

Information Flow Among Transitions of Bounded Equal-Conflict Petri Nets

Federica Adobbati¹, Luca Bernardinello¹, Görkem Kılınc Soylu¹ and Lucia Pomello¹

¹*Dipartimento di Informatica, Sistemistica e Comunicazione, Università degli Studi di Milano - Bicocca, viale Sarca 336 U14, Milano, Italia*

Abstract

In a distributed system, in which an action can be either “hidden” or “observable”, an unwanted information flow might arise when occurrences of observable actions give information about occurrences of hidden actions. A collection of relations, i.e. reveals and its variants, is used to model such information flow among transitions of a Petri net. This paper introduces a parametrised generalisation to the existing reveals relations and provides a formal basis for information-flow analysis of bounded equal-conflict PT systems.

Keywords

Information flow, noninterference, bounded equal-conflict Petri nets, reveals relations, distributed systems.

1. Introduction

In distributed systems, it is often the case that some actions are confidential, and it is not desired that a user is able to deduce information about occurrences of these confidential actions, while still being able to interact with the system. This kind of unwanted information flow can form a security concern for systems. *Noninterference* is a formal notion which guarantees that a system does not suffer from such information flow.

The concept of noninterference was introduced by Goguen and Meseguer for deterministic state machines in [1]. In this concept, a system is viewed as consisting of components at two distinct levels of confidentiality: *high* (hidden) and *low* (observable). A system is then said to be secure with respect to noninterference if a user, who knows the structure of the system, cannot deduce information about high actions by interacting only via low actions. Sutherland and McCullough moved the concept to the nondeterministic and concurrent systems in [2] and [3]. Since then, various noninterference properties have been proposed in the literature based on different system models, e.g., [4, 5, 6, 7]. An overview on information-flow security and noninterference is provided in [8].


Busi and Gorrieri moved the notion of noninterference to 1-safe Petri nets by studying observational equivalences and structural properties in [9]. In [10], the authors investigate structural properties, such as absence of certain places, for defining noninterference in elementary net sys-

PNSE'22: International Workshop on Petri Nets and Software Engineering, Bergen, Norway, 2022

✉ f.adobbati@campus.unimib.it (F. Adobbati); luca.bernardinello@unimib.it (L. Bernardinello); gorkemkln@gmail.com (G. Kılınc Soylu); lucia.pomello@unimib.it (L. Pomello)



© 2022 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

 CEUR Workshop Proceedings (CEUR-WS.org)

tems. In [11], Best et. al. study the decidability of noninterference in Petri nets. In [12], Baldan and Carraro give a characterisation of noninterference in terms of causalities and conflicts on unfoldings of 1-safe Petri nets. In [13], the authors work on the minimal solutions for enforcing noninterference on bounded Petri nets.

The work presented in [14] and [15] distinguish two kinds of information flow in Petri nets, i.e. positive and negative information flow. The first one arises when the occurrence of a low transition gives information about the occurrence of a high transition whereas the second arises when the occurrence of a low transition allows one to deduce the non-occurrence of a high transition. The authors introduce two families of relations, namely *reveals* and *excludes*, to model positive and negative information flow among transitions of a Petri net. Reveals relation was originally defined for the events of an occurrence net by Haar in [16] to be used in fault diagnosis [17]. The reveals relation was then redefined for the transitions of Petri nets in [14, 15] and was used, together with its variants and excludes relation, to define various noninterference properties for distributed systems modelled with Petri nets.

In [18], a formal basis is provided for modelling and analysing information flow with 1-safe free-choice Petri nets by computing reveals and excludes relations. The authors introduce a notion of *maximal-step computation tree*, which represents the behaviour of a 1-safe free-choice net under maximal-step semantics. They define a finite prefix, named *full prefix*, on the tree and provide methods for computing reveals and excludes relations on the prefix.

In this paper, we extend the results in [18] to bounded equal-conflict Petri nets, which are a weighted generalisation of (extended) free-choice nets and were introduced in [19]. We introduce two new parametric reveals relations and provide methods for computing them on bounded equal-conflict PT systems for positive information flow.

The paper is organised as follows. Section 2 provides the necessary background on Petri nets. Section 3 recalls some definitions of reveals relation in the context of information flow and introduces new parametric variants. Section 4 formalises maximal-step computation tree and its full prefix for bounded equal-conflict PT systems and provides methods for computing reveals relations for positive information-flow analysis on full prefix. Section 5 concludes the paper and discusses some possible future work.

2. Petri nets

In this section we recall basic definitions concerning Petri nets and net unfoldings that will be useful in the rest of the paper, see also [20, 21, 22, 23, 24].

A *net* is a triple $N = (P, T, F)$, where P and T are disjoint sets. The elements of P are called *places* and are represented by circles, the elements of T are called *transitions* and are represented by squares, $F \subseteq (P \times T) \cup (T \times P)$ is the *flow relation*, represented by directed arcs. The *pre-set* of an element $x \in P \cup T$ is the set $\bullet x = \{y \in P \cup T : (y, x) \in F\}$; the *post-set* of x is the set $x^\bullet = \{y \in P \cup T : (x, y) \in F\}$. Let $X \subseteq P \cup T$ be a subset of elements, its pre-set is defined as $\bullet X = \{y \in P \cup T : \exists x \in X : (y, x) \in F\}$, and its post-set as $X^\bullet = \{y \in P \cup T : \exists x \in X : (x, y) \in F\}$.

A net (P, T, F) is a *subnet* of another net (P', T', F') if $P \subseteq P'$, $T \subseteq T'$ and F is the restriction of F' to $(P \times T) \cup (T \times P)$.

A net is *finite* if $P \cup T$ is finite, and *infinite* otherwise. A net is *T-restricted* if for any $t \in T$, $\bullet t \neq \emptyset$ and $t^\bullet \neq \emptyset$.

A Place Transition (PT) system, $\Sigma = (P, T, F, W, m_0)$ is defined by a finite net (P, T, F) , a map $W : F \rightarrow \mathbb{N}$ which assigns a positive *weight* to each arc, and an *initial marking* $m_0 : P \rightarrow \mathbb{N}$. We write $W(x, y) = 0$ when $(x, y) \notin F$. The tuple (P, T, F, W) will be called *PT net*. In graphical representations, arcs are inscribed by their weights, if the weight is greater than one, and markings m are represented by $m(p)$ dots, called tokens, in each place p .

Let Σ be a PT net and $m : P \rightarrow \mathbb{N}$ be a marking. A transition t is *enabled* at m , denoted $m[t]$, if, for any $p \in P$, $m(p) \geq W(p, t)$. Let t be enabled at m ; then, t can *occur* (or *fire*) in m producing the new marking m' , denoted $m[t]m'$ and defined as follows: $\forall p \in P$, $m'(p) = m(p) - W(p, t) + W(t, p)$.

A finite or infinite sequence of transitions $t_1 t_2 \dots t_k \dots$ is an *occurrence sequence* enabled at m_0 , denoted $m_0[t_1 \dots t_k \dots]$, if there are intermediate markings $m_1 \dots m_k \dots$ such that: $m_0[t_1]m_1 \dots [t_k]m_k \dots$.

Let m be a marking of Σ , the set $[m]$ is the smallest set of markings such that: $m \in [m]$, and if $m' \in [m]$ and $m'[t]m''$, then $m'' \in [m]$. The set $[m_0]$ is the set of *reachable markings* of Σ .

A *multiset* of transitions $U : T \rightarrow \mathbb{N}$ is *concurrently enabled* at $m \in [m_0]$, denoted $m[U]$, and called *step*, iff $\forall p \in P$, $\sum_{t \in T} U(t) \cdot W(p, t) \leq m(p)$; if U is enabled at m , it can occur producing the new marking m' , denoted $m[U]m'$ and defined as follows: $\forall p \in P$, $m'(p) = m(p) - \sum_{t \in T} U(t) \cdot W(p, t) + \sum_{t \in T} U(t) \cdot W(t, p)$.

A multiset $U : T \rightarrow \mathbb{N}$ is a *maximal-step* at $m \in [m_0]$ iff $m[U]$ and $\forall t' \in T$, $\exists p \in P$ such that: $m(p) < \sum_{t \in T} U(t) \cdot W(p, t) + W(p, t')$.

Analogously to occurrence sequences, a finite or infinite sequence of steps (or maximal-steps) $U_1 U_2 \dots U_k \dots$ is a *step sequence* (or a *maximal-step sequence*) enabled at m_0 , denoted $m_0[U_1 \dots U_k \dots]$, if there are intermediate markings $m_1 \dots m_k \dots$ such that: $m_0[U_1]m_1 \dots [U_k]m_k \dots$.

Two transitions t_1 and t_2 are in *conflict at a marking* $m \in [m_0]$ iff they are both enabled at m , $m[t_1]$ and $m[t_2]$, however they are not concurrently enabled at m , $\neg(m[\{t_1, t_2\}])$, the occurrence of one of them disables the other one.

In some situations concurrency and conflicts overlap in such a way that it is not clear if in the execution of concurrent transitions a conflict has been solved or not. This is a so called situation of ‘confusion’ which has been introduced by C.A. Petri and discussed in several papers as for example in [25, 26], and which can be formalised in the following way. Let $m \in [m_0]$, $t_1, t_2 \in T$ such that: $m[\{t_1, t_2\}]$, then (m, t_1, t_2) is a *confusion* at m if $\text{cfl}(t_1, m) \neq \text{cfl}(t_1, m_2)$, where $\text{cfl}(t_1, m) = \{t' \in T : m[t'] \wedge \neg(m[\{t_1, t'\}])\}$, and m_2 is such that $m[t_2]m_2$. In Fig. 1, the two main situations of confusion are illustrated, namely *asymmetric* confusion, on the left side, and *symmetric* confusion, on the right side. In the particular case of asymmetric confusion, the occurrence of the step $\{t_1, t_2\}$ hides the information whether the conflict in-between t_1 and t_3 has been solved or not; moreover, in this case, maximal-step semantics is not equivalent to step semantics, since transition t_3 will never occur in any sequence of maximal steps, whereas the step sequence $m_0[\{t_2\}]\{b_1, b_3, b_4\}[\{t_3\}]$ may occur.

The relation between step semantics and maximal-step semantics has been studied for example in [27], where it is shown that they are equivalent in the case of systems without asymmetric confusion.

A general class of nets in which confusion cannot arise is the class of equal-conflict PT nets,

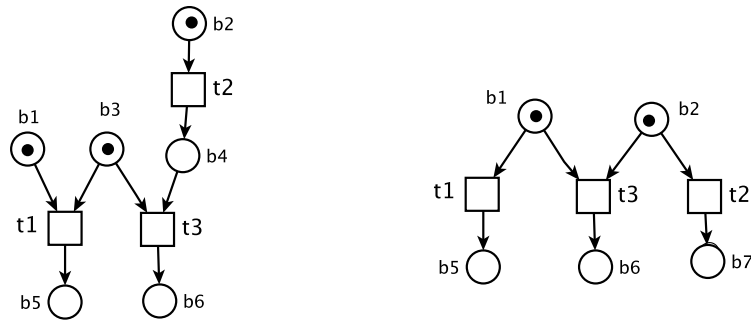


Figure 1: Asymmetric confusion (on the left) and symmetric confusion (on the right side).

studied by various authors, see for example [19]. A PT net (P, T, F, W) is an *equal-conflict PT net*, also called *EC PT net*, iff $\forall t, t' \in T, \bullet t \cap \bullet t' \neq \emptyset \Rightarrow (\bullet t = \bullet t' \wedge \forall p \in \bullet t, W(p, t) = W(p, t'))$.

From the definition, it follows that in EC PT systems, if two transitions share a pre-place, then any given marking enables either both transitions or neither. All the figures in this paper, except for Fig. 1, provide examples of EC PT systems.

A PT system Σ is *bounded* if there is a finite number n such that: for any reachable marking $m \in [m_0]$ and for any place p , $m(p) \leq n$. If Σ is bounded, its set of reachable markings $[m_0]$ is finite. A marking m is a *deadlock* if no transition is enabled in m . Σ is *1-live* if for all $t \in T$ there exists $m \in [m_0]$ such that $m[t]$.

In the rest of the paper, we will consider finite, bounded, and 1-live PT systems, whose underlying nets are T-restricted.

We now introduce two technical relations that will be useful to define the partial order semantics of PT systems. The \prec relation on the elements of a net N is the transitive closure of F and \preceq is the reflexive closure of \prec . Let $x, y \in P \cup T$, $x \# y$, iff there exist $t_1, t_2 \in T : t_1 \neq t_2, t_1 \preceq x, t_2 \preceq y$ and there exists $p \in \bullet t_1 \cap \bullet t_2$.

A net $N = (B, E, F)$, possibly infinite, is an *occurrence net* if the following restrictions hold:

1. $\forall x \in B \cup E : \neg(x \prec x)$
2. $\forall x \in B \cup E : \neg(x \# x)$
3. $\forall e \in E : \{x \in B \cup E \mid x \preceq e\}$ is finite
4. $\forall b \in B : |\bullet b| \leq 1$

In an occurrence net, the elements of B are called *conditions* and the elements of E are called *events*; the transitive and reflexive closure of F , \preceq , forms a *partial order*. The set of minimal elements of an occurrence net N with respect to \preceq will be denoted by $\min(N)$. Since we only consider T-restricted nets the elements of $\min(N)$ are conditions.

A *configuration* of an occurrence net $N = (B, E, F)$ is a, possibly infinite, set of events $C \subseteq E$ which is causally closed (for every $e \in C$, $e' \preceq e \Rightarrow e' \in C$) and free of conflicts ($\forall e_1, e_2 \in C, \neg(e_1 \# e_2)$). C is *maximal* if it is maximal with respect to set inclusion.

On the elements of an occurrence net the relation of concurrency, **co**, is defined as follows: let $x, y \in P \cup T$, x **co** y , if neither $(x \prec y)$ nor $(y \prec x)$ nor $(x \# y)$.

A *B-cut* of N is a maximal set of pairwise concurrent elements of B , and can be intuitively seen as a global state of the net in a certain moment. An *E-cut* of N is a maximal set of pairwise concurrent elements of E , that corresponds to a maximal step on N .

A *branching process* of a bounded 1-live PT system $\Sigma = (P, T, F, W, m_0)$, whose underlying net is T-restricted, is a pair (O, λ) , where $O = (B, E, F')$ is an occurrence net, and λ is a map from $B \cup E$ to $P \cup T$ such that:

1. $\lambda(B) \subseteq P; \lambda(E) \subseteq T$
2. $\forall e \in E, \forall p \in P, W(p, \lambda(e)) = |\lambda^{-1}(p) \cap \bullet e|$ and $W(\lambda(e), p) = |\lambda^{-1}(p) \cap e \bullet|$
3. $\forall p \in P m_0(p) = |\lambda^{-1}(p) \cap \min(O)|$
4. $\forall x, y \in E, \text{if } \bullet x = \bullet y \text{ and } \lambda(x) = \lambda(y), \text{ then } x = y$

We extend the definition of λ to the set of configurations of the branching process: for each configuration C , $\lambda(C)$ is the multiset of the transitions whose occurrences are recorded in C and is called the *footprint* of C ; formally $\lambda(C) = \sum_{e_i \in C} \lambda(e_i)$. If a transition t belongs to the support set of $\lambda(C)$, i.e.: at least an occurrence of t is in C , $\lambda(C)(t) \geq 1$, then we use the notation $t \in \lambda(C)$. If C is infinite, it records the infinite occurrence of some transitions, and the multiset $\lambda(C)$ is such that the multiplicity of those transitions is infinite, whereas its support set is obviously finite, being a subset of T .

A branching process (O_1, λ_1) is a *prefix* of a branching process (O_2, λ_2) if O_1 is a subnet of O_2 containing all minimal elements ($\min(O_2)$) and such that: if $e \in E_1$ and $(b, e) \in F_2$ or $(e, b) \in F_2$ then $b \in B_1$; if $b \in B_1$ and $(e, b) \in F_2$ then $e \in E_1$; and λ_1 is the restriction of λ_2 to $B_1 \cup E_1$.

Any finite PT system $\Sigma = (P, T, F, W, m_0)$ has a unique branching process which is maximal with respect to the prefix relation. This maximal branching process, called the *unfolding* of Σ , will be denoted by $\text{Unf}(\Sigma) = ((B, E, F'), \lambda)$, where λ is the map from (B, E, F') to (P, T, F) .

A *run* records a possible non sequential behaviour of the system, it is a branching process, whose occurrence net is free of conflicts, i.e., its set of events is a configuration. A run is *maximal* if the corresponding configuration is maximal. Let ρ be a run, its footprint $\lambda(\rho)$ is equal to the footprint of its set of events.

Example 1. Figure 2 represents an equal-conflict PT system Σ and a branching process, which is a prefix of the unfolding of Σ . A maximal run of the branching process is the labelled subnet whose elements are grey, the set of its events is the configuration $C = \{e_1, e_2, e_3, e_4, e_5, e_6\}$, The footprint of C is the multiset $\lambda(C) = \{a, e, f, g, 2.h\}$.

3. Formal relations for the analysis of information flow

In [14] and [15], a family of relations, i.e., *reveals* and its variants, were introduced to express information flow among transitions of a Petri net. These relations were then used to define several noninterference properties for specification of security requirements. In this section, after recalling the existing reveals relations we introduce two new variants which are parametric. These new relations give the opportunity to specify security requirements of a distributed system in a way that is more tailored to the needs of the system.

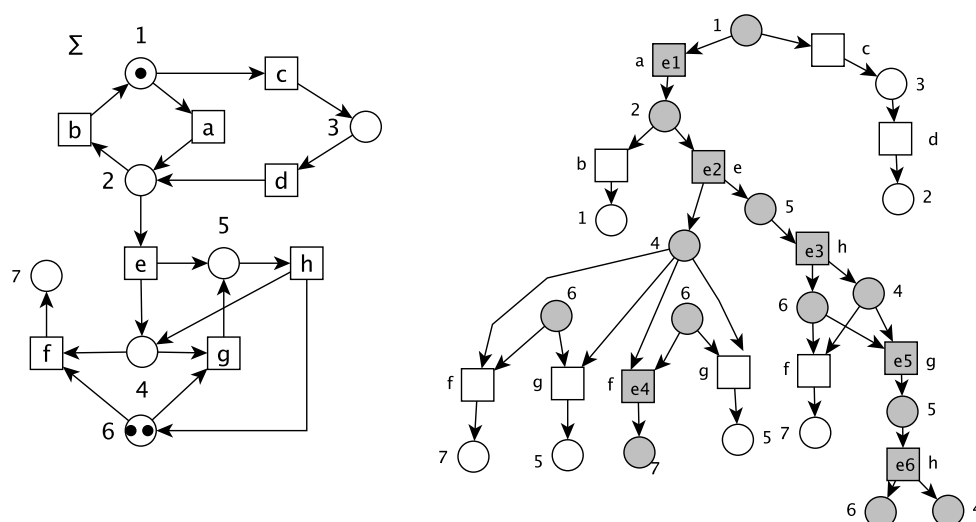


Figure 2: A bounded equal-conflict PT system and a branching process of it (a prefix of $\text{Unf}(\Sigma)$).

Let $\Sigma = (P, T, F, W, m_0)$ be a PT system (bounded, 1-live PT system with T-restricted underlying net) and $\text{Unf}(\Sigma) = ((B, E, F), \lambda)$ be its unfolding. In the following, C_{max} will denote the set of all maximal configurations of $\text{Unf}(\Sigma)$. The set of events of $\text{Unf}(\Sigma)$ corresponding to a specific transition t of a given PT system Σ will be denoted by $E_t = \{e \in E : \lambda(e) = t\}$. We will assume *progress* of the system, i.e., an enabled transition either fires or gets disabled by another transition in conflict with it; the set of configurations satisfying this assumption coincides with C_{max} .

The reveals relation was originally introduced for events of an occurrence net in [16], and in [17] it was applied in the field of fault diagnosis. In [15], reveals relation was redefined for the transitions of a 1-live Petri net in order to express positive information flow: transition t_1 reveals transition t_2 if each maximal configuration which contains an occurrence of t_1 also contains at least one occurrence of t_2 . This means that the occurrence of t_1 implies the occurrence of t_2 , either in the past or in the future. Hence, if a low transition reveals a high transition, this might cause a positive information flow endangering the security of the system.

Definition 1. Let $t_1, t_2 \in T$ be two transitions; t_1 reveals t_2 , denoted $t_1 \triangleright t_2$, iff $\forall C \in C_{max} \ t_1 \in \lambda(C) \implies t_2 \in \lambda(C)$.

Since we consider 1-live nets, there is at least one maximal configuration in which t_1 occurs. If there exists a maximal configuration in which t_1 occurs and t_2 does not occur, then $t_1 \not\triangleright t_2$.

Sometimes, even if a transition alone does not give much information about the occurrence of another transition, a set of transitions together might imply the occurrence of some transitions in another set. This relation was originally defined for the events of an occurrence net in [28], and later in [15] was redefined for the transitions of a Petri net in order to be used in information-flow analysis. Set X extended reveals set Y if each maximal configuration which

has all the transitions of X also has at least one transition of Y . It can violate the security of a system if a group of low transitions extended reveals a group of high transitions.

Definition 2. Let $X, Y \subseteq T$. If there is at least one maximal configuration in which all transitions of X appear, then we say X extended reveals Y , denoted $X \rightarrow Y$, iff $\forall C \in C_{max}$

$$\bigwedge_{t_i \in X} t_i \in \lambda(C) \implies \bigvee_{t_j \in Y} t_j \in \lambda(C).$$

Note that $X \rightarrow Y$ is not defined if the transitions of X never appear together.

Remark 1. Extended reveals relation between two singletons coincides with reveals relation.

The next relation takes the repeated occurrences of a transition into account. In some cases, the occurrence of a transition does not give information while several occurrences of the transition imply the occurrence of another transition. This relation was defined in [15] without progress assumption, thus considering all configurations. Here, we alter the definition for the setting in which maximal configurations are considered. We say that t_1 n -repeated reveals t_2 if each maximal configuration which has at least n occurrences of t_1 , also has at least one occurrence of t_2 . It can violate the security of a system if a number of occurrences of a low transition imply occurrence of a high transition.

Definition 3. Let $t_1, t_2 \in T$ be two transitions, and C_{max} be the set of all maximal configurations of $\text{Unf}(\Sigma)$. Let n be a positive integer. If there exists a maximal configuration in which t_1 occurs at least n times, then we say t_1 n -repeated reveals t_2 , denoted $n.t_1 \triangleright t_2$, iff

$$\forall C \in C_{max} \quad |C \cap E_{t_1}| \geq n \implies C \cap E_{t_2} \neq \emptyset.$$

Note that n -repeated reveals is not defined if there is no maximal configuration in which t_1 occurs at least n times.

The following statements are direct consequences of Def. 3.

Remark 2. Reveals relation (as in Def. 1) coincides with 1-repeated reveals.

Remark 3. If $n.t_1 \triangleright t_2$ and $\exists C \in C_{max}$ such that $|C \cap E_{t_1}| \geq n + 1$ then $(n + 1).t_1 \triangleright t_2$.

Remark 4. $n.t_1 \not\triangleright t_2 \implies (n - 1).t_1 \not\triangleright t_2$.

Example 2. A bounded equal-conflict PT system is illustrated in Fig. 3. In this system, the occurrence of b implies the occurrence of a since every maximal configuration which has b also has a . However, vice versa is not correct since there is a maximal configuration in which a occurs without b . Thus, $b \triangleright a$ and $a \not\triangleright b$. Some other reveals relations are $c \triangleright d$, $d \triangleright c$, $f \triangleright h$, $h \triangleright f$, $d \triangleright a$ and $h \triangleright e$.

Examples of extended reveals are as follows. The occurrence of c alone or f alone is not sufficient to say something about the occurrence of k , but the occurrence of c and f together implies the occurrence of k , i.e., $\{c, f\} \rightarrow \{k\}$. Although $a \not\triangleright c$ and $a \not\triangleright b$, the occurrence of a implies the occurrence of either b or c , i.e., $\{a\} \rightarrow \{b, c\}$. Similarly, $\{e\} \rightarrow \{f, g\}$.

Examples of repeated reveals are as follows. Only one occurrence of a does not give information about d , but if a occurs twice, this implies the occurrence of d , i.e., $2.a \triangleright d$. Similarly, two occurrences of e imply the occurrence of h , i.e., $2.e \triangleright h$.

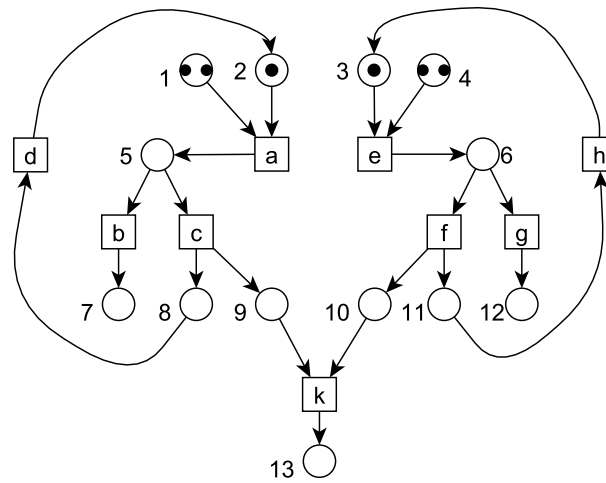
Σ_1


Figure 3: A bounded equal-conflict PT system.

The following variant of reveals relation is parametric and it generalises all the variants above. This relation is more expressive and allows one to specify the expected security requirements in a more tailored fashion.

Definition 4. Let $i, k \geq 1$ and $\{n_1, \dots, n_k\}$ be positive integers. Let $\{t_1, \dots, t_k\}, Y \subseteq T$. If there is at least one maximal configuration in which each transition t_i of $\{t_1, \dots, t_k\}$ occurs at least n_i times, then we say $\{n_1.t_1, \dots, n_k.t_k\}$ extended-repeated reveals Y denoted $\{n_1.t_1, \dots, n_k.t_k\} \rightarrow Y$ iff $\forall C \in C_{max}$

$$\bigwedge_{t_i \in \{t_1, \dots, t_k\}} (|C \cap E_{t_i}| \geq n_i) \implies \bigvee_{t \in Y} (C \cap E_t \neq \emptyset).$$

Note that $\{n_1.t_1, \dots, n_k.t_k\}$ extended-repeated reveals Y is not defined if there is no maximal configuration in which each transition t_i of $\{t_1, \dots, t_k\}$ occurs at least n_i times.

Remark 5. Reveals, extended reveals and repeated reveals can be expressed by Def. 4.

$$\begin{aligned} t_1 \triangleright t_2 &\iff \{1.t_1\} \rightarrow \{t_2\}, \\ \{t_1, t_2\} \rightarrow \{t_3, t_4\} &\iff \{1.t_1, 1.t_2\} \rightarrow \{t_3, t_4\}, \\ n.t_1 \triangleright t_2 &\iff \{n.t_1\} \rightarrow \{t_2\}. \end{aligned}$$

Example 3. In the system net illustrated in Fig. 3, let us examine the relation between the transitions a, e and k . Neither a nor e reveals k alone. There is no extended reveals or repeated reveals

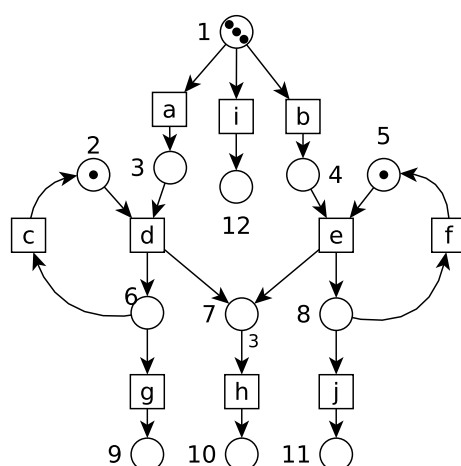


Figure 4: A bounded equal-conflict system.

relation between them either. However, there is still some information flow. Two occurrences of a together with two occurrences of e extended-repeated reveal k , i.e., $\{2.a, 2.e\} \rightarrow \{k\}$. This might cause a security violation in a system where the occurrence of k is supposed to be a secret.

Let us consider a case in which the total number of occurrences of a set of transitions gives information about another set of transitions. In other words, if the total number of occurrences of the transitions in the first set is more than a certain number, this implies that at least one transition of the second set must have occurred or will occur inevitably. The next relation defines such situation. If a set of low transitions collectively reveal some high transitions this might cause a security violation.

Definition 5. Let $n \geq 1$ and $X, Y \subseteq T$. If there is at least one maximal configuration in which the total number of occurrences of the transitions in set X is at least n , then we say X n -collective reveals Y , denoted $n.X \rightarrow Y$, iff $\forall C \in C_{max}$

$$\sum_{t \in X} |C \cap E_t| \geq n \implies \bigvee_{t \in Y} (C \cap E_t \neq \emptyset).$$

Note that X n -collective reveals Y is not defined if there is no maximal configuration in which the total number of occurrences of the transitions in set X is at least n .

Remark 6. Reveals and repeated reveals can be expressed by Def. 5.

$$t_1 \triangleright t_2 \iff 1.\{t_1\} \rightarrow \{t_2\},$$

$$n.t_1 \triangleright t_2 \iff n.\{t_1\} \rightarrow \{t_2\}.$$

Example 4. In the net in Fig. 4, if the total number of occurrences of d and e is at least 3, this implies the occurrence of h , i.e., $3.\{d, e\} \rightarrow \{h\}$.

In the net in Fig. 5, if the total number of occurrences of b and c is at least 2, this implies occurrence of either e or f , i.e., $2.\{b, c\} \rightarrow \{e, f\}$.

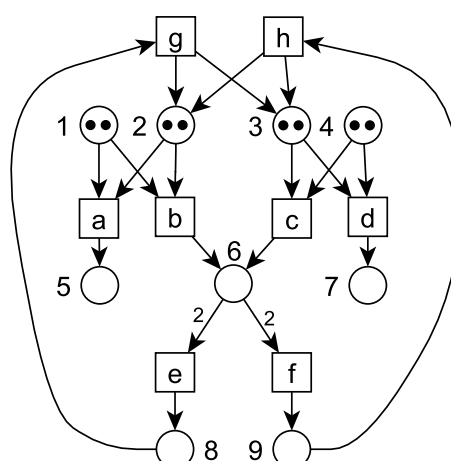


Figure 5: A bounded equal-conflict system.

4. Computing information flow on bounded equal-conflict nets

In this section, we consider bounded equal-conflict PT systems and show how to compute the relations introduced in Sec. 3 by exploiting the equal-conflict structure of the net.

The relations in Sec. 3 consider the unfolding of a system $\text{Unf}(\Sigma)$ and analyse their maximal configurations checking footprints, the multisets of transitions occurring in maximal configurations. Here, we consider maximal-step semantics, since, as recalled in Sec. 2, it is equivalent to step semantics in the case of equal-conflict systems, where there is no asymmetric confusion [27].

Moreover, thanks to the results in [22], it is possible to show a strict relation between footprints of step sequences of a system Σ and footprints of configurations of $\text{Unf}(\Sigma)$.

In fact, in [22], the authors studied the relationships between various classes of non-sequential processes (runs of an unfolding), occurrence sequences and step sequences. They showed that, in the special case of countable, T-restricted nets of finite synchronisation (i.e., where any transition has a finite set of pre- and post-places) with finite initial marking, and then also in the case of the nets here considered, the distinction between occurrence sequences and step sequences disappears. Moreover, for the same class of nets, they showed that it is possible to consider equivalence relations both on the set of occurrence sequences and on the set of processes such that there is a bijection between the induced equivalence classes.

In the case of occurrence sequences, the equivalence relation abstracts w.r.t. the total order arbitrarily chosen when transitions are concurrent; in the case of processes, the equivalence abstracts from the distinction among several tokens on the same place.

From the construction of the equivalence classes both of occurrence sequences and of processes, it is possible to observe that the elements in each class have the same footprints, i.e., the same multiset of transitions which can be observed through the corresponding behaviour, and that elements of an equivalence class are in bijective correspondence with a class of the other sort

have the same footprint too.

Given this correspondence, in the next subsections we introduce a tree, whose paths record the set of maximal-step sequences of Σ , and show that a particular prefix of this tree is sufficient to compute the reveals relations. Note that since we are interested in maximal configurations, we cannot directly work on the marking graph of the net, since its maximal paths are not necessarily associated with maximal configurations in the unfolding.

The tree and its prefix were first introduced in [18] for 1-safe free-choice Petri nets, here we adapt them to the class of bounded equal-conflict PT systems, to study their information flow.

4.1. The tree of maximal-step computations

Definition 6. Let $\Sigma = (P, T, F, W, m_0)$ be a bounded equal-conflict PT system. We define the tree of maximal-step computations, denoted as $\text{msct}(\Sigma)$ or *msc-tree*, as follows.

- Each node of the tree is labelled with a reachable marking.
- The root of the tree is labelled with the initial marking m_0 .
- From each node labelled with marking m , for each maximal step U enabled at m , there is an outgoing arc, labelled with U , and leading to a node labelled with m' , where $m[U]m'$.

A path in $\text{msct}(\Sigma)$ is a sequence $\pi = v_1 U_1 v_2 U_2 \dots$, such that, for each i , there is an arc from v_i to v_{i+1} , labelled with U_i . A path is *initial* if it starts in the root of the tree. A path can be either finite or infinite. Let v, r be two nodes on the same path π , $v < r$ iff v is closer to the root than r . The footprint of the path π is the (multiset) union of all steps occurring in π .

We define an inclusion relation between footprints: let π_1 and π_2 be paths on $\text{msct}(\Sigma)$; then $\lambda(\pi_1) \subseteq \lambda(\pi_2)$ iff $\forall t \in T, \lambda(\pi_1)(t) \leq \lambda(\pi_2)(t)$, i.e., for each transition t , the number of occurrences in the footprint of π_1 is less than or equal to that in the footprint of π_2 .

To simplify the notation, we will write $t \in \lambda(\pi)$ when $\lambda(\pi)(t) \geq 1$, namely when t occurs in some step of path π .

From the considerations in the first part of Sec. 4, each maximal path in $\text{msct}(\Sigma)$ can be associated to at least one maximal configuration of the unfolding, with the same footprint; and vice versa, each configuration of $\text{Unf}(\Sigma)$ is represented in the $\text{msct}(\Sigma)$ by a path with the same footprint.

Example 5. Fig. 6 represents a PT system and a prefix of its maximal computation tree. In the tree, the markings are represented as multisets of places. In each marking, if a place appears only once, its multiplicity is omitted; if a place p appears n times, for the sake of clarity in the figure, it is denoted with p^n . For example, the initial marking of the net is represented with $1, 6^2$. Note that the PT system is the same of Ex. 1, and in Fig. 2 a branching process of Σ is shown.

Let $V = \{v_1, v_2, \dots, v_n, \dots\}$ be the set of nodes in $\text{msct}(\Sigma)$, and $\mu : V \rightarrow [m_0]$ the labelling function, mapping to each node the corresponding marking. Two nodes v_1 and v_2 are equivalent if $\mu(v_1) = \mu(v_2)$. Let $\pi_1 = v_1 U_1 v_2 U_2 \dots v_n U_n \dots$ and $\pi_2 = v'_1 U'_1 v'_2 U'_2 \dots v'_n U'_n \dots$ be two paths on the tree, where the elements v_j represent nodes of the tree, and the elements U_j represent the labels of the arcs. The path π_1 is isomorphic to the path π_2 (in symbols $\pi_1 \simeq \pi_2$) iff for each j , $\mu(v_j) = \mu(v'_j)$, and $\lambda(U_j) = \lambda(U'_j)$. Finally, two subtrees L_1 and L_2 are isomorphic iff for each maximal path π on L_1 there is a maximal path π' in L_2 such that $\pi \simeq \pi'$, and vice versa.

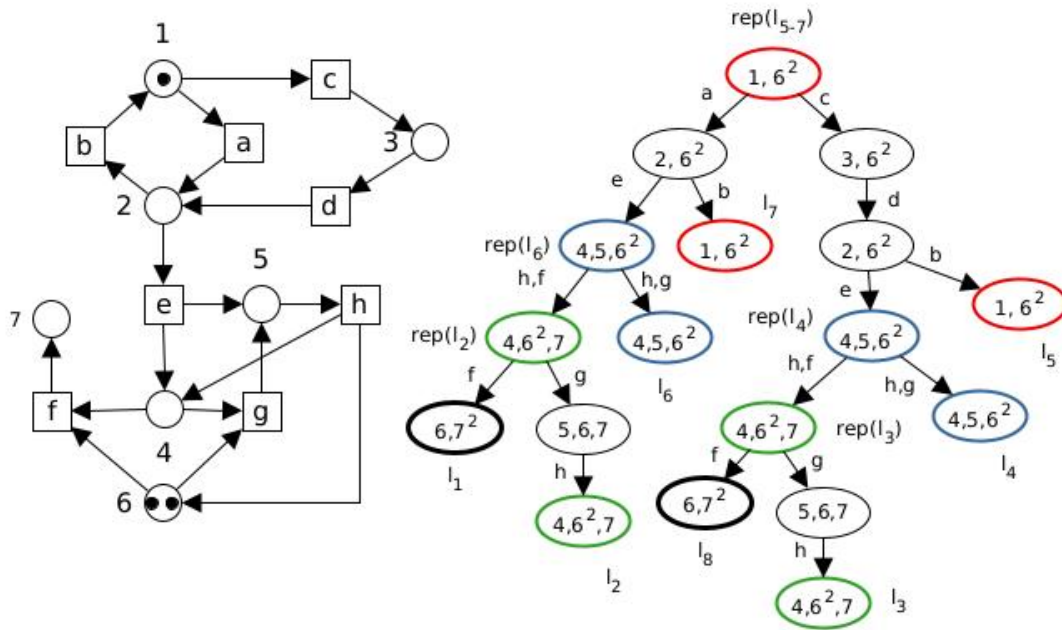


Figure 6: A bounded equal-conflict PT system and a prefix of its associated maximal-step computation tree.

Let π be a maximal path on the tree, and v_n one of its nodes. We denote with $\downarrow \pi(v_n)$ the path from the root to v_n , and with $\uparrow \pi(v_n)$ the subpath π starting from v_n .

Lemma 1. *Let v_n and v_k be two nodes of an msc-tree such that $\mu(v_n) = \mu(v_k)$. The subtree with v_n as root and the one with v_k as root are isomorphic.*

Proof. Since $\mu(v_n) = \mu(v_k)$, the arcs leaving from v_n and from v_k have the same labels. Then, by construction the children of v_n are equivalent to the children of v_k , and the same reasoning can be applied to each of them. \square

On the maximal paths of the msc-tree, we can define a *peeling* operation as follows [18, 17]. Let $\pi = v_1 U_1 v_2 U_2 \dots$ be a maximal path on the msc-tree, v_n and v_k be two nodes in π with $n < k$ and $\mu(v_n) = \mu(v_k)$. Let $\pi_{n,k}$ be the subpath between v_{n+1} and v_k . The peeling of π with respect to $\pi_{n,k}$ is the path $\pi' = \text{peel}(\pi, \pi_{n,k})$ such that $\downarrow(\pi(v_n)) = \downarrow(\pi'(v_n))$, $\pi_{n,k}$ is deleted from π , and $\uparrow(\pi(v_k))$ is equivalent to $\uparrow(\pi'(v_n))$. In words, peeling π means to consider the execution in which the cycle $\pi_{n,k}$ has not been executed. The path π' constructed in this way is also maximal in the msc-tree.

4.2. The full prefix of the maximal-step computation tree

In this section we recall the definition of full prefix given in [18] and some results that will be useful when presenting the algorithm.

Definition 7. Let Σ be a bounded equal-conflict PT system. The full prefix of $\text{msct}(\Sigma)$, denoted $\text{fp}(\Sigma)$, is a labelled rooted tree defined by the following clauses.

1. The root of $\text{fp}(\Sigma)$ is the root of $\text{msct}(\Sigma)$.
2. Let v be a node of $\text{fp}(\Sigma)$. If there is no v' such that $v' < v$ and $\mu(v') = \mu(v)$, then all the children of v in the $\text{msct}(\Sigma)$ are nodes in $\text{fp}(\Sigma)$; otherwise v is a leaf in $\text{fp}(\Sigma)$.

Example 6. The tree in Fig. 6 is a full prefix. Each node is labelled with a marking, and each arc is labelled with a multiset of transitions. The leaves are either deadlocks, and in this case they are denoted with a black thick line, or repeated markings, and in this case the leaves are in the same colour of the node that they repeat. Moreover, some nodes have a second label: in particular, the leaves are labelled with l_i , $i \in \{1, \dots, 8\}$, and, for each leaf corresponding to a repeated marking labelled with l_i , its repetition is labelled $\text{rep}(l_i)$.

The full prefix $\text{fp}(\Sigma)$ is finite, since the length of a path cannot exceed the number of markings of the net system, and every node has a finite number of children.

For each maximal path π on $L = \text{fp}(\Sigma)$, we will denote with $l(\pi)$ the leaf of the path, and, if $l(\pi)$ is not a deadlock, we will denote as $\text{rep}(l(\pi))$ the node preceding $l(\pi)$ and such that $\mu(l(\pi)) = \mu(\text{rep}(l(\pi)))$. In addition, we will denote as $L_{\text{rep}(l(\pi))}$ the subtree of L with root $\text{rep}(l(\pi))$.

Lemma 2. Let π be a path on $\text{msct}(\Sigma)$, and $t \in T$ a transition. If $n \leq \lambda(\pi)(t)$, but $t \notin \pi \cap \text{fp}(\Sigma)$, then there is another path $\pi' \in \text{msct}(\Sigma)$ such that $\lambda(\pi') \subseteq \lambda(\pi)$, $n \leq \lambda(\pi')(t)$, and $t \in \pi' \cap \text{fp}(\Sigma)$.

Proof. Let $l(\pi_1)$ be the leaf of $\pi_1 = \pi \cap \text{fp}(\Sigma)$ and $\text{rep}(l(\pi_1))$ its repetition in the prefix. We can construct the run $\pi'_1 = \text{peel}(\pi, \pi_{\text{rep}(l(\pi_1))}, l(\pi_1))$. If $t \in \pi_2 = \pi'_1 \cap \text{fp}(\Sigma)$ we can stop the construction, otherwise we repeat the procedure by considering $l(\pi_2)$. Since the distance between the root and the first occurrence of t is finite, and every step of the peeling reduces it, after a finite number of peeling operations the thesis must be satisfied. \square

Let π' be a maximal path on $\text{msct}(\Sigma)$, and π its prefix on $\text{fp}(\Sigma)$. If $l(\pi)$ is not a deadlock, there must be a prefix of π' extending π with a segment isomorphic to another segment in $\text{fp}(\Sigma)$ starting from $\text{rep}(l(\pi))$ and arriving to a leaf of $\text{fp}(\Sigma)$. Let π_1 be such a segment; we denote with $\pi + \pi_1$ the prefix of π' obtained by concatenating π with a segment isomorphic to π_1 . If $l(\pi_1)$ is not a deadlock, then $\pi + \pi_1$ can be extended further by looking in $\text{fp}(\Sigma)$ which segments start from $\text{rep}(l(\pi_1))$. Among them, there must be a segment π_2 such that $\pi + \pi_1$ can be extended with a segment isomorphic to π_2 , obtaining a longer prefix of π' : $\pi + \pi_1 + \pi_2$.

Proceeding in this way, we can obtain an arbitrarily long prefix of π' from $\text{fp}(\Sigma)$.

4.3. An algorithm for the collective reveals relations

In this section, we propose an algorithm to compute the collective reveals relation, defined in Def. 5, on a bounded equal-conflict PT system Σ , through the full prefix $\text{fp}(\Sigma)$. Its pseudo-code is presented in Algorithm 1. Since reveals (Def. 1) and repeated reveals (Def. 3) can be expressed as special cases of collective reveals (see Remark 6), the algorithm allows to compute also

these relations. At the end of the section, we will discuss how to modify it to compute also extended-repeated reveals (Def. 4).

Algorithm 1 takes as input the full prefix $L = \text{fp}(\Sigma)$, two sets of transitions, X and Y , and a positive number n . If there is no path in the $\text{msct}(\Sigma)$ with at least n occurrences of transitions of X , then the algorithm returns *undefined*. Otherwise, it returns *true* if $n.X \rightarrow Y$, *false* if $n.X \not\rightarrow Y$. The main function of the algorithm is *repex*. The variable ‘Paths’ includes all the prefixes of the paths of $\text{msct}(\Sigma)$ that we need to check to verify the relation. Initially, it includes all the paths in $\text{fp}(\Sigma)$ with at least an occurrence of a transition of X (this is justified by Lemma 2). The variable ‘empty’ is true if a path with at least n occurrences of transitions of X has not been found, false otherwise. ‘Paths’ and ‘empty’ are the input arguments of the function *extendPaths*. This function checks which input paths already include at least n occurrences of transitions of X , and, if it finds a path with n occurrences of X and none of Y , it sets the value of ‘stop’ to true, and stops the execution. In this case, also the main function will stop, and return false, since the path could be extended to a maximal path of $\text{msct}(\Sigma)$ without adding any new transition label. If the path has at least n occurrences of X and an occurrence in Y , then it does not need to be elongated further, and the algorithm must continue with the analysis of the other paths.

Let π be a path with less than n occurrences of X ; then the algorithm needs to check which extensions could be useful to add more occurrences of transitions of X . If π ends with a deadlock, it is not possible to extend it further, and since there are no n occurrences of X we are not interested in it. Otherwise, $\text{rep}(l(\pi))$ is well defined, and all the paths extending π are labelled as one of the paths starting from $\text{rep}(l(\pi))$. Let *ext* be any path starting from $\text{rep}(l(\pi))$. If *ext* does not have any occurrence of X and ends with a deadlock, then we do not need to consider it, since the concatenation $\pi + \text{ext}$ of π and *ext* does not have n occurrences of X . Also, we do not consider the extension of π with no occurrence of X , and such that $\text{rep}(l(\pi)) \leq \text{rep}(l(\text{ext}))$, since each path with such a prefix and n occurrences of X can be peeled removing *ext* and still have n occurrences of X . All the other extensions are put in the variable ‘newPaths’, and will be considered in the next round. If there are no paths left to analyse, and in all those with at least n occurrences of X there is an occurrence in Y , then the algorithm returns true.

Example 7. Consider the net in Fig. 6 and its full prefix. Assume that we want to check the relation $2.\{a, c\} \rightarrow \{b\}$. We can easily see that the relation is verified. The first part of the relation is satisfied if we observe a or c at least twice, or both a and c once. In all these cases, b must have occurred to bring the token back in 1. We simulate some steps of the algorithm to see how to arrive at this conclusion. First, we need to consider all the paths with at least an occurrence of a or of c . In this case, these are all the paths of the full prefix. Since none of them has two occurrences, we need to check the possible extensions for all of them. The two paths ending in a deadlock cannot be extended further, therefore we can stop to consider them. In what follows, if a transition has multiplicity 1 in a footprint, we will omit the 1. Consider the path with footprint $\{a, e, 2.h, f, g\}$. Its possible extensions are isomorphic to the segments starting with $\text{rep}(l_2)$: these have labels $\{f\}$ and $\{g, h\}$. In none of them an occurrence of a or c appears; the segment labelled $\{f\}$ ends in a deadlock, and the leaf of the segment labelled $\{g, h\}$ (l_2) coincides with $\text{rep}(l_2)$, therefore none of these extensions is useful to observe a second occurrence in $\{a, c\}$ and we can discard the path. Analogously for the path with footprint $\{c, d, e, f, g, 2.h\}$. A similar reasoning

Algorithm 1 Computing collective reveals

```

function repex( $L$ : full prefix,  $X, Y \subseteq T, n \in \mathbf{N}$ )  $\in \{\mathbf{true}, \mathbf{false}, \mathbf{undefined}\}$ 
  Paths = The list of maximal paths in  $L$  with at least an element of  $X$ 
  empty = True
  while Paths  $\neq []$ 
    Paths, empty, stop = extendPaths(Paths, empty)
    if stop == true
      return false
    end if
  end while
  if empty == true
    return undefined
  else
    return true
  end if
end function

function extendPaths(Paths, empty)
  # returns a triple ( $x, y, z$ ), where  $x$  is a list of paths, and  $y$  and  $z$  are boolean values
  newPaths = []
  for  $\pi \in$  Paths
    if  $|\pi \cap X| \geq n$ 
      empty = false
      if  $Y \cap \pi == \emptyset$ 
        return [], false, true
      end if
    else if  $l(\pi)$  is not a deadlock
      for  $ext \in L_{rep}(l(\pi))$ 
        if  $ext \cap X \neq \emptyset \vee (rep(l(ext)) < rep(l(\pi)))$ 
          newPaths.append( $\pi + ext$ )
        end if
      end for
    end if
  end for
  return newPaths, empty, false
end function

#  $|\pi \cap X|$  is equivalent to  $\sum_{t \in X} \lambda(\pi)(t)$ 

```

can be done for the paths with footprints $\{a, e, h, g\}$ and $\{c, d, e, h, g\}$: in these cases, the possible extensions start from the nodes $rep(l_6)$ and $rep(l_4)$ respectively, but none of them has an other occurrence of a or c , and the repetition of the non-deadlock leaves coincides or follows these nodes. Hence, the only paths that we can extend are those ending with the nodes l_7 and l_5 ($\{a, b\}$ and $\{c, d, b\}$ respectively). The extensions of these paths start from the root, therefore in all of them

there is an occurrence of a or one of c , and these extended paths have two occurrences in $\{a, c\}$. Since in all of them there is already an occurrence of b , we can stop, and conclude $2.\{a, c\} \rightarrow \{b\}$.

Lemma 3. *The leaf of every path constructed by the algorithm is a deadlock, or is associated with a marking that is already present in the path.*

Proof. First we observe that every path constructed by the algorithm ends with a node equivalent to a leaf in the prefix tree. For each leaf, the path in the tree starting from the root and arriving to it is unique.

Let r be the root of the tree, $\pi_0\pi'_1\dots\pi'_n$ be a path constructed by the algorithm, where π_0 is a maximal path in the prefix tree and π'_i is an added segment isomorphic to a segment π_i starting from $\text{rep}(l(\pi_{i-1}))$ (the repetition of the leaf $l(\pi_{i-1})$ of the segment π_{i-1}) and ending in $l(\pi_i)$, a leaf of the prefix tree. We have to prove that, if the leaf of the constructed path is not a deadlock, then the repetition $\text{rep}(l(\pi'_n))$ of the leaf $l(\pi'_n)$ of the path belongs to the path itself, i.e., $\text{rep}(l(\pi'_n))$ is in $\pi_0\pi'_1\dots\pi'_n$.

We prove it by induction. Let $l(\pi'_1)$ be the leaf of $\pi_0\pi'_1$; if it is not a deadlock, then $\text{rep}(l(\pi'_1))$ is either in π'_1 or inside $[r, \text{rep}(l(\pi_0))]$, where $[r, \text{rep}(l(\pi_0))]$ is the path from the root r to the repetition of the leaf of π_0 and then it is contained in π_0 . Then $\text{rep}(l(\pi'_1)) \in \pi_0\pi'_1$.

We now assume the constructed path $\pi_0\pi'_1\dots\pi'_i$ ends either with a deadlock, or with a node whose repetition $\text{rep}(l(\pi'_i))$ is either in π'_i or in the segment $[r, \text{rep}(l(\pi'_{i-1}))]$, which is contained in $\pi_0\pi'_1\dots\pi'_{i-1}$, and then $\text{rep}(l(\pi'_i)) \in \pi_0\pi'_1\dots\pi'_i$.

We prove that the path $\pi_0\pi'_1\dots\pi'_{i+1}$ ends either with a deadlock, or with a node whose repetition $\text{rep}(l(\pi'_{i+1}))$ is either in π'_{i+1} or in the segment $[r, \text{rep}(l(\pi'_i))]$, which is contained in $\pi_0\pi'_1\dots\pi'_i$, and therefore $\text{rep}(l(\pi'_{i+1})) \in \pi_0\pi'_1\dots\pi'_{i+1}$. In fact, π'_{i+1} is isomorphic to a segment π_{i+1} in the prefix tree starting from $\text{rep}(l(\pi_i))$, which is between r and $l(\pi_i)$, and ending in a leaf $l(\pi_{i+1})$ of the prefix tree. This last leaf is either a deadlock, or has a repetition, which is either in π_{i+1} or in $[r, \text{rep}(l(\pi_i))]$; since $\text{rep}(l(\pi'_i))$ is by inductive hypothesis in $\pi_0\pi'_1\dots\pi'_i$, we get the thesis. \square

Lemma 4. *Let π be any maximal path in the msc-tree. If π has in total at least n occurrences of transitions belonging to X , then there is at least a path π' analysed by the algorithm with in total at least n occurrences of transitions of X , such that $\lambda(\pi') \subseteq \lambda(\pi)$.*

Proof. We show that we can peel π and obtain a maximal path of the msc-tree such that its prefix is analysed. For Lemma 2, we can peel π and obtain a path π_1 such that at least an occurrence of X appears in its prefix in $\text{fp}(\Sigma)$. Let π'_1 be such prefix. If π'_1 has n occurrences of X , we don't need to proceed further; otherwise π'_1 must be followed in π_1 by a path isomorphic to a path starting from $\text{rep}(l(\pi'_1))$. Let π'_2 be this segment. If π'_2 has at least an occurrence of a transition of X , or $\text{rep}(l(\pi'_2))$ precedes $\text{rep}(l(\pi'_1))$, this elongation of the prefix has been considered by the algorithm. If π'_2 has no occurrences of X and $\text{rep}(l(\pi'_2)) \geq \text{rep}(l(\pi'_1))$, then we can peel π_1 of the part between $\text{rep}(l(\pi'_2))$ and $l(\pi'_2)$, obtaining π_1^2 . Since in π'_2 there are no elements of X , this cannot influence their number in π_1^2 , and $\lambda(\pi_1^2) \subseteq \lambda(\pi_1)$. The path π_1^2 has also a prefix made by π'_1 concatenated with a segment starting from $\text{rep}(l(\pi'_1))$. Let π'_3 be this segment. If $\text{rep}(l(\pi'_3)) \geq \text{rep}(l(\pi'_1))$, we repeat the peeling procedure. Since π_1 has n occurrence of X by hypothesis, after a finite number i of steps, we will obtain a peeled maximal run π_1^i such that

$\text{rep}(l(\pi'_i)) < \text{rep}(l(\pi'_1))$, with π'_i segment starting from $\text{rep}(l(\pi'_1))$ elongating the segment π'_1 , or π'_i has at least an occurrence of X . We can repeat this reasoning until obtaining a prefix with at least n occurrences of X . Since in our steps we never remove any of those, and π includes them by hypothesis, this procedure ends after a finite number of steps. By construction, all the transitions in the constructed prefix are also in π , therefore we produced a prefix as required from the thesis. \square

Theorem 1. *Algorithm 1 is correct.*

Proof. As first step we show that if the algorithm returns false, then $n.X \not\rightarrow Y$. The algorithm returns false if the variable ‘stop’ is true. The value of ‘stop’ is selected into the function *extendPaths*, and it is set to true if a path is found with n occurrences of X and none in Y . Each prefix is constructed so that the final leaf is a deadlock for the path, or it is repeated previously; in the first case the path is already maximal, in the second case, the path can be extended to a maximal path without adding any new transition by repeating infinitely often the segment between the leaf of the prefix and its repetition. The existence of such a repetition is guaranteed by Lemma 3. In both cases there is a maximal run with at least n occurrences of X and none in Y , therefore $n.X \not\rightarrow Y$.

We now show that if the algorithm returns true, then $n.X \rightarrow Y$. This is a consequence of Lemma 4: if there were a path with n occurrences of X and none in Y , Lemma 4 guarantees that we would analyse a prefix with the same feature, but if this happens, the algorithm returns false. If the algorithm returns undefined, then there cannot be any run in the msc-tree with n occurrences of transitions of X as a consequence of Lemma 4. \square

Theorem 2. *Algorithm 1 terminates.*

Proof. The algorithm ends when the variable ‘Paths’ becomes empty, or when a path with n occurrences of X and none in Y was found. We show that ‘Paths’ becomes empty after a finite number of steps. The variable ‘Paths’ is a list of prefixes of paths in the msc-tree. Its content in each iteration of the while loop is entirely determined by the function *extendPaths*, that elongate some of its elements. In particular, the function elongates the paths with less of n occurrences of X . Each path can be elongated in two ways: adding at least an additional occurrence of X , and this happens only finitely many times, since when the path has at least n occurrences of X it is not extended anymore, or with a segment whose repetition of the leaf is strictly closer to the root than the repetition of the previous leaf. Also in this second case the number of extension is finite, since the distance between each node and the root is finite. \square

Remark 7. *To compute the reveals relation, the algorithm needs to analyse only the maximal runs of the prefix tree, without further extensions. This is coherent with the result in [18], where the authors show that for 1-safe free-choice Petri nets, $\forall a, b \in T, a \triangleright b$ iff for each maximal path π in the prefix, if $a \in \pi$, then $b \in \pi$.*

Algorithm 1 can be adapted to compute the relation presented in Def. 4. Here we give only a sketch of how this can be done. Let $X = \{t_1, \dots, t_k\}$ and Y be the input sets. Instead of having just a single threshold n in input as for repeated reveals, the input must include all the thresholds $\{n_1, \dots, n_k\}$ related to transitions of X , and the information about how they are associated to

these transitions. Since the number of observations in which we are interested changes for every transition, when the algorithm needs to decide whether a path can stop or needs to be extended, it must consider all transitions of X separately, each with its threshold. In addition, if a path has already reached the number of required occurrences of a certain transition, we should stop to consider this transition as useful when we evaluate the possible extensions.

Since extended reveals (Def. 2) can be expressed as a special case of extended-repeated reveals, modifying the algorithm as described would allow for its computation.

5. Conclusion

In this paper, we have introduced two new variants of reveals relation, namely extended-repeated reveals and collective reveals. The relations are defined for transitions of general PT nets and they express positive information flow. Existence of reveals relation (or its variants) between transitions means that the occurrence of one gives information about the other one. It can violate security of a system when a low transition reveals a high transition. The new variants introduced in this paper are parametric and they allow one to specify security requirements of a distributed system based on the specific needs of the system. They provide scalability by allowing one to specify different levels of security.

Building upon the results from [18], we have provided a formal basis for information-flow analysis of distributed systems that are modelled with bounded equal-conflict PT nets. We have adapted the formalisation of maximal-step computation tree and its full prefix to bounded equal-conflict PT systems. We have shown that maximal-step computation tree represents the behaviour of a bounded equal-conflict PT system under maximal-step semantics and its full prefix forms an adequate basis for analysis of positive information flow by computing reveals relation and its variants. We have provided an algorithm to compute collective reveals on the full prefix and shown how to adapt the algorithm for computing extended-repeated reveals. The methods provided in this paper cover the computation of all reveals variants and can be used to perform information-flow analysis on bounded equal-conflict PT systems.

One of our next steps will be performing a complexity analysis of the proposed methods and working on improving the efficiency. This includes investigation of a shorter prefix and more efficient algorithms. We plan to work on extending our results to more general classes of Petri nets such as unbounded equal-conflict PT systems. We will explore the practical use of our methods for both information-flow analysis and verification of other desired behavioural properties of complex distributed systems. We will also explore different approaches to information-flow analysis on Petri nets by considering games like in [29, 30].

6. Acknowledgements

This work is partially supported by the Italian MUR.

References

- [1] J. A. Goguen, J. Meseguer, Security policies and security models, in: Proc. IEEE Symp. on Secur. and Privacy, 1982, pp. 11–20.
- [2] D. Sutherland, A model of information, in: Proc. 9th Nat. Comput. Sec. Conf., volume 247, 1986, pp. 175–183.
- [3] D. McCullough, Specifications for multi-level security and a hook-up, in: Proc. IEEE Symp. on Secur. and Privacy, 1987, pp. 161–161. doi:10.1109/SP.1987.10009.
- [4] R. Focardi, R. Gorrieri, A taxonomy of security properties for process algebras, Journal of Computer Security 3 (1995) 5–34.
- [5] J. McLean, A general theory of composition for a class of "possibilistic" properties, IEEE Transactions on Software Engineering 22 (1996) 53–67. doi:10.1109/32.481534.
- [6] G. Boudol, I. Castellani, Noninterference for concurrent programs and thread systems, Theor. Comput. Sci. 281 (2002) 109–130. URL: [https://doi.org/10.1016/S0304-3975\(02\)00010-5](https://doi.org/10.1016/S0304-3975(02)00010-5). doi:10.1016/S0304-3975(02)00010-5.
- [7] H. Mantel, A uniform framework for the formal specification and verification of information flow security, Ph.D. thesis, Saarland University, Saarbrücken, Germany, 2003. URL: <http://scidok.sulb.uni-saarland.de/volltexte/2004/202/index.html>.
- [8] H. Mantel, Information flow and noninterference, in: H. C. A. van Tilborg, S. Jajodia (Eds.), Encyclopedia of Cryptography and Security, 2nd Ed, Springer, 2011, pp. 605–607. URL: https://doi.org/10.1007/978-1-4419-5906-5_874. doi:10.1007/978-1-4419-5906-5_874.
- [9] N. Busi, R. Gorrieri, A survey on non-interference with Petri nets, in: Lectures on Concur. and Petri Nets, Advances in Petri Nets, volume 3098 of LNCS, Springer, 2003, pp. 328–344.
- [10] N. Busi, R. Gorrieri, Structural non-interference in elementary and trace nets, Math. Struct. Comput. Sci. 19 (2009) 1065–1090.
- [11] E. Best, P. Darondeau, R. Gorrieri, On the decidability of non interference over unbounded Petri nets, in: K. Chatzikokolakis, V. Cortier (Eds.), Proc. 8th SecCo, Paris, France, volume 51 of EPTCS, 2010, pp. 16–33. URL: <https://doi.org/10.4204/EPTCS.51.2>. doi:10.4204/EPTCS.51.2.
- [12] P. Baldan, A. Carraro, Non-interference by unfolding, in: G. Ciardo, E. Kindler (Eds.), Proc. PETRI NETS 2014, Tunis, Tunisia, volume 8489 of LNCS, Springer, 2014, pp. 190–209. URL: http://dx.doi.org/10.1007/978-3-319-07734-5_11. doi:10.1007/978-3-319-07734-5_11.
- [13] F. Basile, G. De Tommasi, C. Sterle, Noninterference enforcement via supervisory control in bounded Petri nets, IEEE Trans. Automat. Contr. 66 (2021) 3653–3666. doi:10.1109/TAC.2020.3024274.
- [14] G. Kılınç, Formal Notions of Non-interference and Liveness for Distributed Systems, Ph.D. thesis, Univ. Milano-Bicocca, DISCo, Milano, Italy, 2016.
- [15] L. Bernardinello, G. Kılınç, L. Pomello, Non-interference notions based on reveals and excludes relations for Petri nets, ToPNoC 11 (2016) 49–70. URL: https://doi.org/10.1007/978-3-662-53401-4_3. doi:10.1007/978-3-662-53401-4_3.
- [16] S. Haar, Unfold and cover: Qualitative diagnosability for Petri nets, in: Proc. 46th IEEE Conf. Decis. Control, 2007.
- [17] S. Haar, C. Rodríguez, S. Schwoon, Reveal your faults: It's only fair!, in: Proc. 13th ACSD,

- Barcelona, Spain, 2013, pp. 120–129.
- [18] F. Adobbati, G. Kiliç Soylu, A. Puerto Aubel, A finite prefix for analyzing information flow among transitions of a free-choice net, *IEEE Access* 10 (2022) 38483–38501. URL: <https://doi.org/10.1109/ACCESS.2022.3165185>. doi:10.1109/ACCESS.2022.3165185.
- [19] E. Teruel, M. Silva, Liveness and home states in equal conflict systems, in: M. Ajmone Marsan (Ed.), *Application and Theory of Petri Nets 1993*, Springer Berlin Heidelberg, Berlin, Heidelberg, 1993, pp. 415–432.
- [20] T. Murata, Petri nets: Properties, analysis and applications, *Proceedings of the IEEE* 77 (1989) 541–580. doi:10.1109/5.24143.
- [21] U. Goltz, W. Reisig, The non-sequential behaviour of Petri nets, *Information and Control* 57 (1983) 125–147. URL: <https://www.sciencedirect.com/science/article/pii/S001995883800400>. doi:[https://doi.org/10.1016/S0019-9958\(83\)80040-0](https://doi.org/10.1016/S0019-9958(83)80040-0).
- [22] E. Best, R. Devillers, Sequential and concurrent behaviour in Petri net theory, *Theoretical Computer Science* 55 (1987) 87–136. URL: <https://www.sciencedirect.com/science/article/pii/0304397587900909>. doi:[https://doi.org/10.1016/0304-3975\(87\)90090-9](https://doi.org/10.1016/0304-3975(87)90090-9).
- [23] J. Engelfriet, Branching processes of Petri nets., *Acta Inf.* 28 (1991) 575–591. doi:10.1007/BF01463946.
- [24] J. Esparza, S. Römer, W. Vogler, An improvement of McMillan’s unfolding algorithm, *Formal Methods in System Design* 20 (2002) 285–310. doi:10.1023/A:1014746130920.
- [25] E. Smith, On the border of causality: Contact and confusion, *Theoretical Computer Science* 153 (1996) 245–270. URL: <https://www.sciencedirect.com/science/article/pii/0304397595001239>. doi:[https://doi.org/10.1016/0304-3975\(95\)00123-9](https://doi.org/10.1016/0304-3975(95)00123-9).
- [26] P. S. Thiagarajan, Elementary net systems, in: W. Brauer, W. Reisig, G. Rozenberg (Eds.), *Petri Nets: Central Models and Their Properties, Advances in Petri Nets 1986, Part I, Proceedings of an Advanced Course, Bad Honnef, Germany, 8-19 September 1986*, volume 254 of *Lecture Notes in Computer Science*, Springer, 1986, pp. 26–59. URL: <https://doi.org/10.1007/BFb0046835>. doi:10.1007/BFb0046835.
- [27] R. Janicki, P. E. Lauer, M. Koutny, R. Devillers, Concurrent and maximally concurrent evolution of nonsequential systems, *Theoretical Computer Science* 43 (1986) 213–238.
- [28] S. Balaguer, T. Chatain, S. Haar, Building tight occurrence nets from reveals relations, in: B. Caillaud, J. Carmona, K. Hiraishi (Eds.), *Proc. 11th ACSD, Newcastle Upon Tyne, UK, IEEE, 2011*, pp. 44–53. URL: <http://doi.ieeecomputersociety.org/10.1109/ACSD.2011.16>. doi:10.1109/ACSD.2011.16.
- [29] L. Bernardinello, G. Kiliç, L. Pomello, Weak observable liveness and infinite games on finite graphs, in: W. M. P. van der Aalst, E. Best (Eds.), *Application and Theory of Petri Nets and Concurrency - 38th International Conference, PETRI NETS 2017, Zaragoza, Spain, June 25-30, 2017, Proceedings*, volume 10258 of *Lecture Notes in Computer Science*, Springer, 2017, pp. 181–199. URL: https://doi.org/10.1007/978-3-319-57861-3_12. doi:10.1007/978-3-319-57861-3_12.
- [30] F. Adobbati, L. Bernardinello, L. Pomello, A two-player asynchronous game on fully observable Petri nets, *Trans. Petri Nets Other Model. Concurr.* 15 (2021) 126–149. URL: https://doi.org/10.1007/978-3-662-63079-2_6. doi:10.1007/978-3-662-63079-2_6.