Petri Net Security Checker: Structural Non-interference at Work

Simone Frau[1,⋆], Roberto Gorrieri[2] and Carlo Ferigato[3]

[1] Information Security, ETH Zurich, 8092 Zurich, Switzerland
email: simone.frau@inf.ethz.ch
[2] Dipartimento di Scienze dell'Informazione, Università di Bologna,
Mura A. Zamboni, 7, 40127 Bologna, Italy
email: gorrieri@cs.unibo.it
[3] Joint Research Centre - European Commission,
Via E. Fermi, 1, 21027 Ispra (VA), Italy
email: carlo.ferigato@jrc.it

Abstract. Structural non-interference is a semi-static technique defined over Petri nets to check the absence of illegal information flows. This paper presents the main algorithmic features of this new technique and its implementation in a software tool, called the Petri Net Security Checker.

1 Introduction

Non-interference has been defined in the literature as an extensional property based on some observational semantics: the high part of a system does not interfere with the low part if whatever is done at the high level produces *no visible effect* on the low part of the system. The original notion of non-interference in [10] was defined, using trace semantics, for system programs that are deterministic. Generalized notions of non-interference were then designed to include (nondeterministic) labeled transition systems and finer notions of observational semantics such as bisimulation (see, e.g., [14, 6, 13, 15, 8]). The security properties in this class are based on the dynamics of systems; they are defined by means of one (or more) equivalence check(s); hence, non-interference checking is as difficult as equivalence checking, a well-studied hard problem in concurrency theory.

One relevant property in this class is the bisimulation-based property *BNDC* (Bisimulation Non-Deducibility on Composition) proposed by Focardi and Gorrieri some years ago [6, 8] on a CCS-like [12] process algebra. *BNDC* basically states that a system R is secure if it is bisimilar to R in parallel with any high level process Π w.r.t. the low actions the two systems can perform.

Intuitively, the many definitions of non-interference that have been proposed try to capture the essence of information flow as an extensional property. On

⋆ Frau's work was mainly conducted at the Joint Research Centre of the European Commission, Ispra.

the contrary, one may think that there are clear physical reasons for the occurrence of an information flow, that can be better understood if one exploits a computational model where causality of actions and conflict among actions can be modelled directly. Indeed, this is not the case of labeled transitions systems, a typical example of an *interleaving* model, where parallelism is not primitive.

For this reason, in [1–3] Busi and Gorrieri have shown that these extensional non–interference properties can be naturally defined also on Petri Nets, in particular on Elementary Nets [5], a well-known model of computation where causality and conflict are primitive concepts. More interestingly, they address the problem of defining statically non-interference for Elementary Nets, by looking at the structure of the net systems under investigation:

- in order to better understand the relationship between a flow of information and the causality (or conflict) relation between the activities originating such a flow, hence grounding more firmly the intuition about what is an interference, and
- in order to find more efficiently checkable non-interference properties that are sufficient (sometimes also necessary) conditions for those that have already received some support in the literature, such as *BNDC*.

Structural non-interference is defined on the basis of the absence of particular places in the net. We identify two special classes of places: *causal places*, i.e., places for which there are an incoming high transition and an outgoing low transition; and, *conflict places*, i.e. places for which there are both low and high outgoing transitions. Intuitively, causal places represent potential source of interference because the occurrence of the high transition is a prerequisite for the execution of the low transition. Similarly, conflict places represent potential source of interference because if the low event is not executable, then we can derive that a certain high transition has occurred. The absence of causal and conflict places is clearly a static property that can be easily checked by a simple inspection of the (finite) net structure; interestingly enough, this is a sufficient condition to ensure *BNDC*.

In order to characterize more precisely *BNDC*, the notion of causal place and conflict place is slightly refined, yielding the so-called *active* causal place and *active* conflict place. These new definitions are based also on a limited exploration of the state-space of the net (i.e. of its marking graph), hence, the absence of such places is not a purely structural property, rather a hybrid property. When active causal and active conflict places are absent, we get a property, called *Positive Place–Based Non–Interference* (*PBNI+* for short), which turns out to be equivalent to *BNDC* (proof in [3]). This result is rather surprising because the two properties are defined in a very different way.

1.1 Contribution of this paper

In this paper, we investigate the algorithmic properties of *PBNI+*. First we show that, given an elementary net with p places, n transitions and f arcs,

the complexity of checking for the absence of potential causal/conflict places is $O(f+p)$. Then, once singled out potential causal/conflict places, the check that such a potential place is active takes $O(pn2^{3p})$ in the worst case, because it is necessary to build the whole marking graph (that is exponential in the size of the net). Therefore, depending on the shape of the net, the complexity of *PBNI+* varies in the range between $O(f+p)$ and $O(pn2^{3p})$.

It is interesting to observe that *BNDC* was proved to be decidable in [11] over labeled transitions systems with an algorithm that is *exponential* in the number of the states. Even if the two models are different and so a comparison may be unfair, we point out that our procedure for deciding *BNDC* is *cubic* in the number of states of the marking graph of the net, which in turn is exponential in the number of the places of the net.

These algorithms have been implemented in a software tool, called the Petri Net Security Checker (PNSC for short), which provides functionalities for creating, editing and executing Petri nets, as well as automatically detecting places that are potential/active and causal/conflict.

The paper is organised as follows. In Section 2 we recall the basic definitions about Elementary Net systems, the dynamic non-interference property *BNDC* and the structural property *PBNI+*. In Section 3 we discuss the complexity of checking *PBNI+*. In Section 4 we describe the details of the PNSC tool, its functionalities and its implementation, besides showing its application to a small case study. Finally, some conclusive remarks are drawn in Section 5.

2 Background

2.1 Elementary Net Systems

Here we introduce basic definitions about the class of Petri Nets we use. Some familiarity with Petri net terminology is assumed. More details in [5, 2].

Definition 1. *A transition system is a triple* $TS = (St, E, \rightarrow)$ *where*

- St *is the set of states*
- E *is the set of events*
- $\rightarrow \subseteq St \times E \times St$ *is the transition relation.*

In the following we use $s \xrightarrow{e} s'$ *to denote* $(s, e, s') \in \rightarrow$. *Given a transition* $s \xrightarrow{e} s'$, s *is called the* source, s' *the* target *and* e *the* label *of the transition. A* rooted *transition system is a pair* (TS, s_0) *where* $TS = (St, E, \rightarrow)$ *is a transition system and* $s_0 \in St$ *is the* initial state.

Definition 2. *An* elementary net *is a tuple* $N = (S, T, F)$, *where*

- S *and* T *are the (finite) sets of* places *and* transitions, *such that* $S \cap T = \emptyset$
- $F \subseteq (S \times T) \cup (T \times S)$ *is the flow relation, usually represented as a set of directed arcs connecting places and transitions.*

A subset of S is called a *marking*. Given a marking m and a place s, if $s \in m$ then we say that the place s contains a token, otherwise we say that s is empty.

Let $x \in S \cup T$. The *preset* of x is the set ${}^\bullet x = \{y \mid F(y,x)\}$. The *postset* of x is the set $x^\bullet = \{y \mid F(x,y)\}$. The preset and postset functions are generalized in the obvious way to set of elements: if $X \subseteq S \cup T$ then ${}^\bullet X = \bigcup_{x \in X} {}^\bullet x$ and $X^\bullet = \bigcup_{x \in X} x^\bullet$. A transition t is enabled at marking m if ${}^\bullet t \subseteq m$ and $t^\bullet \cap m = \emptyset$. The firing (execution) of a transition t enabled at m produces the marking $m' = (m \setminus {}^\bullet t) \cup t^\bullet$. This is usually written as $m[t\rangle m'$. With the notation $m[t\rangle$ we mean that there exists m' such that $m[t\rangle m'$.

An *elementary net system* is a pair (N, m_0), where N is an elementary net and m_0 is a marking of N, called *initial marking*. With abuse of notation, we use (S, T, F, m_0) to denote the net system $((S, T, F), m_0)$.

The set of *markings reachable from* m, denoted by $[m\rangle$, is defined as the least set of markings such that

- $m \in [m\rangle$
- if $m' \in [m\rangle$ and there exists a transition t such that $m'[t\rangle m''$ then $m'' \in [m\rangle$.

The set of *firing sequences* is defined inductively as follows:

- m_0 is a firing sequence;
- if $m_0[t_1\rangle m_1 \dots [t_n\rangle m_n$ is a firing sequence and $m_n[t_{n+1}\rangle m_{n+1}$ then also $m_0[t_1\rangle m_1 \dots [t_n\rangle m_n[t_{n+1}\rangle m_{n+1}$ is a firing sequence.

Given a firing sequence $m_0[t_1\rangle m_1 \dots [t_n\rangle m_n$, we call $t_1 \dots t_n$ a *transition sequence*. We use σ to range over transition sequences.

The *marking graph* of a net system N is the transition system

$$MG(N) = ([m_0\rangle, T, \{(m, t, m') \mid m \in [m_0\rangle \wedge t \in T \wedge m[t\rangle m'\})$$

A net is *transition simple* if the following condition holds for all $x, y \in T$: if ${}^\bullet x = {}^\bullet y$ and $x^\bullet = y^\bullet$ then $x = y$. A marking m contains a *contact* if there exists a transition $t \in T$ such that ${}^\bullet t \subseteq m$ and $not(m[t\rangle)$. A net system is *contact–free* if no marking in $[m_0\rangle$ contains a contact. A net system is *reduced* if each transition can occur at least one time: for all $t \in T$ there exists $m \in [m_0\rangle$ such that $m[t\rangle$. In the following we consider contact-free elementary net systems that are transition simple and reduced.

2.2 A Dynamic Non-interference Property: *BNDC*

Our aim is to analyse systems that can perform two kinds of actions: high level actions, representing the interaction of the system with high level users, and low level actions, representing the interaction with low level users. We want to verify if the interplay between the high user and the high part of the system can affect the view of the system as observed by a low user. We assume that the low user knows the structure of the system, and we check if, in spite of this, he is not able to infer the behavior of the high user by observing the low view of

the execution of the system. Hence, we consider nets whose set of transitions is partitioned into two subsets: the set H of high level transitions and the set L of low level transitions. To emphasize this partition we use the following notation. Let L and H be two disjoint sets: with (S, L, H, F, m_0) we denote the net system $(S, L \cup H, F, m_0)$.

Among the many non-interference properties defined by Focardi and Gorrieri in [6–8], here we consider *BNDC* (Bisimulation Non-Deducibility on Composition). To properly define it over Petri nets, we need some auxiliary definitions: the operations of parallel composition (in TCSP-like style [4]) and restriction (in CCS-like style [12]), as well as a notion of low-view bisimulation.

Definition 3. *Let $N_1 = (S_1, L_1, H_1, F_1, m_{0,1})$ and $N_2 = (S_2, L_2, H_2, F_2, m_{0,2})$ be two net systems such that $S_1 \cap S_2 = \emptyset$ and $(L_1 \cup L_2) \cap (H_1 \cup H_2) = \emptyset$. The parallel composition of N_1 and N_2 is the net system*

$N_1 \mid N_2 = (S_1 \cup S_2, L_1 \cup L_2, H_1 \cup H_2, F_1 \cup F_2, m_{0,1} \cup m_{0,2})$

Note that synchronization occurs over those (low or high) transitions that are shared by the two nets, i.e., a transition t that occurs both in N_1 and N_2 has preset (postset), in $N_1 \mid N_2$, given by the union of the disjoint presets (postsets) in N_1 and N_2, respectively.

Definition 4. *Let $N = (S, L, H, F, m_0)$ be a net system and let U be a set of transitions. The restriction on U is defined as $N \backslash U = (S, L', H', F', m_0)$, where:*

$L' = L \setminus U$
$H' = H \setminus U$
$F' = F \setminus (S \times U \cup U \times S)$

The non-interference property we are going to introduce is based on some notion of *low* observability of a system, i.e., what can be observed of a system from the point of view of low users. The low view of a transition sequence is nothing but the subsequence where high level transitions are discarded.

Definition 5. *Let $N = (S, L, H, F, m_0)$ be an elementary net system. The* low view *of a transition sequence σ of N is defined as follows:*

$$\Lambda_N(\varepsilon) = \varepsilon$$
$$\Lambda_N(\sigma t) = \begin{cases} \Lambda_N(\sigma)t & \text{if } t \in L \\ \Lambda_N(\sigma) & \text{otherwise} \end{cases}$$

Then, a variant of bisimulation [12] can be defined in such a way that only the low behaviour is considered.

Definition 6. *Let $N_1 = (S_1, L_1, H_1, F_1, m_{0,1})$ and $N_2 = (S_2, L_2, H_2, F_2, m_{0,2})$ be two net systems. A* low–view bisimulation *from N_1 to N_2 is a relation R on $\mathcal{P}(S_1) \times \mathcal{P}(S_2)$ such that if $(m_1, m_2) \in R$ then for all $t \in \bigcup_{i=1,2} L_i \cup H_i$:*

- *if $m_1[t\rangle m_1'$ then there exist σ, m_2' such that $m_2[\sigma\rangle m_2'$, $\Lambda_{N_1}(t) = \Lambda_{N_2}(\sigma)$ and $(m_1', m_2') \in R$*

- *if* $m_2[t\rangle m_2'$ *then there exist* σ, m_1' *such that* $m_1[\sigma\rangle m_1'$, $\Lambda_{N_2}(t) = \Lambda_{N_1}(\sigma)$ *and* $(m_1', m_2') \in R$

If $N_1 = N_2$ *we say that* R *is a low–view bisimulation on* N_1.

We say that N_1 *is* low–view bisimilar *to* N_2, *denoted by* $N_1 \overset{\Lambda}{\approx}_{bis} N_2$, *if there exists a low–view bisimulation* R *from* N_1 *to* N_2 *such that* $(m_{0,1}, m_{0,2}) \in R$.

Now we are ready to define *BNDC*.

Definition 7. *Let* $N = (S, L, H, F, m_0)$ *be a net system.* N *is BNDC iff for all high-level nets* $K = (S_K, \emptyset, H_K, F_K, m_{0,K})$: $N\backslash H \overset{\Lambda}{\approx}_{bis} (N \mid K)\backslash(H \setminus H_K)$.

The left-hand term $N\backslash H$ represents the system N when isolated from high-level users (hence, the low view of N in isolation), while the right-hand term expresses the low view of N interacting with the (common transitions of the) high environment K (note that the activities resulting from such interactions are invisible by the definition of low view equivalence). *BNDC* is a very intuitive property: whatever high level system K is interacting with N, the low effect is unobservable. However, it is difficult to check this property because of the universal quantification over high systems.

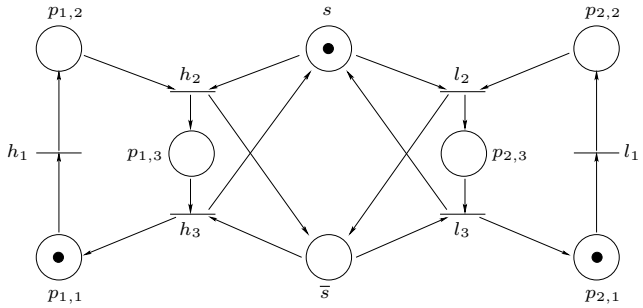

Fig. 1. The net system for a mutually exclusive access to a shared resource.

Example 1. As a simple case study and running example, consider the net in Figure 1, which represents a mutually exclusive access to a shared resource (represented by the token in s) by a high-user (left part of the net) and a low-user (right part of the net). Even if it might appear, at first sight, that the system is secure (and indeed, it is *BSNNI* (Bisimulation Strong Nondeterministic Non-Interference) [8]), actually it is not secure because a low level user can detect if a high-level user has deadlocked the system. Indeed, if the high-level user represented in the net K in Figure 2 wants to interact with the user in Figure 1, then a deadlock is reached after performing the sequence $h_1 h_2$ and the low level user can detect this because (s)he is not able to interact with the net. As a matter of fact, $BNDC$ is not satisfied, as K makes invalid the equivalence check in the definition of $BNDC$.

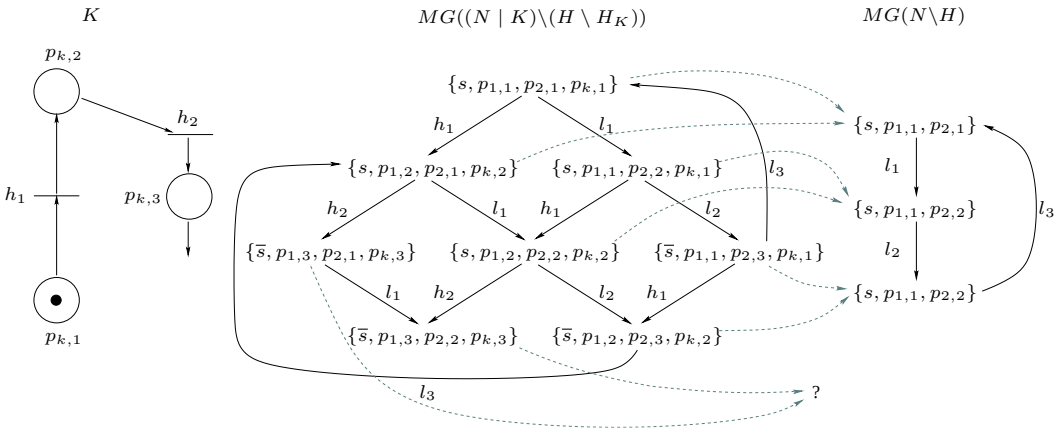

Fig. 2. The shared resource net system is not *BNDC*.

2.3 Structural Non-interference

Consider a net system $N = (S, L, H, F, m_0)$. Consider a low level transition l of the net: if l can fire, then we know that the places in the preset of l are marked before the firing of l; moreover, we know that such places become unmarked after the firing of l. If there exists a high level transition h that produces a token in a place s in the preset of l (see the system N_1 in Figure 3), then the low level user can infer that h has occurred if he can perform the low level transition l. We note that there exists a causal dependency between the transitions h and l, because the firing of h produces a token that is consumed by l. In this case we will say s is a potential causal place.

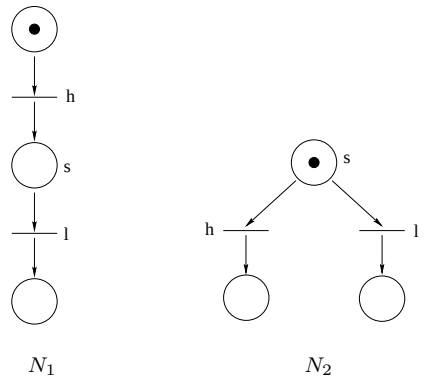

Fig. 3. Examples of net systems containing causal and conflict places.

Consider now the situation illustrated in the system N_2 of Figure 3: in this case, place s is in the preset of both l and h, i.e., l and h are competing for the use of the resource represented by the token in s. Aware of the existence of such a place, a low user knows that the high-level action h has been performed, if he is not able to perform the low-level action l. Place s represents a conflict between transitions l and h, because the firing of h prevents l from firing. In this case we will call s a potential conflict place.

In order to avoid the definition of a security notion that is too strong, and that rules out systems that do not reveal information on the high-level actions that have been performed, we need to refine the concepts illustrated above. In particular the potential causal place is an active causal place if there is an execution where the token produced by the high level transition is eventually consumed by the low level transition. Similarly, a potential conflict place is active if the token that could be consumed immediately by a high level transition can be later on also consumed by a low level transition. The formal definitions follow.

Definition 8. *Let $N = (S, L, H, F, m_0)$ be an elementary net system. Let s be a place of N such that $s^\bullet \cap L \neq \emptyset$ (i.e., a token in s can be consumed by a low transition).*

The place $s \in S$ is a potentially causal place *if ${}^\bullet s \cap H \neq \emptyset$ (i.e., a token in s can be produced by a high transition). A potentially causal place s is an* active causal place *if the following condition holds: there exist $l \in s^\bullet \cap L$, $h \in {}^\bullet s \cap H$, $m \in [m_0\rangle$ and a transition sequence σ such that $m[h\sigma l\rangle$ and $s \notin t^\bullet$ for all $t \in \sigma$.*

The place $s \in S$ is a potentially conflict place *if $s^\bullet \cap H \neq \emptyset$ (i.e., the token in s can be consumed also by a high transition). A potentially conflict place is an* active conflict place *if the following condition holds: there exist $l \in s^\bullet \cap L$, $h \in s^\bullet \cap H$, $m \in [m_0\rangle$ and a transition sequence σ such that $m[h\rangle$, $m[\sigma l\rangle$ and $s \notin t^\bullet$ for all $t \in \sigma$.*

Definition 9. *Let $N = (S, L, H, F, m_0)$ be an elementary net system. We say that N is PBNI+ (*positive Place Based Non-Interference*) if, for all $s \in S$, s is neither an active causal place nor an active conflict place.*

The following non-trivial result, proved in [3], states that the behavioural non-interference property *BNDC* is equivalent to the semi-static, structural property *PBNI+*.

Theorem 1. *Let $N = (S, L, H, F, m_0)$ be an elementary net system. Then N is PBNI+ iff N is BNDC.*

An obvious consequence is that if N has no *potentially causal* and *potentially conflict* places, then N is *BNDC*. Hence, a simple strategy to check if N is *BNDC* is to first identify potential causal/conflict places, a procedure that we show in the next section to be of complexity $O(f + p)$ in the size of the net (p is the number of places and f of arcs). If no place of these sorts is found, then N is *PBNI+*, hence *BNDC*. Otherwise, any such a candidate place should be better studied to check if it is actually an *active* causal/conflict place, a procedure that requires a limited exploration of the marking graph.

Observe that the net in Figure 1 of our running example is not *PBNI+* because place s is an active conflict (and also active causal) place.

3 *PBNI+* verification algorithms

Verification of *PBNI+* requires two separate steps: first, detection of potential causal places and potential conflict ones; then, checking if such places are active (causal/conflict) places.

We assume to use certain data structures. Precisely, a Net will be a structure containing an ordered list of *places*, an ordered list of *transitions* and a list of places (subset of the above mentioned *places* list) representing the *initial marking*. We also assume places in the *initial marking* list maintain the same order they have in the *places* list, so that all operations on sets of places (such as union, intersection and difference) can be done in $O(p)$.

Each place (each transition) has associated its own *preset* and *postset*, that are represented by lists of the opposite elements (transitions or places, respectively). As for the *initial marking* list in a Net, we will assume the nodes in the *preset* and *postset* lists appear in the same order they do in the lists they are taken from, so to be able to perform all operations on sets in linear time w.r.t. the number of nodes ($O(p)$ or $O(n)$, respectively). For convenience, we will assume a Net also contains a list of *arcs* (as specified by the flow relation F, $|F| = f$), thus that we can occasionally cycle on it in $O(f)$ time rather than $O(np)$ (inherent upper bound for f).

3.1 Potential places detection

Detecting potential places is a purely structural procedure, easy and computationally light-weight. Let us consider detection of potential causal places in a net N with p places and f arcs, and each place has three dedicated boolean variables for keeping track of the examined arcs: $highPre$, $lowPost$ and $highPost$. This consists of the following steps (each one annotated with an estimate of its worst-case computational cost):

- for each arc a in the net N: – $O(f)$ times
  - if a's source is a transition t – $O(1)$
    - if t is high then set $highPre$ of a's target as $true$ – $O(1)$
  - else
    - if t is low then set $lowPost$ of a's source as $true$ – $O(1)$
- for each place p in the net N: – $O(p)$ times
  - if p's $highPre$ and $lowPost$ are true, then add p to the set of computed potential causal places – $O(1)$

Detection of potential conflict places differs slightly, in that it will only set $highPost$ instead of $highPre$.

As all inner instructions cost $O(1)$, the final procedure cost will be the sum of the two cycles, namely $O(f + p)$.

3.2 Active places detection

Differently from the above, detection of active places is a complex (hence, also heavier) procedure because it has to analyze – though partially – the dynamic behaviour of the net.

First of all, we need a procedure to build the marking graph, i.e., the state space of the net. [1] We represent such a graph as a list of structures. Each of these structures is composed of a marking m, (where each marking is a set of places represented in the same fashion as the *initial marking* in Net), and of a list of pairs (t, m'), where t is an enabled transition and m' is the marking reached by firing t, i.e., $m[t\rangle m'$.

Under these modelling assumptions, the algorithm is composed of the following instructions (each annotated with an estimate of its worst-case, computational cost):

- create the marking list $list$ with the initial marking as its only element – $O(1)$
- for each marking m in $list$: – $O(2^p)$ times
  - for each transition t: – $O(n)$ times
    - if t is enabled at m: – $O(p)$
      - compute the marking m' reachable from m by firing t – $O(p)$
      - if m' is not already in $list$, add $(m', emptylist)$ to $list$ – $O(2^p)$
      - add to the list associated to m the new pair (t, m') – $O(1)$

The procedure acts mostly as a breadth-first visit: we add first the initial marking, and start a cycle exploring the graph. For each marking in the list, we compute the marking every enabled transition leads to and add the corresponding pair (enabled transition, reached marking) to the currently examined marking. When we meet a new marking, we add it to the queue and this will be examined later as the cycle proceeds.

Since the heaviest operation in the innermost cycle is checking whether the marking graph already contains a marking ($O(2^p)$), the procedure's cost is bound to the product of this by the weight of the nested cycles over the places and transitions of the net. Therefore the procedure's final cost will be $O(n2^{2p})$ [2].
Notice also that this procedure can take any marking as initial marking, which means that it can compute every possible subgraph rooted in the given marking. Furthermore, also a procedure for creating a marking graph restricted on a set of transitions can be easily obtained from the above. It is easy to see this trivially involves including only one more check and does not change computational costs.

[1] Notice that, since each marking is a set of places, the marking graph can contain up to 2^p states. Hence, the state space we are dealing with is inherently exponential in the number p of places.

[2] A further optimization could be using a search tree instead of a list for representing the *marking graph*, once an appropriate order on the places is introduced such that induces an order on the markings as well. That would reduce look up time from $O(2^p)$ to $O(p)$, and the whole procedure would cost $O(np2^p)$.

We can now introduce a procedure that searches for active causal places over the net. Intuitively, we do the following steps:

- find potentially causal places – $O(f + p)$
- for each place s among these: – $O(p)$ times
  - scan the markings in the marking graph, and single out only those which are reached through high transitions in ${}^\bullet s$. – $O(2^{2p})$
  - for each marking m among these: – $O(2^p)$ times
    - create a new marking graph rooted in m and restricted on all transitions containing s in their postset – $O(n2^{2p})$
    - search among its markings for one enabling any low transition in $s^\bullet$. If any is found, add s to the list of active causal places returned – $O(2^p)$

The active places so found perfectly comply with Definition 8. Indeed, for each potential causal place we single out all markings reached through a high transition h in ${}^\bullet s$, that is $m[h\rangle m'$. Then, for each of these, we create a marking graph rooted in m and restricted on all transitions belonging to ${}^\bullet s$. In such a marking graph every marking is reached through a sequence of transitions that do not produce new tokens in s ($s \notin t^\bullet$ for all $t \in \sigma$), therefore if we find one that enables a low transition $l \in s^\bullet$, we have $m'[\sigma l\rangle$, and hence $m[h\sigma l\rangle$.

The procedure is as heavy as computing the restricted marking graph ($O(n2^{2p})$) for each marking ($O(2^p)$) and each place in the net ($O(p)$), therefore it has a final cost of $O(pn2^{3p})$ (or $O(p^2n2^{2p})$ if the optimization in footnote 2 is implemented).

Finally, a procedure verifying *PBNI+* would just call the previous one and the one to detect active conflict places (which, intuitively enough, has same computational costs). Needless to say, procedure's final cost, in the worst case, is $O(pn2^{3p})$ as well. Note that, since the number of states is $O(2^p)$, the procedure for verifying *PBNI+* (hence *BNDC*) is actually *cubic* in the number of states.

Note, moreover, that in practice, the cost of checking *PBNI+* is much lower: (*i*) it might be the case that there are no potential causal/conflict places and so in this case the complexity is $O(f + p)$; (*ii*) the number of potential places is usually small w.r.t. to p; and, in particular, (*iii*) the number of reachable markings of the marking graph is generally much lower than 2^p.

4 Petri Net Security Checker

The tool, named *Petri Net Security Checker* (*PNSC* for short) [9], was written in Java [16], using the Eclipse development platform [17].

Figure 4 shows the tool's interface, which provides the user, in a single working environment, with different functionalities, that can be grouped into three main categories: editing, execution and net properties checking.

4.1 Editing

First of all, *PNSC* allows the user to create, save and open Petri nets. For these operations the tool uses the *Petri Net Markup Language* format [18],

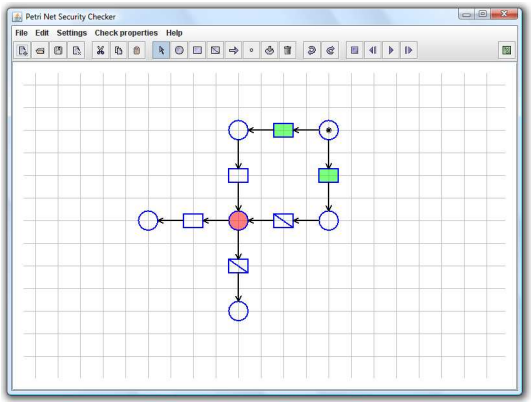

Fig. 4. Petri Net Security Checker main window

the standard format for Petri nets interchange, thus ensuring compatibility with external programs for further analysis of the nets (e.g. PIPE2 [19]).
By means of an intuitive toolbar, the user can draw the net. This includes basic operations as drawing places and (both high and low) transitions, draw arcs between them and set the initial marking of the net. Furthermore, it is possible to select portions of the net to carry out further operations as deleting, cutting, copying and pasting.
Being designed for incremental editing of nets in conjunction with checking their security, we developed different view modes to make comparison easier. In fact, nets can be cloned and placed side by side to be edited and compared concurrently, as shown in Figure 5.

In addition, the tool keeps track of all editing steps, so that each one can be undone/redone.

4.2 Execution

It is also possible to graphically simulate net executions (commonly referred to as *token game animation*).

The user can either fire one of the currently enabled transitions (highlighted in green as in Figure 4 and in Figure 5) by double clicking on it, or he can start a timed random execution, which consists of firing, at regular time intervals, a random transition among the enabled ones. In this case also, the tool keeps track of all firing steps, so that it is possible to step back (and, afterwards, forward) to previous (respectively, following) markings.

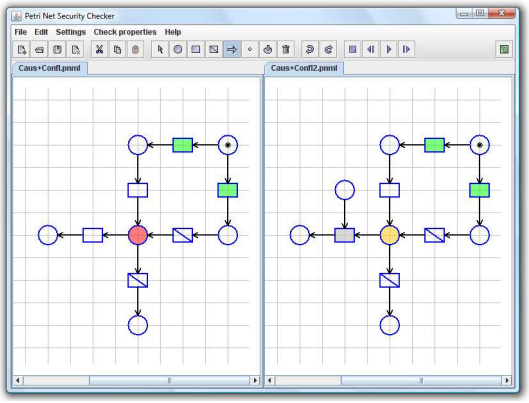

Fig. 5. Comparison mode

4.3 Properties check

Finally, the most distinctive functionalities of $PNSC$ pertain to the verification of the net's properties.

First, it is possible to check whether a net is simple, reduced and contact free[3]. Whenever one of these does not hold, all nodes that do not comply with it are highlighted in grey (as in Figure 6).

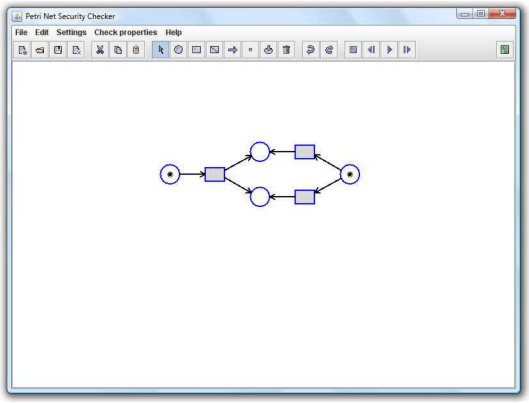

Fig. 6. A not contact-free net

[3] These checks are of minor complexity, namely $O(np^2)$, $O(n2^p)$ and $O(pn2^p)$, respectively.

The main functionality of our tool though is finding both potential and active causal/conflict places in the net, using the algorithms described in Section 3. When these checks are activated, potential causal/conflict places will be highlighted in orange, while active causal/conflict ones will be highlighted in red, as shown in Figure 7, which depicts the net already discussed in Figure 1.

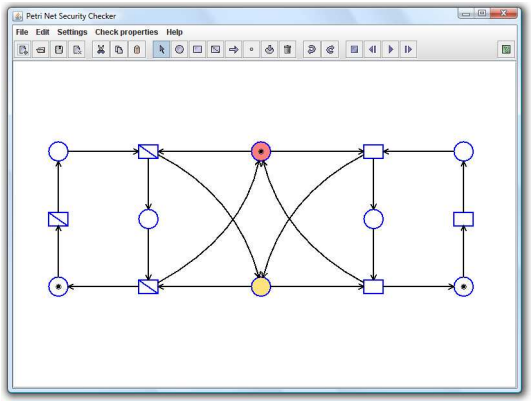

Fig. 7. Potential causal/conflict places will be highlighted in orange, active ones in red

Furthermore, for each potential/active place found it is possible to pinpoint and highlight the transitions and markings involved in the causality or conflict situation from the rest of the net, as in Figure 8, allowing a better inspection of the problem.

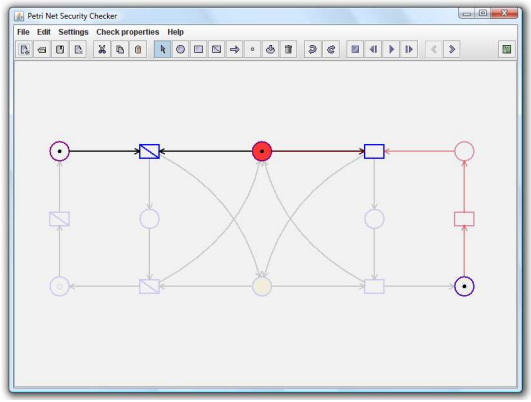

Fig. 8. Focus on the active conflict situation

5 Conclusion

In this paper we presented the tool *Petri Net Security Checker* for building Petri nets with transitions of two different confidentiality levels and check a structural security property on them, namely *PBNI+*.

The tool can actually check also some behavioural security properties, such as *SBNDC* and *BSNNI* [2]. Interestingly enough, *PBNI+* is proved to be equivalent to the behavioural property *BNDC* which is not obviously decidable; hence, the algorithms we presented in Section 3 offer a decidability proof for *BNDC* over Elementary Net Systems. The only paper we know offering a decidability result for *BNDC* is [11] where an exponential (in the number of states) procedure is presented for labeled transition systems. Our result is actually for a rather different model (unlabeled elementary net systems) and so it might be unfair to make a comparison; nonetheless, our decision procedure is cubic in the number of states of the marking graph (in turn exponential in the number of places).

We considered, for theoretical reasons in the implementation of our tool, only *Elementary Net Systems*, where places can contain at most one token. A natural generalization of this approach is to consider Place/Transition systems, where each place can contain more than one token. Such a class of nets is particularly interesting because the marking graph associated to a finite P/T net system may be infinite. In [3] Busi and Gorrieri claim that *PBNI+* can be easily defined also on this richer class of nets and checked in a finite amount of time, and keep on being the same as *BNDC* and *SBNDC* also for P/T net systems. This is particularly interesting because bisimulation is not decidable over P/T nets, hence *BNDC* as well as *SBNDC* are not checkable at all! This extension would also possibly provide the first result of decidability of a behavioural information flow security property, like *BNDC*, on a class of infinite state systems. It is likely that an extension of *PBNI+* to cover P/T nets also could be easily followed by a corresponding extension of the tool.

Acknowledgements

The authors would like to thank the anonymous referees for helpful comments.

References

1. N. Busi and R. Gorrieri. A Survey on Non-Interference with Petri Nets. *Advanced Course on Petri Nets 2003*, Springer LNCS 3098:328-344, 2004.
2. N. Busi and R. Gorrieri. Positive Non-Interference in Elementary and Trace Nets. Proc. 25th Int.l Conf. on Application and Theory of Petri Nets, Springer LNCS 3099:1-16, 2004.
3. N. Busi and R. Gorrieri. Structural Non-Interference in Elementary and Trace Nets. Accepted for publication in *Mathematical Structures in Computer Science*, 2008. Available at http://www.cs.unibo.it/∼gorrieri/Papers/bg08.ps on 2008/4/3.
4. S.D.Brooks, C.A.R.Hoare, A.W.Roscoe. A Theory of Communicating Sequential Processes. *Journal of the ACM* 31(3):560-599, 1984.

5. J.Engelfriet and G. Rozenberg. Elementary Net Systems, *Lectures on Petri Nets I: Basic Models*, Springer LNCS 1491, 1998.
6. R. Focardi, R. Gorrieri. A Classification of Security Properties. *Journal of Computer Security* 3(1) pp.5-33, 1995.
7. R. Focardi, R. Gorrieri. The Compositional Security Checker: A Tool for the Verification of Information Flow Security Properties, *IEEE Transactions on Software Engineering* 23(9):550-571, 1997.
8. R. Focardi, R. Gorrieri. Classification of Security Properties (Part I: Information Flow), *Foundations of Security Analysis and Design* - Tutorial Lectures (R. Focardi and R. Gorrieri, Eds.), Springer LNCS 2171:331-396, 2001.
9. S. Frau. Uno strumento sotware per l'analisi di proprietà di sicurezza su reti di Petri. Master thesis (in Italian), University of Bologna, March 2008.
10. J.A. Goguen, J. Meseguer. Security Policy and Security Models, Proc. of Symposium on Security and Privacy, IEEE CS Press, pp. 11-20, 1982.
11. F. Martinelli. Partial Model Checking and Theorem Proving for Ensuring Security Properties, Proc. of Computer Security Foundations Workshop, IEEE CS Press, pp. 44-52, 1998.
12. R.Milner. *Communication and Concurrency*, Prentice-Hall, 1989.
13. A.W. Roscoe. CSP and Determinism in Security Modelling, Proc. of IEEE Symposium on Security and Privacy, IEEE CS Press, pp. 114-127, 1995.
14. P.Y.A. Ryan. Mathematical Models of Computer Security, *Foundations of Security Analysis and Design* - Tutorial Lectures (R. Focardi and R. Gorrieri, Eds.), Springer LNCS 2171:1-62, 2001.
15. P.Y.A. Ryan, S. Schneider. Process Algebra and Noninterference, Proc. of 12th Computer Security Foundations Workshop, IEEE CS Press, pp. 214-227, 1999.
16. Java Technology. Sun Microsystems. Available at http://java.sun.com/ on 2008/4/3.
17. Eclipse.org. Eclipse Foundation, Available at http://www.eclipse.org/ on 2008/4/3.
18. Available at http://www2.informatik.hu-berlin.de/top/pnml/download/about/PNML_LNCS.pdf on 2008/4/3.
19. Platform Independent Petri Net Editor. Available at http://pipe2.sourceforge.net/ on 2008/4/3.