow net systems.

Proposition 4.10 (Configuration equivalence and equality of $\mathbb{BP}^{c}$).
Let $S := (N, M)$, $N := (P, T, F, \Lambda, \lambda)$, and $S' := (N', M')$, $N' := (P', T', F', \Lambda, \lambda')$, be two unlabeled WF-flow net systems such that there exists a bijection $\gamma : T \to T'$, $\lambda(t) = \lambda'(\gamma(t))$, $t \in T$. Then, it holds that:

$$\mathbb{BP}^{c}(S) = \mathbb{BP}^{c}(S')\ \Leftrightarrow\ S \approx_{conf} S'.$$

⌟



(a) Net system S_6 (b) Branching process

Fig. 6. A net system (a) and its branching process (b).

Proof. ($\Rightarrow$) Firstly, let us show that if $\mathbb{BP}^c(S) = \mathbb{BP}^c(S')$ then $S \approx_{conf} S'$.

Suppose that $\mathbb{BP}^c(S) = \mathbb{BP}^c(S')$, but $\neg(S \approx_{conf} S')$. By Corollary 4.4, we have $\neg(\mathcal{E}(S) \approx_{conf} \mathcal{E}(S'))$. Furthermore, by Proposition 4.9, for any pair of transitions $t, t' \in T$ it holds: $(S, (\lambda(t), \lambda(t'))) \in \oplus \Leftrightarrow t \# t'$ and $(S, (\lambda(t), \lambda(t'))) \in \nrightarrow \Leftrightarrow t \lessdot t'$, and analogously for any pair of transitions in T'.

Assume a configuration $C \subseteq T$ in $\mathcal{E}(S)$ and its mapping $C' = \{\gamma(t') \mid t' \in C\}$ in $\mathcal{E}(S')$, such that C' is not a configuration. By Definition 2.12, the configuration C of a FES (i) is conflict free, (ii) for all $t' \in C$ and $t \notin C$, s.t., $t \lessdot t'$ there exists an $t'' \in C$ s.t. $t \# t'' \lessdot t'$, and (iii) has no flow cycles.

Therefore, we must consider the following cases:

(i) Conflict freeness. Since C is a configuration in $\mathcal{E}(S)$, then for any $t, t' \in C$ it holds $\neg(t \# t')$ and, in consequence, $(S, (\lambda(t), \lambda(t'))) \notin +$ by Proposition 4.9(1). Then, by the assumption on the equality of the $\mathbb{BP}^c$'s, $\exists t_1, t_1' \in C' : \gamma(t) = t_1 \wedge \gamma(t') = t_1'$, such that $(S, (\lambda(t_1), \lambda(t_1'))) \notin +$ and thus $\neg(t_1 \# t_1')$. So, C' is also conflict free if C is conflict free, and for every pair of $t_1, t_1' \in C'$ then $(S, (\lambda(t), \lambda(t'))) \in |||$, $(S, (\lambda(t), \lambda(t'))) \in \nleftarrow$ or $(S, (\lambda(t), \lambda(t'))) \in \nrightarrow$.

(ii) For any $t_1'' \in C'$ and $t_1 \notin C'$, s.t., $t_1 \lessdot t_1''$, there exists an $t_1' \in C' : t_1 \# t_1' \lessdot t_1''$. Suppose that there is an event $t_1 \notin C'$, such that $\exists t_1'' \in C' : t_1 \lessdot t_1''$ and $\forall t_1' \in C' : \neg(t_1 \# t_1')$. Given that $t_1 \lessdot t_1''$, then $(S, (\lambda(t_1), \lambda(t_1''))) \in \nrightarrow$ (in fact, $(S, (\lambda(t_1), \lambda(t_1''))) \in \nrightarrow$), and since $\neg(t_1 \# t_1')$ then $(S, (\lambda(t_1), \lambda(t_1'))) \notin +$ for any $t_1' \in C'$ by Proposition 4.9. Hence, by the equality of $\mathbb{BP}^c$'s, $\exists t \in T, t'' \in C : t \notin C \wedge \gamma(t) = t_1 \wedge \gamma(t'') = t_1'' \wedge (S, (\lambda(t), \lambda(t''))) \in \nrightarrow$, and for any $t' \in C$ it holds $(S, (\lambda(t), \lambda(t'))) \notin +$. However, the last contradicts the fact that C is a configuration in $\mathcal{E}(S)$, because t would necessarily be in C. Henceforth, condition 2 also holds for C'.

(iii) Free of flow cycles. The only case remaining, so that C' is not a configuration in $\mathcal{E}(S')$, is when C' contains cycles, i.e., $\lessdot_{C'}^*$ is not a partial order. This case simply cannot happen because WF-flow nets are acyclic and any occurrence sequence contains at most one occurrence of each activity.

Therefore, if C is a configuration in $\mathcal{E}(S)$, then C' must also be a configuration in $\mathcal{E}(S')$.

($\Leftarrow$) The opposite case, $S \approx_{conf} S' \Rightarrow \mathbb{BP}^c(S) = \mathbb{BP}^c(S')$, follows directly from the construction of the $\mathbb{BP}^c$, see Definition 3.4. ∎

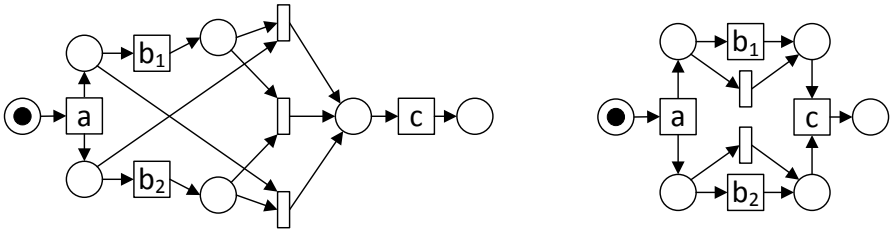
Armed with the above, one can conclude that $\{\nleftarrow, \nrightarrow, +, |||\}$ is equivalence preserving on the class of unlabeled WF-flow net systems. This fact is captured in the following Corollary.

Corollary 4.11 (Classic behavioral profiles and equivalence preservation).
The repertoire of behavioral predicates $\{\nleftarrow, \nrightarrow, +, |||\}$, cf. Definition 3.4, preserves configuration equivalence, cf. Definition 2.10, on the set of all unlabeled WF-flow net systems. ⌟

4.3. Discussion on the limitations of behavioral profiles

Fig. 6(a) shows net system S_6 that demonstrates one limitation of classic behavioral profiles. Note that for this net system, task labels `b` and `e` are in the interleaving relation, i.e., it holds that `b` $|||$ `e`. This, however, does not capture the fact that in some configurations `b` always precedes `e`. In particular, in all the configurations of S_6 that contain transitions labeled with `b`, `e`, and `f`, it is always the case that `b` precedes `e`.

A solution to disambiguate this situation can be to reason not in terms of task labels, but in terms of



(a) Flow net system S_7 (b) Flow net system S_8

Fig. 7. Two flow net systems that have equal causal behavioral profiles.

their occurrences. In this regard, alternative representations of net systems, e.g., by means of branching processes [6], can result useful. However, the price to pay is that a branching process can contain several instances of a single task that must be treated as different label instances in corresponding behavioral profiles. Hence, representation of a behavioral profile as a matrix of size $|\Lambda|^2$ is no longer guaranteed. For example, Fig. 6(b) shows the maximal branching process of net system S_6 from Fig. 6(a). Note that this branching process contains two occurrences of label `e` and two occurrences of label `g`.

Another approach to cope with lack of expressive power of classic behavioral profiles is to rely on other repertoires of behavioral predicates, as proposed in [26]. This latter work studies causalities with different look-aheads. The authors claim that the use of more behavioral predicates improves expressiveness of the induced behavioral profiles. They show that 1-look-ahead causalities induce trace equivalence for a restricted family of Petri nets. However, as it is shown below, repertoires of behavioral predicates used by causal behavioral profile [25] and the 4C spectrum [11], which contain more behavioral predicates than the repertoire used by classic behavioral profiles, do not preserve trace equivalence, even on acyclic net systems.

Proponents of classic behavioral profiles seek to provide a representation that only considers the distinct labels of the observable tasks yet is equivalence preserving. When it comes to representing net systems, the common approach is to ignore the silent transitions and to consider all the transitions with the same label as the same action. This design choice, however, comes with a loss of accuracy. Acknowledging the limitations of classic behavioral profiles, several extensions have been proposed, which incorporate other repertoires of behavioral predicates. For example, *causal behavioral profiles* [25], in addition to the behavioral predicates of classic behavioral profiles, include one additional behavioral predicate.

Definition 4.12 (Co-occurrence relation, cf. [25]).
The *co-occurrence* behavioral predicate is the set of pairs $\gg := \{(((P,T,F,\Lambda,\lambda),M),(\alpha,\beta)) \in \mathbb{S}\times(\mathbb{\Lambda}^2) \mid \alpha,\beta \in \Lambda \wedge (\forall \sigma \in \Theta(((P,T,F,\Lambda,\lambda),M)) : (\exists\, i \in [1..|\sigma|] : \lambda(\sigma_i)=\alpha) \Rightarrow (\exists\, j \in [1..|\sigma|] : \lambda(\sigma_j)=\beta)\}$. ⌟

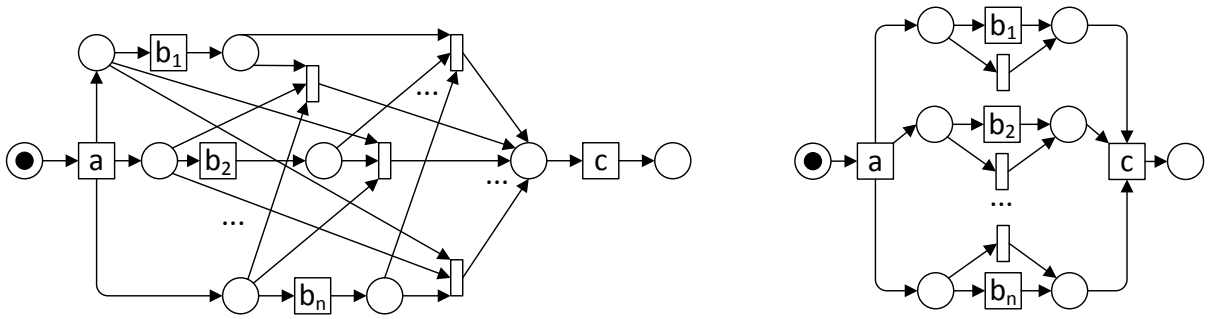
Then, the causal behavioral profile of a net system is defined as follows.

Definition 4.13 (Causal behavioral profile, cf. [25]).
The *causal behavioral profile* of a net system $S \in \mathbb{S}$, denoted by $\mathbb{BP}^{causal}(S)$, is the behavioral profile of S induced by the repertoire of behavioral predicates $\Pi := \{\nleftarrow, \nrightarrow, +, |||, \gg\}$, i.e., $\mathbb{BP}^{causal}(S) := \mathbb{R}[S,\Pi]$, where $\gg$ is the co-occurrence behavioral predicate, and the other behavioral predicates are defined in Definition 3.4. ⌟

Despite adding one additional behavioral predicate, causal behavioral profiles still fall short of preserving configuration equivalence on the class of WF-flow net systems. Two flow net systems S_7 and S_8 in Fig. 7 are not configuration equivalent. Interestingly, there is only one configuration that distinguishes the two systems, namely $\{\mathtt{a},\mathtt{c}\}$; note that we restrict configurations to observable transitions. However, their causal and, hence, classic behavioral profiles are equal, i.e., $\mathbb{BP}^{causal}(S_7) = \mathbb{BP}^{causal}(S_8)$ and $\mathbb{BP}^{c}(S_7) = \mathbb{BP}^{c}(S_8)$.

Another class of behavioral profiles can be induced by the behavioral predicates of the 4C spectrum [11]. The 4C spectrum defines a repertoire of eighteen behavioral predicates that capture such behavioral phenomena as co-occurrence, conflict, causality, and concurrency. Due to the non-mutually-exclusive nature of the predicates, two task labels of a system can be in several behavioral relations of the 4C spectrum at the same time, i.e., the repertoire of behavioral predicates of the 4C spectrum is non-basic, cf. Definition 3.5.

Despite the large size of the repertoire of behavioral predicates of the 4C spectrum, this repertoire does not preserve configuration equivalence on the class of WF-flow net systems. Indeed, the two WF-flow net systems in Fig. 7 are not configuration equivalent but have equal behavioral profiles induced by the predicates of the 4C spectrum; we refer the reader to [11] for the precise definition of the 4C spectrum. Fig. 8 shows two



(a) S_9 (b) S_{10}

Fig. 8. Generalization of the net systems in Fig. 7.

constructions that generalize the net systems in Fig. 7. For any fixed value of $n \in \mathbb{N}$, system S_9 in Fig. 8(a) comprises the set of configurations $\{\{\mathtt{a}, \mathtt{b}_1, \mathtt{b}_2, \ldots, \mathtt{b}_n, \mathtt{c}\}\} \cup \{\{\mathtt{a}, \mathtt{b}_m, \mathtt{c}\} \mid m \in [1\,..\,n]\}$. However, S_9 has the same representation as system S_{10} in Fig. 8(b) over the relations of the 4C spectrum. Note that system S_9 encodes $n+1$ configurations, whereas system S_{10} describes 2^n configurations. Therefore, there exist two net systems for which there is an exponential number of configurations that are indistinguishable when using the representation based on the behavioral predicates of the 4C spectrum; specifically, $2^n - n - 1$ indistinguishable configurations for systems in Fig. 8.

The above observations confirm that existing behavioral profiles are lossy behavioral representations of net systems. Thus, if one relies on the existing behavioral profiles in the context of process model comparison, then one must tolerate inaccurate diagnosis. To address this problem, one can either look for new and more accurate behavioral profiles or, alternatively, explore behavior representations in terms of occurrences of actions. However, the size of such latter representations can be larger than $|\Lambda|^2$.

5. Behavioral Profiles and Regular Languages

Under the notion of *trace equivalence*, a dynamic system is fully characterized by the language consisting of all the strings that the system can *recognize*, where the symbols of the alphabet of this language represent actions/tasks that the system can perform. Each string in this language describes a possible computation, or *instance*, of the system, i.e., a sequence of actions that can be performed by the system.

Taking this viewpoint, this section examines the ability of behavioral profiles to discriminate net systems that accept regular languages. We start by introducing notation related to formal languages, followed by a definition of net systems that induce regular languages. Then, we show that there exist regular net systems that cannot be distinguished by behavioral profiles, regardless of the specific set of predicates employed.

5.1. Strings and languages

An *alphabet* is any non-empty finite set. The elements of an alphabet are also called *symbols*. The following are three examples of alphabets:

- $\Sigma_1 := \{\mathtt{0}, \mathtt{1}\}$
- $\Sigma_2 := \{\mathtt{0}, \mathtt{1}, \mathtt{2}, \mathtt{3}, \mathtt{4}, \mathtt{5}, \mathtt{6}, \mathtt{7}, \mathtt{8}, \mathtt{9}, \mathtt{a}, \mathtt{b}, \mathtt{c}, \mathtt{d}, \mathtt{e}, \mathtt{f}\}$
- $\Sigma_3 := \{\mathtt{a}, \mathtt{b}, \mathtt{c}, \mathtt{d}, \mathtt{e}, \mathtt{f}, \mathtt{g}, \mathtt{h}, \mathtt{i}, \mathtt{j}, \mathtt{k}, \mathtt{l}, \mathtt{m}, \mathtt{n}, \mathtt{o}, \mathtt{p}, \mathtt{q}, \mathtt{r}, \mathtt{s}, \mathtt{t}, \mathtt{u}, \mathtt{v}, \mathtt{w}, \mathtt{x}, \mathtt{y}, \mathtt{z}\}$

A *string* over some alphabet is a finite sequence of symbols from that alphabet. The symbols of a string are usually written next to one another. For example, $q := \mathtt{101011}$, $r := \mathtt{2b}$, and $s := \mathtt{abababahalamaha}$, are strings over Σ_1, Σ_2, and Σ_3, respectively; q can also be seen as a string over alphabet Σ_2. If x is a string, the *length* of x, denoted by $|x|$, is the number of symbols that x contains, e.g., $|q| = 6$, $|r| = 2$, and $|s| = 13$. The string of length zero is called the *empty string* and is denoted by ϵ.

If a string x over an alphabet Σ has length n, we can write $x = x_1 x_2 \ldots x_n$, where each $x_i \in \Sigma$, $i \in [1..n]$.

Given a string x over an alphabet Σ and a set $X \subseteq \Sigma$, by $x|_X$ we denote x *restricted to* X, i.e., the string obtained from x by deleting all symbols of x that are not members of X without changing the order of the remaining symbols, e.g., $\texttt{ababahalamaha}|_{\{\texttt{a},\texttt{b},\texttt{h}\}} = \texttt{abababhaaaha}$.

The *concatenation* of a string x of length n and a string y of length m is the string obtained by appending y to the end of x, as in $x_1 \dots x_n y_1 \dots y_m$, denoted by $x \circ y$. We use the superscript notation x^z, $z \in \mathbb{N}$, to denote the string obtained by concatenating x with itself z times. For example, $x^1 = x$, $y^2 = yy$, and $\texttt{1}^3 = \texttt{111}$.

Finally, a (formal) *language* over an alphabet Σ is a set of strings over Σ.

5.2. Regular languages and regular net systems

A *regular language* is a formal language that can be expressed using a regular expression (for the strict form of regular expressions used in theoretical computer science [13]). Alternatively, a regular language can be defined as a language recognized by a *finite automaton* as per Kleene's theorem [27].

Definition 5.1 (Finite automaton).
A *finite automaton* is a 5-tuple $(Q, \Sigma, \delta, q_0, K)$, where
- Q is a finite non-empty set of *states*,
- Σ is an *alphabet*,
- $\delta : Q \times (\Sigma \cup \{\tau\}) \to Q$ is the *transition function*, where τ is a special symbol, $\tau \notin \Sigma$,
- $q_0 \in Q$ is the *start state*, and
- $K \subseteq Q$ is the *set of accept states*.

⌟

Next, we give a formal definition of a computation of a finite automaton.

Definition 5.2 (Computation).
A *computation* of a finite automaton $(Q, \Sigma, \delta, q_0, K)$ is either the empty string ϵ or a string $s := s_1 s_2 \dots s_n$, $n \in \mathbb{N}$, where every s_i is a member of $\Sigma \cup \{\tau\}$, $i \in [1..n]$, and there exists a sequence of states $q := \langle q_0, q_1, \dots, q_n \rangle$, where every q_j is a member of the set of states Q, $j \in [1..n]$, such that for every position k in s it holds that $\delta(q_{k-1}, s_k) = q_k$.

⌟

We say that s *leads to* q_n. By convention, the empty string always leads to the start state. A finite automaton $A := (Q, \Sigma, \delta, q_0, K)$ *accepts* a sting s iff s is a computation of A that leads to an accept state $q \in K$ of A.

Definition 5.3 (Language of a finite automaton).
The *language* of a finite automaton $A := (Q, \Sigma, \delta, q_0, K)$ is denoted by $L(A)$ and is the set of all strings that A accepts restricted to Σ, i.e., $L(A) := \{s \in \Sigma^* \mid \exists\, r \in (\Sigma \cup \{\tau\})^* : (A \textit{ accepts } r) \wedge (s = r|_\Sigma)\}$.

⌟

We say that finite automaton A *recognizes* language $L(A)$.

Let $S := (N, M)$, where $N := (P, T, F, \Lambda, \lambda)$, be a net system. Function λ can be lifted to a homomorphism from T^* to Λ^* in the canonical way as follows: $\lambda(\langle\rangle) := \langle\rangle$ and $\lambda(\sigma \circ \langle t \rangle) := \lambda(\sigma) \circ \lambda(t)$ for $\sigma \in T^*$ and $t \in T$.

For a given net system (N, M) and a finite set of markings $\mathbb{M}$ of N, the triple $(N, M, \mathbb{M})$ is a *net system with accept marking set*. The set of all occurrence sequences in (N, M) that lead to some marking $M' \in \mathbb{M}$ induce the language of the system.

Definition 5.4 (Language of a net system).
The language of a net system with accept marking set $S := (N, M, \mathbb{M})$, $N := (P, T, F, \Lambda, \lambda)$, is denoted by $L(S)$ and is the set of strings $\{s \in \Lambda^* \mid \exists\, \sigma \in \Omega((N, M)) : \lambda(\sigma)|_\Lambda = s \wedge (\exists\, M' \in \mathbb{M} : (N, M)[\sigma\rangle(N, M'))\}$.

⌟

We say that S *recognizes* $L(S)$. For every regular language there exists a system that recognizes this language.

Proposition 5.5 (Finite automata and net systems).
For every finite automaton A there exists a net system with accept marking set S that recognizes the language of A, i.e., $L(S) = L(A)$.

⌟

Indeed, given a finite automaton, one can obtain a system that recognizes the same language via a straightforward construction. Let $A := (Q, \Sigma, \delta, q_0, K)$ be a finite automaton. Then, the net system with accept marking set $S := ((Q, \delta, F, \Sigma, \lambda), M, \mathbb{M})$, where $F := \{(q, (q, x, q')) \in Q \times \delta \mid (q, x, q') \in \delta\} \cup \{((q, x, q'), q') \in \delta \times Q \mid (q, x, q') \in \delta\}$, $\lambda := \{((q, x, q'), x) \in \delta \times (\Sigma \cup \{\tau\}) \mid (q, x, q') \in \delta\}$, M is the marking that puts one token at place $q_0 \in Q$ and no tokens elsewhere, and for every $k \in K$ it holds that $\mathbb{M}$ contains the marking that puts one token at place $k \in Q$ and no tokens elsewhere, recognizes the language of A, i.e., $L(A) = L(S)$.

A regular net system is a net system that recognizes a regular language.

Definition 5.6 (Regular net system).
A regular net system is a net system with accept marking set that recognizes a regular language. ⌟

Note that there exist net systems that are not bounded but recognize regular languages [14].

5.3. Behavioral profiles and regular net systems

By $\mathbb{S}_{reg}$, we denote the class of all regular net systems. It is well-known that the class of all net systems properly contains the class of all regular net systems, i.e., it holds that $\mathbb{S}_{reg} \subset \mathbb{S}$. It is also known that the class of regular net systems properly contains the class of all bounded net systems [14].

Let $\Lambda \subseteq \mathbb{A}$ be a set of labels. By $\mathbb{S}^{\Lambda}$, we denote the set of all net systems over labels in Λ, i.e., $\mathbb{S}^{\Lambda} := \{S \in \mathbb{S} \mid \mathit{labels}(S) = \Lambda\}$. Finally, we introduce the main result of this section, which states that behavioral profiles cannot distinguish between regular net systems with different languages.

Theorem 5.7 (Two regular net systems with distinct languages may have identical profiles).
For every repertoire of behavioral predicates Π there exist two regular net systems $S_1, S_2 \in \mathbb{S}_{reg}$ such that:

$$\mathbb{R}[S_1, \Pi] = \mathbb{R}[S_2, \Pi] \wedge L(S_1) \neq L(S_2).$$

⌟

Proof. Note that the set of all regular languages over an alphabet Σ is infinite. This follows from the following two facts: (i) for every symbol $\mathtt{a} \in \Sigma$ it holds that language $\{\mathtt{a}\}$, i.e., the language with exactly one string composed of a single symbol $\mathtt{a}$, is a regular language, and (ii) the class of regular languages is closed under the concatenation operation [13]. Hence, the set $\{\{\mathtt{a}^n\} \mid n \in \mathbb{N}\}$, where $\mathtt{a} \in \Sigma$, i.e., the set that for every natural number $n \in \mathbb{N}$ contains the language composed of exactly one string that is a concatenation of string $\mathtt{a}$ with itself n times, is, indeed, an infinite set of regular languages over Σ.

Furthermore, note that for every alphabet Σ and every repertoire of behavioral predicates Π, it holds that the set of all behavioral profiles of net systems in $\mathbb{S}^{\Sigma}$ induced by Π, i.e., $\{\mathbb{R}[S, \Pi] \mid S \in \mathbb{S}^{\Sigma}\}$, is finite. This follows immediately from the following two facts: (i) Π is finite, and (ii) there exist exactly $2^{|\Sigma|^n}$ distinct behavioral relations of net systems in $\mathbb{S}^{\Sigma}$ induced by n-ary behavioral predicates.

Let us assume that there exists a repertoire of behavioral predicates Π such that for every two regular net systems $S_1, S_2 \in \mathbb{S}_{reg}$ it holds that if $\mathbb{R}[S_1, \Pi] = \mathbb{R}[S_2, \Pi]$ then $L(S_1) = L(S_2)$. Let Σ be some alphabet and let R be the equivalence relation on $\mathbb{S}_{reg} \cap \mathbb{S}^{\Sigma}$ such that $S_1 \, R \, S_2$ iff $\mathbb{R}[S_1, \Pi] = \mathbb{R}[S_2, \Pi]$, $S_1, S_2 \in \mathbb{S}_{reg} \cap \mathbb{S}^{\Sigma}$. Note that the set of all equivalence classes of $\mathbb{S}_{reg} \cap \mathbb{S}^{\Sigma}$ by R is finite, as the set $\{\mathbb{R}[S, \Pi] \mid S \in \mathbb{S}^{\Sigma}\}$ is finite. Let f be a function that maps every regular language over Σ to its equivalence class of $\mathbb{S}_{reg} \cap \mathbb{S}^{\Sigma}$ by R that contains regular net systems that recognize this language; such an equivalence class always exists because of Proposition 5.5 and the fact that one can always construct a net system with accept marking set that recognizes the same language as a given finite automaton. Note that f is non-injective; recall that the set of all regular languages over Σ is infinite. Then, there exist two distinct regular languages L_1 and L_2 over Σ such that $f(L_1) = f(L_2)$. Hence, there exist two regular net systems S_1 and S_2 such that $\mathbb{R}[S_1, \Pi] = \mathbb{R}[S_2, \Pi]$ and $L(S_1) \neq L(S_2)$. We have reached a contradiction. ∎

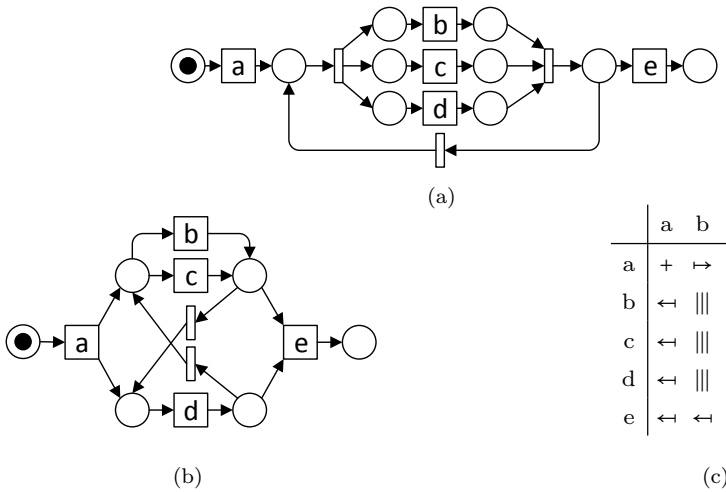
As a consequence of Theorem 5.7, one cannot rely on behavioral profiles – regardless of the specific set of behavioral relations employed – as the basis for deciding whether two regular language systems, e.g., finite automata [13] or regular net systems [14], recognize the same language, i.e., they are trace equivalent [5].

Corollary 5.8 (Regular net systems and equivalence preservation).
There exist no repertoire of behavioral predicates that preserves trace equivalence on $\mathbb{S}_{reg}$. ⌟

For example, the classic behavioral profiles of the two net systems in Figs. 9(a) and 9(b) are identical to the one shown in Fig. 9(c). However, the languages of these net systems are different (assuming that the accept marking set of each of the systems is composed of a single marking that puts one token in the only place in the postset of the transition with label $\mathtt{e}$).

6. Conclusion

In summary, this paper has shown that:



	a	b	c	d	e
a	+	↦	↦	↦	↦
b	↤	‖	‖	‖	↦
c	↤	‖	‖	‖	↦
d	↤	‖	‖	‖	↦
e	↤	↤	↤	↤	+

Fig. 9. Two net systems (a) and (b), and their classic behavioral profile (c).

1. The repertoire of behavioral predicates used to induce classic behavioral profiles proposed in [20] preserves configuration equivalence on the class of unlabeled WF-flow nets. This provides evidence that behavioral profiles can be used to characterize the behavior of a restricted class of systems.
2. There exists no known repertoire of behavioral predicates that preserves some well-known notion of behavioral equivalence on the class of acyclic labeled WF-net systems.
3. For any finite set of behavioral predicates, behavioral profiles induced by this set are strictly less expressive than regular language systems, i.e., the set does not preserve trace equivalence on systems that recognize regular languages.

These results leave open the question of possible trade-offs between expressiveness and convenience of use of behavioral profiles. One assumption underpinning behavioral profiles is that the size of the matrix, when one considers binary behavioral relations, is $|\Lambda|^2$ where Λ is a set of task labels. In contrast, other behavioral representations, such as event structures [10], capture relations between label occurrences, which entails a larger representation as the same label type may have multiple occurrences, to the benefit of expressiveness.

A possible trade-off between behavioral profiles and event structures would be to reduce the size of event structures, for example via reduction rules, while retaining expressiveness. In separate work, we have proposed a notion of canonically reduced event structures as a means to achieve this trade-off [1, 2]. However, the extent to which reduction can be achieved for different classes of net systems is still an open question.

An interesting line of future research is the analysis of other existing behavioral relations, e.g., the relations in [7] and [26], that can potentially offer more fine-grained insights into the expressiveness of behavioral profiles. It would be particularly interesting to define behavioral predicates that preserve trace equivalence or configuration equivalence (or any other well-known notion of behavioral equivalence) on a more substantial family of net systems rather than unlabeled WF-flow net systems.

References

[1] A. Armas-Cervantes, P. Baldan, M. Dumas, and L. García-Bañuelos. Behavioral comparison of process models based on canonically reduced event structures. In *Business Process Management – 12th International Conference, BPM 2014, Haifa, Israel, September 7–11, 2014. Proceedings*, volume 8659 of *Lecture Notes in Computer Science*, pages 267–282. Springer, 2014.

[2] A. Armas-Cervantes, P. Baldan, and L. García-Bañuelos. Reduction of event structures under history preserving bisimulation. *Journal of Logical and Algebraic Methods in Programming*, 2015. (In press).

[3] A. Armas-Cervantes, M. Dumas, L. García-Bañuelos, and A. Polyvyanyy. On the suitability of generalized behavioral profiles for process model comparison. In *Web Services and Formal Methods – 11th International Workshop, WS-FM 2014, Eindhoven, The Netherlands, September 11–12, 2014. Proceedings. (Accepted on 10 July 2014)*.

[4] G. Boudol. Flow event structures and flow nets. In *Semantics of Systems of Concurrent Processes, LITP Spring*

School on Theoretical Computer Science, La Roche Posay, France, April 23–27, 1990, Proceedings, volume 469 of *Lecture Notes in Computer Science*, pages 62–95. Springer, 1990.

[5] J. Engelfriet. Determinacy ⟶ (observation equivalence = trace equivalence). *Theoretical Computer Science (TCS)*, 36:21–25, 1985.

[6] J. Engelfriet. Branching processes of Petri nets. *Acta Inf.*, 28(6):575–591, 1991.

[7] S. Haar, C. Kern, and S. Schwoon. Computing the reveals relation in occurrence nets. *Theoretical Computer Science (TCS)*, 493:66–79, 2013.

[8] M. Kunze, M. Weidlich, and M. Weske. Behavioral similarity — A proper metric. In *Business Process Management – 9th International Conference, BPM 2011, Clermont-Ferrand, France, August 30 – September 2, 2011. Proceedings*, volume 6896 of *Lecture Notes in Computer Science*, pages 166–181. Springer, 2011.

[9] T. Murata. Petri nets: Properties, analysis and applications. *Proceedings of the IEEE*, 77(4), 1989.

[10] M. Nielsen, G. D. Plotkin, and G. Winskel. Petri nets, event structures and domains, Part I. *Theoretical Computer Science (TCS)*, 13:85–108, 1981.

[11] A. Polyvyanyy, M. Weidlich, R. Conforti, M. L. Rosa, and A. H. M. ter Hofstede. The 4C spectrum of fundamental behavioral relations for concurrent systems. In *Application and Theory of Petri Nets and Concurrency – 35th International Conference, PETRI NETS 2014, Tunis, Tunisia, June 23–27, 2014. Proceedings*, volume 8489 of *Lecture Notes in Computer Science*, pages 210–232. Springer, 2014.

[12] A. Polyvyanyy, M. Weidlich, and M. Weske. Isotactics as a foundation for alignment and abstraction of behavioral models. In *Business Process Management – 10th International Conference, BPM 2012, Tallinn, Estonia, September 3–6, 2012. Proceedings*, volume 7481 of *Lecture Notes in Computer Science*, pages 335–351. Springer, 2012.

[13] M. Sipser. *Introduction to the Theory of Computation*. Cengage Learning, 3rd edition, 2012.

[14] R. Valk and G. Vidal-Naquet. Petri nets and regular languages. *Journal of Computer and System Sciences (JCSS)*, 23(3):299–325, 1981.

[15] W. M. P. van der Aalst. Verification of workflow nets. In *Application and Theory of Petri Nets 1997, 18th International Conference, ICATPN '97, Toulouse, France, June 23-27, 1997, Proceedings*, volume 1248 of *Lecture Notes in Computer Science*, pages 407–426. Springer, 1997.

[16] W. M. P. van der Aalst. Workflow verification: Finding control-flow errors using Petri-net-based techniques. In *Business Process Management, Models, Techniques, and Empirical Studies*, volume 1806 of *Lecture Notes in Computer Science*, pages 161–183. Springer, 2000.

[17] R. J. van Glabbeek. *The Linear Time-Branching Time Spectrum*, volume 458 of *Lecture Notes in Computer Science*. Springer, 1990.

[18] R. J. van Glabbeek and U. Goltz. Equivalence notions for concurrent systems and refinement of actions. In *Mathematical Foundations of Computer Science 1989, MFCS'89, Porabka-Kozubnik, Poland, August 28 – September 1, 1989, Proceedings*, volume 379 of *Lecture Notes in Computer Science*, pages 237–248. Springer, 1989.

[19] R. J. van Glabbeek and U. Goltz. Refinement of actions and equivalence notions for concurrent systems. *Acta Informatica (ACTA)*, 37(4/5):229–327, 2001.

[20] M. Weidlich, J. Mendling, and M. Weske. Efficient consistency measurement based on behavioral profiles of process models. *IEEE Transactions on Software Engineering (TSE)*, 37(3):410–429, 2011.

[21] M. Weidlich, J. Mendling, and M. Weske. A foundational approach for managing process variability. In *Advanced Information Systems Engineering – 23rd International Conference, CAiSE 2011, London, UK, June 20–24, 2011. Proceedings*, volume 6741 of *Lecture Notes in Computer Science*, pages 267–282. Springer, 2011.

[22] M. Weidlich, A. Polyvyanyy, N. Desai, and J. Mendling. Process compliance measurement based on behavioural profiles. In *Advanced Information Systems Engineering, 22nd International Conference, CAiSE 2010, Hammamet, Tunisia, 2010. Proceedings*, volume 6051 of *Lecture Notes in Computer Science*, pages 499–514. Springer, 2010.

[23] M. Weidlich, A. Polyvyanyy, N. Desai, J. Mendling, and M. Weske. Process compliance analysis based on behavioural profiles. *Information Systems (IS)*, 36(7):1009–1025, 2011.

[24] M. Weidlich, A. Polyvyanyy, J. Mendling, and M. Weske. Efficient computation of causal behavioural profiles using structural decomposition. In *Applications and Theory of Petri Nets, 31st International Conference, PETRI NETS 2010, Braga, Portugal, June 21–25, 2010. Proceedings*, volume 6128 of *Lecture Notes in Computer Science*, pages 63–83. Springer, 2010.

[25] M. Weidlich, A. Polyvyanyy, J. Mendling, and M. Weske. Causal behavioural profiles — efficient computation, applications, and evaluation. *Fundamenta Informaticae (FUIN)*, 113(3-4):399–435, 2011.

[26] M. Weidlich and J. M. E. M. van der Werf. On profiles and footprints – relational semantics for Petri nets. In *Application and Theory of Petri Nets – 33rd International Conference, PETRI NETS 2012, Hamburg, Germany, June 25–29, 2012. Proceedings*, volume 7347 of *Lecture Notes in Computer Science*, pages 148–167. Springer, 2012.

[27] S. Yu. Regular languages. In G. Rozenberg and A. Salomaa, editors, *Handbook of Formal Languages*, pages 41–110. Springer Berlin Heidelberg, 1997.