UNIVERSITÀ DEGLI STUDI DI MILANO - BICOCCA
DIPARTIMENTO DI INFORMATICA, SISTEMISTICA E COMUNICAZIONE
DOTTORATO DI RICERCA IN INFORMATICA - CICLO XXVIII

Ph.D. Thesis

Formal Notions of Non-interference and Liveness for Distributed Systems

Author: Görkem Kılınç

Supervisors: Prof. Lucia Pomello
Dr. Luca Bernardinello

Tutor: Prof. Paola Bonizzoni

February 2016

This thesis is dedicated to my parents.
For their endless love, support and encouragement...

Acknowledgements

First and foremost, I would like to express my sincere gratitude to my advisors Lucia Pomello and Luca Bernardinello for their continuous support, patience and motivation. They have both generously devoted me time and helped me in each step of my studies with their tremendous knowledge and vision. They have always believed in me and encouraged me. I could not have imagined having better advisors and mentors for my Ph.D study. They also have been like a family to me during these years that I have been away from my country and my beloved ones. I consider myself very lucky to have the chance to know such special people and have them as my thesis advisors.

I will forever be thankful to my former supervisor Carlo Ferigato who never stopped supporting me since my days in JRC-Ispra. He has been a great influence to me in choosing my research line. I also would like to thank his family to invite me during every Easter holidays and not let me be alone.

My sincere thanks also go to Jörg Desel who has provided me the possibility to spend four months in Fernuniversität in Hagen and work in his team. He had shared his time and his knowledge with me. He has given me ideas and has worked on them together with me which play an important role in this thesis. He has been a great support and guidance to me even after I left Hagen.

I would also like to thank Stefan Haar who has inspired me with his work and has given me the chance to visit him in ENS Cachan. He has helped me a lot with his valuable comments and suggestions.

I am also grateful to Thomas Chatain for sharing his valuable time and visiting us in Milano. His insightful comments and suggestions have been very helpful for my studies.

My sincere thanks also goes to my tutor Paola Bonizzoni for her valuable guidance and insightful comments.

I am grateful to Elisabetta Mangioni who has spared me time for a discussion that helped me on my work. I also would like to thank her and her family for being very kind to me and hosting me in their house during my Ph.D admissions.

During this thesis, as for everything else in my life, my best friends Burcu Ezgi Özdemir and Anu Baby have always been there for me. I am glad to have them in my life.

I would like to thank a very special person, Egemen Soylu. He has been a big support and encouragement with his love. He has been very understanding and patient. He is the person who has tolerated all my emotional crisis and stress explosions during this study.

Last but not the least, I would like to thank my parents Şengül Kılınç and Hasan Kılınç, my aunt Aysun Çulhacı and my cousin Merve Çulhacı to be always by my side even when I am far away from them. They have been the greatest support in my life for everything I do. I am very much lucky to have such an amazing, loving family.

Contents

Chapter 1

Introduction

The main goal of this thesis is to propose new approaches for defining secure and serviceable systems, provide model checking methods and explore formal techniques for designing secure and serviceable systems. The thesis examines distributed systems equipped with an interface through which users can control the system and get service. On one hand, an interface provides possibilities to a user to control the system, whereas on the other hand, it can create certain unwanted situations in terms of security. In general, maximizing security will reduce serviceability of the system and vice versa. The thesis studies these two notions separately and provides a common formal framework in which the two notions can be put together to reach a reasonable compromise.

In modeling distributed systems, it is assumed that there are actions that can be performed. Some of these actions are observable by the users, e.g., interactions between the user and the system, whereas the others are hidden internal actions. Some observable actions are controllable by the user. The aim of this work is to achieve secure and serviceable systems under this setting. The thesis focuses in particular on serviceability and information flow in distributed systems modeled with Petri nets. In addition to examining and comparing the existing notions of serviceability and information flow, the thesis proposes new notions and formal techniques. Potential application areas of this work include industrial control systems, critical infrastructures and security protocols.

1.1 The Setting

In our setting, we consider two kinds of entities for a distributed system: service providers and clients. A service provider is responsible for reliable service to the clients as well as for protection of sensible information. Assuming that one or more of the clients can be attackers, two possible scenarios can be considered. In the first scenario, the attacker can try to break down the system. The thesis provides a special kind of liveness notion for Petri net models which guarantees that the service provider will continue to serve. In the second scenario, the attacker can try to get hidden information about the other clients or about the internal actions of the service provider. The thesis studies non-interference notion in order to prevent unwanted information flow. For formal modeling, Petri nets and their unfoldings are used.

1.2 Liveness and Serviceability

The term 'serviceability' is defined as follows in this thesis: if a system is serviceable, a user is always able to get the expected service from the system by controlling it through the interface. Liveness is one of the properties which can be used for expressing serviceability, however the traditional liveness notion in Petri net theory is not suitable for partially observable and controllable systems. Chapter 3 introduces a new notion called *observable liveness* for expressing serviceability.

In the considered Petri net models, only some transitions are observable, and only a subset of these can be controlled by a user. Observable transitions represent the outputs (services) a user can get whereas the controllable transitions represent the actions that the user can control, i.e., user can choose which one to perform or even choose not to perform any. Intuitively, a system is observably live, if all the observable actions can be forced to eventually fire by choosing and performing the right controllable actions on the basis of the observed behavior. Section 3.1 formally defines the new liveness notion and explores its properties. It also compares observable liveness with traditional liveness and shows that the new notion generalizes traditional liveness in various ways. In particular, liveness of a 1-safe Petri net implies observable liveness, provided the only conflicts that can appear are between controllable transitions. This assumption refers to applications in which the uncontrollable part models a deterministic machine (or several deterministic machines), whereas the user of the machine is modeled by the controllable part and can behave arbitrarily. In general, observable liveness does not imply traditional liveness and traditional liveness does not imply observable liveness. In Section 3.2, we introduce *weak observable liveness* and show that observable liveness of a transition implies weak observable liveness of it.

Section 3.3 discusses a game theoretic method for checking weak observable liveness. The problem of checking if a transition is weakly observably live can be translated into a game. In particular, it can be translated into an infinite game that is played on finite graph with two players. These kind of games are described in [38, 50]. Weak observable liveness can be considered as a game between a user and a system. Intuitively, a transition is weakly observably live if the user has a winning strategy for the game. We discuss this idea on a concrete example by using Streett games.

1.3 Security and Non-interference

Chapter 4 is dedicated to the formal notion of non-interference. There is a rich literature which studies information flow in distributed systems. One of the formalizations of information flow is non-interference. Several notions of non-interference have been proposed in the literature to study the problem of confidentiality in nondeterministic and concurrent systems. The definition of non-interference is based on some observational semantics: assuming the system is composed of two distinct parts, classified as *high* (hidden) and *low* (observable), the high part of the system is non-interfering with the low part if whatever is done at the high level part produces no visible effect on the low part of the system. The original notion of non-interference was defined in [36] using trace semantics, for deterministic programs. Generalized notions of non-interference were then designed to include (non-deterministic) labeled transition systems and finer notions of observational semantics such as bisimulation. Busi and Gorrieri brought this approach in the field of Petri nets [19]. In Section 4.1, we discuss the general notions of information flow and non-interference, followed by a brief survey on the existing non-interference notions in the Petri net literature.

Similar to Busi and Gorrieri [19], this thesis analyzes systems that can perform high and low level actions and checks if deducing information about the high actions is possible by observing low actions. However, the new notions introduced in this thesis consider the information flow both about the past and future actions whereas the other existing notions mainly deal with deducing information about past occurrences of hidden actions. In addition, the new notions introduced in this thesis are formalized on unfoldings and can be checked on finite prefixes of unfoldings. In fact, the unfolding based approach is gaining interest for its efficiency comparing to the methods based on reachability graphs. In recent research, new approaches have been explored for checking the existing non-interference notions on unfoldings.

Non-interference deals with two kinds of information flow: *positive* and *negative*. The thesis introduces two main relations and their variants in order to deal with positive and negative information flow and to build the new non-interference notions on: *reveals* and *excludes*. These two new relations are introduced in Section 4.3 and Section 4.4 respectively.

A transition *reveals* another transition if the occurrence of the first means that the second one has already occurred or will inevitably occur in the future. The thesis analyzes and formalizes this relation between transitions of a Petri net and introduces two more parametric reveals relations: *extended-reveals* and *repeated-reveals*. In [39], reveals and extended-reveals are defined for events of processes and in [40] they are used for fault-diagnosis. In this thesis, reveals and extended-reveals relations are adapted to Petri nets by defining them for transitions in order to use in non-interference. In addition, a new parametric reveals relation based on repeated occurrence of a specific transition is introduced, i.e., repeated-reveals. On the basis of reveals relation and its variants, the thesis introduces new non-interference notions to catch positive information flow: *Reveals based Non-Interference*, *k-Extended Reveals based Non-Interference* and *n-Repeated Reveals based Non-Interference*. These new non-interference notions catch positive information flow both about the past and future occurrences of hidden transitions.

In order to deal with negative information flow, a new relation called *excludes* is introduced in the thesis. A transition *excludes* another transitions if they can never occur in the same run. A transition can exclude another transition only if it excludes the other in the past and in the future. By using excludes relation together with reveals relation, a new non-interference notion is introduced in this thesis which catches both positive and negative information flow both in the past and in the future. This new non-interference notion is called *Positive/Negative Non-Interference*.

Section 4.5 introduces the new non-interference notions on the basis of reveals and excludes relations with their variants, discusses examples and compares them with each other. In Section 4.6, the new non-interference notions are compared with some known non-interference notions existing in the Petri net literature.

Checking a net for the non-interference notions which are introduced in this thesis requires to compute reveals and excludes relations which are defined on unfoldings of Petri nets. Unfoldings are possibly infinite acyclic nets which express all possible behavior of a Petri net. They are preferred in model checking because they are more efficient as compared to reachability graphs. In Section 4.7, two methods have been proposed for checking reveals/excludes based non-interference notions. The first method is to translate reveals and excludes relations to Linear Temporal Logic (LTL) in order to express non-interference properties in LTL and to use LTL model checking methods. In the literature there are efficient LTL model checking techniques on the unfoldings such as the one introduced in [28]. The second method that we propose adapts the diagnosis algorithm introduced in [40] to the problem of checking non-interference. This method requires to compute only a finite prefix of the unfolding in order to compute all reveals and excludes relations. The method is based on translating reveals and excludes problems to diagnosis problem and with the help of this method it is possible to check the reveals/excludes based non-interference notions on a finite prefix of unfolding.

In Section 4.8, reveals and excludes relations and the new non-interference notions based on these relations are generalized to formal languages.

Chapter 5 concludes the thesis and discusses some open problems and possible improvements.

Chapter 2

Basic Definitions

In this section we collect preliminary definitions and set the notation which will be used in the rest of the thesis.

Let $R \subseteq I \times I$ be a binary relation, the transitive closure of R is denoted by R^+; the reflexive and transitive closure of R is denoted by R^*.

A *net* is a triple $N = (B, E, F)$, where B and E are disjoint sets, and $F \subseteq (B \times E) \cup (E \times B)$ is called the *flow relation*. The pre-set of an element $x \in B \cup E$ is the set ${}^\bullet x = \{y \in B \cup E : (y, x) \in F\}$. The post-set of x is the set $x^\bullet = \{y \in B \cup E : (x, y) \in F\}$.

An (ordinary) Petri net $N = (P, T, F, m_0)$ is defined by a net (P, T, F), and an initial marking $m_0 : P \to \mathbb{N}$. The elements of P are called *places*, the elements of T are called *transitions*. A net is finite if the sets of places and of transitions are finite.

A *marking* is a map $m : P \to \mathbb{N}$. A marking m is safe if $m(p) \in \{0, 1\}$ for all $p \in P$. Markings represent global states of a net.

A transition t is *enabled* at a marking m if, for each $p \in {}^\bullet t$, $m(p) > 0$. We write $m[t\rangle$ when t is enabled at m. A transition enabled at a marking can *fire*, producing a new marking. Let t be enabled at m; then, the firing of t in m produces the new marking m', defined as follows:

$$m'(p) = \begin{cases} m(p) - 1 & \text{for all } p \in {}^\bullet t \setminus t^\bullet \\ m(p) + 1 & \text{for all } p \in t^\bullet \setminus {}^\bullet t \\ m(p) & \text{in all other cases} \end{cases}$$

We will write $m[t\rangle m'$ to mean that t is enabled at m, and that firing t in m produces m'.

If $m_1[t_1\rangle m_2[t_2\rangle m_3[t_3\rangle m_4\cdots$, then $\sigma = t_1\,t_2\,t_3\,t_4\ldots$ is called *occurrence sequence*, enabled at the marking m_1. If an occurrence sequence σ is finite, i.e., $\sigma = t_1\,\ldots t_n$, then we write $m_1[\sigma\rangle m_{n+1}$ and call m_{n+1} *reachable* from m_1. This includes the case $n = 0$, i.e., each marking is reachable from itself.

The set of markings reachable from m will be denoted by $[m\rangle$.The set of *reachable markings of the Petri net N* is the set of markings reachable from its initial marking m_0. If all the markings reachable from m_0 are safe, then $N = (P, T, F, m_0)$ is said to be *1-safe* (or, shortly, safe). N is called *1-live* iff $\forall t \in T \;\; \exists m \in [m_0\rangle$ such that $m[t\rangle$.

A Petri net is *bounded* if its set of reachable markings is finite. Equivalently, it is bounded if and only if there exists a bound b such that each reachable marking m satisfies, for each place p, $m(p) \leqslant b$. It is called 1-bounded if this condition holds for $b = 1$.

A Petri net is *live* if, for each reachable marking m and each transition t, there exists a marking m' reachable from m that enables t. Equivalently, it is live if and only if, for each transition t and each finite occurrence sequence σ enabled at the initial marking m_0, there exists a transition sequence τ containing an occurrence of t such that $\sigma\,\tau$ is an occurrence sequence enabled at m_0. Notice that in order to append two sequences, the left hand one is supposed to be finite; when writing $\sigma\,\tau$ we implicitly express that σ is finite.

Two transitions t_1 and t_2 of a Petri net are in *structural conflict* if they share an input place, i.e., if ${}^\bullet t_1 \cap {}^\bullet t_2 \neq \emptyset$. t_1 and t_2 are in *behavioral conflict* for a given marking $m \in [m_0\rangle$, if they are both enabled at marking m and the firing of one disables the other.

Let $N = (B, E, F)$ be a net, and $x, y \in B \cup E$. If there exist $e_1, e_2 \in E$, such that $e_1 \neq e_2$, $e_1 F^* x$, $e_2 F^* y$, and there is $b \in {}^\bullet e_1 \cap {}^\bullet e_2$, then we write $x\#y$.

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

1. $\forall x \in B \cup E : \neg(xF^+x)$
2. $\forall x \in B \cup E : \neg(x\#x)$
3. $\forall e \in E : \{x \in B \cup E :\; xF^*e\}$ is finite
4. $\forall b \in B : |{}^\bullet b| \leqslant 1$

The set of minimal elements of an occurrence net N with respect to F^* will be denoted by ${}^\circ N$. The elements of B are called *conditions* and the elements of E are called *events.* If in an occurrence net $x\#y$, then we say that x and y are in conflict. Let $e \in E$ be an event in an occurrence net; then the *past* of e is the set of events preceding e in the partial order given by F^*: $\uparrow e = \{t \in E : tF^*e\}$ and the *future* of e is the set of events succeeding e in the partial order given by F^*: $\downarrow e = \{t \in E : eF^*t\}$.

Two elements of an occurrence net, x and y, are *concurrent*, denoted by x **co** y, indicating that x and y may occur at the same time in some reachable marking, if they are not causally dependent or in conflict with each other, defined as: x **co** y iff $\neg(x\#y)$ and $\neg(xF^*y)$ and $\neg(yF^*x)$. A subset of nodes $X \subseteq B$ pairwise concurrent will be called a *co-set*: $\forall x, y \in X$, x **co** y. A co-set formed by elements of B will be called a *B-co-set.* A maximal co-set with respect to set inclusion is called a *cut.*

An occurrence net represents the alternative histories of a process; therefore its underlying graph is acyclic and paths branching from a condition, corresponding to a choice between alternative behaviors, never converge.

A *run* of an occurrence net $N = (B, E, F)$ is a set R of events which is closed with respect to the past, and it is free of conflicts: (1) for each $e \in R$, $\uparrow e \subseteq R$; (2) for each $e_1, e_2 \in R$, $\neg(e_1\#e_2)$. A run is maximal if it is maximal with respect to set inclusion.

Let $N_i = (P_i, T_i, F_i)$ be a net for $i = 1, 2$. A map $\pi : P_1 \cup T_1 \to P_2 \cup T_2$ is a morphism from N_1 to N_2 if:

1. $\pi(P_1) \subseteq P_2$; $\pi(T_1) \subseteq T_2$
2. $\forall t \in T_1$ the restriction of π to ${}^\bullet t$ is a bijection from ${}^\bullet t$ to ${}^\bullet \pi(t)$
3. $\forall t \in T_1$ the restriction of π to $t^\bullet$ is a bijection from $t^\bullet$ to $\pi(t)^\bullet$

In the rest of the thesis, we will consider only 1-live Petri nets, in which all transitions have non-empty presets, i.e., all have input places. Moreover, except for occurrence nets, the considered Petri nets will have finite underlying nets. Of course, Petri nets may have infinite behavior.

A *branching process* of a Petri net $N = (P, T, F, m_0)$ is a pair (O, π), where $O = (B, E, G)$ is an occurrence net, and π is a morphism from O to N such that:

1. $\forall p \in P$ $m_0(p) = |\pi^{-1}(p) \cap {}^\circ O|$
2. $\forall x, y \in E$, if ${}^\bullet x = {}^\bullet y$ and $\pi(x) = \pi(y)$, then $x = y$

A branching process $\Pi_1 = (O_1, \pi_1)$ is a prefix of $\Pi_2 = (O_2, \pi_2)$ if there is an injective morphism f from O_1 to O_2 which is a bijection when restricted to ${}^\circ O_1$, and such that $\pi_1 = \pi_2 f$.

Any finite Petri net N has a unique branching process which is maximal with respect to the prefix relation. This maximal process, called the *unfolding* of N, will be denoted by $\mathrm{Unf}(N) = ((B, E, F), \lambda)$, where λ is the morphism from (B, E, F) to N [27]. In Figure 2.1, a Petri net with its infinite unfolding is illustrated.

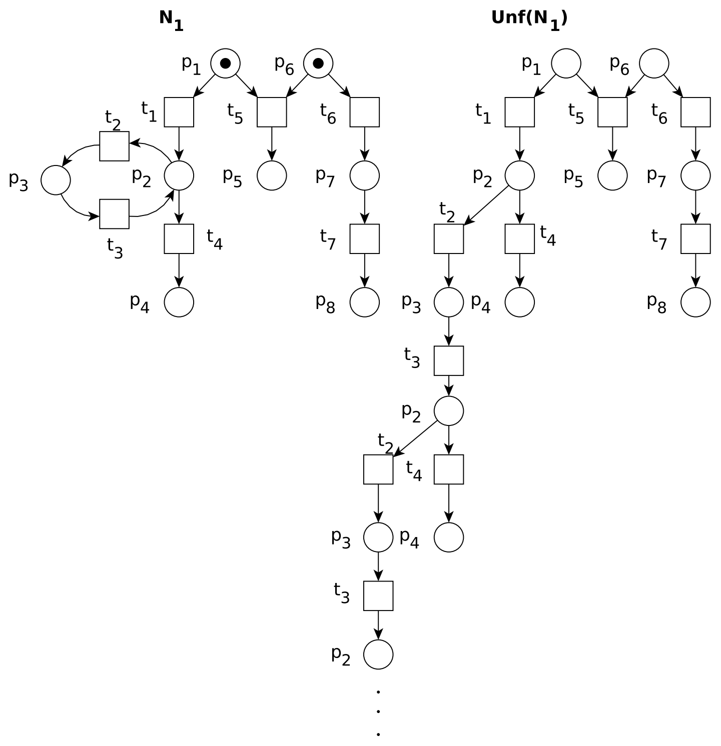

FIGURE 2.1: A Petri net and its unfolding

The following definition will be used in the rest of the thesis to denote the set of events of an unfolding corresponding to a specific transition of a given Petri net.

Definition 2.0.1. Let $N = (P, T, F, m_0)$ be a Petri net, $\mathrm{Unf}(N) = ((B, E, F), \lambda)$ be its unfolding and $t \in T$, the set of events corresponding to t is denoted $E_t = \{e \in E : \lambda(e) = t\}$.

An *alphabet* is a finite set $\mathbb{X}$, whose elements are called *letters*.

Chapter 3

Liveness and Serviceability

In this chapter we will considers systems in which users can interact with the system through an interface. Some actions are observable to the users whereas some other actions are hidden internal actions of the system. Among the observable actions, some are controllable by the users. The traditional liveness property for Petri nets guarantees that each transition of a Petri net can always occur again, whereas observable liveness requires that, from any reachable marking, each observable transition can be forced to fire by choosing appropriate controllable transitions. Hence, it is defined for Petri nets with distinguished observable and controllable transitions. Section 3.1 introduces *observable liveness*, examines its properties, compares it with the traditional liveness notion in Petri net theory. Some results on observable liveness are published in [25, 26]. Section 3.2 introduces *weak observable liveness* discusses the intuition on the basis of examples and show that observable liveness implies weak observable liveness. The last section of the chapter discusses a game theoretic method for checking weak observable liveness.

3.1 Observable Liveness for Serviceability

Liveness and boundedness have turned out to be the most prominent behavioral properties of Petri nets; a Petri net is considered to behave well if it is live and bounded. This claim is supported by many publications since decades, and, in particular, by the nice correspondences between live and bounded behavior of a Petri net and its structure; see e.g. [23, 51]. Nowadays workflow (Petri) nets receive a particular interest, and with them the behavioral soundness property. As shown in [69], soundness of a workflow net is identical to the combination of liveness and boundedness of the Petri net obtained by adding a feedback place between the final and the initial transition of the workflow net.

This way, these behavioral properties are also applied to models of processes that have a start and an end action, such as business processes.

Observable liveness expresses serviceability of a distributed system. This means that, having a machine producing observable and unobservable actions, a user (or controller) can always get the expected services (observable actions) by using the buttons (controllable transitions) on the machine. The buttons on the machine form an interface (controllable part of the machine) to the user to request service. Observable liveness guarantees that the machine will respond with the requested service, if the right buttons are pressed.

This chapter concentrates on liveness, but looks at yet another scenario: we consider Petri nets with transitions that can be observable or unobservable (silent transitions), and can be controllable or not. These nets are inspired by Petri net applications in control theory [22, 43, 66], but can also be seen as a generalization of Petri nets with silent transitions. We provide a notion of liveness which is tailored to Petri nets with observable and controllable transitions, or to the systems modeled by these nets.

The traditional definition of liveness is as follows: A transition t of a Petri net is *live* if, for each reachable marking m, there is a marking m' reachable from m that enables t. A Petri net is *live* if all its transitions are live.

A characterization of liveness is based on the view that the system can be controlled, i.e., that someone can choose which activities to perform once they are enabled. In such a setting, liveness means that, no matter which state was reached, we can construct a run, i.e., an occurrence sequence of activities, that ends with a given desired activity. Thus, instead of asking for the existence of such a run, we take the view that we can enforce such a run by performing its activities.

However, what happens if not each activity is controllable, i.e., if some activities can be controlled and others occur autonomously? What if, moreover, the assumed user does not have knowledge about the current state because he can observe only a part of the activities? In this contribution, we study a liveness notion for such an environment. As usual for models of Discrete Event Systems (DES), we distinguish observable and unobservable activities and, among the observable ones, controllable and uncontrollable activities. Therefore, we consider Petri net models where only some transitions are observable, and only a subset of these can be controlled.

The new introduced liveness notion is still in accordance with the idea that, for a given transition t, no matter which marking m was reached, an occurrence sequence can be constructed which includes t. However, in contrast to the traditional definition of liveness,

- we only consider observable transitions t (i.e., if a transition t cannot be observed then it does not have to be live),
- we assume that instead of constructing the entire sequence, the user can just control the net by choosing controllable transitions whenever they are enabled, whereas the net is always free to fire uncontrollable transitions,
- the net is weakly fair with respect to all uncontrollable transitions, i.e., once an uncontrollable transition is enabled, it either eventually occurs or a transition in structural conflict with this transition occurs.

If a controllable transition is in structural conflict with an uncontrollable transition, the user cannot prevent the occurrence of the uncontrollable one. However, if the user decides not to fire the controllable transition, then, by weak fairness, the uncontrollable one eventually fires (provided no other conflicts are involved).

3.1.1 The setting

When defining observable liveness, several design decisions have to be made. We have a particular setting of a modeled system in mind, which motivates our choices. This section aims at explicating this setting and thus justifying our design decisions.

The generic system to be modeled consists of a machine (or several machines), a user interface to this machine, and perhaps of activities and conditions which do not belong to the machine. The user can observe and control all activities outside the machine, but he can neither control nor observe any activities inside the machine. Concerning the user interface, there are activities that the user can only observe but not control, whereas other interface activities might be both observable and controllable.

One might argue that, instead of activities, only local states of machines are observable, for example a light which can be on or off. Then, instead of observing this state, in our setting we observe the activities that cause the changes of the state. In terms of Petri nets, instead of observing a place, we observe the (occurrences of) transitions in the pre- or post-set of the place.

A controllable activity can be unconnected to the machine, or it can be an activity of the machine's interface. Whereas a controllable activity outside the machine is clearly also observable, one might argue that this is not obvious for controllable interface activities. In fact, if the activity can be caused by pressing a button, the user cannot be sure that with every use of this button the activity takes place. An additional prerequisite is that the activity is enabled by the machine, whereas buttons can always be pressed. So we implicitly assume that the user sees whether a controllable transition is enabled or not and can thus distinguish activities from non-activities caused by buttons.

Assume that a user wants to enforce an observable activity a after some previous run of the system. Then, depending on what he has observed so far, he should have a strategy to control activities in such a way that eventually he can observe a. By translating activities to transitions, the same holds for the Petri net model. The strategy is formalized by a function that maps a sequence of observable transitions to a set of controllable transitions: if the sequence was observed, then any of these controllable transitions can be chosen and fired by the user. Since the domain of this function is infinite in general, and its co-domain finite (theoretically exponential in the number of controllable transitions, but usually linear), different sequences are mapped to the same set. We assume that the user can effectively compute this function by using, e.g., only a finite history or an automata based approach. For generality of our approach, we nevertheless consider a strategy to be an arbitrary function as above.

There might be states in which controllable activities and uncontrollable ones are enabled, i.e., both the machine and the user can do something. In such a state, we cannot expect that the user is able to do his controllable activity first. This means that, in the case of competition between activities, the user only has full control if only controllable activities are involved.

For each observably live activity, we demand that the user can enforce its occurrence. Therefore, we provide an appropriate behavioral model of Petri nets. Clearly, the user can only enforce any reaction from the machine if the machine obeys some progress assumption: we do not consider runs in which an enabled uncontrollable transition, which is not in structural conflict with any other occurring transition, does not occur. However, progress is not assumed for controllable transitions.

Throughout the chapter, a controllable transition is illustrated as a black filled rectangle of a Petri net, an uncontrollable observable transition is illustrated as a bold rectangle, while unobservable ones are drawn by plain rectangles. Incoming and outgoing arcs, which are not connected to any place or transition, are used when only a part of a Petri net is shown.

Example 3.1.1. The net shown in Figure 3.1 models a vending machine with coffee and tea options. The user can operate the machine by inserting a coin and using three buttons (*insert coin, choose coffee, choose tea* and *take money back* are controllable transitions). The transitions *output coffee*, *output tea* and *output money* are observable, but not controllable. However, the user can force these transitions to occur by firing the controllable transitions. In other words, each observable transition of the net is

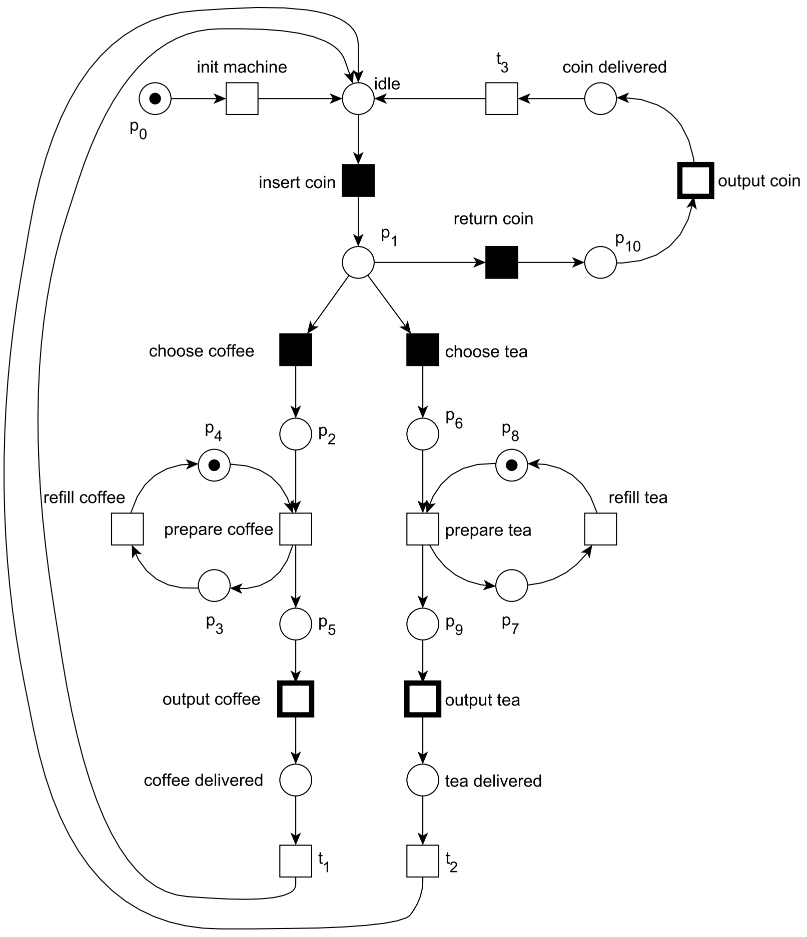

FIGURE 3.1: An observably live net which represents a vending machine.

observably live, and so the entire net is observably live. If there is no more coffee or tea, the machine needs a refill operation. In this case, the user has to wait until the refill operation is done. Due to the progress assumption, the transitions *refill coffee* and *refill tea*, both being not in conflict with any other transition, will fire eventually once they are enabled.

The entire net is not live because the unobservable part includes a transition which can only fire once (*init machine*). However, this behavior does not affect our notion of observable liveness, since all the observable transitions can be forced to occur.

Considering such a machine, observable liveness is a useful notion to express the serviceability of a machine via an interface. In this sense, observable liveness expresses the liveness of a system from the user's point of view. □

In this chapter we consider Petri nets where transitions can be observable or non-observable, and can be controllable or non-controllable. If T is the set of all transitions of a Petri net, we denote by $O \subseteq T$ the set of *observable* transitions and by $C \subseteq O$ the set of *controllable* ones. Given an occurrence sequence σ of the Petri net, its projection $\overline{\sigma}$ to the observable transitions is called *observable occurrence sequence*. In turn, a sequence $t_1\, t_2\, t_3 \ldots$ of observable transitions is an observable occurrence sequence if and only if there are finite sequences $\sigma_1, \sigma_2, \sigma_3, \ldots$ of unobservable transitions such that $\sigma_1\, t_1\, \sigma_2\, t_2\, \sigma_3\, t_3 \ldots$ is an occurrence sequence of the Petri net.

A finite or infinite occurrence sequence $\sigma = t_1\, t_2\, t_3 \ldots$ enabled at some marking m is called *weakly fair* with respect to some transition t if, whenever an occurrence sequence $t_1 t_2 \ldots t_k t$ is enabled at m (where $t_1 t_2 \ldots t_k$ is a prefix of σ), then either t or a transition u in structural conflict with t appears in $t_{k+1}\, t_{k+2} \ldots$ (once t is enabled, it either occurs, or a transition in conflict with t occurs). Notice that this definition is slightly weaker than the usual definition of weak fairness which demands that transition t eventually occurs once it is persistently enabled, even if transitions in conflict with t occur. There are many different fairness notions for Petri nets (and previously for other models). The one we adapt here - often called progress assumption - was first mentioned in [57].

A *run* of a Petri net with observable and controllable transitions is an observable occurrence sequence $\overline{\sigma}$ where σ is an occurrence sequence, enabled at the initial marking, which is weakly fair with respect to all uncontrollable transitions.

3.1.2 Observable liveness

Throughout the rest of this chapter, let N be a Petri net with initial marking m_0, T its set of transitions, O its set of observable transitions, and C its set of controllable transitions. In order to define observable liveness, we first stick to observable liveness of a single transition, which has to be observable, and later define observable liveness of the Petri net N as observable liveness of all its observable transitions.

Consider a single observable transition t which might be moreover controllable or not. Let m be a marking reached from m_0 by the occurrence of an occurrence sequence σ_0. Assume that, from m, a user wants to enforce transition t by selecting appropriate controllable enabled transitions. If this is always (for each reachable marking m) possible, then we call t observably live.

From the marking m, the net first proceeds arbitrarily and autonomously, i.e., some occurrence sequence σ_1 without controllable transitions occurs. We assume that this occurrence sequence is either

a) finite and leads to a deadlock, or

b) finite and leads to a marking that enables at least one controllable transition, or

c) infinite and weakly fair with respect to all uncontrollable transitions.

Notice that, after a proper prefix of σ_1, a controllable transition might be enabled, but then σ_1 continues with an uncontrollable transition. This includes the case where a controllable and an uncontrollable transition are in conflict relation. In case b), the obtained marking might also enable uncontrollable transitions which, however, do not fire after the sequence.

In case b), the user fires a controllable transition t_1 after σ_1, and then the net proceeds as before with a next autonomous sequence σ_2, and so on. This either yields an infinite sequence $\sigma_1\ t_1\ \sigma_2\ t_2 \ldots$, or eventually leads to a deadlock (case a)), or to an infinite sequence σ_i (case c)).

Our liveness notion should express that, in case of observable liveness, after any sequence σ_i of uncontrollable transitions, there is (at least one) suitable enabled controllable transition that can be fired, such that eventually t occurs. If, after a sequence σ_i, controllable and uncontrollable transitions are enabled, the user cannot avoid that the uncontrollable one fires. However, if a controllable transition is enabled after a σ_i and no transition in conflict with this controllable transition occurs in the sequel, then an additional fairness constraint demands that the user is able to fire this transition eventually.

To formalize this, and to avoid an infinite alternation of $\forall$ and $\exists$, we introduce a response function φ which delivers a set of controllable transitions as possible responses of the user to the sequence he has observed so far. Notice that an observed sequence does neither determine the actual occurrence sequence nor the reached marking because unobservable transitions might have occurred as well, affecting the marking but not the observed sequence.

Definition 3.1.1. Let $\varphi\colon O^* \to 2^C$ be a function, called *response function*, and let $m_0[\sigma_0\rangle m$ be an occurrence sequence. We call an occurrence sequence σ, enabled at m, *φ-fair* if it is

1. weakly fair w.r.t. all transitions in $T \setminus C$,
2. weakly fair w.r.t. all controllable transitions $t \in \varphi(\overline{\sigma_0\sigma})$, if $\overline{\sigma}$ is finite,
3. weakly fair w.r.t. all controllable transitions t satisfying $t \notin \varphi(\overline{\sigma_0\sigma'})$ for only finitely many prefixes σ' of σ (after some finite prefix, t is persistently responded by φ), if $\overline{\sigma}$ is infinite, and

4. an infinite composition $\sigma = \sigma_1\, t_1\, \sigma_2\, t_2\, \sigma_3\, t_3\, \ldots$, or a finite composition $\sigma = \sigma_1 t_1 \sigma_2 t_2 \ldots \sigma_k t_k \sigma_{k+1}$, where $k \geqslant 0$, such that, for $i \geqslant 1$, the sequence σ_i contains no controllable transitions and $t_i \in \varphi(\overline{\sigma_0 \sigma_1 t_1 \sigma_2 t_2 \ldots \sigma_i})$ (t_i is a controllable transition and a possible response to the sequence observed so far).

Each σ_i in the above definition can be empty. If the sequence σ ends with σ_{k+1} then this sequence σ_{k+1} can be infinite or finite; in the latter case, the marking reached either is a deadlock or only enables controllable transitions which are not in $\varphi(\overline{\sigma_0 \sigma})$.

Lemma 3.1.1. *Let σ be a φ-fair occurrence sequence enabled at a reachable marking m of N. If $\sigma = \sigma_1\, \sigma_2$ and $m[\sigma_1\rangle m_1$ then σ_2 is a φ-fair occurrence sequence enabled at m_1.*

Proof. The claim follows immediately from the definition of φ-fair occurrence sequences. □

Definition 3.1.2. An observable transition t of N is *observably live* if there is a response function $\varphi_t \colon O^* \to 2^C$ such that, for each $m_0[\sigma_0\rangle m$, each φ_t-fair occurrence sequence enabled at m contains an occurrence of t. The Petri net N is *observably live* if all its observable transitions are observably live.

In this definition, "an occurrence of t" can be replaced by "infinitely many occurrences of t" like in the definition of traditional liveness, as shown next.

Theorem 3.1.2. *An observable transition t of N is observably live if and only if there is a response function $\varphi_t \colon O^* \to 2^C$ such that, for each $m_0[\sigma_0\rangle m$, each φ_t-fair occurrence sequence enabled at m contains infinitely many occurrences of t.*

Proof. We only have to prove $\Rightarrow$ because each occurrence sequence with infinitely many occurrences of t has at least one t-occurrence.

So assume observable liveness of t, i.e., a response function $\varphi_t \colon O^* \to 2^C$ such that, for each $m_0[\sigma'_0\rangle m'$, each φ_t-fair occurrence sequence enabled at m' contains an occurrence of t (notice that we replaced σ_0 by σ'_0 and m by m').

Let $m_0[\sigma_0\rangle m$ and let σ be a φ_t-fair occurrence sequence enabled at m. We will show that σ contains infinitely many occurrences of t. By assumption, we know that σ contains at least one occurrence of t. Let σ_1 be the prefix of σ that ends after the first occurrence of t and let $\sigma = \sigma_1\, \sigma_2$. Then $m_0[\sigma_0 \sigma_1\rangle m_1$ for some marking m_1. This marking m_1 enables the φ_t-fair occurrence sequence σ_2 by Lemma 3.1.1. Again using the assumption, σ_2 contains an occurrence of t. The arbitrary repetition of this argument yields arbitrarily many occurrences of t in σ. □

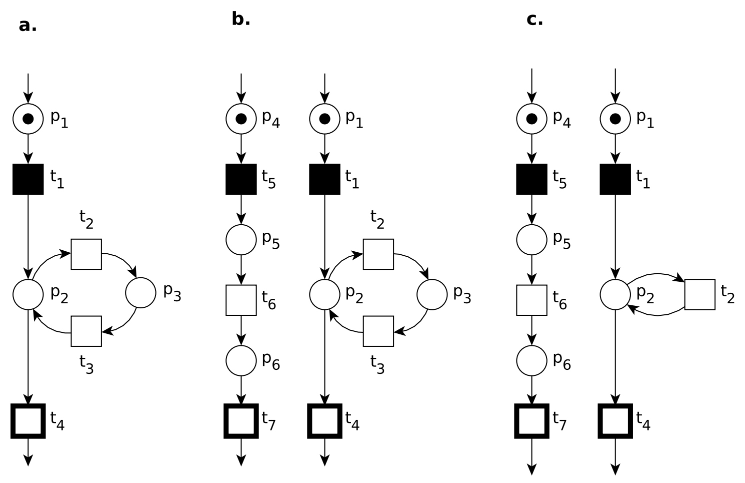

FIGURE 3.2: Some example nets

Example 3.1.2. Figures 3.2.a, 3.2.b, and 3.2.c illustrate the weak fairness notion employed in the definition of φ-fair occurrence sequences.

In the net shown in Figure 3.2.a, after the controlled occurrence of t_1, the system can choose between t_2 and t_4. It can even always prefer t_2, and then t_4 never occurs. Only strong fairness would imply that eventually t_4 can be observed, but our chosen notion of weak fairness does not. So t_4 is not observably live.

In Figure 3.2.b, the net of Figure 3.2.a is extended by a concurrent sequence. Our weak fairness assumption implies that the left branch proceeds even if the right stays in an infinite loop. So transition t_7 is observably live.

Figure 3.2.c illustrates the difference between our weak fairness and a similar notion often used in the literature (e.g. in [58]). We do not expect that t_4 eventually occurs although it remains enabled at each marking reached after the occurrence of t_1. Since t_2 and t_4 share the input place p_2, we have a conflict here. So, again, t_7 is observably live, and t_4 is not.

In the net fragment shown in Figure 3.3.a, there is conflict between t_3 and t_4. After the occurrence of transition t_1 or t_2, both t_3 and t_4 are enabled. In this situation, even if the response function φ tells us to fire t_4, we cannot be sure that t_4 actually fires because the unobservable transition t_3 might fire. Therefore, we also cannot force t_5 to fire, whence t_5 is not observably live. $\square$

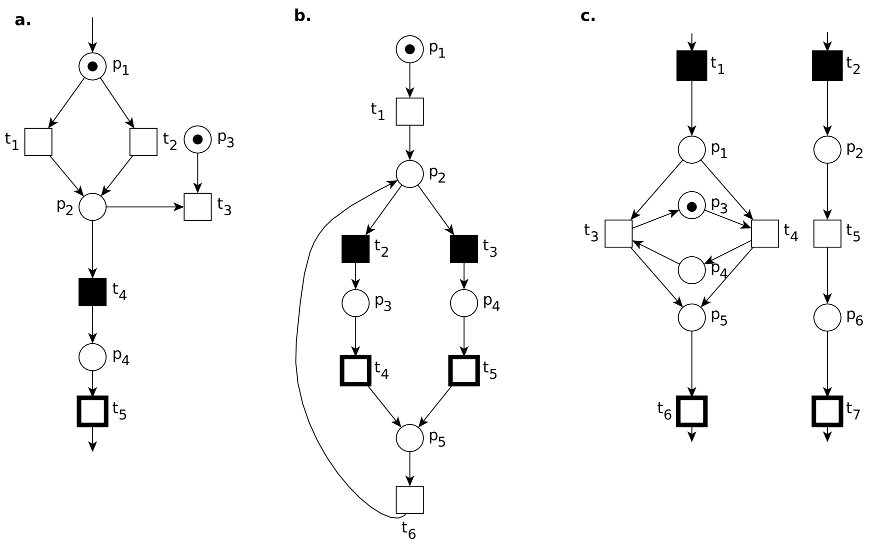

FIGURE 3.3: Example nets

3.1.3 Properties of observable liveness

In this section, we provide some properties of observable liveness and relations between observable liveness and traditional liveness.

Lemma 3.1.3. *For each response function φ and each $m_0[\sigma_0\rangle m$, there is a φ-fair occurrence sequence enabled at m.*

Proof. In order to construct a φ-fair occurrence sequence, we proceed iteratively. Assume that we constructed a finite occurrence sequence σ', enabled at m, and let $m[\sigma'\rangle m'$. If m' enables an uncontrollable transition t or a controllable one which is in the current response set $\varphi(\overline{\sigma_0\sigma'})$, then we append t to σ'. If there is more than one such candidate, we choose one which has not been chosen yet (if such a transition exists) or the least recently chosen one (otherwise) in order to ensure weak fairness. This is either always possible and leads to an infinite sequence, or it is eventually not possible, in which case no transition is appended to σ'. In both cases, the obtained sequence satisfies all fairness requirements formulated in Definition 3.1.1. The decomposition of this occurrence sequence required in Definition 3.1.1 is obtained by separating controllable and uncontrollable transitions. □

Proposition 3.1.4. *Each observably live transition of N is live.*

Proof. Let t be an observably live transition. Then there is a response function φ_t such that, for each $m_0[\sigma_0\rangle m$, each φ_t-fair occurrence sequence enabled at m includes t. By Lemma 3.1.3, there exists a φ_t-fair occurrence sequence. This implies that each reachable marking m enables an occurrence sequence which includes t, and so t is live. □

Corollary 3.1.5. *If N is observably live and all its transitions are observable, then N is live.*

Corollary 3.1.5 does not hold without the assumption that all transitions are observable. The net shown in Figure 3.3.b is observably live, but it is not live since t_1 can occur only once.

A live transition is not necessarily observable live. Consider that the net in Figure 3.2.a is changed such that p_1 is the post-condition of t_4, then the net is live. However, it is not observably live because the observable transition t_4 is live, but not observably live.

If all transitions are controllable, then liveness of a transition t implies its observable liveness, as shown next:

Proposition 3.1.6. *If $O = C = T$ then observable liveness of a transition t coincides with its liveness.*

Proof. By Proposition 3.1.4, we only have to show the implication $\Leftarrow$.

Assume that t is live. We have to show that there is a response function $\varphi_t \colon O^* \to 2^C$ such that, for each reachable marking m, each φ_t-fair occurrence sequence enabled at m contains an occurrence of t. So let $m_0[\sigma_0\rangle m$. Since t is live, there exists a finite occurrence sequence σ enabled at m which includes an occurrence of t.

Since $O = T$, $\sigma_0\,\sigma = \overline{\sigma_0\sigma}$. Let $\sigma_0\,\sigma = t_1 t_2 t_3 \dots t_k$. We choose any response function satisfying $\varphi_t(t_1\,t_2 \dots t_i) = \{t_{i+1}\}$ for $i = 0, 1, \dots, k-1$, which is possible because $C = T$. Again since all transitions are controllable, the unique φ_t-fair occurrence sequence consists of only controllable transitions. The σ_i (for $i = 1, 2, 3, \dots$) of Definition 3.1.1 are thus empty sequences, and so there is only one φ_t-fair occurrence sequence enabled at m. □

Corollary 3.1.7. *If $O = C = T$ then N is observably live if and only if N is live.*

Proposition 3.1.8. *Assume that N is observably live. Let σ be an infinite occurrence sequence without controllable transitions, enabled at a reachable marking m of N. If σ is weakly fair with respect to all uncontrollable transitions, then each observably live transition occurs in σ.*

Proof. Let t be an observably live transition. There is a response function φ_t such that each φ_t-fair occurrence sequence enabled at m contains an occurrence of t. By definition of φ_t-fairness, this holds in particular for the sequence σ. □

Corollary 3.1.9. *Assume that N is observably live and has no controllable transitions. Let σ be an infinite occurrence sequence, enabled at a reachable marking of N, and assume that σ is weakly fair with respect to all transitions. Then σ contains occurrences of all transitions of N.*

3.1.4 Conflict-free transitions

As seen before, a live Petri net is not necessarily observably live. The main reason is that, given a live Petri net, we can always choose an appropriate occurrence sequence including some transition t, whereas for observable liveness this choice is only possible for enabled controllable transitions (which are not in conflict with unobservable ones), and the Petri net behaves arbitrarily elsewhere.

In Section 3.1.5, we will show that the situation is different if the only choices to be made are among controllable transitions. This is not an unrealistic setting; the automated part of a system often behaves deterministically (but still concurrently), whereas the user might have alternatives. Consequently, the only choices in the system's model concern its part of the controllable user interface. i.e., its controllable transitions.

Formally, deterministic behavior is given in terms of the conflict-free property, to be defined next. Intuitively, a transition is conflict-free if it is never in conflict with any other transition; if both are enabled then they are enabled concurrently. Since "never" refers to reachable markings, the definition of conflict-freeness applies to a Petri net with an initial marking and its state space, and not to its structure. So we distinguish *structural conflicts* between two transitions, as defined before, and *behavioral conflicts*. However, each two transitions that are ever in behavioral conflict necessarily share an input place and are therefore also in structural conflict. With concurrent behavior we mean that two transitions do not compete for tokens. If a place carries more than one token, one could argue that two transitions in its post-set can occur concurrently (see [70]). We take the stricter view that every two enabled transitions with a common input place (which can carry one or more tokens) are considered in behavioral conflict and not concurrent.

This section is devoted to results on conflict-free transitions. These results will be used in the next section.

Definition 3.1.3. A transition u of N is *conflict-free* if, for each reachable marking m enabling u, every other transition v enabled at m satisfies ${}^\bullet u \cap {}^\bullet v = \emptyset$ (i.e., u and v are not in structural conflict and can therefore occur concurrently).

Figure 3.3.c on page 20 shows a net fragment where all transitions are conflict-free. Notice that there is a structural conflict between t_3 and t_4, but no reachable marking enables these two transitions.

The following lemma will be used frequently in the sequel. It follows immediately from the occurrence rule.

Lemma 3.1.10. *Let u and v be two distinct transitions of N, both enabled at some marking m of N, which are not in structural conflict. Then m enables uv as well as vu, and both occurrence sequences lead to the same marking.*

The next lemmas generalize well-known results for conflict-free nets [47], i.e. nets where all transitions are conflict-free, to nets with some conflict-free transitions.

Lemma 3.1.11. *If a transition u of N is conflict-free and some reachable marking m of N enables u as well as an occurrence sequence σu, where u does not occur in σ, then m also enables the occurrence sequence $u\sigma$, and the occurrences of σu and of $u\sigma$ lead to the same marking.*

Proof. By induction on the length of σ.

Base: If σ is the empty sequence then nothing has to be shown.

Step: Assume that $\sigma = v\sigma'$. We have $u \neq v$ because u does not occur in σ. By conflict-freeness of u and since m enables both u and v, these transitions are concurrent. By Lemma 3.1.10, m also enables the occurrence sequence $v\,u$. Let $m[v\rangle m'$. The induction hypothesis can be applied to the marking m' enabling u and $\sigma' u$, yielding the occurrence sequence $u\sigma'$ enabled at m'. So $vu\sigma'$ is enabled at m. Again since u and v are concurrent and by Lemma 3.1.10, m also enables $uv\sigma'$, which is identical with $u\sigma$.

Since each transition occurs in σu and in $u\sigma$ the same number of times, and by the occurrence rule, the occurrences of these sequences lead to the same marking. □

Lemma 3.1.12. *If a transition u of N is conflict-free and some reachable marking m of N enables u as well as a finite occurrence sequence σ in which u does not occur, then m also enables the occurrence sequence σu.*

Proof. By induction on the length of σ.

Base: If σ is the empty sequence then nothing has to be shown.

Step: Assume that $\sigma = v\sigma'$. We have $u \neq v$ because u does not occur in σ. By conflict-freeness of u and since m enables both u and v, these transitions are concurrent. By

Lemma 3.1.10, m also enables the occurrence sequence vu. Let $m[v\rangle m'$. The induction hypothesis can be applied to the marking m' enabling u and σ', yielding the occurrence sequence $\sigma' u$ enabled at m'. So $v\sigma' u$ is enabled at m. Since $v\sigma' = \sigma$, this finishes the proof. □

Lemma 3.1.13. *Let σ and τ be two finite occurrence sequences of N, both enabled at a reachable marking m of N, such that no transition occurs both in σ and in τ. If all transitions in σ are conflict-free then m enables the occurrence sequence $\sigma\tau$.*

Proof. By induction on the length of σ.

Base: If σ is the empty sequence then nothing has to be shown.

Step: Assume that $\sigma = v\sigma'$. Since v is conflict-free and since m enables τ, Lemma 3.1.12 can be applied and yields that m enables τv. By Lemma 3.1.11, m also enables $v\tau$. Let $m[v\rangle m'$. Then m' enables σ' as well as τ. By the induction hypothesis, m' enables $\sigma'\tau$, whence m enables $v\sigma'\tau = \sigma\tau$. □

Definition 3.1.4. An occurrence sequence σ of N, enabled at a marking m of N, is called *minimal towards* t, where t is a transition, if $\sigma = \sigma' t$, t does not occur in σ', and σ cannot be divided as $\sigma = \mu' u \mu''$ for some transition u, $u \neq t$, such that $\mu'\mu''$ is enabled at m, too (i.e., σ ends with t, contains no other occurrence of t, and no transition in σ can be postponed).

It is easy to see that every occurrence sequence enabled at a marking m and containing a transition t with minimal length is also minimal towards t. However, the converse does not necessarily hold. As an example, assume occurrence sequences abt and ct enabled at m. If neither t nor at or bt are enabled at m then abt is minimal towards t, although it is longer than ct.

A transition u can only occur if its input places carry tokens, and another transition v might have to occur before transition u because it produces a token consumed by u. We then call the occurrence of v a causal predecessor of the occurrence of u. A minimal occurrence sequence towards a transition t contains one occurrence of t, its causal predecessors, the predecessors of these predecessors etc., and nothing else. In partially ordered runs, where causal dependencies between transition occurrences are explicitly modeled by means of a partial order, this corresponds to a run containing the occurrence of t and all transition occurrences that precede the occurrence of t with respect to the partial order.

Definition 3.1.5. Given a sequence $\sigma = t_1 t_2 t_3 \dots$ with indices $I = \{1, 2, 3 \dots\}$, any (ordered) subset $J = \{j_1, j_2, j_3, \dots\}$ of I generates a *subsequence* $t_{j_1} t_{j_2} t_{j_3} \dots$ of σ. Its *complementary subsequence* is the subsequence generated by the complementary index set $I \setminus J$.

This definition captures the case $\sigma = \sigma'\sigma''$ where σ' is a subsequence and σ'' is its complementary subsequence (and vice versa), but is more general. For example, if $\sigma = t_1, t_2, \ldots, t_{2n}$, the sequence $t_1, t_3, \ldots, t_{2n-1}$ is a subsequence, and $t_2, t_4, \ldots, t_{2n}$ its complementary subsequence.

Lemma 3.1.14. *Let m be a reachable marking of N enabling an occurrence sequence $\sigma = \sigma_1 t$ such that σ_1 contains only conflict-free transitions (the transition t is not necessarily conflict-free) and moreover does not contain t. Then there exists a subsequence σ' of σ, enabled at m, which is minimal towards t. Moreover, if σ'' is its complementary subsequence, then m enables the occurrence sequence $\sigma'\sigma''$, which leads to the same marking as σ.*

Proof. Assume that σ can be divided as $\sigma = \mu' u \mu''$ such that μ'' is not the empty sequence, u does not occur in μ'' (and thus $u \neq t$), and $\mu'\mu''$ is enabled at m. Transition u is conflict-free because σ_1 contains only conflict-free transitions. By Lemma 3.1.12, we can shift u behind μ'' and thus obtain the occurrence sequence $\mu'\mu'' u$ which leads to the same marking as σ. Still t occurs only once in this sequence, now as the last transition in μ''.

Let u_1 be the rightmost transition occurrence in σ_1 for which such a division $\sigma = \mu_1' u_1 \mu_1''$ is possible, and let $\mu_1'\mu_1'' u_1$ be the occurrence sequence obtained by shifting u_1 to the end. Then μ_1'' ends with t. Let $\sigma_2 t = \mu_1'\mu_1''$. Let u_2 be the rightmost occurrence of a transition with the above property in σ_2, i.e., $\sigma_2 t = \mu_2' u_2 \mu_2''$ and $\mu_2'\mu_2''$ is enabled at m. The same argument as above yields the occurrence sequence $\mu_2'\mu_2'' u_2$, leading to the same marking as $\sigma_2 t$ whence u_1 can be appended to it.

Exhaustive repetition of this procedure yields smaller and smaller sequences μ_i' to be considered and eventually the occurrence sequence $\mu_k'\mu_k'' u_k u_{k-i} \ldots u_1$ such that no further transition to be postponed can be found in $\mu_k'\mu_k''$. So $\mu_k'\mu_k''$ is minimal towards t. By construction, it is a subsequence of σ, and $u_k u_{k-i} \ldots u_1$ is its complementary subsequence. The entire occurrence sequence is a permutation of σ, whence, by the occurrence rule, it leads to the same marking as σ. □

3.1.5 Deterministic uncontrollable behavior

In this section, we require 1-safeness. The main result of this section applies only to 1-safe Petri nets where behavioral conflicts can occur only between controllable transitions, i.e., to Petri nets with all uncontrollable transitions being conflict-free. This restriction rules out conflicts between two uncontrollable transitions as well as conflicts between controllable and uncontrollable transitions. Under this restriction, we will prove that liveness implies observable liveness. As a preparation of this result that will be given in Theorem 3.1.18, we need a couple of lemmas.

Lemma 3.1.15. *Let t be a transition of N, and let m be a reachable marking of N. All occurrence sequences σt enabled at m such that σ contains only conflict-free transitions and σt is minimal towards t lead to the same marking and have the same length.*

Proof. Consider two occurrence sequences σ_1 and σ_2, both enabled at m, both containing only conflict-free transitions, such that both $\sigma_1 t$ and $\sigma_2 t$ are minimal towards t. We will show that σ_1 and σ_2 lead to the same marking and have equal length and proceed by induction on the length of σ_1.

Base: The occurrence sequence σ_1 is empty if and only if t is enabled at m. In this case, σ_2 is empty as well.

Step: Assume that t is not enabled at m.

We will first show that there is a transition u, enabled at m and thus satisfying $u \neq t$, which appears in σ_1 as well as in σ_2. We proceed indirectly and assume the contrary.

We decompose $\sigma_2 t$ as $\sigma_2' \sigma_2''$ such that no transition of σ_2' occurs in $\sigma_1 t$ and the first transition in σ_2'', say v, occurs in $\sigma_1 t$. The occurrence sequence σ_2'' is not empty because both $\sigma_1 t$ and $\sigma_2 t$ contain t. If $v = t$ then v is not enabled at m by assumption of the induction step. If $v \neq t$ then v is not enabled at m by the assumption of the indirect proof. We decompose $\sigma_1 t$ as $\sigma_1' \sigma_1''$ such that σ_1'' begins with the first occurrence of v in $\sigma_1 t$.

Since v is not enabled at m, some place $s \in {}^\bullet v$ satisfies $m(s) = 0$. Since v is enabled after σ_1' and also after σ_2', s carries a token after the occurrence of σ_1', and it carries a token after the occurrence of σ_2'.

The sets of occurring transitions in σ_1' and σ_2' are disjoint by definition of σ_2', and both occurrence sequences are enabled at m. By conflict-freeness of all transitions in σ_1' and by Lemma 3.1.13, m enables $\sigma_1' \sigma_2'$. Since both σ_1' and σ_2' add a token to the place s, the marking obtained by $\sigma_1' \sigma_2'$ assigns two tokens to s, contradicting 1-safeness. So we can conclude that some transition u is enabled at m and occurs in both sequences σ_1 and σ_2.

Now let $\sigma_1 = \mu_1' u \mu_1''$ and $\sigma_2 = \mu_2' u \mu_2''$ such that u neither occurs in μ_1' nor in μ_2'. By conflict-freeness of u (all transitions in σ_1 are conflict-free) and by Lemma 3.1.11, the marking m enables the occurrence sequences $u\mu_1'\mu_1''$ and $u\mu_2'\mu_2''$. Clearly, the occurrence sequence $u\mu_1'\mu_1''t$ is also minimal towards t, because every transition that could be postponed in it could be postponed in $\sigma_1 t$ as well, and the same holds for $u\mu_2'\mu_2''t$. Moreover, if $m[u\rangle m'$, both occurrence sequences $\mu_1'\mu_1''t$ and $\mu_2'\mu_2''t$ are minimal towards t with respect to the marking m'. So we can apply the induction hypothesis to m' and to $\mu_1'\mu_1''$ and $\mu_2'\mu_2''$ and obtain that both occurrence sequences lead from m' to the same marking and have equal length. The result follows by the occurrence rule because $\sigma_1 = \mu_1'u\mu_1''$ and $\sigma_2 = \mu_2'u\mu_2''$. □

We aim at proving that liveness of a 1-safe Petri net implies observable liveness, provided the only conflicts that can appear are between controllable transitions. Although this result might seem obvious at first sight, its proof is surprisingly involved. The core argument of the proof is that, in a live Petri net, for each transition t, every reachable marking m enables an occurrence sequence σ_m with an occurrence of t. If t is observable, then observable liveness requires that we can force t to occur by providing a suitable response function φ_t which controls the behavior whenever there is a conflict. So an obvious idea is to define φ_t in such a way that, at each reached marking, the next transition in σ_m is responded, if this transition is controllable. However, φ_t does not depend on markings, but on observed sequences. That means, instead of the marking m, the user only knows the sequence of observable transitions within the initially enabled occurrence sequence σ_0 that leads to m. For this observed sequence, there might exist many occurrence sequences including unobservable transitions, and hence many different reached markings m, and so also many different occurrence sequences σ_m. Therefore, instead of the unknown marking m or the unknown occurrence sequence σ_0, we consider the set of all occurrence sequences μ_0 satisfying $\overline{\mu_0} = \overline{\sigma_0}$. Among these sequences we concentrate on minimal ones, in a sense to be defined more precisely. We will show in Lemma 3.1.16 that all these minimal occurrence sequences lead to the same marking, which we will call $m_{\overline{\sigma_0}}$. We will define a response function φ_t in such a way that all φ_t-fair occurrence sequences enabled at $m_{\overline{\sigma_0}}$ eventually contain t. Finally we will use that m, the marking reached by the occurrence of σ_0, is reachable from $m_{\overline{\sigma_0}}$ without firing any observable transition. Therefore, each concatenation of an occurrence sequence without observable transitions from $m_{\overline{\sigma_0}}$ to m and a φ_t-fair occurrence sequence enabled at m is a φ_t-fair occurrence sequence enabled at $m_{\overline{\sigma_0}}$. So the entire sequence contains an occurrence of t. Since the prefix from $m_{\overline{\sigma_0}}$ to m contains only unobservable transitions, t occurs in the second part of the sequence, which will establish the result.

We continue with a definition and two technical lemmas which are used in the proof of Theorem 3.1.18. The first generalizes the notion of minimality towards a transition.

Definition 3.1.6. Let σ be a finite occurrence sequence of N, enabled at a reachable marking m_1 of N. Assume $\overline{\sigma} = u_1 u_2 \dots u_k$ and $\sigma = \sigma_1 u_1 \sigma_2 u_2 \dots u_k \sigma_{k+1}$. The occurrence sequence σ is called *minimal towards observable transitions* if, for $1 \leqslant i \leqslant k$ and $m_1[\sigma_1 u_1 \dots \sigma_i u_i\rangle m_{i+1}$, the occurrence sequence $\sigma_i u_i$ enabled at m_i is minimal towards u_i.

Lemma 3.1.16. *Assume that all uncontrollable transitions of N are conflict-free. If a reachable marking m of N enables an occurrence sequence σ then m also enables an occurrence sequence $\tau\tau'$ such that*

- $\overline{\tau} = \overline{\sigma}$,
- *τ is minimal towards observable transitions, and*
- *both occurrence sequences σ and $\tau\tau'$ lead to the same marking.*

Proof. Let $\overline{\sigma} = u_1 u_2 \dots u_k$ and $\sigma = \sigma_1 u_1 \sigma_2 u_2 \dots u_k \sigma_{k+1}$.

For each i, $1 \leqslant i \leqslant k$,

> we repeat the following transformation exhaustively:
>
> If a transition in σ_i can be postponed, i.e., if $\sigma_i = \sigma_i' v \sigma_i''$ for some transition v such that $\sigma_i' \sigma_i'' u_i$ can also occur after $\sigma_1 u_1 \dots \sigma_{i-1} u_{i-1}$, then we replace $\sigma_i u_i$ in σ by $\sigma_i' \sigma_i'' u_i v$. This sequence is enabled at the marking reached by $\sigma_1 u_1 \dots \sigma_{i-1} u_{i-1}$, and it leads to the same marking as $\sigma_i u_i$, by Lemma 3.1.14.

Since each instance of this transformation shifts an observable transition to the left, the transformation can be performed only finitely often. Once it cannot be applied anymore, the minimal prefix including all observable transitions, τ, is obviously minimal towards observable transitions. Let τ' be the remaining sequence. Since each single transformation is a simple permutation of the considered sequence, $\tau\tau'$ is a permutation of σ. Therefore, by the occurrence rule, both occurrence sequences lead to the same marking. □

The next lemma roughly states that, if a reachable marking m enables any occurrence sequence of conflict-free transitions followed by an arbitrary transition t, then, for an appropriate response function φ, every φ-fair occurrence sequence enabled at m contains t.

Lemma 3.1.17. *Assume that all uncontrollable transitions of N are conflict-free. Let $m_0[\sigma_0\rangle m[\sigma\rangle m'[t\rangle m''$ be an occurrence sequence of N such that σ contains only uncontrollable transitions and σ does not contain t (t can be controllable or uncontrollable). Let φ be a response function which assigns, for each prefix σ_1 of σ, the singleton set $\{t\}$ to $\overline{\sigma_0\sigma_1}$ if t is controllable, and the empty set to $\overline{\sigma_0\sigma_1}$ if t is not controllable. If μ is a φ-fair occurrence sequence enabled at m then μ eventually contains an occurrence of t.*

Proof. We proceed by induction on the length of σ.

Base: If σ is the empty sequence then m enables t. Either t is uncontrollable and thus conflict-free, or t is controllable and the only transition selected by the response function φ. By φ-fairness of μ, t occurs in μ.

Step: Assume that σ is not the empty sequence. Then σ begins with an uncontrollable and hence conflict-free transition u; let $\sigma = u\sigma'$. By φ-fairness, u also occurs in μ; let $\mu = \mu' u \mu''$ such that u does not occur in μ'. By Lemma 3.1.11, m also enables $u\,\mu'\,\mu''$. Now we apply the induction hypothesis to the marking obtained by firing u at m and to the occurrence sequence σ' and derive that t occurs in $\mu'\mu''$, and therefore also in μ. □

Theorem 3.1.18. *Let N be a 1-safe Petri net with observable and controllable transitions such that all uncontrollable transitions are conflict-free. If N is live then it is observably live.*

Proof. We assume that N is live and have to prove observable liveness, i.e., observable liveness of all observable transitions. So let t be an observable transition. To show observable liveness of t, we have to provide a response function φ_t such that, for each $m_0[\sigma_0\rangle m$, each φ_t-fair occurrence sequence σ enabled at m eventually contains t.

Since N is live, each reachable marking enables an occurrence sequence μ such that t occurs in μ. Hence, each reachable marking has a well-defined *distance* to t, being the minimal number of controllable transitions that occur in such sequences μ. If t is controllable, then this distance is at least one for each reachable marking.

We assume an arbitrary fixed total order $\prec$ on the set of controllable transitions, i.e., if u and v are distinct controllable transitions then either $u \prec v$ or $v \prec u$. Let m' be a reachable marking with distance $k \geqslant 1$ to transition t. We now consider all occurrence sequences enabled at m' with k controllable transitions and an occurrence of t. Let U be the set of controllable transitions that occur in any of these sequences as the respective first controllable transition and let u be the smallest (w.r.t. $\prec$) transition in U. Then the function $Next$ assigns transition u to the marking m'. Intuitively speaking, once m' is reached, u is the next controllable transition to be fired.

Let μ_0 be an occurrence sequence enabled at m_0. By Lemma 3.1.16, m_0 also enables an occurrence sequence $\tau\tau'$ such that $\overline{\mu_0} = \overline{\tau}$ and τ is minimal towards observable transitions. Let $\overline{\tau} = u_1 \dots u_k$. Then $\tau = \tau_1\, u_1 \dots \tau_k\, u_k$ such that, for $1 \leqslant i \leqslant k$, the sequence $\tau_i u_i$ is minimal towards u_i. By repeated application of Lemma 3.1.15, the markings reached after $\tau_1\, u_1$, after $\tau_1 u_1 \tau_2 u_2, \dots$, and finally after τ only depend on their observable subsequences. In particular, the marking reached by τ only depends on $\overline{\tau}$, which equals $\overline{\mu_0}$. For each initially enabled occurrence sequence μ_0, we will denote this unique marking by $m_{\overline{\mu_0}}$.

Now we are ready to define φ_t: $\varphi_t(\overline{\mu_0}) = \{u\}$ if the distance of $m_{\overline{\mu_0}}$ to t is at least 1 and $Next(m_{\overline{\mu_0}}) = u$, and $\varphi_t(\overline{\mu_0}) = \emptyset$ if the distance of $m_{\overline{\mu_0}}$ to t is 0.

If $Next(m) = u$ for a reachable marking m then, by the definition of $Next$, there is an occurrence sequence $m[\tau u\rangle m'$ such that only uncontrollable transitions occur in τ. Assume that m also enables an uncontrollable transition v. If v occurs in τ, i.e., if $\tau = \tau' v \tau''$ with no occurrence of v in τ', then m also enables $v\tau'\tau'' u$ by Lemma 3.1.11. If v does not occur in τ then m also enables $\tau u v$ by Lemma 3.1.12. Therefore, the value of the $Next$ function does not change by the occurrence of v. As a consequence, the value of φ_t is also only changed by the occurrence of controllable transitions.

Coming back to the core of the proof, now let $m_0[\sigma_0\rangle m$. As mentioned before, we have to show that each φ_t-fair occurrence sequence σ enabled at m contains an occurrence of t. We proceed in three steps:

(1) We will show that there exists a φ_t-fair occurrence sequence enabled at $m_{\overline{\sigma_0}}$ which contains an occurrence of t and is moreover minimal towards observable transitions.

(2) We will show that each φ_t-fair occurrence sequence enabled at $m_{\overline{\sigma_0}}$ contains an occurrence of t.

(3) We will show that, for each φ_t-fair occurrence sequence σ enabled at m, there is an occurrence sequence τ which does not contain an occurrence of t, leading from $m_{\overline{\sigma_0}}$ to m, such that $\tau\,\sigma$ is φ_t-fair as well.

By (2) and (3), each φ_t-fair occurrence sequence σ enabled at m is a suffix of a φ_t-fair occurrence sequence enabled at $m_{\overline{\sigma_0}}$ and contains an occurrence of t, which establishes the result.

It remains to provide proofs of the three claims above.

(1) Since N is live, $m_{\overline{\sigma_0}}$ enables an occurrence sequence μ which contains t. We can assume that t occurs only once in μ and that it is the last transition that occurs in μ. We moreover assume without loss of generality that μ has a minimal

number of controllable transitions and that these transitions occur according to the $Next$-function, i.e., after each prefix μ_1 of μ, the next controllable transition that occurs in μ is equal to the single element in $\varphi_t(\overline{\sigma_0\mu_1})$. Recall that, if t is not controllable, then φ_t returns the empty set after firing the last controllable transition. We can moreover assume that μ is minimal towards observable transitions according to Lemma 3.1.16 (otherwise we transform the sequence as in the proof of Lemma 3.1.16). After the occurrence of μ, we continue with an arbitrary φ_t-fair occurrence sequence. By the definition of φ_t-fairness, the entire occurrence sequence is φ_t-fair.

(2) Let μ be the occurrence sequence defined in the proof of (1), and let $\mu = \mu_1 u_1 \mu_2 u_2 \ldots \mu_k u_k \mu_{k+1}$ such that $u_1 \ldots u_k$ are controllable transitions and $\mu_1 \ldots \mu_{k+1}$ are sequences of uncontrollable transitions.

If t is controllable then $t = u_k$ and μ_{k+1} is the empty sequence. Otherwise, t is the last transition in μ_{k+1}. It is possible that $k = 0$; then μ contains no controllable transitions at all and $\mu = \mu_1$.

Let τ be an arbitrary φ_t-fair occurrence sequence enabled at $m_{\overline{\sigma_0}}$. We claim that τ also contains t.

First we show that all the controllable transitions $u_1 \ldots u_k$ of μ also occur in τ. If $k = 0$, nothing has to be shown. So assume that $k \geqslant 1$.

Since $\varphi_t(\overline{\sigma_0}) = \{u_1\}$ and thus $\varphi(\overline{\sigma_0\tau'}) = \{u_1\}$ for each prefix τ' of τ without controllable transitions, only u_1 can be the first controllable transition in τ. By Lemma 3.1.17, u_1 actually occurs in τ. Let $\tau = \tau_1\, u_1\, \tau_2$ such that τ_1 contains no controllable transitions. By Lemma 3.1.14, we can decompose $\tau_1 u_1$ in subsequences τ_1' and τ_1'' such that $\tau_1'\, u_1$ is minimal towards u_1 and $m_{\overline{\sigma_0}}$ enables $\tau_1' u_1 \tau_1'' \tau_2$. The occurrence sequence μ is minimal towards observable transitions. So its prefix $\mu_1\, u_1$ is minimal towards u_1. By Lemma 3.1.15, the occurrence sequences $\tau_1' u_1$ and $\mu_1 u_1$ lead to the same marking. The same argument can now be applied to $\tau_1'' \tau_2$, enabled at this marking, yielding subsequences τ_2' and τ_2'' such that $\tau_2' u_2$ leads to the same marking as $\mu_2 u_2$. Repeating this argument, we eventually obtain the occurrence sequence $\tau_1' u_1 \tau_2' u_2 \ldots \tau_k' u_k \tau_k'' \tau_{k+1}$, still a permutation of τ, enabled at $m_{\overline{\sigma_0}}$. If t is controllable then we are finished because, in this case, $t = u_k$.

So it remains to consider the case that t is not controllable. By the above arguments, the marking reached after $\tau_1' u_1 \ldots \tau_k' u_k$ equals the marking reached after $\mu_1 u_1 \ldots \mu_k u_k$. We have $\varphi_t(\overline{\sigma_0 \tau_1' u_1 \ldots \tau_k' u_k}) = \emptyset$, and the same holds when uncontrollable transitions are appended to this occurrence sequence. By Lemma 3.1.17 and since μ_{k+1} contains an occurrence of t, so does $\tau_k'' \tau_{k+1}$.

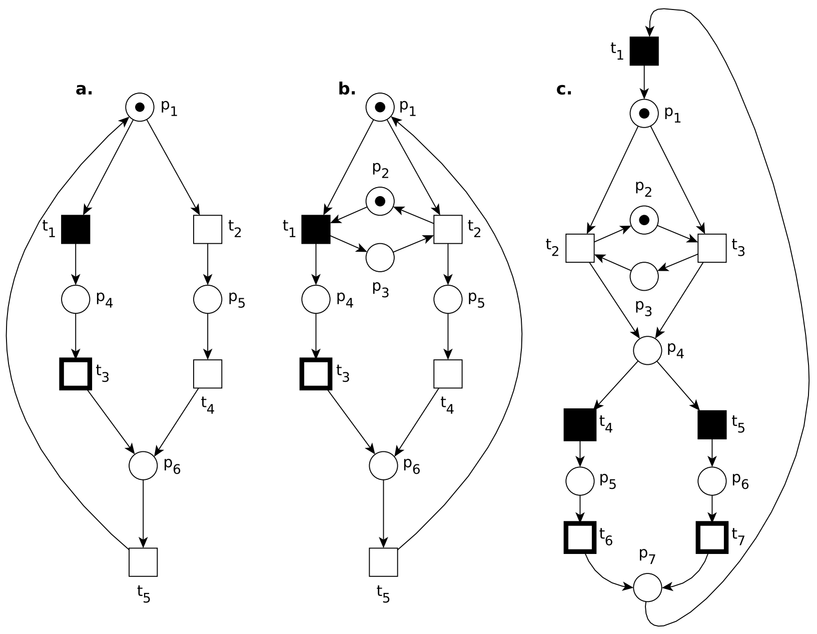

FIGURE 3.4: **a**: a net with a conflict, **b**: a conflict-free net, **c**: a net which is conflict-free w.r.t. its uncontrollable transitions

(3) Let σ be a φ_t-fair occurrence sequence enabled at m. By Lemma 3.1.16, σ_0 can be replaced by a permutation $\sigma_0'\sigma_0''$ such that σ_0' is minimal towards observable transitions, σ_0'' contains only unobservable transitions, and $\sigma_0'\sigma_0''$ also leads to m. The marking reached by σ_0' is $m_{\overline{\sigma_0}}$ because σ_0' contains all observable transitions of σ. Therefore, since σ_0'' contains only unobservable transitions and by the definition of φ_t-fairness, $\sigma_0''\sigma$ is a φ_t-fair occurrence sequence enabled at $m_{\overline{\sigma_0}}$. Again since σ_0'' contains only unobservable transitions, σ_0'' does not contain the observable transition t.

□

Example 3.1.3. In Figure 3.4, we see one net with a conflict and a conflict-free net. The net shown in Figure 3.4.a includes a conflict between a controllable transition and an uncontrollable transition (which is also unobservable). Although the net is live, since we cannot force t_1 to fire, both t_1 and t_3 are not observably live and so the net is not observably live. The net shown in Figure 3.4.b has no conflicts. It is both live and observably live. The net shown in Figure 3.4.c is conflict-free w.r.t. all its uncontrollable transitions, but there is a conflict between the controllable transitions t_4 and t_5. We can choose the related controllable transition in order to observe the occurrence of any observable transitions. The uncontrollable part of the machine behaves deterministically. This net is both live and observably live. □

3.1.6 Observable liveness and supervisory control

The introduced approach has many similarities to supervisory control of Discrete Event Systems (DESs). A DES is discrete in time and (usually) in state space. It is asynchronous or event-driven and it may be nondeterministic: that is, capable of choices by internal chance or other mechanisms not necessarily modeled by the system analyst. DESs require control and coordination to ensure they behave well and satisfy some properties. Being controlled (or potentially controllable) dynamic systems, discrete event systems are a subject for control theory. The most interesting and original approach to solve the control problem of discrete event systems is Supervisory Control Theory originated by the work of Ramadge and Wonham [56]. According to the paradigm of Supervisory Control, a discrete event system is a language generator whose behavior i.e., its language, is given by a set of traces of events. Given a legal language, the basic control problem is to design a supervisor that restricts the closed loop behavior of the plant, by disabling only controllable events. The events whose occurrence cannot be disabled are called uncontrollable. It is usually required that the closed loop system satisfies additional properties such as non-blocking. This notion can be considered to be similar to observable liveness, but the two notions have significant differences. A system is called non-blocking if a final state (of a given set of final states) is always reachable, whereas observable liveness refers to observable transitions. If a deadlock state is final, a non-blocking system can even run in this deadlock state, whereas an observably live Petri net can not (provided it has at least one observable transition).

Although our liveness notion is defined on systems with distinguished controllable and observable actions, an important difference with the above approach is that we do not aim at controlling a system to avoid certain illegal states or to restrict the observable language. Moreover, the typical approaches of supervisory control are based on a state space analysis of the unobservable part of a system whereas we do not assume that this state space is finite in general. Conceptually, the main difference is that we do not aim at constructing a control such that the composition of the original system and the control is a composed system with correct behavior, but we rather consider the system without control and the input-output behavior of its unobservable parts.

3.1.7 Further discussions on observable liveness

We provided a novel liveness notion, called observable liveness for a variant of Petri nets with observable transitions, where an observable transition can also be controllable. In analogy to the usual definition of liveness of a Petri net, observable liveness roughly means that a user can always enforce the occurrence of any observable transition, only by stimulating the net by choosing appropriate enabled controllable transition. It is necessary to assume that the uncontrollable part of a net proceeds, i.e., we assume that the net behaves weakly fair with respect to uncontrollable transitions.

Our approach has some similarities to *T-liveness* as presented in [45] which, roughly speaking, equals to liveness of a subset of transitions, but does not consider observable or controllable transitions. However, we do not require liveness of (only) observable transitions, but observable liveness of these transitions.

Observable liveness expresses serviceability of distributed systems. What we call serviceability is related to the responsiveness notion studied in [73]. The difference is that responsiveness is defined for open systems. It guarantees that an open system and its environment always have the possibility to communicate.

With observable liveness, we also get a kind of diagnosability property which is related to – but not the same as – the diagnosability notion in the literature (see [22, 64]). For explaining our use of diagnosability, we provide a small example. Assume that we have a machine which is modeled and verified to be observably live. When the machine stops working, the system administrator can test and find the faulty part by using the buttons of the machine (controllable transitions). Since the machine is verified to be observably live, there is always a way to observe each observable transition. The combination of buttons to be pressed to observe each observable transition only depends on the previously observed activities. In [64], Sampath et al. require that the system is live and that additionally there exists no loop of unobservable events. By observable liveness, we weaken this requirement on loops of unobservable events. Instead, an unobservable loop is only allowed when it does not prevent firing of any observable transition.

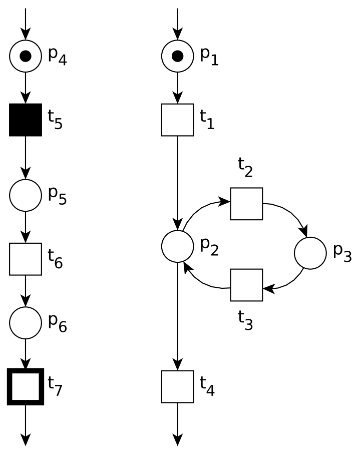

FIGURE 3.5: a net fragment with an unobservable loop.

Example 3.1.4. In Figure 3.5, the net fragment is observably live because the unobservable loop in the right part does not affect any of the observable transitions, while the existence of the unobservable loop in Figure 3.2.a on page 19 violates weak fairness and observable liveness. □

Our notion of observable liveness is based on occurrence sequences, i.e., on a sequential view of the behavior of concurrent systems. This has the advantage of a simple definition of behavior, and it is justified because concurrency of events plays no particularly important role for liveness of single transitions. However, a concurrent view also has several advantages. For example, a partial order view would have obvious advantages to capture the progress assumption (that we called weak fairness) in a natural way [24, 42]; a maximal process net models a weakly fair net by definition because even an infinite process net can be extended by a persistently enabled transition. For liveness, as it is considered in this thesis, we need to consider also the branching structure of behavior. Therefore, a partial order view of single runs, as provided e.g. by the well-known notion of Petri net processes, would not be sufficient. To additionally capture the branching structure of behavior, Petri net unfoldings (see [27] and [46]) are considered. Formally, an unfolding is an acyclic graph with nodes representing transition occurrences and tokens, where the single elements are either sequential, concurrent or in conflict. Using this notion, our admittedly cumbersome notion of a φ-fair occurrence sequence would be translated to a maximal unfolding starting with a given marking and respecting the respective response function. As mentioned before, the concurrent view also helps to understand the concepts in the section on deterministic uncontrollable behavior. Actually, determinism itself, i.e., lack of conflicts between uncontrollable transitions, is very easy to define for unfoldings: no forward branching condition contains an event in its post-set representing the occurrence of an uncontrolled transition.

We will consider generalizations of Theorem 3.1.18 in the future work. It still holds when there is some limited nondeterminism in the uncontrolled part; for example, if two alternative uncontrollable transitions cause the same marking transformation, the result is not spoiled. More generally, we aim at defining an equivalence notion on nets, based on the respective observed behavior, which preserves observable liveness. Reduction rules, as defined e.g. in [12], [42] and [23] but also in many other papers, could be applied to the uncontrollable part, leading to simpler but equivalent nets. However, there are obvious additional rules. For example, a rule that deletes an unobservable dead transition is sound with respect to the equivalence because dead uncontrollable transitions do not contribute to the observable liveness or non-liveness of the considered net.

Finally, equivalence notions on (arbitrary) models are frequently evaluated with respect to certain relevant properties in the following sense: if a model is replaced by an equivalent one, then the property should either hold for both models or for none. This is particularly relevant for models which are part of other models, i.e., in compositional settings. Then the requirement is that replacing a sub-model by an equivalent one respects the property of the entire model. If this holds, the equivalence is called congruence with respect to the considered property. Among these congruences, one is often interested in the coarsest one, which canonically corresponds to the property under consideration. An example of this kind of result is given in [72], which proves that the equivalence provided by Failure Semantics preserves deadlock-freeness. In our context, we aim at an equivalence between observable nets such that two equivalent nets are either both observably live or both not observably live. An example of [55] shows that to this end we need at least Testing Equivalence (defined for classical 1-bounded nets in [55]) which distinguishes loops of unobservable sequences (which often spoil observable liveness) from absence of these loops.

3.2 Weak Observable Liveness

One might argue that observable liveness notion is too strong in some cases such as the one explained in the example below.

Example 3.2.1. In the net in Figure 3.6, u is not observably live since there exists a reachable marking, $m_0[\sigma_0\rangle m$ from which u cannot fire. However, if the user chooses to fire t_1 then u fires infinitely often. And since t_1 is in conflict with a controllable transition (not an uncontrollable one), this choice is really up to the user.

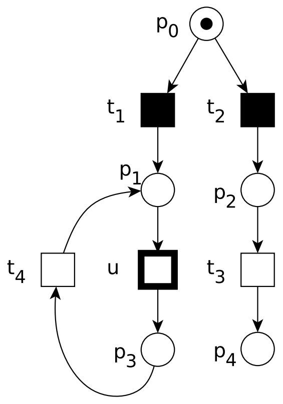

FIGURE 3.6: A not observably live net

□

In observable liveness, we consider that a user can decide to fire an observable transition t at any reachable marking, thus we have to consider the possibility of forcing the system to fire t from each reachable marking. The net in Figure 3.6 is not observably live because the user might decide to fire u after he/she reaches the marking $\{p_4\}$ from which there is no possibility to fire u. However, it can be the case that the user knows that he/she wants to fire t at the beginning. So, given a net N and $t \in O$, the user decides to fire t at the beginning (at m_0), and whenever the user can make a choice he/she makes the right choice in order to fire t eventually. Thus, with this assumption, a marking which does not respect the response function is never reached. In this case, we do not have to consider each reachable marking as we do for deciding observable liveness of t.

We relax the observable liveness notion for such cases by defining a weaker notion assuming that one can be interested in observable liveness of a transition under the assumption of knowing the aimed transition at the beginning and always respecting to the response function. Here we introduce weak observable liveness which relies on the assumption that the user knows at the beginning that he/she wants to fire t and always respects to the response function. In this case, we will only consider the markings that are reached by the sequences with respect to the response function.

Let $N = (P, T, F, m_0)$ be a 1-safe Petri net, such that $O \subseteq T$ is the set of all observable transitions and $C \subseteq O$ is the set of all controllable transitions. Let $\varphi : O^* \to 2^C$ be the response function which formalizes the strategy of a user who aims to force the net to fire a specific observable transition.

Definition 3.2.1. Let $\sigma = u_0 u_1 \dots u_i \dots$ be an infinite sequence of transitions, enabled at m_0, with $m_0[u_0\rangle m_1[u_1\rangle \dots m_i[u_i\rangle \dots$. Let t be a transition. We say that t is *finally postponed* in σ if, from some point on, it is always enabled, but never fires:

$$\exists j \geqslant 0 : \forall k \geqslant j : m_k[t\rangle \text{ and } u_k \neq t$$

Let $\sigma_k = u_0 u_1 \dots u_k$ be the finite prefix of σ of length $k+1$. We say that $t \in C$ is *finally eligible* in σ if

$$\exists j \geqslant 0 : \forall k \geqslant j : t \in \varphi(\overline{\sigma_k})$$

Define $\underline{\sigma}$ as the projection of σ on the set C of controllable transitions.

Definition 3.2.2. A transition sequence σ is *consistent with* a response function φ if it satisfies the three following clauses:

1. σ is weakly fair with respect to $T \setminus C$
2. for each prefix $\sigma_k t$ of σ, if $t \in C$, then $t \in \varphi(\overline{\sigma_k})$

3. if $\underline{\sigma}$ is finite, then no controllable t is finally postponed and finally eligible in σ

Definition 3.2.3. An observable transition $u \in O$ is *weakly observably live* iff there exists a response function φ such that in each maximal transition sequence that is consistent with φ, u appears infinitely often. N is weakly observably live iff each $u \in O$ is weakly observably live.

Example 3.2.2. Transition u in Figure 3.6 is not observably live, whereas it is weakly observably live. Let ϵ be the empty transition sequence. There is a response function φ such that $\varphi(\epsilon) = \{t_1\}$ such that each transition sequence that is consistent with it includes infinitely often u. In this example there is only one maximal transition sequence that is consistent with φ and it is $t_1(ut_4)^\omega$. So u is weakly observably live. Let us consider

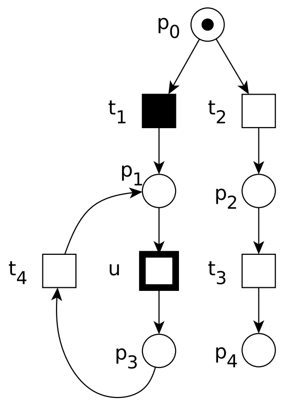

FIGURE 3.7: A not observably live net with a weakly observably live transition

the same net but change the controllability of transition t_2 as in Figure 3.7. Now u is not weakly observably live anymore since t_2t_3 is a consistent transition sequence and it does not include u. $\square$

Theorem 3.2.1. *Let σ be an infinite transition sequence enabled at m_0 and let φ be a response function. If σ is consistent with φ then σ is φ-fair at m_0.*

Proof. Let σ be consistent with φ. We will show that σ satisfies clauses (1)-(4) of Definition 3.1.1. Condition (1) is satisfied by the definition of "consistent".

Suppose $\overline{\sigma}$ is finite. $\sigma = \sigma_b t_f \sigma_a$ where t_f is the last observable transition in σ. Then $\overline{\sigma_a} = \epsilon$. This implies that also $\underline{\sigma_a} = \epsilon$. Let $t \in \overline{\sigma} = \overline{\sigma_b t_f}$. By clause (3) of Definition 3.2.2 σ is weakly fair with respect to any $t \in \varphi(\overline{\sigma})$. This proves clause (2) of Definition 3.1.1.

Suppose $\overline{\sigma}$ is infinite and let t be finally eligible in σ. By clause (3) of Definition 3.2.2 t is not finally postponed which implies that σ is weakly fair with respect to t. This proves clause (3) of Definition 3.1.1.

Considering condition (4) let σ be an infinite composition $\sigma_1 t_1 \sigma_2 t_2 \dots$ such that no σ_i includes any controllable transitions and $t_i \in C$ for $i \geqslant 1$. Since σ is consistent with φ, $t_i \in \varphi(\overline{\sigma_1 t_1 \sigma_2 t_2 \dots \sigma_i})$; let now σ be a finite composition $\sigma_1 t_1 \sigma_2 t_2 \dots t_k \sigma_{k+1}$ such that no σ_i includes any controllable transitions and $t_i \in C$ for $i \geqslant 1$. Again since σ is consistent with φ, $t_i \in \varphi(\overline{\sigma_1 t_1 \sigma_2 t_2 \dots \sigma_i})$.

□

Corollary 3.2.2. *If t is weakly observably live starting from any $m \in [m_0\rangle$ then t is observably live.*

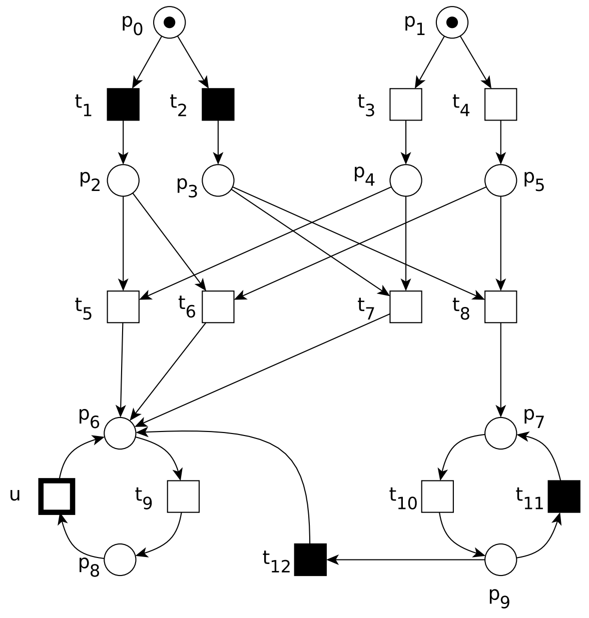

Figure 3.8: An example net for weak observable liveness

Example 3.2.3. Let us consider the observable transition u in the net in Figure 3.8. u is observably live and thus also weakly observably live. There is a response function φ such that for each reachable marking $m_0[\sigma_0\rangle m$, each φ-fair occurrence sequence enabled at m includes u infinitely often. In other words, from each reachable marking, user can force the system to fire u by using controllable transitions. This fact already implies that there exists a response function φ such that each maximal transition sequence that is consistent with φ includes u infinitely often. More intuitively, for weak observable liveness we only consider the markings reached by the transition sequences that are consistent with φ.

Let us consider the observable liveness (as it is defined in Section 3.1) and weak observable liveness of u for the case in which t_{12} is deleted from the net. When t_{12} is deleted, there are reachable markings in the net from which the user cannot force the occurrence of u. However, u is still weakly observably live, since those markings which violate the observable liveness are not consistent with u and hence it is not required that they must include occurrence of u infinitely often. Since the transition sequences starting with t_2 are not consistent with φ the user will not choose to fire t_2, and then, whatever the system does, u will occur infinitely often. □

3.3 Checking Observable Liveness

In the previous section two new notions, observable liveness and weak observable liveness, are introduced. These notions allow one to express serviceability of partially observable/controllable distributed systems. Intuitively, a system is observably live if, no matter what marking is reached, all the observable actions can be forced to eventually fire by choosing and performing the right controllable actions on the basis of the observed behavior. In other words, a system is observably live, if the user has a strategy, from each reachable marking, for forcing each observable transition to eventually fire by using the controllable transitions. In weak observable liveness, we weaken the requirement of having a strategy from each reachable marking, and we consider the cases in which the user decides to fire an observable transition at the beginning and respects the strategy from the beginning, i.e. he/she never chooses to fire any transition which is not consistent with the response function. Thus, we do not consider each reachable marking, but only the ones reached by the transition sequences which are consistent with the response function. So having a strategy that leads an observable transition t to fire eventually is enough to say that t is weakly observably live. In this section, we describe a potential application of infinite games that are played on finite graphs for checking weak observable liveness. The method only applies to a restricted class of Petri nets which is closer to the setting of two-person infinite games. In the following we will first give a basic background for two-person infinite games and ω-automata on which these games are formalized. Later we will discuss the idea of using two-person infinite games for checking weak observable liveness on a concrete example. The discussion will be informal and it is not fully developed yet.

3.3.1 Infinite games on finite graphs

The definitions in this section are adapted from [38].

Streett games are infinite games that are played on finite graphs with two players [38, 50]. The game is based on Streett automata which are non-deterministic ω-automata [17].

An ω-automaton is a type of finite automaton that runs on infinite strings as input. ω-automata have a variety of acceptance conditions, they are useful for specifying behavior of systems that are not expected to terminate, such as hardware, operating systems and control systems. They are well suited for specifying properties in the form: for each request, eventually an acknowledgement is given.

Here we fix some notation that will be used in the sequel. Σ denotes a finite alphabet. Σ^* denotes the set of all finite words over Σ whereas Σ^ω denotes the set of infinite words (ω-words). A set of ω-words over a given alphabet is called *ω-language*.

Classes of ω-automata include Rabin automata, Streett automata, parity automata, Büchi automata and Muller automata. For each, deterministic or non-deterministic types exist. These classes of ω-automata differ only in terms of the acceptance condition (acceptance component). In this section, we will examine only non-deterministic Streett automata.

Definition 3.3.1. An *ω-automaton* is a quintuple $(Q, \Sigma, \delta, q_0, Acc)$ where Q is a finite set of states, Σ is a finite alphabet, δ: $Q \times \Sigma \to 2^Q$ is the non-deterministic state transition function, $q_0 \in Q$ is the initial state and Acc is the *acceptance component*. The acceptance component varies according to the type of automata.

Definition 3.3.2. Let $\mathscr{A} = (Q, \Sigma, \delta, q_0, Acc)$ be an ω-automaton. A *run* on an ω-word $\alpha = a_1a_2... \in \Sigma^\omega$ is an infinite state sequence $\varrho = \varrho(0)\varrho(1)\varrho(2)... \in Q^\omega$, such that the following conditions hold:

- $\varrho(0) = q_0$,
- $\varrho(i) \in \delta(\varrho(i-1), a_i)$ for $i \geqslant 1$ if $\mathscr{A}$ is non-deterministic,

Inf(ϱ) denotes the set of infinitely often visited states during the run ϱ.

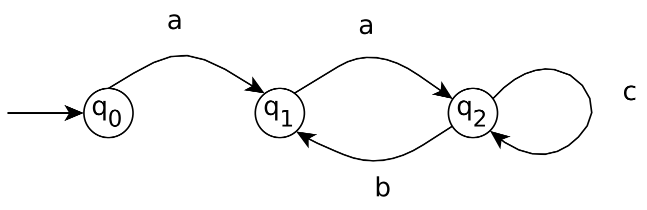

FIGURE 3.9: Example for a run on a ω-automaton.

Example 3.3.1. Let's consider the automaton in Figure 3.9 and the ω-word, $\alpha_1 = aac(ba)^\omega$. The run on α_1 is $\varrho_1 = q_0q_1q_2q_2(q_1q_2)^\omega$ and the infinitely often visited states during ϱ_1 are $\text{Inf}(\varrho_1) = \{q_1, q_2\}$.

Another example run is $\varrho_2 = q_0q_1q_2q_1q_2(q_2)^\omega$ which corresponds to the ω-word $\alpha_2 = aaba(c)^\omega$. The only infinitely often visited state during ϱ_2 is q_2, $\text{Inf}(\varrho_2) = \{q_2\}$. □

Definition 3.3.3. An ω-automaton $\mathscr{A} = (Q, \Sigma, \delta, q_0, Acc)$ with acceptance component $Acc = \{(E_1, F_1), \ldots, (E_k, F_k)$ with $E_i, F_i \subseteq Q\}$ is called Streett automaton if it is used with the following Streett acceptance condition:

A word α is accepted by $\mathscr{A}$ if there exists a run ϱ of $\mathscr{A}$ on α such that

$$\forall (E, F) \in Acc, \mathrm{Inf}(\varrho) \cap F \neq \emptyset \implies \mathrm{Inf}(\varrho) \cap E \neq \emptyset$$

Example 3.3.2. In Figure 3.10, an ω-automaton $\mathscr{A}$ is given. Let $Acc = \{(\{q_1\}, \{q_0\}), (\{q_0\}, \{q_1\})\}$ be the acceptance component. If $\mathscr{A}$ is used with Streett acceptance condition, it recognizes the ω-language L over alphabet $A = \{a, b\}$ which consists of infinite words that include infinitely many a's and infinitely many b's. $\square$

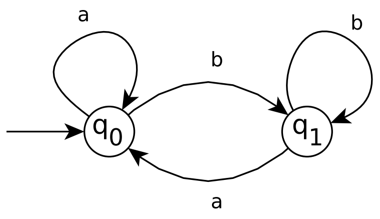

FIGURE 3.10: An ω-automaton

In order to define Streett games, we first briefly recall what is a game and how it is played formally. A game is composed of an arena and a winning condition.

Definition 3.3.4. An *arena* is a triple $A = (V_0, V_1, E)$, where $V_0 \cap V_1 = \emptyset$. Elements of V_0 are called 0-vertices and elements of V_1 are called 1-vertices. $V = V_0 \cup V_1$. Edges represent possible moves: such that $E \subseteq V \times V$. The set of successors of $v \in V$ is defined by $E(v) = \{v' \in V : (v, v') \in E\}$.

In this section, we focus on games with two players. In the sequel we will refer to the two players as *Player*-0 and *Player*-1.

A play of a game on an arena A is played in the following way:

Let $\sigma = \{0, 1\}$

- A token is placed on an initial vertex $v \in V$
- Let v be a σ-vertex,
  1. Player-σ moves the token from v to $v' \in E(v)$

2. v' can be a 0-vertex or a 1-vertex. In each case the corresponding player moves the token from v' to $v'' \in E(v')$
3. Repeated
   - either infinitely often
   - or until a dead end is reached (v is a dead end if $E(v) = \emptyset$)

Note that the same player can make several (even infinite) consecutive moves.

A play on arena A can be:

1. an infinite path $\pi = v_0v_1v_2... \in V^\omega$ with $v_{i+1} \in E(v_i)$ for all $i \in \mathbb{N}$ (infinite play) or
2. a finite path $\pi = v_0v_1v_2...v_l \in V^+$ with $v_{i+1} \in E(v_i)$ for all $i < l$, and $E(v_l) = \emptyset$ (finite play)

Let A be an arena and $\text{Win} \subseteq V^\omega$ be the *winning set*; $G = (A, \text{Win})$ is called a *game.*

Player-0 is the *winner of a play* π in the game G iff

- π is a finite play $\pi = v_0v_1...v_l \in V^+$ and v_l is a 1-vertex where Player-1 can't move anymore or
- π is an infinite play and $\pi \in \text{Win}$

Player-1 wins if Player-0 does not win π.

In general, an arena might consist of an infinite number of vertices. Hence, in order to use the acceptance conditions for ω-automata, we have to map the vertices of the arena to a finite set. These previously mentioned acceptance conditions only make sense when they are used with finite state automata since a run of an infinite state automaton might have no recurring state. For this reason, the vertices of an arena are colored by a *coloring function* which is defined as $\chi : V \to C$ where V is the set of vertices of arena A and C is a finite set of colors. Let $\pi = v_0v_1v_2...$ be a play, its coloring, $\chi(\pi)$ is $\chi(\pi) = \chi(v_0)\chi(v_1)\chi(v_2)....$ When C is viewed as the state set of a finite ω-automaton and *Acc* is an acceptance condition for this automaton, then $W_\chi(Acc)$ is the winning set consisting of all infinite plays π where $\chi(\pi)$ is accepted according to *Acc.*

Several winning conditions are defined based on acceptance conditions of finite ω-automata, here we will recall only Streett condition. *Streett winning condition* is defined as:

$$Acc = \{(E_1, F_1), \dots, (E_m, F_m)\} \text{ such that}$$
$$\pi \in W_\chi(Acc) \text{ iff } \forall k \in \{1, \dots, m\}, \mathrm{Inf}(\chi(\pi)) \cap F_k \neq \emptyset \implies \mathrm{Inf}(\chi(\pi)) \cap E_k \neq \emptyset.$$

Definition 3.3.5. Let A be an arena, $\sigma \in \{0, 1\}$ and $f_\sigma : V^*V_\sigma \to V$ a partial function. A prefix $v_0v_1...v_l$ of a play is said to be *conform* with f_σ if $\forall i$ such that $0 \leqslant i < l$ and $v_i \in V_\sigma$, the function f_σ is defined at $v_0v_1...v_i$ and we have $v_{i+1} \in f_\sigma(v_0v_1...v_i)$. This implies $v_{i+1} \in E(v_i)$.

A play is conform with f_σ if each of its prefixes is conform with f_σ.

Definition 3.3.6. Let A be an arena, $\sigma \in \{0, 1\}$ and $f_\sigma : V^*V_\sigma \to V$ a partial function. f_σ is a *strategy* for Player-σ on $U \subseteq V$ if it is defined for every prefix of a play which is conform with it, starts in a vertex from U , and does not end in a dead end of Player-σ. When U is a singleton $\{v\}$, we say f_σ is a strategy for Player-σ in v.

Definition 3.3.7. Let $G = (A, \mathrm{Win})$ be a game, and f_σ a strategy for Player-σ on U. The strategy f_σ is said to be a *winning strategy* for Player-σ on U if all plays which are conform with f_σ and start in a vertex from U are wins for Player-σ.

3.3.2 Checking weak observable liveness by Streett games

In the previous section we recalled ω-automata, Streett automata and above them Streett games. In this section, a method for checking observable liveness is proposed. The method is based on transforming weak observable liveness problem to a Streett game in which the two players are the system and the user.

ω-automata are well known and widely used for description of systems that are expected to run continuously. They are powerful for specifying properties like "for each request, eventually an acknowledge is given". Observable liveness property also requires that a system continues to run without termination. Moreover, in an observably live system, each request of service must be eventually satisfied. Since observable liveness property is based on the existence of a strategy for a user to control the system in order to force the occurrence of observable transitions, it is natural to think that observable liveness can be expressed by means of an infinite game on a finite graph with two players i.e., the system and the user.

As it has been mentioned in the previous section, there are several acceptance conditions defined for ω-automata which are later adapted to games as winning conditions. Among those, Streett games seem to be the closest translation of weak observable liveness problem as a game.

Consider a 1-safe Petri net with distinguished controllable and observable transitions as described in Section 3.1.1. We will assume progress for the behavior of uncontrollable transitions, whereas for controllable transitions we do not assume progress, i.e., a user can choose not to fire an enabled controllable transition. Moreover, the user can decide which controllable transition occurs among the enabled ones. Of course, we cannot assume that the user will be fast enough to fire a controllable transition if it is in conflict with an uncontrollable one. An observable transition t is weakly observably live if there exists a strategy for the user to force the system starting from the initial marking in order to fire t by choosing the right controllable transitions. We assume that the user will always respect the strategy while choosing controllable transitions.

Here we propose a method for translating the problem of checking weak observable liveness of an observable transition into a Streett game.

Let $N = (P, T, F, m_0)$ be a 1-safe Petri net where $O \subseteq T$ is the set of *observable* transitions and $C \subseteq O$ the set of *controllable* ones. Let $[m_0\rangle$ be set of all reachable markings of N. Let $m \in [m_0\rangle$, T_m denotes the set of all transitions that are enabled at m, $T_m = \{t \in T : [m\rangle t\}$. Let N be such that $\forall m \in [m_0\rangle, T_m \cap C = \emptyset \vee T_m \cap (T \setminus C) = \emptyset$, i.e., at each reachable marking, enabled transitions are either all controllable or all uncontrollable. The idea is to create an arena from the marking graph of N, $MG(N)$, and on this arena encode the problem of deciding whether a transition is weakly observably live as a Streett game.

The notions of observable liveness and weak observable liveness are based on observation of occurrences of transitions. However the considered games are based on moving between vertices which correspond to the states of an automaton and winning conditions are based on visiting some specific vertices infinitely often. The marking graph of a Petri net consists of states that are corresponding to reachable markings in the net whereas an arc from one state to another state corresponds to a transition that changes the marking. Occurrence of a transition cannot be deduced from the states since there might be several transitions that yield the same marking. In order to encode weak observable liveness we need to be able to represent transition occurrences as states. This can be done by dividing each arc in the marking graph into two by adding an additional state for each arc as illustrated in Figure 3.11.

After applying the transformation for each arc of $MG(N)$, we get a transformed marking graph $MG(N)'$ on which we can see the occurrences of any transition of N. Note that we assume weak fairness with respect to the uncontrollable transitions in the setting of observable liveness, i.e., each consistently enabled uncontrollable transition will eventually fire unless another transition which is in conflict fires. We assume the same behavior for controllable transitions since otherwise the user will lose because of its inactivity. Consider t in Figure 3.11, it is safe to say if state s_t is reached then t has fired

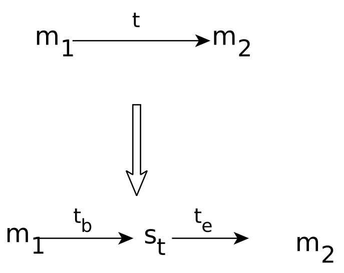

FIGURE 3.11: Transition transformation

since it cannot pause its occurrence and stay in the middle state forever. We will call a sequence of states weakly fair if the corresponding transition sequence is weakly fair.

To create an arena from $MG(N)'$ we do not need coloring because we consider only 1-safe Petri nets which always have finite marking graphs. However, $MG(N)'$ still cannot be an arena because to be an arena the states of $MG(N)'$ must be divided into two distinct sets, V_0 and V_1, which belong to Player-0 and Player-1 respectively. Since in our restricted class, we only have states in which either the user or the system can decide, it is easy to distinguish the states in $MG(N)'$. Let Player-0 be the system, and Player-1 be the user, in each state either Player-0 or Player-1 can make the move since we only consider the nets in which at any reachable marking (state of MG(N)') either enabled transitions are all controllable or all uncontrollable. So at each state in $MG(N)$ all outgoing arcs correspond to either all controllable transitions or all uncontrollable transitions. Any state in $MG(N)'$ that is added after the transformation, belongs to the same player as its predecessor state.

Now we have an arena A that is created from N through the marking graph. Let $t \in O$, it is possible to define a Streett game on A such that t is weakly observably live iff the user has a winning strategy on the vertex corresponding to the initial marking of N. For the simplicity, we define winning condition of Player-0 as: Let S_t be the new added states during the transformation corresponding to transition t.

$$\text{Win} = \{w \in V^\omega : \ w \text{ is weakly fair AND } S_t \text{ is not infinitely often visited}\}.$$

Transition t is weakly observably live iff Player-1 has a winning strategy for the game (A, Win).

Example 3.3.3. Let us consider the net in Figure 3.12. Transition E is weakly observably live. The user has a choice at the initial marking between two controllable transitions. E will fire infinitely often if the user chooses F. So the strategy of the user

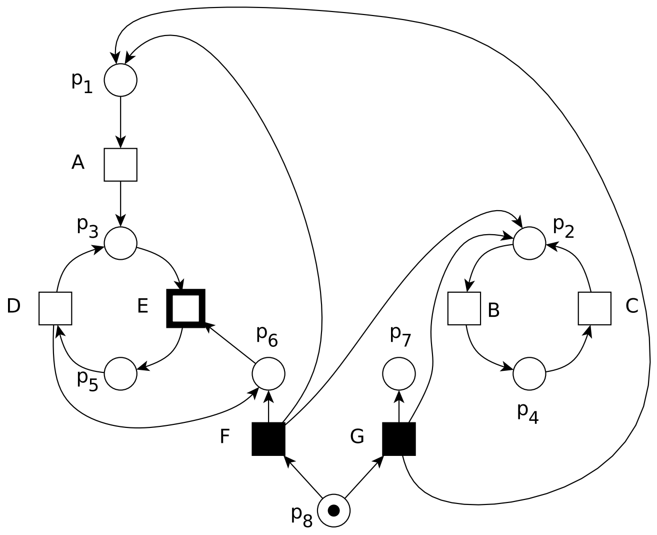

FIGURE 3.12: Is E weakly observably live?

should be choosing F, i.e., let ϵ be the empty transition sequence, $\varphi(\epsilon) = \{F\}$. Each transition sequence that is consistent with φ includes E infinitely often.

Now let us consider the problem of deciding "Is E weakly observably live?" as a Streett game. In order to define a Streett game, we first need to define the arena. For this, we create the marking graph of the net and apply the transformation explained in Figure 3.11 for each transition. After the transformation, we get the marking graph with additional states illustrated in Figure 3.13. Note that occurrence of each transition is represented with a state on this transformed marking graph. In order to have an arena we assign each state to a player. Here, each state belongs to Player-0 except the initial state. So, $V_1 = \{1\}$ and all other vertices belong to V_0.

We set the winning component for Player-0 as:

$$\begin{aligned} Acc = \{&(\emptyset, \{12, 13\}),\\ &(\{16, 17\}, \{14\}),\\ &(\{16, 17\}, \{15\}),\\ &(\{12, 13\}, \{18\}),\\ &(\{12, 13\}, \{19\}),\\ &(\{22, 23\}, \{20\}),\\ &(\{22, 23\}, \{21\})\} \end{aligned}$$

We assume that the user will follow the winning strategy and choose F at the initial marking. So, we do not consider the states reached by choosing G instead of F, since the user makes the first move and then those states will be never reached.

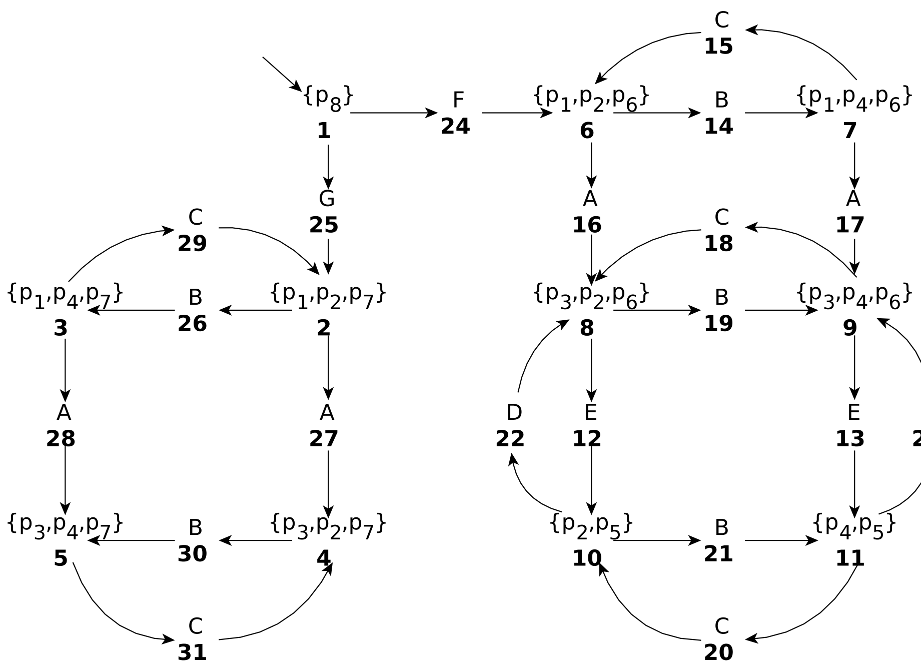

FIGURE 3.13: Transformed marking graph of the net in Figure 3.12

The first condition in *Acc* says that if Player-0 visits 12 or 13 infinitely often he will lose. Since 12, 13 correspond to occurrences of E, Player-0 must avoid to infinitely often visit one or both of them in order to win. The other conditions are to force weakly fair behavior. For example consider infinitely often visiting 14 (firing transition B of the net N); this implies that the path goes forever through 14 and 15; then in the net, transition A is consistently enabled so it must be fired. Indeed in the net we see that A and B are concurrent transitions.

Player-0 cannot have a winning strategy in this game if Player-1 chooses F at the initial marking, and thus E is weakly observably live. □

In the literature there are algorithms for solving two person infinite games. For example in [44], two algorithms are presented for computing strategies for both players. [18] provides a method of strategy construction in infinite games with Streett and Rabin chain winning conditions. Also [53] provides algorithms for Streett and Rabin games.

Our idea of using two person infinite games for checking weak observable liveness is restricted to the above explained class of Petri nets for now. We plan to continue to work on this idea, explore other possible games and generalize to 1-safe Petri nets.

Chapter 4

Non-interference

In this chapter, we study formal notions of unwanted information flow, based on a general notion of *non-interference*, within the theory of Petri nets. We compare our approach with existing approaches and discuss methods for checking non-interference notions.

4.1 Information Flow and Non-interference

In distributed systems, information flows among components. The flow can be used to rule the behavior of the system, to guarantee the correct synchronization of tasks, to implement a communication protocol, and so on.

In some cases, a flow of information from one component to another is actually a *leakage*: that piece of information should not have passed from here to there. Such unwanted flows can endanger the working of the system.

Information flow is studied in different areas such as logic, linguistics, information theory, software engineering, distributed systems, etc.

In [5] it is stated that information flow results from regularities in a distributed system. By describing a system in which information flows as distributed, we naturally assume that information flows from a source to a destination. Thus, we are considering a system which is divided into two main parts. In abstract systems, the relation of whole to part is also abstract and the metaphor of "flow" is interpreted even more loosely. There is no restriction on what is considered as part, the choice of parts determines the way information flows. Presence of regularities links the parts of a system together in a way that permits the flow of information. Regularities in a distributed system makes the behavior predictable. However, the behavior of the system does not have to be entirely predictable for information to flow. Yet, as a general rule, the more random the system the less information will flow.

From information theoretical point of view, information flow is the transfer of information from a variable x to a variable y in a given process. Not all information flow is desirable. For example, an information flow which leads the system to leak any secret to public observers is absolutely not wanted. Usually, each variable is assigned a security level. The basic model consists of two distinct levels: low and high. Low corresponds to publicly observable information, while high corresponds to secret information. For the sake of confidentiality, flowing information from high to low variables should not be allowed.

There are several formal notions concerning information flow such as non-deducibility [68], opacity [15, 16, 48], anonymity [65] and non-interference.

Non-interference was first defined for deterministic programs [36]. Later, several adaptations were proposed for more abstract settings, like transition systems, usually related to observational semantics [32, 33, 59, 62, 63].

Broadly speaking, these approaches assume that the actions performed in a system belong to two types, conventionally called *high* (hidden) and *low* (observable). A system is then said to be free from interference if a user, by interacting only via low actions, cannot deduce information about which high actions have been performed. This approach was formalized in terms of 1-safe Petri nets in [19], relying on known observational equivalences, including bisimulation.

In [61] a special kind of non-interference, called *intransitive non-interference*, is introduced in which there are not only two kinds of actions but also an intermediate kind called *downgrading*. The idea of having downgrading actions is that whenever one of such actions occurs it declassifies the high actions executed before it. Intransitive non-interference has been formalized in elementary net systems in [37] and studied in [13, 14] on Place/Transition nets.

This thesis adapts the first setting with only high and low actions and analyzes systems that can perform high and low level actions and we check if an observer, who knows the structure of the system, can deduce information about the high actions by observing low actions. We rely on a progress assumption which was ignored in non-interference notions in the literature.

There are mainly two kinds of information flows that non-interference notions deal with. These are *positive information flow* and *negative information flow*. A positive information flow arises when the occurrence of a high level transition can be deduced from the low level behavior of the system, whereas a negative information flow is concerned with the non-occurrences of a high transition.

In this chapter, new notions of non-interference for ordinary Petri nets are introduced. They deal with positive information flow as well as negative information flow, regarding both past and future occurrences and are based on unfoldings and on reveals and excludes relations which are formally defined in Section 4.3 and Section 4.4. *Reveals* was originally defined as a relation between events of an occurrence net in [39] and applied in fault diagnosis. Here, we adapt this relation to transitions of Petri nets. Intuitively, a transition t_1 reveals another transition t_2 if, by observing the occurrence of t_1, it is possible to deduce the occurrence of t_2. *Excludes* is a new relation between transitions of a Petri net, which is introduced in order to detect negative information flow. A transition t_1 excludes another transition t_2 if, by observing the occurrence of t_1, it is possible to deduce that t_2 has not yet occurred and will not occur in the future, i.e., they never appear together in the same run. We also introduce two variants of the excludes relation, namely *past excludes* and *future excludes*.

The first notion of non-interference we introduce is called *Reveals based Non-Interference (RNI)* and it states that a net is secure if no low transition reveals any high transition. This new notion is introduced in Section 4.5.1. We also propose more restrictive notions called *k-Extended-Reveals based Non-Interference (k-ERNI)* and *n-Repeated-Reveals based Non-Interference (n-ReRNI)*, they are based on observation of multiple occurrences of low transitions. These two parametric non-interference notions are introduced and discussed in Section 4.5.2 and Section 4.5.3. In Section 4.5.4, *Positive/Negative Non-Interference (PNNI)* is introduced on the basis of both the reveals and excludes relations between low and high transitions capturing both positive and negative information flow. In Section 4.5.5, *PNNI* is improved by using two variants of the excludes relation. The new notions are discussed and compared with each other while they are introduced. In Section 4.6, we compare, on the basis of examples, the new introduced notions with the ones already introduced in the literature and mentioned at the beginning of Section 4.2. Section 4.7 is devoted to the methods for checking non-interference. After briefly investigating some methods used for checking the most popular non-interference notions in the literature, we explore two approaches for checking the non-interference notions based on reveals and excludes relations introduced in this thesis.

4.2 Non-interference Notions with Petri Nets

In this section, before introducing the new notions, we briefly recall the most used non-interference notions in the literature and discuss our motivation for introducing new non-interference notions based on reveals and excludes relations. The notions recalled in the following are based on some notion of low observability of a system. It is what can be observed of a system from the point of view of low users.

In the following, we will use acronyms to denote the set of nets satisfying the corresponding security notion.

The less restrictive notion, introduced in [32, 33] and also studied on 1-safe Petri nets in [19], is *Strong Nondeterministic Non-Interference (SNNI)*. It is a trace-based property (trace as sequence of event occurrences), that intuitively says that a system is secure if what the low part can see does not depend on what the high level part does. If a net system N is *SNNI* secure, then it should offer, from the low point of view, the same traces as the system where the high level transitions are prevented. In *SNNI* secure systems, information can flow from low to high but not from high to low. A different characterization of the same notion, called *Non-Deducibility on Composition (NDC)*, is given in [19].

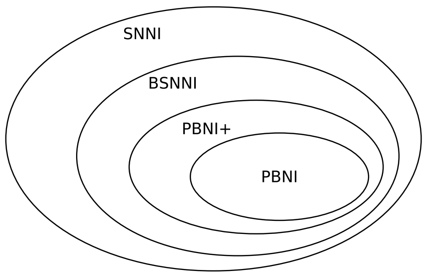

FIGURE 4.1: Relation between some existing interference notions in the literature

While *SNNI* is based on trace equivalence, the more restrictive notions *Bisimulation based Strong Nondeterministic Non-Interference (BSNNI)* and *Bisimulation based Non-Deducible on Composition (BNDC)* are based on bisimulation.

NDC and *BNDC* are also defined for unbounded P/T nets in [14] both for the systems with only high and low transitions and for the systems which also has downgrading transitions.

Strong Bisimulation based Non-Deducible on Composition (SBNDC) is an alternative characterization of *BNDC* [32, 33]. In fact, Busi and Gorrieri in [19] show that *BNDC* is equivalent to *SBNDC*, and it is stronger than *BSNNI*.

Another non-interference notion called *Place Based Non-Interference (PBNI)* was introduced in [19]. It is based on the absence of some kinds of specific places in the net, namely causal and conflict places. A causal place is a place between a low transition and a high transition such that the low transition consumes the token from the place which

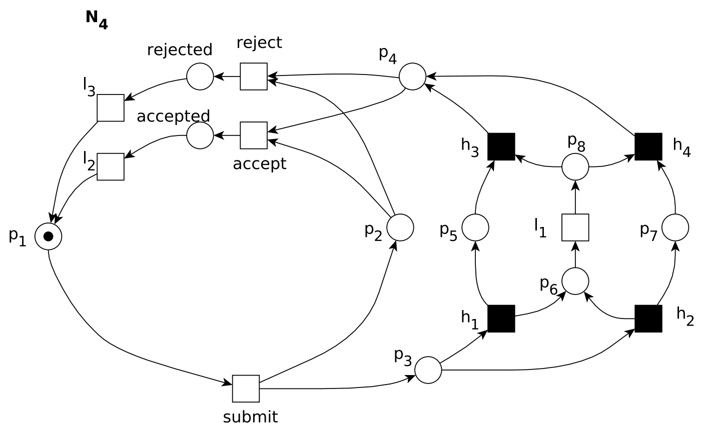

FIGURE 4.2: A net modeling paper submission and evaluation.

was produced by the high transition. A conflict place is a place such that at least one low transition and one high transition consume a token from it. A net is considered to be *PBNI* secure in the absence of such places. In [19], it is shown that if a net is *PBNI* secure then it is also *SBNDC* secure.

In [20], a similar notion, called *Positive Place Based Non-Interference (PBNI+)*, is proposed by introducing the notions of active causal and active conflict places. *PBNI+* is weaker than *PBNI* and it coincides with *SBNDC*.

The overall relationship between these mentioned notions is illustrated in Figure 4.1. In [19], the following relation is proved for 1-safe Petri nets: *SNNI*≡*NDC*, *BSNNI* ⊆*SNNI*, *SBNDC* ≡ *BNDC* ≡ *PBNI+* ⊆ *BSNNI*, *PBNI*⊆*PBNI+* . In the rest of the chapter, we will refer only to these notions for discussing the similarities and the differences with our notions we introduce here.

With respect to the above mentioned different kinds of information flow, *SNNI*, *BSNNI* and *PBNI+* deal with positive information flow, whereas *PBNI* deals also with negative information flow.

All these notions seem to aim mainly at deducing past occurrences of high transitions, for example they all consider system N_6 in Figure 4.11 on page 67 secure, whereas, by considering progress, after the occurrence of l, a low user deduces that h is inevitable and therefore N_6 is not secure with respect to the ability of deducing information about the future behavior.

Differently from the previous notions, the ones we are going to propose do not only capture information flow about past occurrences of high transitions, but also information flow about inevitable or impossible future occurrences of high transitions.

In some cases, the mere ability to deduce that some high transition has occurred is not a security threat, provided the low user cannot know which one occurred.

Let us illustrate this issue with the help of an example. The net in Figure 4.2 represents a system in which a user can repeatedly submit a paper to a committee, each time receiving a judgment (accept or reject). The black squares represent high transitions. The review process can follow either of two paths, and we do not want the user to know which one was chosen. When the user receives an answer, he knows that some high transition occurred, however he cannot infer which one.

For this reason, the new notions we are going to introduce in the following will consider such a system secure, whereas it is not secure with respect to *SNNI*, and the other above recalled notions.

4.3 Reveals Relations

This section briefly summarizes a group of relations originally defined on occurrence nets in [1, 39] and then adapts them to Petri nets as in [9]. Intuitively, an event of an occurrence net *reveals* another event, if the occurrence of the latter is inevitable, i.e., it must have occurred in the past or will eventually occur in the future, once the first event occurs. Moreover, another relation that will be explored in this section is *extended-reveals*. The original notion describes a relation between two sets of events of an occurrence net. Set A extended-reveals set B if the occurrence at least one of the elements in set B is inevitable provided that all elements of set A occurred. In this section these two relations are studied and then adapted to the transitions of a P/T net. Moreover, a new relation which is based on the fact that repeated occurrence of a transition can give information about past or inevitable future occurrence of another transition is introduced.

4.3.1 Reveals relations on occurrence nets

Definition 4.3.1. [39]Let $O = (B, E, F)$ be an occurrence net, $\Omega \subseteq 2^E$ be the set of its maximal runs, and e_1, e_2 be two of its events. Event e_1 reveals e_2, denoted $e_1 \rhd e_2$, iff $\forall \sigma \in \Omega : \ e_1 \in \sigma \implies e_2 \in \sigma$

Example 4.3.1. To give a simple example on the original reveals and extended-reveals notions, we examine the occurrence net in Figure 4.3. In this occurrence net, e_5 reveals e_1 since e_5 cannot occur if e_1 does not occur before. However e_1 does not reveal e_5 since, instead of e_2, e_3 can fire so that e_5 cannot fire anymore. Another reveals relation on this occurrence net is $e_2 \rhd e_4$, because when e_2 occurs e_3 cannot occur anymore and so e_4 has to occur. $\square$

Definition 4.3.2. [1]Let $O = (B, E, F)$ be an occurrence net, $\Omega \subseteq 2^E$ be the set of its maximal runs, and A, B two sets of events. A extended-reveals B, denoted $A \twoheadrightarrow B$, iff $\forall \omega \in \Omega : \; A \subseteq \omega \implies B \cap \omega \neq \emptyset$.

In other words, a set of events, A, extended-reveals another set of events B, iff every maximal run that contains A also hits B. The reveals relation can be expressed as extended-reveals relation between singletons: $a \rhd b$ can be written as $\{a\} \twoheadrightarrow \{b\}$.

Example 4.3.2. In the occurrence net given in Figure 4.3, $e_2 \rhd e_4$ and $e_4 \rhd e_2$.

In the same occurrence net, the occurrence of e_1 does not necessarily mean that e_5 will occur, whereas e_1 together with e_2 extended-reveals e_5, denoted as $\{e_1, e_2\} \twoheadrightarrow \{e_5\}$. The occurrence of e_4 reveals neither e_6 nor e_7. However, it reveals that either e_6 or e_7 will occur, denoted as $\{e_4\} \twoheadrightarrow \{e_6, e_7\}$. $\square$

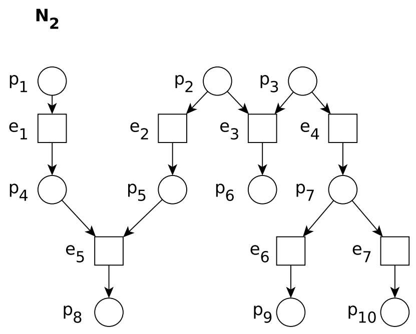

FIGURE 4.3: An occurrence net.

Remark 4.3.1. In general, reveals relation is not symmetrical. As an example, $e_6 \rhd e_4$ but $e_4 \not\rhd e_6$ since after e_4, e_7 can occur instead of e_6.

4.3.2 Reveals Relations on Petri Nets

In this section, we define a *reveals* and an *extended-reveals* relation on the set of transitions of a Petri net, relying on the corresponding relations on occurrence nets. Moreover, we introduce a new parametric relation, called *repeated-reveals*, again on the set of transitions of a Petri net. Reveals, extended-reveals and repeated-reveals relations will be used to detect positive information flow, however they can also be applied in other areas, e.g. fault diagnosis as explored in [39] by using original reveals relation on occurrence nets.

In the following, except for the repeated reveals relation, we assume progress in the behavior of the nets, which means that a constantly enabled transition occurs if it is not disabled by another transition. This means that we consider only maximal runs in the unfolding.

For the rest of this section, we fix the notation as in the following. In the sequel, $N = (P, T, F, m_0)$ will denote a Petri net, and $\text{Unf}(N) = ((B, E, F), \lambda)$ the unfolding of N. R will denote the set of all runs of N whereas Ω the set of all maximal runs. Let $t_i \in T$ and $E_{t_i} = \{e \in E : \lambda(e) = t_i\}$ for $i \in \{1, 2, \ldots\}$ denotes the set of events corresponding to the occurrences of transition t_i in the unfolding.

4.3.2.1 Reveals

Definition 4.3.3. Let $t_1, t_2 \in T$ be two transitions, t_1 *reveals* t_2, denoted $t_1 \rhd_{tr} t_2$, iff $\forall \omega \in \Omega : E_{t_1} \cap \omega \neq \emptyset \implies E_{t_2} \cap \omega \neq \emptyset$.

Transition t_1 reveals transition t_2 if and only if each maximal run which contains an occurrence of t_1 also contains at least one occurrence of t_2. This means that for each observation of t_1, t_2 has been already observed or will be observed.

Remark 4.3.2. The reveals relation on transitions is *reflexive* and *transitive*, i.e., let $t_1, t_2, t_3 \in T$, then from Definition 4.3.3 it directly follows that $t_1 \rhd_{tr} t_1$, and $(t_1 \rhd_{tr} t_2 \wedge t_2 \rhd_{tr} t_3) \implies t_1 \rhd_{tr} t_3$.

Example 4.3.3. In the net N_1, in Figure 4.4, t_3 reveals both t_2 and t_1. It is easy to notice that to be able to fire t_3 we must first fire t_1 and t_2. In fact, in the unfolding $\text{Unf}(N_1)$, for each occurrence of t_3 there is at least one occurrence of t_2 and similarly, for each occurrence of t_3 there is at least one occurrence of t_1. However, t_1 does not reveal t_2 or t_3, since there is a run in which t_1 occurs and neither t_2 nor t_3 occurs. If an observer, who knows the structure of N_1, can only observe t_1 he cannot have information about t_2 or t_3, however if he is able to observe t_3, he can deduce that both t_2 and t_1 must have occurred.

Transition t_1 also reveals transition t_6 because when t_1 fires, t_5 cannot fire anymore and, since the net progresses, t_6 must fire. Since we do not assume strong fairness, $t_1 \not\rhd_{tr} t_4$, after the occurrence of t_1, t_2 and t_3 can loop forever. Reveals relation is not only about past occurrences but also about future occurrences. Observing t_1 does not tell us when t_6 fires. It might have fired already or it will fire in the future. $t_1 \rhd_{tr} t_6$ tells us that when t_1 occurs, an occurrence of t_6 is inevitable. □

Remark 4.3.3. Reveals relation is neither symmetric nor antisymmetric. For example, in Figure 4.4, $t_2 \rhd_{tr} t_3$ and $t_3 \rhd_{tr} t_2$, however $t_2 \rhd_{tr} t_1$ and $t_1 \not\rhd_{tr} t_2$.

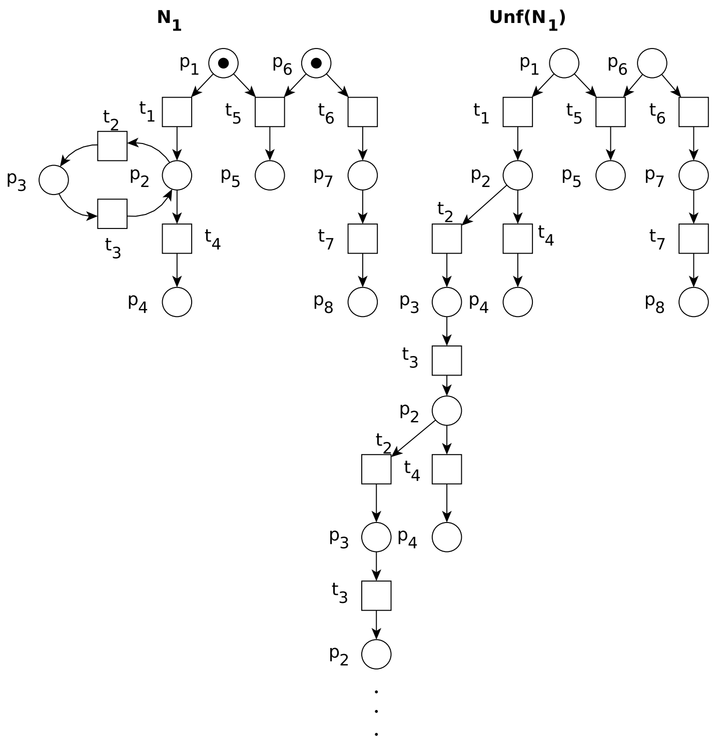

FIGURE 4.4: A Petri net and its unfolding

Reveals relation between transitions, i.e., $\rhd_{tr}$ can be equivalently redefined by using the extended-reveals relation between the events of an occurrence net, i.e., $\multimap$. Intuitively, transition t_1 reveals transition t_2 iff each occurrence of t_1 extended-reveals some occurrences of t_2.

Definition 4.3.4. Let $t_1, t_2 \in T$ be two transitions, $t_1 \hat{\rhd}_{tr} t_2$ iff $\forall e_1 \in E_{t_1} : \{e_1\} \multimap E_{t_2}$.

Example 4.3.4. In Figure 4.5, $t_1 \rhd_{tr} t_6$. In accordance with the Definition 4.3.3, each maximal run which contains an occurrence of t_1 also contains at least one occurrence of t_6. There are two maximal runs in the unfolding which contains an occurrence of t_1 that is e_1. One of the runs contains event e_6 whereas the other contains e_7 both of which are occurrences of t_6. In other words, $e_1 \multimap \{e_6, e_7\}$ which satisfies the definition of $\hat{\rhd}_{tr}$ since e_1 is the only event corresponding to transition t_1. □

Proposition 4.3.4. *Let $t_1, t_2 \in T$ be two transitions, $t_1 \rhd_{tr} t_2 \iff t_1 \hat{\rhd}_{tr} t_2$.*

Proof. Here we prove that Definition 4.3.3 and Definition 4.3.4 are equivalent by showing the above implication holds in both directions. We first show that if $t_1 \hat{\rhd}_{tr} t_2$ then $t_1 \rhd_{tr} t_2$.

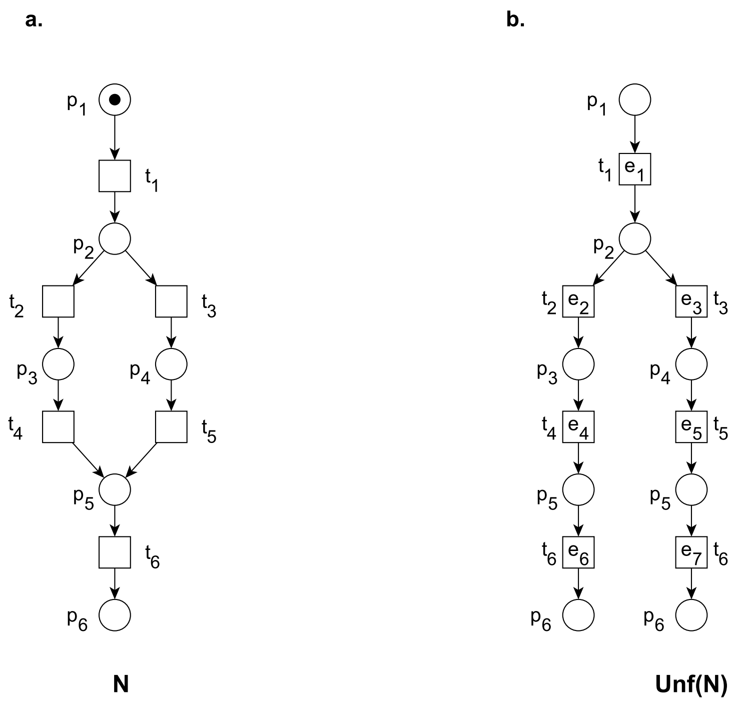

FIGURE 4.5

Definition 4.3.3 states that $\forall\ \omega \in \Omega: \quad E_{t_1} \cap \omega \neq \emptyset \implies E_{t_2} \cap \omega \neq \emptyset$ whereas Definition 4.3.4 states that $\forall\, e_1 \in E_{t_1}: \ \{e_1\} \twoheadrightarrow E_{t_2}$.

By using the definition of extended-reveals between events, Definition 4.3.2, we rewrite

$$\forall\, e_1 \in E_{t_1}: \ \{e_1\} \twoheadrightarrow E_{t_2} \ \text{ into}$$

$$\forall\, e_1 \in E_{t_1}, \forall\omega \in \Omega: \ \{e_1\} \subseteq \omega \implies E_{t_2} \cap \omega \neq \emptyset.$$

We can move the universal quantifier $\forall\omega \in \Omega$ to the beginning of the statement. $\forall e_1 \in E_{t_1}: \ \{e_1\} \subseteq \omega \implies E_{t_1} \cap \omega \neq \emptyset$, so the statement can be rewritten as:

$$\forall\omega \in \Omega: \ E_{t_1} \cap \omega \neq \emptyset \implies E_{t_2} \cap \omega \neq \emptyset.$$

With this, we have achieved the definition of $t_1 \ \rhd_{tr} \ t_2$.

To this point, it is shown that if $t_1 \ \hat{\rhd}_{tr} \ t_2$ then $t_1 \ \rhd_{tr} \ t_2$. Now we have to show the implication also works for the other direction, i.e., if $t_1 \ \rhd_{tr} \ t_2$ then $t_1 \ \hat{\rhd}_{tr} \ t_2$. Starting from $t_1 \ \rhd_{tr} \ t_2$, we write:

$$\forall\omega \in \Omega: \ E_{t_1} \cap \omega \neq \emptyset \implies E_{t_2} \cap \omega \neq \emptyset.$$

The above statement can be rewritten equivalently as:

$$\forall \omega \in \Omega, \forall e_1 \in E_{t_1} : \ \{e_1\} \cap \omega \neq \emptyset \implies \exists e_2 \in E_{t_2} : \ \{e_2\} \cap \omega \neq \emptyset.$$

This means that each occurrence of t_1 appears together with at least one occurrence of t_2 in the same maximal run. So the statement becomes:

$$\forall \omega \in \Omega, \forall e_1 \in E_{t_1} : \ \{e_1\} \cap \omega \neq \emptyset \implies E_{t_2} \cap \omega \neq \emptyset.$$

Note that for each $e_1 \in E_{t_1}$ it is possible to have different occurrences of t_2 for different runs. This is in accordance with the definition of extended-reveals between events. Consequently we can write:

$$\forall e_1 \in E_{t_1} : \ \{e_1\} \twoheadrightarrow E_{t_2}.$$

With this proof we have shown that Defitinion 4.3.3 and Definition 4.3.4 are equivalent so from now on we can use the same symbol, $\triangleright_{tr}$, for both definitions. □

4.3.2.2 Extended-reveals

In some cases, one transition alone does not give much information about the behavior of the net whereas a set of transitions together can give some information about the behavior of the net. This relation is defined as in the following.

Definition 4.3.5. Let $W, Z \subseteq T$. W *extended-reveals* Z, denoted $W \twoheadrightarrow_{tr} Z$, iff $\forall\, \omega \in \Omega$:

$$\bigwedge_{t \in W} (\omega \cap E_t \neq \emptyset) \implies \bigvee_{t \in Z} (\omega \cap E_t \neq \emptyset)$$

We say that a set of transitions W extended-reveals another set of transitions Z, if and only if each maximal run, which contains at least an occurrence of each transition in W, also contains at least an occurrence of a transition in Z.

The reveals relation on transitions, $t_1 \triangleright_{tr} t_2$, corresponds to the extended-reveals relation between singletons, $\{t_1\} \twoheadrightarrow_{tr} \{t_2\}$.

Example 4.3.5. In the net shown in Figure 4.6, t_2 alone does not reveal t_5, whereas t_2 and t_3 together tell us that t_5 will fire, denoted as $\{t_2, t_3\} \twoheadrightarrow_{tr} \{t_5\}$. In the same net, the occurrence of t_5 tells us that either t_8 or t_9 will fire, denoted as $\{t_5\} \twoheadrightarrow_{tr} \{t_8, t_9\}$. Similarly, $\{t_7, t_8\} \twoheadrightarrow_{tr} \{t_{10}\}$, i.e., there is no maximal run which includes occurrences of t_7, t_8 and not t_{10}. □

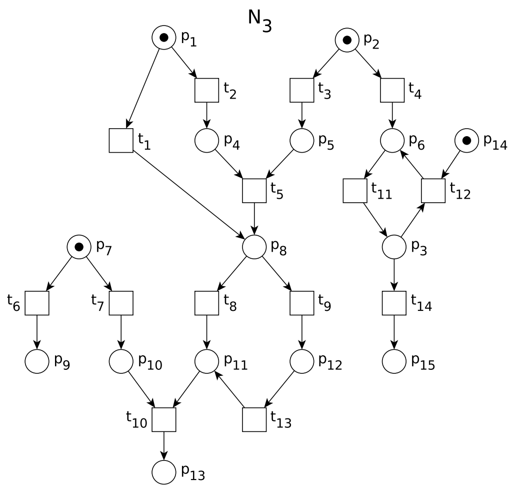

FIGURE 4.6

Remark 4.3.5. Let $A, B, C \subseteq T$. From Definition 4.3.5, it immediately follows: if $A \rightarrowtriangle_{tr} B$ and $B \subseteq C$, then $A \rightarrowtriangle_{tr} C$.

Remark 4.3.6. The extended-reveals relation is *neither symmetric nor antisymmetric.* For example: $\{t_2, t_3\} \rightarrowtriangle_{tr} \{t_8, t_9\}$, however $\{t_8, t_9\} \not\rightarrowtriangle_{tr} \{t_2, t_3\}$; indeed $\{t_8, t_9\} \rightarrowtriangle_{tr} \emptyset$, since there is no run containing an occurrence of both t_8 and t_9, as it will be also discussed later on. On the other side, $\{t_2, t_3\} \rightarrowtriangle_{tr} \{t_5, t_6\}$ and $\{t_5, t_6\} \rightarrowtriangle_{tr} \{t_2, t_3\}$.

The extended-reveals relation on transitions is *not transitive* since the left part of the relation is a conjunction whereas the right part is a disjunction. For example $\{t_2, t_3\} \rightarrowtriangle_{tr} \{t_5, t_7, t_8\}$ and $\{t_5, t_7, t_8\} \rightarrowtriangle_{tr} \{t_{10}\}$, however $\{t_2, t_3\} \not\rightarrowtriangle_{tr} \{t_{10}\}$.

Proposition 4.3.7. *Let $N' = (P', T', F', m'_0)$ be a 1-safe Petri net, whose underlying net is an acyclic graph and $t_1, t_2 \in T$. If there exists a marking $m \in [m'_0\rangle$ such that t_1 is in conflict with t_2 at m, then $\{t_1, t_2\} \rightarrowtriangle_{tr} \emptyset$.*

Since the previous proposition considers acyclic 1-safe nets, its proof can be directly derived from the corresponding one for the events of an occurrence net as given in [2].

4.3.2.3 Repeated-reveals

In some cases, repeated occurrences of the same transition can give more information about the behavior of a net than only one occurrence of that transition. A relation based on this fact is defined in the following.

Definition 4.3.6. Let $t_1, t_2 \in T$ be two transitions of N, and n be a positive integer. Let $R^n_{t_i} = \{\omega \in R : \ |\omega \cap E_{t_i}| = n\}$ and $\Omega^n_{t_i}$ denotes the set of maximal runs in $R^n_{t_i}$ with respect to set inclusion (i.e., $\Omega^n_{t_i} \subseteq R^n_{t_i}$ such that if $u, v \in \Omega^n_{t_i} \ \wedge \ u \subseteq v$ then $u = v$).

If $\Omega^n_{t_1} \neq \emptyset$ then t_1 *n-repeated reveals* t_2, denoted $t_1 \rhd^n_{tr} t_2$, iff $\forall \omega \in \Omega^n_{t_1} : E_{t_2} \cap \omega \neq \emptyset$.

If $\Omega^n_{t_1} = \emptyset$ then neither $t_1 \rhd^n_{tr} t_2$ nor $t_1 \not\rhd^n_{tr} t_2$ is defined.

Notation. $t_1 \not\rhd^n_{tr} t_2$ will denote that there is at least one run in $\Omega^n_{t_1}$ such that t_1 appears n times and t_2 does not appear. $\neg(t_1 \rhd^n_{tr} t_2)$ will denote that either $t_1 \rhd^n_{tr} t_2$ is not defined, or $t_1 \not\rhd^n_{tr} t_2$.

Example 4.3.6. Let us consider N_3 in Figure 4.6. Transition t_{11} does not reveal t_{12}, however if the occurrence of t_{11} is observed twice then it is evident that t_{12} occurred, therefore t_{11} 2-Repeated reveals t_{12}, denoted $t_{11} \rhd^2_{tr} t_{12}$, whereas $t_{11} \not\rhd^1_{tr} t_{12}$ since after the first occurrence of t_{11}, t_{14} can fire instead of t_{12}.

Note that $t_{11} \rhd^3_{tr} t_{12}$ and $t_{11} \not\rhd^3_{tr} t_{12}$ are both not defined since t_{11} can fire at most twice, therefore in this case $\neg(t_{11} \rhd^3_{tr} t_{12})$. □

Proposition 4.3.8. *Let $t_1, t_2 \in T$ be two transitions of N,*

$$t_1 \rhd^1_{tr} t_2 \implies t_1 \rhd_{tr} t_2$$

Proof. Let $R^1_{t_1} = \{\omega \in R : |\omega \cap E_{t_1}| = 1\}$ and $\Omega^1_{t_1}$ be the set of maximal runs in $R^1_{t_1}$. If $t_1 \rhd^1_{tr} t_2$, then $\Omega^1_{t_1} \neq \emptyset$ and $\forall \omega \in \Omega^1_{t_1} : \omega \cap E_{t_2} \neq \emptyset$. Let σ be an arbitrary maximal run of Unf(N). Suppose that $\sigma \cap E_{t_1} \neq \emptyset$ then we can always take a run $\omega \in \Omega^1_{t_1}$ such that $\omega \subseteq \sigma$. Then we know that σ contains at least one occurrence of t_2 and so $t_1 \rhd_{tr} t_2$. □

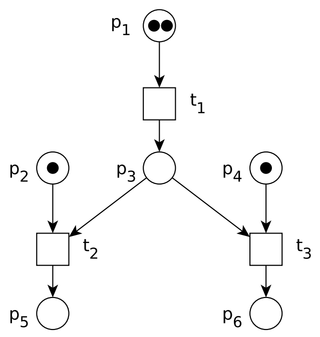

FIGURE 4.7

However, the implication of the previous proposition does not hold in the other direction. In fact, consider the net in Figure 4.7, $t_1 \rhd_{tr} t_2$, $t_1 \rhd_{tr} t_3$, $t_1 \not\rhd^1_{tr} t_2$ and $t_1 \not\rhd^1_{tr} t_3$. The main difference is that we consider only maximal runs for reveals relation. For this net there is only one maximal run which contains t_1 (twice), t_2 and t_3. However, there is a run in $\Omega^1_{t_1}$ in which t_1 appears and t_2 does not appear, as well as a run in which t_1 appears and t_3 does not appear. All runs in $\Omega^2_{t_1}$, i.e., including t_1 twice, contain both t_2 and t_3, i.e., $t_1 \rhd^2_{tr} t_2$ and $t_1 \rhd^2_{tr} t_3$.

Proposition 4.3.9. *Let $t_1, t_2 \in T$ be two transitions, if $t_1 \rhd^n_{tr} t_2$ and $\Omega^{n+1}_{t_1} \neq \emptyset$ then $t_1 \rhd^{n+1}_{tr} t_2$.*

Proof. Let $R^n_{t_1} = \{\omega \in R : |\omega \cap E_{t_1}| = n\}$ and $\Omega^n_{t_1}$ be the set of maximal runs in $R^n_{t_1}$. If $t_1 \rhd^n_{tr} t_2$, then $\Omega^n_{t_1} \neq \emptyset$ and for all $\omega \in \Omega^n_{t_1}$ $\omega \cap E_{t_2} \neq \emptyset$. Let $\sigma \in \Omega^{n+1}_{t_1}$, we can always choose a run $\omega \in \Omega^n_{t_1}$ such that $\omega \subseteq \sigma$. Then we know that $\sigma \cap E_{t_2} \neq \emptyset$, so $t_1 \rhd^{n+1}_{tr} t_2$.

□

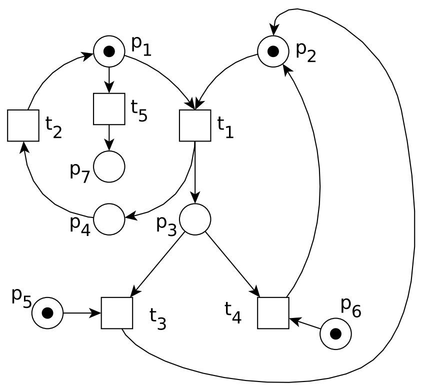

FIGURE 4.8

Example 4.3.7. In Figure 4.8, t_1 reveals neither t_3 nor t_4, however if the occurrence of t_1 is observed twice then it is evident that either t_3 or t_4 has occurred or will inevitably occur. Assume that t_3 has occurred, before t_1 can occur again, t_4 has to occur. In other words, $R^2_{t_1}$ includes both t_3 and t_4, i.e., $t_1 \rhd^2_{tr} t_3$ and $t_1 \rhd^2_{tr} t_4$.

Note that $R^3_{t_1}$ is not empty and so by Proposition 4.3.9, $t_1 \rhd^3_{tr} t_3$ and $t_1 \rhd^3_{tr} t_4$. □

4.4 Excludes Relation

In this section, we introduce a new relation between transitions, called *excludes*, which will be used to detect negative information flow, yet there are other possible application areas such as fault diagnosis.

In the sequel, $N = (P, T, F, m_0)$ will denote a Petri net, and $\mathrm{Unf}(N) = ((B, E, F), \lambda)$ the unfolding of N. R will denote the set of all runs of N whereas Ω the set of all maximal runs. Let $t_i \in T$, $E_{t_i} = \{e \in E : \lambda(e) = t_i\}$ for $i \in \{1, 2, \ldots\}$.

4.4.1 A new relation: Excludes

The new introduced notion *excludes* is based on the intuition of the impossibility of the two transitions to fire together in a run. In other words, a transition excludes another transition if they can never appear in the same run. Thus, observation of one tells that the other did not occur and will never occur in the future.

Definition 4.4.1. Let $t_1, t_2 \in T$ be two transitions, t_1 *excludes* t_2, denoted $t_1 \; \underline{\text{ex}} \; t_2$, iff $\forall \omega \in \Omega : \quad E_{t_1} \cap \omega \neq \emptyset \implies E_{t_2} \cap \omega = \emptyset$, i.e., they never appear in the same run.

It is easy to see that excludes is a symmetric relation and it is not transitive as well as obviously not reflexive.

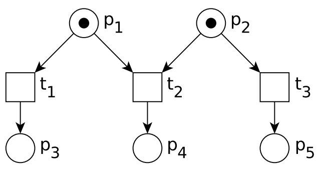

FIGURE 4.9

Example 4.4.1. In Figure 4.9, $t_1 \; \underline{\text{ex}} \; t_2$ since the occurrence of t_1 disables t_2 as well as $t_2 \; \underline{\text{ex}} \; t_3$ whereas $\neg(t_1 \; \underline{\text{ex}} \; t_3)$. □

In the case of Petri nets whose underlying net is an acyclic graph, if two transitions are in conflict at a marking, i.e., they are both enabled and the firing of one disables the other one, then one excludes the other. However, in general, transitions which are in conflict can still appear in the same maximal run and therefore they could be in not-excludes relation.

Example 4.4.2. The transitions t_2 and t_4 of N_1 in Figure 4.4 on page 57 are in conflict whereas $\neg(t_2 \; \underline{\text{ex}} \; t_4)$. In the unfolding in the same figure, it is possible to see a maximal run including occurrences of both.

$t_5 \; \underline{\text{ex}} \; t_4$ although they are not in conflict.

$t_7 \; \underline{\text{ex}} \; t_5$, $t_5 \; \underline{\text{ex}} \; t_1$ and $\neg(t_7 \; \underline{\text{ex}} \; t_1)$, indeed the relation is not transitive. □

Proposition 4.4.1. *Let* $t_1, t_2 \in T$, $t_1 \; \underline{\text{ex}} \; t_2 \iff \{t_1, t_2\} \twoheadrightarrow_{tr} \; \emptyset$.

Proof. $t_1 \; \underline{\text{ex}} \; t_2$ means $\forall \omega \in \Omega : \; \omega \cap E_{t_1} \neq \emptyset \implies \omega \cap E_{t_2} = \emptyset$. Directly from the definition of extended reveals, $\{t_1, t_2\} \twoheadrightarrow_{tr} \; \emptyset$ implies that $\forall \omega \in \Omega : \; \omega \cap E_{t_1} \neq \emptyset \; \wedge \; \omega \cap E_{t_2} \neq \emptyset$ is false. Thus, either $\omega \cap E_{t_1} = \emptyset$ or $\omega \cap E_{t_2} = \emptyset$ for all maximal runs. This means that t_1 and t_2 can never appear in the same run, i.e., $t_1 \; \underline{\text{ex}} \; t_2$. □

Considering non-interference, we are interested in capturing the existence of a high and a low transition in a Petri net that can never appear together in the same maximal run. Existence of such transitions causes negative information flow, i.e., an observer can deduce that a high transition did not occur and will not occur in the future by observing occurrence of a low transition. Excludes relation will be used in this work for catching negative information flow.

4.4.2 Future/Past excludes

Excludes relation as introduced in Section 4.4.1 is a symmetric relation and it basically tells that two transitions can never appear in the same maximal run together. This refers to all past and future occurrences of the two transitions. However, we are not only interested that two transitions exclude each other in general, but also in situations like occurrence of a transition guarantees that another transition will never appear in the future, although it might have occurred in the past, or that it might occur in the future and it did not occur in the past.

Here, we define two versions of excludes relation, *future-excludes* and *past-excludes*. The first one focuses on future occurrences while the latter focuses on the past occurrences.

Definition 4.4.2. Let $e \in E$, $\downarrow e = \{e' \in E :\ e < e'\}$. Let $t_1, t_2 \in T$ be two transitions, t_1 *future-excludes* t_2, denoted $t_1\ \underline{\text{ex}}_f\ t_2$, iff $\forall e \in E_{t_1} :\ \downarrow e \cap E_{t_2} = \emptyset\ \wedge\ \nexists e' \in E_{t_2}$ such that e **co** e', i.e., t_2 never occurs after t_1 or concurrently with t_1.

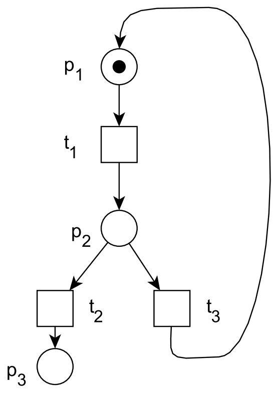

FIGURE 4.10

Example 4.4.3. In Figure 4.10, $t_2\ \underline{\text{ex}}_f\ t_3$ since occurrence of t_2 disables t_3 forever, however after occurrence of t_3, t_2 can still fire in the future, therefore $\neg(t_3\ \underline{\text{ex}}_f\ t_2)$. Similarly, in this net, $t_2\ \underline{\text{ex}}_f\ t_1$. It is easy to see that t_1 can occur many times until t_2 occurs, and after the occurrence of t_2, it can never occur again. □

Note that, unlike excludes, future-excludes is not symmetric.

Definition 4.4.3. Let $e \in E$, $\uparrow e = \{e' \in E : e' < e\}$. Let $t_1, t_2 \in T$ be two transitions, t_1 *past-excludes* t_2, denoted $t_1 \underline{\text{ex}}_p\, t_2$, iff $\forall e \in E_{t_1} : \ \uparrow e \cap E_{t_2} = \emptyset \ \wedge \ \nexists e' \in E_{t_2}$ such that $e \ \mathbf{co}\ e'$, i.e., t_2 never occurs before t_1 or concurrently with t_1.

Example 4.4.4. In Figure 4.10, $t_3 \ \underline{\text{ex}}_p\ t_2$ since occurrence of t_3 means that t_2 did not fire in the past since after t_2, t_3 can never fire again. However $\neg(t_2 \ \underline{\text{ex}}_p\ t_3)$. Similarly, in this net, $t_1 \ \underline{\text{ex}}_p\ t_2$. It is easy to see that t_1 and t_3 can occur many times until t_2 occurs, so $\neg(t_2 \ \underline{\text{ex}}_p\ t_1)$, however after occurrence of t_2, they can never occur again, thus t_2 cannot appear in the past of t_1 or t_3. □

Proposition 4.4.2. *Let $t_1, t_2 \in T$.*

$$t_1 \ \underline{ex}_f\ t_2 \Longleftrightarrow t_2 \ \underline{ex}_p\ t_1.$$

Proof. We first prove that $t_1 \ \underline{\text{ex}}_f\ t_2 \implies t_2 \ \underline{\text{ex}}_p\ t_1$ by contradiction. For this we assume that $t_1 \ \underline{\text{ex}}_f\ t_2$ and $\neg(t_2 \ \underline{\text{ex}}_p\ t_1)$. So, $\forall e_1 \in E_{t_1} : \ \downarrow e_1 \cap E_{t_2} = \emptyset$ and $\forall e_1 \in E_{t_1}$, $\forall e_2 \in E_{t_2} : \ \neg(e_1 \ \mathbf{co}\ e_2)$. Since $t_1 \ \underline{\text{ex}}_f\ t_2$ implies that $\neg(e_1 \ \mathbf{co}\ e_2)$ for all occurrences of t_1 and t_2, the unsatisfied requirement for $t_2 \ \underline{\text{ex}}_p\ t_1$ is that there exists $e_2 \in E_{t_2}$ such that $\uparrow e_2 \cap E_{t_1} \neq \emptyset$. This means that:

$$\exists e_1' \in E_{t_1},\ \exists e_2' \in E_{t_2} : \ e_1' < e_2',$$

i.e., there is an occurrence of t_1 in the past of an occurrence of t_2, which is in contradiction with:

$$\forall e_1 \in E_{t_1} : \ \downarrow e_1 \cap E_{t_2} = \emptyset,$$

which means that t_2 cannot occur after t_1. By this we prove that $t_1 \ \underline{\text{ex}}_f\ t_2 \implies t_2 \ \underline{\text{ex}}_p\ t_1$.

Now we show that the implication holds also for the other direction. We assume that $t_2 \ \underline{\text{ex}}_p\ t_1$ and $\neg(t_1 \ \underline{\text{ex}}_f\ t_2)$. So, $\forall e_2 \in E_{t_2} : \ \uparrow e_2 \cap E_{t_1} = \emptyset$ and $\forall e_2 \in E_{t_2}$, $\forall e_1 \in E_{t_1}$: $\neg(e_2 \ \mathbf{co}\ e_1)$. Since $t_2 \ \underline{\text{ex}}_p\ t_1$ implies that $\neg(e_2 \ \mathbf{co}\ e_1)$ for all occurrences of t_1 and t_2, the unsatisfied requirement for $t_1 \ \underline{\text{ex}}_f\ t_2$ is that there exists $e_1 \in E_{t_1}$ such that $\downarrow e_1 \cap E_{t_2} \neq \emptyset$. This means that:

$$\exists e_1' \in E_{t_1},\ \exists e_2' \in E_{t_2} : e_1' < e_2',$$

i.e., there is an occurrence of t_1 in the past of an occurrence of t_2, which is in contradiction with:

$$\forall e_2 \in E_{t_2} : \ \uparrow e_2 \cap E_{t_1} = \emptyset,$$

which means that t_1 cannot occur before t_2. By this we prove that $t_2 \; \underline{\mathrm{ex}}_p \; t_1 \implies t_1 \; \underline{\mathrm{ex}}_f \; t_2$. □

Proposition 4.4.3. *Let $t_1, t_2 \in T$ be two transitions.*

$$t_1 \; \underline{ex} \; t_2 \iff t_1 \; \underline{ex}_f \; t_2 \; \wedge \; t_1 \; \underline{ex}_p \; t_2.$$

Proof. By Definition 4.4.1 $t_1 \; \underline{\mathrm{ex}} \; t_2$ means

$$\forall \omega \in \Omega : \quad E_{t_1} \cap \omega \neq \emptyset \implies E_{t_2} \cap \omega = \emptyset.$$

This means that t_1 can never be in the same run with t_2. So,

$$\forall e_1 \in E_{t_1}, \; \forall e_2 \in E_{t_2} : \quad \neg(e_1 \mathbf{co} e_2) \; \wedge \; \neg(e_1 < e_2) \; \wedge \; \neg(e_2 < e_1).$$

By using Proposition 4.4.2, we can rewrite the above statement as:

$$t_2 \; \underline{\mathrm{ex}}_p \; t_1 \; \wedge \; t_1 \; \underline{\mathrm{ex}}_p \; t_2$$

We can also write the same statement as:

$$t_1 \; \underline{\mathrm{ex}}_f \; t_1 \; \wedge \; t_2 \; \underline{\mathrm{ex}}_f \; t_1$$

Both translation means that the two transitions cannot occur concurrently or one after the other. So it is easy to see that the statements $t_1 \; \underline{\mathrm{ex}} \; t_2$ and $(t_1 \; \underline{\mathrm{ex}}_f \; t_2 \; \wedge \; t_1 \; \underline{\mathrm{ex}}_p \; t_2)$ are equivalent. □

4.5 Reveals and Excludes Based Non-interference Notions

In this section we introduce new non-interference notions based on the reveals and excludes relations and their variants which are introduced in Section 4.3 and Section 4.4. Some results of this section are also published in [9]. The new notions aim to catch positive and negative information flow, about the past and future events. We consider a system which is modeled by ordinary Petri nets, such that a transition can be high (hidden) or low (observable). The intuition is that in an ideally secure (interference free) system, observation of low transitions should not give information about the occurrences of high transitions.

In the sequel, $N = (P, T, F, m_0)$ will denote a Petri net such that $T = H \cup L$, $H \cap L = \emptyset$, $L, H \neq \emptyset$, where H is the set of high transitions and L is the set of low transitions. $\text{Unf}(N) = ((B, E, F),\lambda)$ will denote the unfolding of N. R will denote the set of all runs of N whereas Ω the set of all maximal runs. Let $t_i \in T$, $E_{t_i} = \{e \in E : \ \lambda(e) = t_i\}$ for $i \in \{1, 2, \dots\}$.

4.5.1 Non-interference based on reveals

Reveals-based Non-Interference accepts a net as secure if no low transition reveals any high transition.

Definition 4.5.1. N is secure with respect to *Reveals-based Non-Interference (RNI)* iff

$$\forall l \in L, \forall h \in H : \ l \ \ntriangleright_{tr} \ h.$$

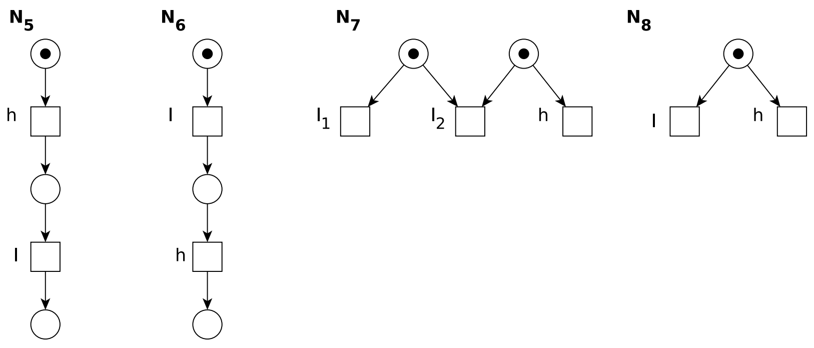

FIGURE 4.11

Example 4.5.1. N_4 in Figure 4.2 on page 53 is *RNI* secure. However, N_5 and N_6 in Figure 4.11 are not secure with respect to *RNI*, since in both nets a low transition reveals a high transition, i.e., $l \ \triangleright_{tr} \ h$. An observer who knows the structure of the net can deduce that h has already fired in N_5 by observing l. For N_6, again by observing l, he can deduce that h will fire. N_7 in Figure 4.11 is also not secure in this context because the observation of l_1 tells the observer that h has already fired or will fire since l_2 cannot fire anymore. □

With *RNI*, we are able to capture positive information flow. Moreover, we not only capture past occurrences of high transitions but also future occurrences, and this is because of the progress assumption.

Although it is useful to capture positive information flow, *RNI* is not able to capture negative information flow. N_8 in Figure 4.11 is considered to be secure with respect to *RNI* since it cannot capture the flow between h and l. However, an observer could deduce that h has not fired and will not fire in the future by observing the occurrence of l. In Section 4.5.4 and Section 4.5.5 we introduce a notion based on both reveals and excludes relations which deals with this kind of information flow.

4.5.2 Non-interference based on extended-reveals

As explained in Section 4.3.2, in some cases, a transition does not tell much about the behavior of the net, whereas a set of transitions together gives some more information. Extended-reveals deals with this relation between transitions of a Petri net. We propose to use this relation in order to define a new non-interference notion in which the occurrences of a set of low transition together give information about some high transitions.

Definition 4.5.2. Let k be a positive integer such that $|L| \geqslant k \geqslant 1$. N is secure with respect to *k-Extended-Reveals based Non-Interference (k-ERNI)* iff

$$\forall h \in H, \forall A \subseteq L \text{ such that } |A| \leqslant k \ \wedge \ \exists \omega \in \Omega : \ A \subseteq \lambda(\omega), \text{ it holds } A \not\twoheadrightarrow_{tr} \{h\}.$$

In other words, a net is secure with respect to *k-ERNI* iff no subset of L with at most k elements, such that they can all appear in the same run, extended reveals an $h \in H$. Therefore, we do not consider the subsets consisting of transitions which can never appear together in a run.

N is *ERNI* secure if it satisfies the above condition for $k = |L|$.

Intuitively, we say that a net is *k-ERNI* secure, if an attacker is not able to deduce information about the hidden part of the net by observing occurrences of k low transitions. If a net is *k-ERNI* secure then it is secure with respect to all *n-ERNI* where $1 \leqslant n \leqslant k$.

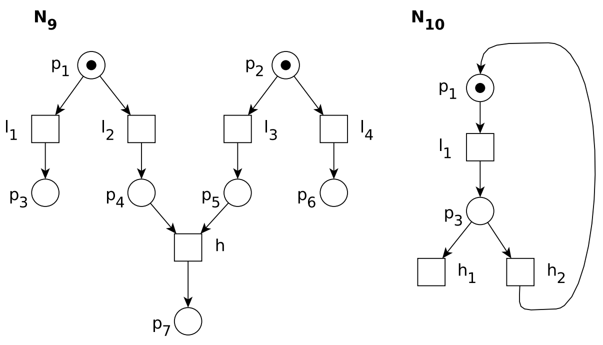

FIGURE 4.12

Example 4.5.2. N_9 in Figure 4.12 is not secure with respect to *2-ERNI*. When l_2 and l_3 occur, a low level observer can deduce that h will occur, i.e., $\{l_2, l_3\} \multimap_{tr} \{h\}$. In this net, the occurrence of only one low transition does not give sufficient information about any high transitions, whereas the occurrence of two low transitions together does. In the net in Figure 4.13, no low transition alone reveals a high transition as well as

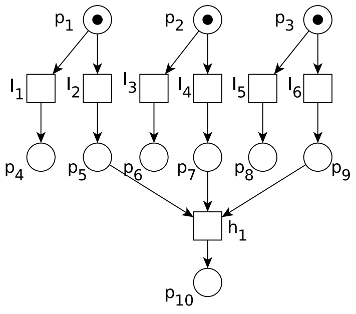

FIGURE 4.13

no pair of low transitions reveals a high transition. However, $\{l_2, l_4, l_6\} \multimap_{tr} \{h_1\}$, i.e., a low user, observing that all these three transitions occurred, can deduce that h_1 will inevitably occur. Thus, this net is 2-*ERNI* secure whereas it is not 3-*ERNI* secure. $\square$

Obviously, 1-*ERNI* coincides with *RNI*, where no low transition alone reveals a high transition. Moreover, k-*ERNI* $\subseteq$ *RNI*, for $k \geqslant 1$. N_9 is *RNI* secure since none of the low transitions reveals a high transition alone.

4.5.3 Non-interference based on repeated-reveals

Another case can be the one in which an attacker is not able to deduce information by observing low transitions and this is because only repeated occurrence of a low transition gives information about the hidden part of the net. Thus, we assume that the attacker can count the occurrences of low transitions and so he can deduce information about the high transitions.

Definition 4.5.3. Let $n > 0$.

N is secure with respect to *n-Repeated-Reveals based Non-Interference (n-ReRNI)* iff

$$\forall l \in L,\ \forall h \in H,\ \forall m \leqslant n \text{ such that } \Omega_l^m \neq \emptyset : \quad \neg(l \ \triangleright_{tr}^{m}\ h),$$

where Ω_l^m is the set of maximal runs containing n occurrences of l.

N is *ReRNI*, iff it is n-*ReRNI* for all $n > 0$.

Proposition 4.5.1. n-ReRNI $\implies$ $(n-1)$-ReRNI

The proof follows from the definition.

Example 4.5.3. N_{10} in Figure 4.12 is not 2-*ReRNI* secure. Although the first occurrence of l_1 does not reveal a high transition, by observing its second occurrence an observer can deduce that h_2 occurred. However, the net is *RNI* secure as well as *ERNI* secure. In the net in Figure 4.14, an observer cannot infer about the high transitions

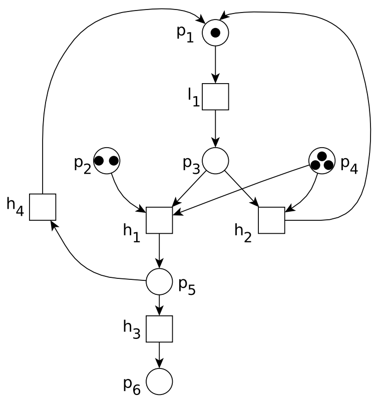

FIGURE 4.14

by observing l_1 occurring only once. Also the second occurrence of l_1 does not tell the observer which high transition occurred or will occur. However, the observer can deduce that h_2 has already occurred or will occur inevitably if he observes three occurrences of l_1. Therefore, this net is 2-*ReRNI* secure but it is not 3-*ReRNI* secure. Note that if the transition h_3 was absent then every maximal run would include at least one occurrence of h_2 and then, even without observing l_1, the occurrence of h_2 would be inevitable. $\square$

The following proposition is directly derived from Proposition 4.3.8.

Proposition 4.5.2. *If a net is* RNI *secure then it is* 1-ReRNI *secure.*

However, the previous implication does not hold in the opposite direction. Consider the net in Figure 4.7 on page 61 and let t_1 be a low transition, t_2 and t_3 be high transitions. This net is 1-*ReRNI* secure since the first occurrence of t_1 does not reveal information about t_2 and t_3, as discussed in Example 4.3.6. However, since we consider progress, i.e., maximal runs, the net is not *RNI* secure in fact $t_1 \rhd_{tr} t_2$ and $t_1 \rhd_{tr} t_3$. Note that this net is not secure with respect to 2-*ReRNI* since the second occurrence of t_1 reveals both t_2 and t_3, i.e. $t_1 \rhd^2_{tr} t_2$ and $t_1 \rhd^2_{tr} t_3$.

k-ERNI and *n-ReRNI* are in general not comparable since they are parametric notions which are based on observing different aspects: *k-ERNI* considers the observation of

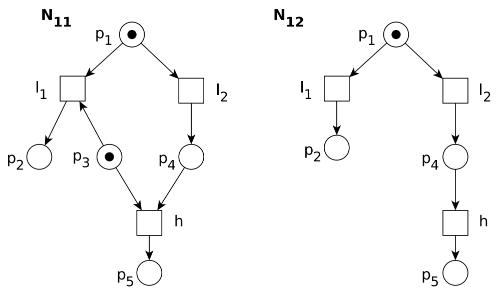

FIGURE 4.15

occurrences of different low transitions together, whereas *n-ReRNI* considers the observation of multiple occurrences of the same low transition. However, there are nets which are secure with respect to both notions and nets which are secure with respect to only one of them.

Both *k-ERNI* and *n-ReRNI* catch positive information flow about the past or future occurrences of high transitions, whereas they allow negative information flow. In the following we will introduce a notion considering both positive and negative information flow.

4.5.4 Positive/negative non-interference based on reveals and excludes

Until now we explored positive information flow on Petri nets. In order to catch negative information flow which is related to non-occurrence of high transitions, we need to consider the excludes relation between low and high transitions, as introduced in Definition 4.4.1.

Definition 4.5.4. N is secure with respect to *Positive/Negative Non-Interference (PNNI)* iff

$$\forall l \in L, \forall h \in H: \ l \not\triangleright_{tr} \ h \ \land \ \neg(l \ \underline{\text{ex}} \ h).$$

If in a Petri net N, no low transition reveals a high transition and no low transition excludes a high transition, N is considered to be *PNNI* secure. *PNNI* is stronger than *RNI*, i.e., *PNNI* $\subseteq$ *RNI*, and this follows directly from the definitions. In order to be *PNNI* secure, a net has to be *RNI* secure (no low transition reveals a high transition) and to satisfy an additional requirement (no low transition excludes a high transition).

Example 4.5.4. Both N_{11} and N_{12} in Figure 4.15 are not *PNNI* secure since a low transition l_1 excludes a high transition h. Thus, by observing occurrence of l_1, an observer can deduce that h did not and will not occur.

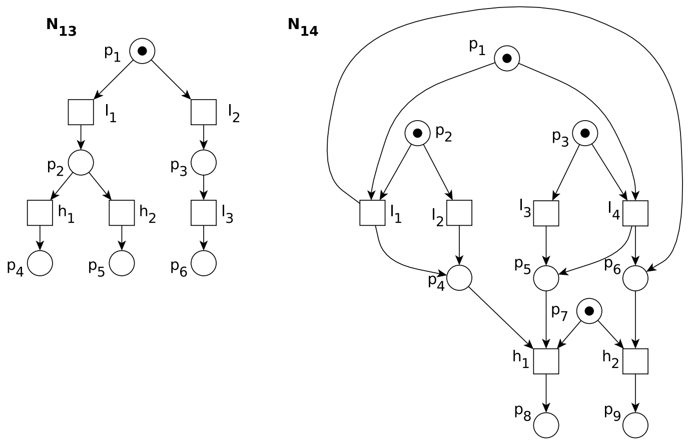

FIGURE 4.16

N_{13} in Figure 4.16 is not secure with respect to *PNNI* because of the negative information flow, i.e., l_2 excludes h_1 as well as it excludes h_2. An observer can deduce that none of the high transitions occurred and they will not occur in the future by observing l_2 or l_3. This net is *RNI*, *ERNI* and *ReRNI* secure.

In the same figure, N_{14} is a *PNNI* secure Petri net. No low transition reveals a high transition as well as no low transition excludes a high transition. However an observer is able to deduce that h_1 will occur inevitably by observing the occurrences of both l_2 and l_3, i.e., $\{l_2, l_3\} \multimapdotinv_{tr} \{h_1\}$. In other words, this net is not 2-*ERNI* while it is *RNI* and *ReRNI* secure. $\square$

As seen in the previous example, *PNNI* is strictly stronger than *RNI*.

PNNI and k-*ERNI* are intersecting for any k, *PNNI* $\cap$ k-*ERNI* $\neq \emptyset$, *PNNI* $\setminus$ k-*ERNI* $\neq \emptyset$, k-*ERNI* $\setminus$ *PNNI* $\neq \emptyset$. None of them is stronger than the other one. The net N_{15} in Figure 4.17 is both *ERNI* and *PNNI* secure, whereas N_{16} in Figure 4.17 is not *PNNI* secure, however it is *ERNI* secure. N_{14} of Figure 4.16 is *PNNI* secure, whereas it is not secure with respect to 2-*ERNI* as it is discussed in Example 4.5.4.

PNNI and n-*ReRNI* are also intersecting for any n. A net which is both *PNNI* and *ReRNI* secure is the one in Figure 4.2 on page 53. The net in Figure 4.14 on page 70 is not secure with respect to 3-*ReRNI* whereas it is *PNNI* secure. If we add to the net another low transition l_2 which consumes a token from p_5, the net becomes not secure with respect to *PNNI* as well as with respect to *RNI*, since l_2 reveals h_1 and moreover $l_2 \underline{\text{ex}}\, h_3$.

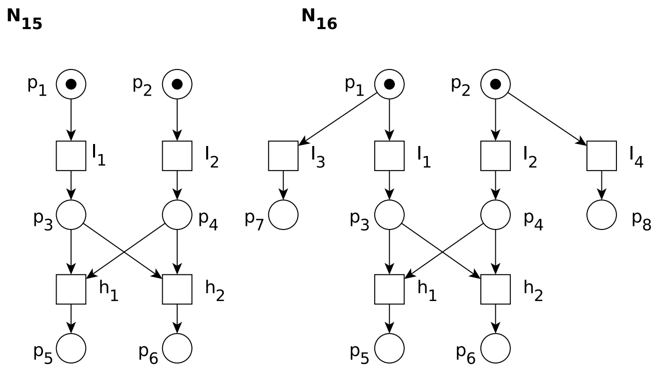

FIGURE 4.17

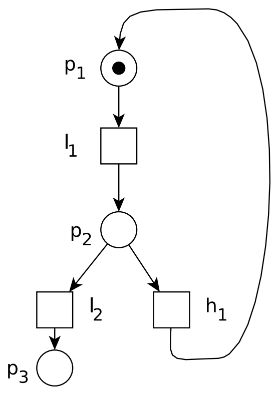

FIGURE 4.18

4.5.5 Improved positive/negative non-interference based on reveals and future/past excludes

Here we show that although *PNNI* is capable of capturing both positive and negative information flow, it needs to be improved since there is a certain flow which is ignored. The intuition is explained on the example below.

Example 4.5.5. Let us consider the net in Figure 4.18. This net is *PNNI* secure since none of the low transitions reveals or excludes h_1. In fact, a transition excludes another transition only if they can never occur together in the same run. In this example, l_1 can occur together with h_1 as well as l_2 can. However, there is a certain flow that is ignored by *PNNI* which is a transition can exclude another transition in the future (resp. past) whereas it does not exclude in the past (resp. future).

As we can see in Figure 4.18, h_1 can occur in the past of l_2 whereas it cannot occur in the future. Consequently, an observer can deduce that h_1 will never occur in the future once l_2 occurs, although h_1 might have occurred in the past. $l_2 \underline{\text{ex}}_f\ h_1$ whereas $\neg(l_2\ \underline{\text{ex}}\ h_1)$. $\square$

As explained in the Example 4.5.5, *PNNI* misses a certain kind of negative information flow which is due to the fact that:

$$\begin{aligned}\neg(t_1 \ \underline{\mathrm{ex}} \ t_2) &= \neg(t_1 \ \underline{\mathrm{ex}}_f \ t_2 \ \wedge \ t_1 \ \underline{\mathrm{ex}}_p \ t_2 \ ) \\ &= \neg(t_1 \ \underline{\mathrm{ex}}_f \ t_2 \ ) \ \vee \ \neg(t_1 \ \underline{\mathrm{ex}}_p \ t_2 \ ).\end{aligned}$$

In order to fix this weakness of *PNNI*, we must look for non-existence of both $\underline{\mathrm{ex}}_f$ and $\underline{\mathrm{ex}}_p$ between high and low transitions instead of only looking for $\underline{\mathrm{ex}}$.

Definition 4.5.5. N is secure with respect to *Improved-Positive/Negative Non-Interference (I-PNNI)* iff

$$\forall l \in L, \ \forall h \in H : \ \ l \ \not\rhd_{tr} \ \ h \ \wedge \ \neg(l \ \underline{\mathrm{ex}}_f \ h) \ \wedge \ \neg(l \ \underline{\mathrm{ex}}_f \ h).$$

Proposition 4.5.3. I-PNNI $\implies$ PNNI.

Proof. The proof is quite straightforward. Let $t_1, t_2 \in T$, by Proposition 4.4.3 $t_1 \ \underline{\mathrm{ex}} \ t_2 \iff t_1 \ \underline{\mathrm{ex}}_f \ t_2 \ \wedge \ t_1 \ \underline{\mathrm{ex}}_p \ t_2$. If a net is *I-PNNI* then, besides the reveals requirement, $\forall l \in L, \ \forall h \in H : \ \neg(l \ \underline{\mathrm{ex}}_f \ h) \wedge \neg(l \ \underline{\mathrm{ex}}_p \ h)$ which implies that $\neg(l \ \underline{\mathrm{ex}}_f \ h) \vee \neg(l \ \underline{\mathrm{ex}}_p \ h) = \neg(l \ \underline{\mathrm{ex}} \ h)$ which is the excludes requirement of *PNNI*, note that reveals requirements are the same in *PNNI* and *I-PNNI*.

Clearly, the implication does not hold the other direction and *I-PNNI* is strictly stronger than *PNNI*. □

Example 4.5.6. The net Figure 4.18 is not secure with respect to *I-PNNI* whereas it is *PNNI* secure. N_{13} in Figure 4.16 on page 72 is not *PNNI* and so not *I-PNNI* secure. On the other hand, N_{14} of Figure 4.16 is *I-PNNI* secure hence it is also *PNNI* secure. □

4.6 Comparison of Non-interference Notions with the Ones in the Literature

We have introduced new notions of non-interference for Petri nets. These notions are based on the reveals and the excludes relations and on the progress assumption.

One major difference between these notions with the existing ones, recalled in Section 4.2, is that the new notions explicitly consider the information flow both about the past and the future occurrences of high transitions. For example, if a low user can tell that the occurrence of a high transition is inevitable in the future, such a system is considered to be not secure according to the notions we have here introduced, whereas it is considered secure by the old notions such as *SNNI*, *BSNNI*, *PBNI+* and *PBNI*. Similarly, for the negative information flow, we consider both past and future non-occurrences of high transitions.

Another important difference is shown by N_4 in Figure 4.2 on page 53. This net is not secure according to *SNNI* even if a low user cannot infer which high transitions actually occurred. On the other hand, it is secure with respect to all non-interference notions based on reveals and excludes, since these require the capability of differentiating among the high transitions.

Moreover, the notions recalled in Section 4.2 are defined for 1-safe Petri nets, whereas *RNI*, *k-ERNI*, *n-ReRNI* and *PNNI* are defined for general Petri nets. Figure 4.19

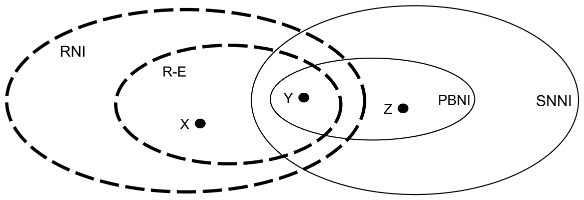

FIGURE 4.19

illustrates the relation between our notions and the other notions we have discussed so far. For the sake of simplicity, we only consider the weakest (*SNNI*) and the strongest (*PBNI*) notions from the ones recalled in Section 4.2. with the weakest of the new notions, i.e., *RNI*, and with the intersection set, denoted *R-E* in Figure 4.19, of the new notions *RNI*, *k-ERNI*, *n-ReRNI* and *I-PNNI*.

We will examine three examples to discuss the differences of these classes.

A net which is secure with respect to all notions based on reveals and excludes and which is not secure with respect to *SNNI* is denoted by X in Figure 4.19 and it is the one in Figure 4.2 on page 53. We consider this net secure since an observer cannot differentiate among the high transitions even if he can know some high actions have been performed (or will be performed). However, this net is not secure with respect to *SNNI*.

The net denoted by Y in Figure 4.19 is secure with respect to all non-interference notions based on reveals and excludes as well as with respect to *PBNI*. This net can be N_{15} in Figure 4.17 on page 73. This net is secure since no low transition reveals a high transition (alone or together with another transition) as well as no low transition excludes a high transition. Thus there is neither positive nor negative information flow. It is also secure with respect to *PBNI* due to the fact that there is no active causal or active conflict place.

Two nets which are secure with respect to *PBNI* but not secure with respect to any of the non-interference notions based on reveals and excludes, denoted by Z in Figure 4.19, are for example N_6 in Figure 4.11 on page 67 and N_{12} in Figure 4.15 on page 71.

4.7 Checking Non-interference

In this section, after briefly discussing the methods used for checking the non-interference notions for Petri nets in the literature, we will introduce two methods for checking the non-interference notions introduced in Section 4.5. The first method is based on translating the reveals and excludes relations to Linear Temporal Logic (LTL), and to use LTL model checking methods for checking reveals/excludes based non-interference notions. The second method adapts the diagnosis algorithm proposed in [40].

4.7.1 The methods in the literature

In [14], the authors prove the decidability of *NDC* and *BNDC* in two settings: with only high and low transitions and with also downgrading transitions. [13] generalizes results of [14] by considering "selective declassification", i.e.: each downgrading transition can declassify just a subset of high transitions. The paper introduces a notion of *Intransitive Non Interference with Selective Declassification (INISD)* on the basis of *NDC* with downgrading transitions. Moreover, it proves decidability of this new property *INISD* by an algorithm to check this property.

[67] introduces an algorithm for checking *BNDC* in elementary nets on the basis of some specific places in the net. The algorithm, which is also implemented as a tool, uses the fact that *PBNI+* and *BNDC* are proved to be equivalent in elementary nets [21].

In [3, 4], the authors examine *BNDC* on unfoldings of 1-safe Petri nets and show that it admits some characterizations in terms of causalities and conflicts between high and low level events. On the basis of these characterizations, the authors provide an algorithm and a tool for checking *BNDC* in 1-safe Petri nets.

4.7.2 LTL model checking

One of the approaches we propose for checking the non-interference notions that are introduced in Section 4.5 is based on Linear Temporal Logic (LTL) model checking.

The introduced notions are based on two main relations and their variants. Thus, deciding whether a net is secure requires computation of reveals and excludes relations depending on the specific non-interference notion.

A popular technique for system verification is model checking. Given a model of a system, model checking is based on exhaustively and automatically checking whether the given model satisfies a given specification. The specification to be checked is expressed with a logic formula. LTL [54] is one of the widely used propositional logics and there is a rich literature on model checking with LTL.

[71] introduces an automata theoretic approach to model checking with LTL by pointing out the relation between LTL and ω-words. In [35] the authors provide a tableau-based algorithm for obtaining an automaton from a temporal logic formula for being used in model checking in an "on-the-fly" fashion.

However, the traditional model checking methods, which are based on exploring the reachable states of a system, face the state space explosion problem.

On the other hand, an alternative method for model checking distributed systems is based on unfoldings. Unfoldings are well-studied partial order semantics for Petri nets and in [74] the authors show that the finite complete prefix (as defined in [49]) of an unfolding is suited for model checking linear-time temporal properties. The method is based on the so-called automata-theoretic approach to model checking. They propose a technique based on finite prefixes of unfoldings of 1-safe Petri nets. Unfortunately, as explained in [28], although the algorithm has been applied with success to a variety of examples, it is not satisfactory because, after constructing the complete prefix, construction of an additional graph is required. This graph can be exponentially larger than the complete prefix itself. In [28], the authors propose an unfolding based LTL model checking method which overcomes the problem. With the new method, the model checking can be done directly on the prefix without requiring the additional construction of the possibly exponential graph. The authors propose a new unfolding method for creating a prefix which is similar to the old algorithm for the complete prefix except with a new cut-off criterion which is studied in more details in [29]. The new prefix has a larger size than McMillan's complete prefix. The theoretical upper bound on the number of events in the new prefix is $O(K^2)$ events where K is the number of reachable states for 1-safe Petri nets. In practice, the new prefix is much smaller than the state space.

In this section, we will introduce a method for translating our reveals and excludes relations to LTL, so that the model checking methods can be applied for checking new non-interference notions introduced in Section 4.5.

4.7.2.1 LTL translation of reveals and excludes relations

In order to use the existing LTL model checking methods for checking reveals and excludes relations (and so non-interference notions based on these relations), we propose a method for translating these relations to LTL formulas.

In the sequel $N = (P, T, F, m_0)$ will denote a 1-safe, 1-live Petri net and $MG(N)$ will denote the marking graph of N.

LTL formulas are based on atomic propositions which correspond to local properties in a state of a transition system. We will use a standard notation, where $\Box$ denotes the *always* operator, $\Diamond$ denotes the *eventually* operator, and $\mathcal{U}$ denotes the *until* operator. In order to represent reveals and excludes relations in LTL, we first need to express the occurrence of a transition as a state. In this way, occurrence of a transition will set the value of a specific atomic proposition true. For this purpose, we use the marking graph of the net which represents the reachable markings and their relations with the transitions. However, one cannot tell if a transition has occurred by looking at a reachable marking since for example occurrence of different transitions may lead to the same marking.

Here we propose a method for translating reveals and excludes relations between two transitions of a 1-safe 1-live Petri net to LTL formulas. In other words, we construct an LTL language for expressing the new introduced relations. This requires a transformation of the net N for the reason explained above.

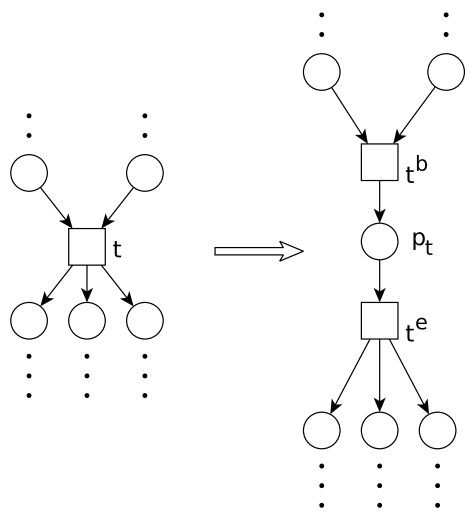

FIGURE 4.20

As illustrated in Figure 4.20, we replace transition t with two transitions t^b and t^e which stand for the beginning and end of transition t. In other words, we divide the occurrence of t into two phases and add a place p_t in between which is marked only when t is being fired. Since t and t_b has the exact same preconditions, t^b occurs whenever t occurs. Moreover, considering progress assumption, t^e will inevitably occur once t_b occurs because the only precondition of t^e is the only postcondition of t^b which is not used by any other transition except t^e. In other words, t^b reveals t^e.

Let $t_1, t_2 \in T$ be the two transitions to be checked. From $N = (P, T, F, m_0)$ we construct another 1-safe 1-live Petri net $N' = (P \cup \{p_{t_1}, p_{t_2}\}, (T \setminus \{t_1, t_2\}) \cup \{t_1^b, t_1^e, t_2^b, t_2^e\}, F', m_0)$. Let $\tilde{F}$ be the set of all arcs that are related to neither t_1 nor t_2. Let F_{new} be the set of all the arcs from ${}^\bullet t_i$ to t_i^b, from t_i^b to p_{t_i}, from p_{t_i} to t_i^e, and from t_i^e to $t_i^\bullet$, for $i = \{1, 2\}$. $F' = \tilde{F} \cup F_{new}$. The initial marking m_0 is not affected by the transformation. This new net behaves as N except that t_1 and t_2 are divided into two phases and the net includes two additional places.

Let $m \in [m_0\rangle_{N'}$ be a reachable marking in N' (so a state in $MG(N')$). Let $A = \{p_{t_1}, p_{t_2}\}$ be the set of atomic propositions such that $m \vDash p_{t_i}$ iff $p_{t_i} \in m$. After the transformation explained above, we can translate reveals and excludes relations between two transitions of a 1-safe 1-live Petri net to LTL formulas as in the following:

$t_1 \ \triangleright_{tr} \ t_2$: $\Diamond p_{t_1} \implies \Diamond p_{t_2}$

$t_1 \ \underline{\text{ex}}\ t_2$: $\Diamond p_{t_1} \implies \Box \neg p_{t_2}$

$t_1 \ \underline{\text{ex}}_f\ t_2$: $((\neg p_{t_1})\mathcal{U}(p_{t_1} \ \wedge \ \Box \neg p_{t_2})) \ \vee \ \Box \neg p_{t_1}$

$t_1 \ \underline{\text{ex}}_p\ t_2$: $\Box((\neg p_{t_2})\mathcal{U}(p_{t_1})) \ \vee \ \Box \neg p_{t_1}$

Now we can check the reveals and excludes relations in N by model checking the corresponding formulas in $MG(N')$.

Example 4.7.1. Let $N = (P, T, F, m_0)$ be the net in Figure 4.21 such that $T = H \cup L$, $H \cap L = \emptyset$, $L, H \neq \emptyset$. Let us check if this net is secure with respect to *I-PNNI*. *I-PNNI* requires that $\forall l \in L, \ \forall h \in H : \ l \ \not\triangleright_{tr} \ h \ \wedge \ \neg(l \ \underline{\text{ex}}_f \ h) \ \wedge \ \neg(l \ \underline{\text{ex}}_p \ h)$. N is secure with respect to *I-PNNI* iff $\forall l \in L, \ \forall h \in H$, setting $\phi_r = \Diamond p_l \implies \Diamond p_h$, $\phi_f = ((\neg p_{t_1})\mathcal{U}(p_{t_1} \wedge \Box \neg p_{t_2})) \vee \Box \neg p_{t_1}$ and $\phi_p = \Box((\neg p_{t_2})\mathcal{U}(p_{t_1})) \vee \Box \neg p_{t_1}$, $\quad MG(N') \nvDash \phi_r \ \wedge \ MG(N') \nvDash \phi_f \ \wedge \ MG(N') \nvDash \phi_p$.

In this example, we can see that $MG(N') \vDash \phi_f$ for l_1 and h_1 which means that $l_1 \ \underline{\text{ex}}_f \ h_1$. So, N is not secure with respect to *I-PNNI*. $\square$

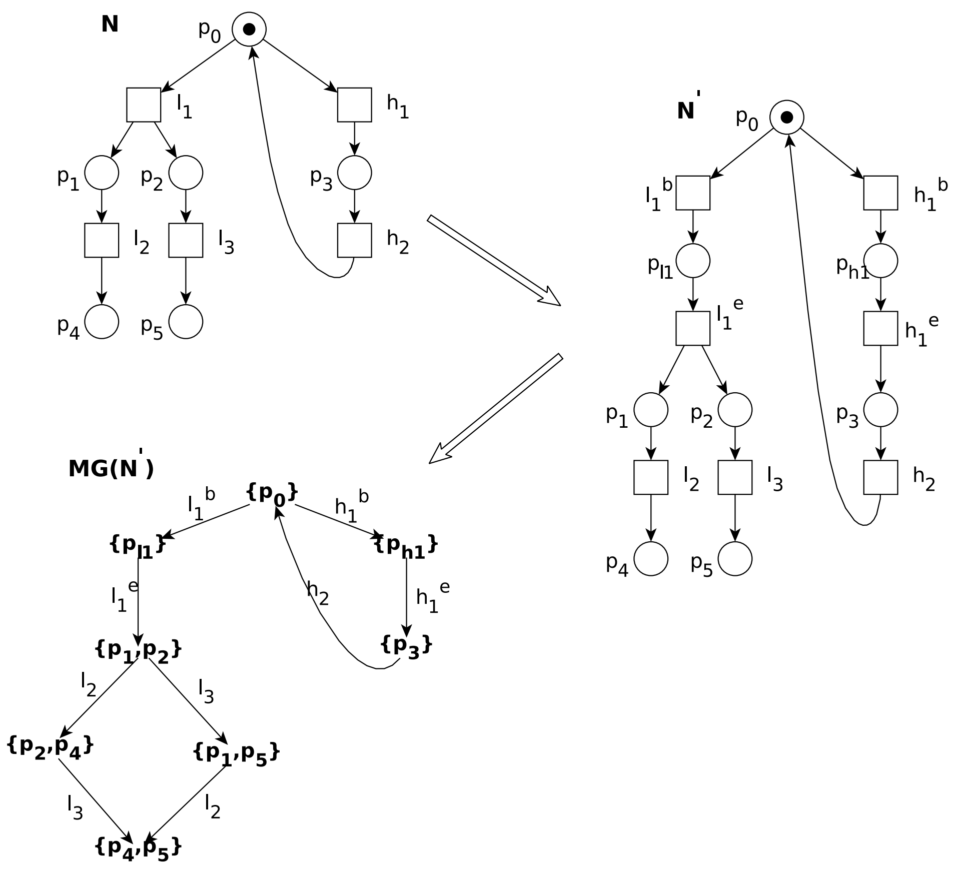

FIGURE 4.21

4.7.3 Diagnosis approach

In [40], Haar et al. present a method for fault diagnosis in concurrent, partially observable systems under a weak fairness assumption. The question to be answered in this diagnosis problem is whether an observable behavior allows to determine that a non-observable fault will inevitably occur or has already occurred. Their approach is based on computing a finite compact prefix of the unfolding of a 1-safe Petri net that carries sufficient information for the diagnosis. This approach can also be applied to checking reveals and excludes relations on Petri nets and thus is useful for checking non-interference on finite prefixes of unfoldings of safe Petri nets. The method captures indirect revealed dependencies such as inevitable occurrences of concurrent or future events by extending and generalizing the unfolding-based diagnosis approaches by Benveniste et al. [6] as well as by Esparza and Kern [30]. In [40], the authors provide an algorithmic solution to the diagnosis problem by translating it into a SAT instance, which we adapt for our problem of checking non-interference on finite prefixes of unfoldings. Here, after briefly recalling their solution, we show how to adapt it to non-interference check.

4.7.3.1 Fault diagnosis with extended-reveals

We recall the algorithm for the fault diagnosis problem proposed in [40], and include some essential formal background in order to be self-contained. The algorithm is based on extended-reveals and finite prefixes of unfoldings.

Definition 4.7.1. [40] Let $N = (P, T, F, m_0)$ be a 1-safe Petri net. A *spoiler* of a transition $t \in T$ is any $t' \in T$ such that ${}^{\bullet}t \cap {}^{\bullet}t' \neq \emptyset$, and spoilers$(t)$ denotes the set of all spoilers of t. Note that $t \in$ spoilers(t).

An infinite transition sequence $\sigma = t_1 t_2 ... \in T^{\omega}$ of N is *weakly fair* if the corresponding marking sequence $m_0, m_1, ...$ satisfies that for all $i \in \mathbb{N}$ and all $t \in T$, if m_i enables t, then there exists $j > i$ such that $t_j \in$ spoilers(t).

A *labelled partial order (LPO)* over a finite alphabet $\mathbb{X}$ is a tuple $\alpha = (S, <, \ell)$ where $< \subseteq S \times S$ is an irreflexive and transitive relation on S, and $\ell : S \to \mathbb{X}$ is a labeling map. The size of α, denoted by $|\alpha|$, is $|S|$. Let $\alpha' = (S', <', \ell')$ be an LPO over $\mathbb{X}$. A *homomorphism* from α to α' is a function $h : S \to S'$ satisfying

- $\ell(a) = \ell'(h(a))$, and
- $a < b$ implies $h(a) <' h(b)$ for all $a, b \in S$.

An *isomorphism* between α and α' is a bijective homomorphism h from α to α' where h^{-1} is a homomorphism from α' to α.

α is *compatible* with α' if there exists a bijective function $f : S \to S'$ such that

- $\ell(a) = \ell'(f(a))$, and
- $a < b$ implies $\neg(f(b) <' f(a))$ for all $a, b \in S$.

compat(α) denotes the set of LPOs compatible with α. Compatibility is a symmetric relation.

The approach proposed in [40] aims to diagnose whether every weakly fair run [1], that is compatible with the observations so far, contains a fault occurrence. Thus, they consider all weakly fair runs that contain an "explanation" (the formal definition of explanation will be given later) of the current observation as a prefix.

[1] In [40], what we call run here is called a *configuration*, whereas 'run' stands for an interleaving of a configuration.

In the sequel, $N = (P, T, F, m_0)$ will denote a finite, 1-safe Petri net with low (observable) and high (unobservable) transitions such that $T = L \cup H$, $L \cap H = \emptyset$, $L, H \neq \emptyset$ where L denotes the low transitions and H denotes the high transitions. $\mathrm{Unf}(N)$ will denote the unfolding of N: $\mathrm{Unf}(N) = ((B, E, F, c_0), \lambda)$, $\lambda : B \cup E \to P \cup T$, and $E_L = \lambda^{-1}(L)$ will denote the low events whereas $E_H = \lambda^{-1}(H)$ will be the set of high events. Let $\mathbb{X}$ denote a non-empty *observation alphabet* and ε be a symbol such that $\varepsilon \notin \mathbb{X}$ which will be used to label unobservable transitions. Let $\ell : T \to \mathbb{X} \cup \{\varepsilon\}$ be a mapping such that $\ell(H) = \{\varepsilon\}$ and let $\phi \in H$ be the unique *fault* transition. We will denote the set of *fault events* by $E_\phi = \lambda^{-1}(\phi)$.

Observations are LPOs over an observation alphabet. A *finite observation pattern* α is an LPO, $\alpha = (S_\alpha, <_\alpha, \ell_\alpha)$ over the observation alphabet $\mathbb{X}$, such that S_α is finite.

The goal is to determine, for a given N and an observation α, whether any weakly fair execution corresponding to α contains a fault. Thus, all weakly fair runs of $\mathrm{Unf}(N)$ that are compatible with α are needed to be considered. For this purpose, each run $\omega \in R$ of $\mathrm{Unf}(N)$, where R is the set of all runs, is associated with an LPO $\mathit{lpo}(\omega) = (S, <', \ell')$, where $S = \omega \cap E_L$ are the observable events in ω, $<'$ is the restriction of $<$ and $\ell' : S \to \mathbb{X}$ is the restriction of ℓ to S. Since $<'$ and ℓ' are restrictions of $<$ and ℓ, we can use them interchangeably.

The set of *observations* of ω is defined as $\mathit{obs}(\omega) = \mathit{compat}(\mathit{lpo}(\omega))$. A run ω *explains* observation α if $\alpha \in \mathit{obs}(\omega)$ and the set of explanations of α is defined as $\mathit{expl}(\alpha) = \{\omega \in R : \alpha \in \mathit{obs}(\omega)\}$.

Definition 4.7.2. [40] An observation pattern α *diagnoses* ϕ iff

$$\forall \omega \in \mathit{expl}(\alpha) : \quad \omega \twoheadrightarrow E_\phi.$$

Example 4.7.2. In the net illustrated in Figure 4.22, let transitions $l_1, \ldots, l_6$ be observable, $h_1, \ldots, h_4$ and ϕ be unobservable where ϕ is the fault transition. The observation of l_1 and l_2 diagnoses the fault ϕ. Each run which allows to observe l_1 and l_2 extended-reveals the fault, e.g., $\{e_1, e_2\} \twoheadrightarrow \{e_3, e_4\}$. Each maximal run in $\mathrm{Unf}(N)$ which contains occurrences of both l_1 and l_2 must also contain an occurrence of ϕ. □

The *diagnosis problem* is to decide whether the observation α diagnoses ϕ in N. Haar et al. show that if $\omega \in \mathit{expl}(\alpha)$ is finite, then $\omega \twoheadrightarrow E_\phi$ can be verified on a bounded extension of ω. For our problem of non-interference, ω is always finite, as it will be clear later on.

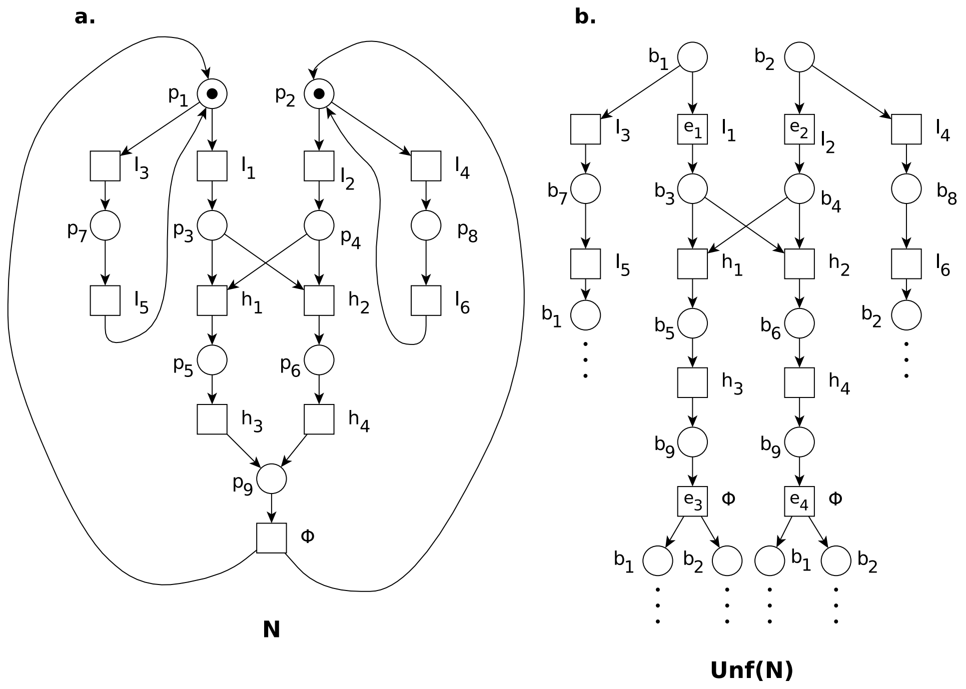

FIGURE 4.22: A Petri net with its unfolding

Definition 4.7.2 can be equivalently rephrased as:

$$\forall \omega_{max} \in \Omega : \; (\exists \omega \in \; expl(\alpha) : \omega \subseteq \omega_{max}) \implies E_\phi \cap \omega_{max} \neq \emptyset.$$

The algorithm proposed in [40] and recalled here decides the negation of the above statement. In general, $expl(\alpha)$ can be infinite due to the infinite unobservable loops. However, this obstacle is solved by showing that it is sufficient for deciding the diagnosis problem to search for ω within a finite subset of $expl(\alpha)$, instead of the entire set of explanations. Since the subset is finite, there exists an unfolding prefix, denoted $\mathcal{P}_\alpha$, which contains the entire subset. The algorithm checks if ω can be extended to a fault-free weakly fair maximal run, $\omega_{max} \in \Omega$, once such ω is found. This is the case iff two runs of the unfolding $\omega_1 \subset \omega_2$ exist such that both reach the same marking, i.e., $mark(\omega_1) = mark(\omega_2)$, both are fault-free, ω_2 disables every event enabled by ω_1 and $\omega \subseteq \omega_1$. By "$\omega_2$ disables every event enabled by ω_1" we mean the following: Let $\#[e] = \{\tilde{e} \in E : \; e\#\tilde{e}\}$. $\forall e \in E$ such that $^\bullet e \subseteq mark(\omega_1)$ it holds that $e \in \omega_2 \; \vee \; \#[e] \cap \omega_2 \neq \emptyset$. However, ω_2 can be unboundedly large. To fix this, two finite unfolding prefixes, $\mathcal{P}_N^1$ and $\mathcal{P}_N^2$ are defined in a way that $\mathcal{P}_N^1$ is contained in $\mathcal{P}_N^2$, i.e., $\mathcal{P}_N^1 \sqsubseteq \mathcal{P}_N^2$, and ω_1, ω_2 exist iff $\tilde{\omega}_1 \in R(\mathcal{P}_N^1)$ and $\tilde{\omega}_2 \in R(\mathcal{P}_N^2)$ exist and satisfy that $mark(\tilde{\omega}_1) = mark(\tilde{\omega}_2)$, both are fault-free, $\tilde{\omega}_2$ disables every event enabled by $\tilde{\omega}_1$ and $\omega \subseteq \tilde{\omega}_1$, where $R(\mathcal{P}_N^1)$ and $R(\mathcal{P}_N^2)$ are the sets of all runs in $\mathcal{P}_N^1$ and $\mathcal{P}_N^2$.

Below, a finite subclass of explanations of α which is sufficient for deciding the diagnosis problem is given.

Definition 4.7.3. A run $\omega \in R$ is *verbose* if it contains two events e, e' such that (1) $e' < e$, (2) $mark([e]) = mark([e'])$, and (3) $obs([e]) = obs([e'])$ where $[x] = \{y \in E : y \leqslant x\}$, i.e.,$[x]$ is the causal past. If ω is not verbose, it is *succinct*.

In order to define a finite prefix $\mathcal{P}_N^2$ that includes $\mathcal{P}_N^1$ and preserves not only reachable markings but also the capability of a run to spoil previously enabled events, a cut-off criterion called sp-cutoff is defined in [40] as below.

Definition 4.7.4. [40] Let $\mathcal{P}_N^1 = (B_1, E_1, F_1, \tilde{c_0})$. Event $e \in E$ is an *sp-cutoff* if there exists $e' \in E$, $e' < e$, such that, setting $D = [e] \setminus [e']$, the following holds:

(1) $\lambda({}^\bullet D \setminus D^\bullet) = \lambda(D^\bullet \setminus^\bullet D)$

(2) $B_1 \cap {}^\bullet D = \emptyset$.

The first condition implies that $mark([e]) = mark([e'])$.

$\mathcal{P}_N^2 = (B_2, E_2, F_2, \tilde{c_0})$ is defined as the unfolding prefix whose events are exactly all non sp-cutoff events, i.e., $E_2 = \{e \in E : e \text{ is not sp-cutoff}\}$ and it is shown in [40] that $\mathcal{P}_N^2$ is finite.

The theorem below states a set of necessary and sufficient conditions that characterize whether or not a given observation α diagnoses a fault ϕ.

Theorem 4.7.1. *[40] Observation α does not diagnose ϕ iff there exist runs $\omega, \omega_1' \in R$, $\omega_1 \in R(\mathcal{P}_N^1)$, $\omega_2 \in R(\mathcal{P}_N^2)$ satisfying that*

1. *ω is a succinct explanation of α*
2. *$\omega \subseteq \omega_1'$*
3. *$\omega_1 \subseteq \omega_2$*
4. *$mark(\omega_1') = mark(\omega_1) = mark(\omega_2)$*
5. *$\forall e \in E \;:\; \omega_1 \text{ enables } e \implies \text{spoilers}(e) \cap \omega_2 \neq \emptyset$*
6. *neither ω_1' nor ω_2 contains an occurrence of ϕ.*

Given the observation α, in order to decide whether or not α diagnoses ϕ, all conditions of Theorem 4.7.1 must be checked. For this, it is enough to construct three unfolding prefixes: $\mathcal{P}_\alpha$, containing all succinct explanations of α, and $\mathcal{P}_N^1 \sqsubseteq \mathcal{P}_N^2$ to check for the existence of a weakly fair run starting from a given marking and whether one marking is reachable from another. Note that $\mathcal{P}_\alpha$ depends only on the observation whereas $\mathcal{P}_N^2$ depends only on N and can be constructed offline.

Well known algorithms for computing unfolding prefixes such as [31] aim to obtain a *marking-complete* prefix. They start with the initial marking of $\mathrm{Unf}(N)$, then add events to the prefix until each branch reaches a cutoff event. Haar et al. in [40] replace the cutoff criteria with the one in Definition 4.7.4.

The construction must be restricted to the explanations of α and all succinct explanations must be preserved whereas verbose ones are eliminated. Thus, the considered $\omega \in \mathit{expl}(\alpha)$ are always finite. For this, N is synchronized with a net representing $\alpha = (S, <, \ell)$. Let S_{min} (resp. S_{max}) be the elements without predecessor (resp. successor) in S. α is translated into an occurrence net $O_\alpha = (P_\alpha, S, F_\alpha, m_\alpha)$, whose events are S and whose causal relation is $<$. $P_\alpha = P_{min} \uplus P_{mid} \uplus P_{max}$ is partitioned into three sets, where P_{max} (resp. P_{min}) is the postset (resp. preset) conditions of S_{max} (resp. S_{min}). Then $N = (P, T, F, m_0)$ and O_α are composed into a net $N_\alpha = (P', T_o \cup T_u, G, m_0')$, where:

- $P' = P \cup P_\alpha$
- $T_o = \{(t, s) : t \in L, s \in S, \ell(t) = \ell(s)\}$
- $T_u = H \setminus \{\phi\}$
- $\forall (t, s) \in T_o, {}^\bullet(t, s) = {}^\bullet t \cup {}^\bullet s$ and $(t, s) = t^\bullet \cup s^\bullet$
- $\forall t \in T_u, {}^\bullet t$ and $t^\bullet$ remain as in N
- $m_0' = m_0 \cup m_\alpha$.

$\mathcal{P}_\alpha$ is the finite prefix of $\mathrm{Unf}(N_\alpha)$ which is cut at any event e such that there exists $e' < e$ such that $\mathit{mark}([e']) = \mathit{mark}([e])$. Projecting each event labeled with a tuple (t, s) to t instead, each run ω of $\mathrm{Unf}(N_\alpha)$ is also a run of $\mathrm{Unf}(N)$.

For constructing $\mathcal{P}_N^2$, first $\mathcal{P}_N^1$ is obtained by the usual unfolding methods for marking-complete prefixes. Then it is extended by additional events using the sp-cutoff criterion.

The authors also propose an encoding of the diagnosis problem into SAT in [40]. This allows one to use a SAT solver to solve the diagnosis problem.

4.7.3.2 Checking reveals and excludes relations by the diagnosis approach

In Section 4.5, we have introduced and examined new non-interference notions which are based on two main relations and their variations, i.e., *reveals* and *excludes* defined and studied in Section 4.3.2 and Section 4.4. In this section, we will show how to check reveals and excludes relations on finite prefixes of unfoldings following the solution of Haar et al. for diagnosis problem proposed in [40] and recalled in Section 4.7.3.1.

In [40], the authors propose an algorithm for deciding whether observation α diagnoses fault ϕ, i.e., $\forall \omega \in \mathit{expl}(\alpha) \;\; \omega \twoheadrightarrow E_\phi$ where observation α is an LPO whereas ϕ is the fault transition. The reveals relation between transitions can be rephrased as a diagnosis problem with the help of Definition 4.3.4. Intuitively, transition t_1 reveals transition t_2 iff each occurrence of t_1 extended-reveals some occurrence of t_2. If we assume that all transitions are unobservable except t_1 and assume t_2 is the fault ϕ, then to decide whether t_1 reveals t_2 will be equivalent to decide if observation of t_1 diagnoses t_2, i.e., each maximal run, in which t_1 is observed, contains an occurrence of t_2.

The next theorem shows that the statement $t_1 \;\rhd_{tr}\; t_2$ can be equivalently translated to "the observation of t_1 diagnoses t_2" if all transitions except t_1 are unobservable.

Theorem 4.7.2. *Let $N = (P, T, F, m_0)$ be a 1-safe Petri net, and $\mathrm{Unf}(N) = ((B, E, F), \lambda)$, be the unfolding of N. Let R be the set of all runs and Ω be the set of all maximal runs of N. Let $t_1, t_2 \in T$. Assume that all transitions of N are unobservable except t_1. Then $t_1 \;\rhd_{tr}\; t_2$ iff α diagnoses t_2 for $\alpha = (\{s_0\}, <, \ell)$ where $\ell(s_0) = t_1$.*

Proof. We first show that $t_1 \;\rhd_{tr}\; t_2 \implies \alpha$ diagnoses t_2. Since $t_1 \;\rhd_{tr}\; t_2$, directly from Definition 4.3.4 we get $\forall e \in E_{t_1} \; \{e\} \twoheadrightarrow E_{t_2}$, and by using the definition of $\twoheadrightarrow$ we write $\forall \omega_{max} \in \Omega \;\; \forall e \in E_{t_1} : \;\; \{e\} \subseteq \omega_{max} \implies E_{t_2} \cap \omega_{max} \neq \emptyset$. By definition, "$\alpha$ diagnoses t_2" means that $\forall \omega \in \mathit{expl}(\alpha) : \;\; \omega \twoheadrightarrow E_{t_2}$. The explanation set of α is defined as $\mathit{expl}(\alpha) = \{\omega \in R : \alpha \in \mathit{obs}(\omega)\}$ where $\mathit{obs}(\omega) = \mathit{compat}(\mathit{lpo}(\omega))$ and two observations are compatible if there exists a bijective function as explained in Section 4.7.3.1. Assuming that all transitions except t_1 are unobservable, for $\alpha = (\{s_0\}, <, \ell)$, $\ell(s_0) = t_1$, the explanation set consists of runs in $\mathrm{Unf}(N)$ which contain exactly one occurrence of t_1, i.e., $\mathit{expl}(\alpha) = \{r \in R : \; |r \cap E_{t_1}| = 1\}$. Since each element ω of $\mathit{expl}(\alpha)$ contains one occurrence of t_1, each maximal run extending ω contains at least one occurrence of t_1. We assumed that $t_1 \;\rhd_{tr}\; t_2$, so $\forall \omega_{max} \in \Omega$ for which $\exists \omega \in \mathit{expl}(\alpha)$ such that $\omega \subseteq \omega_{max}$, it holds that $\omega_{max} \cap E_{t_2} \neq \emptyset$. This proves that $t_1 \;\rhd_{tr}\; t_2 \implies \alpha$ diagnoses t_2.

Now we show that α diagnoses $t_2 \implies t_1 \;\rhd_{tr}\; t_2$. Assuming that α diagnoses t_2 we get $\forall \omega \in \mathit{expl}(\alpha) : \;\; \omega \twoheadrightarrow E_{t_2}$. This means that all maximal runs which extend ω include an occurrence of t_2. As explained above, each $\omega \in expl(\alpha)$ contains exactly one occurrence of t_1 since α has a single point which corresponds to an occurrence of t_1. Since ω is a run and it is defined as a causally closed conflict free set of events on the unfolding, each element ω of $expl(\alpha)$ contains exactly one occurrence of t_1 and of course this is the first occurrence of t_1 for any extension of ω. Since the occurrences of a given transition are totally ordered, this implies that $t_1 \ \rhd_{tr} \ t_2$. $\square$

As a result of Theorem 4.7.2, the reveals relation between two transitions of a 1-safe Petri net N can be checked on a finite prefix of $\text{Unf}(N)$. Let us now explain how to adapt the algorithm presented in [40] for the problem of checking reveals relation between two transitions on an example.

Example 4.7.3. Let $N = (P, T, F, m_0)$ be the net in Figure 4.23, and $\text{Unf}(N) = ((B, E, F), \lambda)$, be the unfolding of N which is illustrated in Figure 4.24. Let R be the set of all runs and Ω be the set of all maximal runs of N. In order to check if $t_0 \ \rhd_{tr} \ t_1$ we assume that all transitions of N are unobservable except t_0. We define $\alpha = (\{s_0\}, <, \ell)$ where $\ell(s_0) = t_0$ and $\phi = t_1$. If all the properties of Theorem 4.7.1 hold for α and ϕ, it means that α does not diagnose ϕ, i.e., $t_0 \ \not\rhd_{tr} \ t_1$.

We need to calculate three unfolding prefixes, $\mathcal{P}_\alpha$ and $\mathcal{P}_N^1 \sqsubseteq \mathcal{P}_N^2$. $\mathcal{P}_\alpha$ contains all succinct explanations of α and it depends on the observation whereas $\mathcal{P}_N^1 \sqsubseteq \mathcal{P}_N^2$ only depend on the net.

We construct $\mathcal{P}_N^1 \sqsubseteq \mathcal{P}_N^2$ in two steps: first we construct $\mathcal{P}_N^1$ which is the marking complete prefix of $\text{Unf}(N)$; then we extend it according to sp-cutoff criterion explained in Definition 4.7.4. $\mathcal{P}_N^1$ is shown in Figure 4.24 with a dotted line.

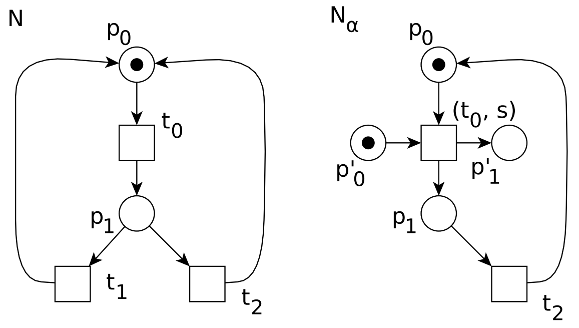

FIGURE 4.23

$\mathcal{P}_N^2$ extends $\mathcal{P}_N^1$ according to the sp-cutoff criterion. Thus, for each run we have to find an sp-cutoff event. Let us consider e_3 and check if it can be an sp-cutoff event.

Let $e = e_3$, e is an sp-cutoff event if there is $e' \in E$ such that, setting $D = [e] \setminus [e']$, we have $e' < e$ and (1) and (2) of Definition 4.7.4 hold. Assume $e' = e_1$, then $D = \{e\}$,

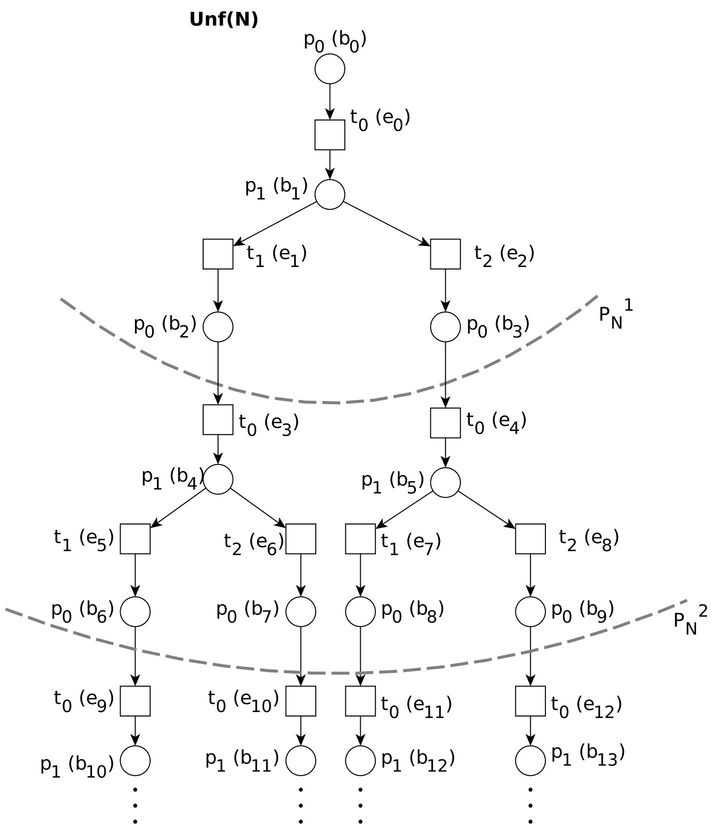

FIGURE 4.24

${}^\bullet D = \{b_2\}$ and $D^\bullet = \{b_4\}$. $\lambda({}^\bullet D \setminus D^\bullet = \lambda(b_2) = p_0$ and $\lambda(D^\bullet \setminus^\bullet D = \lambda(b_4) = p_1$. Since $\lambda(b_2) \neq \lambda(b_4)$, (1) is violated and so e' cannot be e_1. Of course this does not mean that e_3 is not an sp-cutoff event, unless there is no appropriate e' satisfying the requirements.

Let us try another event for e' . Let $e = e_3$ and assume $e' = e_0$, then $D = \{e_1, e_3\}$, ${}^\bullet D = \{b_1, b_2\}$ and $D^\bullet = \{b_2, b_4\}$. $\lambda({}^\bullet D \setminus D^\bullet = \lambda(b_1) = p_1$ and $\lambda(D^\bullet \setminus^\bullet D = \lambda(b_4) = p_1$. (1) is satisfied. However (2) is violated since $B_1 \cap^\bullet D \neq \emptyset$. So also e_0 is not suitable for e'. This shows us that e_3 cannot be an sp-cutoff event since there is no e' in its past to satisfy the requirements of sp-cutoff criterion.

Let $e = e_9$ and assume $e' = e_3$, then $D = \{e_5, e_9\}$, ${}^\bullet D = \{b_4, b_6\}$ and $D^\bullet = \{b_6, b_10\}$. $\lambda({}^\bullet D \setminus D^\bullet = \lambda(b_4) = p_1$ and $\lambda(D^\bullet \setminus^\bullet D = \lambda(b_10) = p_1$. (1) is satisfied. (2) is also satisfied since $B_1 \cap {}^\bullet D \neq \emptyset$. So also e_3 is suitable for e' for e_9 and e_9 is an sp-cutoff event. Note that cutoff events are not included in the prefix.

By computing all sp-cutoff events as explained above, we construct $\mathcal{P}_N^2$ which is illustrated in Figure 4.24, separated with a dotted line. All reveals relations in N can be computed on this prefix of the unfolding.

Now let us check if $t_0 \ \rhd_{tr} \ t_1$ by translating it to the diagnosis problem. For this, we set $\alpha = (\{s_0\}, <, \ell)$ where $\ell(s_0) = t_0$ and let t_1 the fault ϕ. All the conditions of Theorem 4.7.1 are satisfied if $t_0 \ \not\rhd_{tr} \ t_1$. We first translate α to an occurrence net O_α which in this case consists of only one event corresponding to t_0. Then we compose N and O_α into a net N_α which is illustrated in Figure 4.23. In this way, observable transitions of N and O_α are synchronized to ensure that no run contradicts α or adds further observable events, and faults are excluded.

Consider $\mathrm{Unf}(N_\alpha)$. If each event is mapped to a transition t instead of a tuple (t, s), each run ω of $\mathrm{Unf}(N_\alpha)$ is also a run of $\mathrm{Unf}(N)$. In addition, ω explains α iff $mark(\omega)$ contains P_{max}. By Definition 4.7.3, we construct $\mathrm{Unf}(N_\alpha)$ by cutting it at any event e such that there is another event $e' < e$ with $mark([e']) = mark([e])$. This yields a finite prefix $\mathcal{P}_\alpha$ which is illustrated in Figure 4.25, separated with a dotted line. Note that the corresponding events are labeled with same symbols as in $\mathcal{P}^1_N \sqsubseteq \mathcal{P}^2_N$ so that $\mathcal{P}_\alpha$ can be seen as a subnet of $\mathcal{P}^1_N \sqsubseteq \mathcal{P}^2_N$ with two additional places, namely $p'_0(b'_0)$ and $p'_1(b'_1)$.

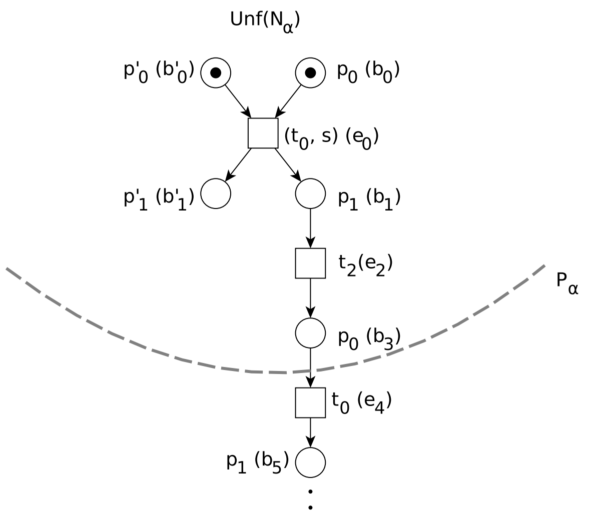

FIGURE 4.25

By Theorem 4.7.1, α does not diagnose t_1 (so that by Theorem 4.7.2 $t_0 \ \not\rhd_{tr} \ t_1$) iff there exist runs $\omega, \omega'_1 \in R$, $\omega_1 \in R(\mathcal{P}^1_N)$, $\omega_2 \in R(\mathcal{P}^2_N)$ satisfying the six conditions in Theorem 4.7.1. Let $\omega = \{e_0\}$, $\omega'_1 = \{e_0, e_2\}$. ω is a succinct explanation of α, $\omega \subseteq \omega'_1$ and ω'_1 is fault free i.e., it does not include any instance of t_1. We have to find ω_1 such that $mark(\omega_1) = mark(\omega'_1)$. Let $\omega_1 = \{e_0, e_2\}$ then $mark(\omega_1) = mark(\omega'_1) = b_3$. Now we have to find fault free run $\omega_2 \in R(\mathcal{P}^2_N)$ which yields the same marking with ω_1 and ω'_1. Let $\omega_2 = \{e_0, e_2, e_4, e_8\}$, ω_2 is fault free and $mark(\omega_2) = b_3$. So far all conditions except number 5 are satisfied. The remaining condition to be checked is if

$\forall e \in E \ : \ \omega_1$ enables $e \implies \text{spoilers}(e) \cap \omega_2 \neq \emptyset$. There is only one event which is enabled by ω_1 that is e_4. e_4 is included in ω_2 so $\text{spoilers}(e_4) \cap \omega_2 \neq \emptyset$. Consequently, all six conditions of Theorem 4.7.1 are satisfied which means t_0 does not diagnose t_1, i.e., $t_0 \ \not\triangleright_{tr} \ t_1$. □

We have shown that reveals problem can be translated to diagnosis problem in Theorem 4.7.2 and how the algorithm is adapted on an example. The variants of reveals relation such as extended-reveals and repeated-reveals can also be translated in a similar way to the diagnosis problem in order to use the diagnosis algorithm for checking non-interference notions that are introduced in this thesis.

Another relation we use in the new introduced non-interference notions is excludes relation, $\underline{\text{ex}}$. Let N be a 1-safe Petri net, $t_1, t_2 \in T$, $t_1 \ \underline{\text{ex}} \ t_2$ means that the two transitions can never appear in the same run together whereas $t_1 \ \triangleright_{tr} \ t_2$ means that if t_1 occurs, t_2 has already occurred or will inevitably occur in the future. Thus, proving that $\neg(t_1 \ \underline{\text{ex}} \ t_2)$ requires to find a run in which t_1 and t_2 occurs together whereas proving $t_1 \ \not\triangleright_{tr} \ t_2$ requires to find a run in which t_1 occurs but t_2 does not. Setting $\alpha = (\{s_0\}, <, \ell)$ where $\ell(s_0) = t_1$ and $\phi = t_2$, Theorem 4.7.1 looks for such run for $t_1 \ \not\triangleright_{tr} \ t_2$ on a finite unfolding of N.

Here we propose a method for adapting Theorem 4.7.1 to the problem of checking $\neg(t_1 \ \underline{\text{ex}} \ t_2)$. As in the reveals problem of two transitions, we assume all transitions except t_1 are unobservable. Observation pattern is $\alpha = (\{s_0\}, <, \ell)$ where $\ell(s_0) = t_1$. We are interested that ω being a succinct explanation of α, either its extension $\omega_1' \in R$ or $\omega_2 \in R(\mathcal{P}_N^2)$ includes an instance of t_2.

The corollary below follows from Theorem 4.7.1 and the fact explained above.

Corollary 4.7.3. *Let N be a 1-safe Petri net and $t_1, t_2 \in T$. Let $\alpha = (\{s_0\}, <, \ell)$ where $\ell(s_0) = t_1$. $\neg(t_1 \ \underline{ex} \ t_2)$ iff there exist runs $\omega, \omega_1' \in R$, $\omega_1 \in R(\mathcal{P}_N^1)$, $\omega_2 \in R(\mathcal{P}_N^2)$ satisfying that*

1. *ω is a succinct explanation of α*
2. *$\omega \subseteq \omega_1'$*
3. *$\omega_1 \subseteq \omega_2$*
4. *$mark(\omega_1') = mark(\omega_1) = mark(\omega_2)$*
5. *$\forall e \in E \ : \ \omega_1$ enables $e \implies \text{spoilers}(e) \cap \omega_2 \neq \emptyset$*
6. *either ω_1' or ω_2 contains an occurrence of t_2.*

Example 4.7.4. In Example 4.7.3, we have shown that $t_0 \not\triangleright_{tr} t_1$. In order to show that $\neg(t_0 \underline{\text{ex}}\, t_1)$ we need to follow the same procedure but this time we have to check the conditions of Corollary 4.7.3. In the net N in Example 4.7.3, $\neg(t_0 \underline{\text{ex}}\, t_1)$ and all the conditions of Corollary 4.7.3 are satisfied for $\omega = \{e_0\}$, $\omega_1' = \{e_0, e_1\}$, $\omega_1 = \{e_0, e_1\}$ and $\omega_2 = \{e_0, e_1, e_3, e_5\}$. □

There are two variants of excludes relation: future excludes and past excludes which are introduced in Section 4.4.2. These two relations can also be adapted to the diagnosis problem in a similar way.

4.7.3.3 Checking non-interference notions based on reveals and excludes

We have shown how to check reveals and excludes relations on finite prefixes of 1-safe Petri nets by using the diagnosis approach. Now we continue to apply the solution for checking the non-interference notions introduced in Section 4.5.

In Section 4.5, several non-interference notions are introduced on the basis of reveals and excludes relations and their variants. Let us consider the first notion introduced in Section 4.5 which is Reveals-based Non-Interference, *RNI*. Deciding if a given net is secure with respect to *RNI* is based on checking if any low transition reveals a high transition in the given net. By Theorem 4.7.2, *RNI* security problem can be translated as:

Let $N = (P, T, F, m_0)$ be 1-safe Petri net where $T = H \cup L$ and $H \cap L \neq \emptyset$. N is secure with respect to *RNI* iff $\forall l \in L$, $\forall h \in H$, setting $\alpha = (\{s_0\}, <, \ell)$ where $\ell(s_0) = l$, α does not diagnose h.

As discussed before, given a 1-safe Petri net, the diagnosis, and so the reveals problem, is solvable on a finite prefix of the unfolding of the given net. In order to check if the net is secure, we have to compute $\mathcal{P}_N^1$ and its extension $\mathcal{P}_N^2$ just once; this computation depends only on the given net. However, $\mathcal{P}_\alpha$ must be computed for each $l \in L$ since $\mathcal{P}_\alpha$ depends on the observation α. For each reveals relation $l \triangleright_{tr} h$ to be checked, α corresponds to the occurrence of that specific l. Thus, for each $l \in L$, a new $\mathcal{P}_\alpha$ must be computed. However, $\mathcal{P}_\alpha$ is expected to be small in size because it corresponds to the explanation of occurrence of one transition, namely l.

Considering Positive/Negative Non-Interference, *PNNI*, given a 1-safe Petri net $N = (P, T, F, m_0)$ where $T = H \cup L$ and $H \cap L \neq \emptyset$, the net is secure iff $\forall l \in L$, $\forall h \in H$, setting $\alpha = (\{s_0\}, <, \ell)$ where $\ell(s_0) = l$, both Theorem 4.7.1 and Corollary 4.7.3 apply for all α and h.

By adapting the diagnosis algorithm to the variants of reveals and excludes, all introduced non-interference notions can be checked for 1-safe Petri nets on finite prefixes of unfoldings.

4.8 Non-interference with Languages

In Chapter 4, we have introduced new non-interference notions based on reveals and excludes relations on Petri nets. Although Petri nets are useful for modeling distributed and concurrent systems, those notions can also be used for detecting information flow in other models and for designing secure systems. In this chapter, we generalize our non-interference notions with the underlying relations by carrying them to formal languages.

In [41, 52] Petri nets have been used in the study of formal languages and they are used to model the flow of information and control of actions in a system. Usually, each transition of a Petri net is associated with a symbol for naming the transition, i.e., labeled. The number of transitions in a Petri net is finite, hence a finite alphabet can be defined for labeling all the transitions. Let Σ be a finite alphabet, a labeling function λ maps transitions to symbols of the alphabet, i.e., $\lambda = T \rightarrow \Sigma$. A labeled marked Petri net defines a set of words over Σ, each word corresponding to a possible execution of the net. The set of all possible words corresponding to the possible interleaved executions of a marked labeled Petri net defines a Petri net language.

Since the notions introduced in Chapter 4 are mostly based on behavioral observations of Petri nets and closely associated to the executions of nets, it is very natural to consider carrying these notions to the Petri net languages. Moreover, we are not restricted to Petri net languages. It is also possible to lift the notions to general formal languages. Considering the fact that Petri net languages are only a restricted class of languages, carrying our non-interference and underlying reveals and excludes relations to the general class of formal languages might lead to a wider application area and give more freedom for improvement.

In the following sections we will redefine reveals and excludes relations, with their variants, on formal languages. Later, we will carry the non-interference notions that are introduced in Section 4.5 to formal languages.

Let $A = \{a_1, \ldots, a_n\}$ be a finite alphabet. A^* denotes the set of finite words over A. $A^\infty = A^* \cup A^\omega$, where $A^\omega = \{\phi : \mathbb{N} \rightarrow A\}$, is the set of infinite words over A. Consider a language $L \subseteq A^\infty$, the set of maximal words of L is $L_M = \{\alpha \in L \ : \ \forall \beta \in L \ \beta \neq \alpha\gamma, \gamma \in A^\omega\}$.

A word $\alpha \in L$ can also be seen as a multiset if the order of the letters is ignored. Hence, in the reveals/excludes based non-interference notions, except *I-PNNI* we are not interested in the order of the transitions, we can define our words as multisets. A *multiset* on A is a function $m : A \to \mathbb{N}$. For each $a \in A$, the *multiplicity* of a is the number of its occurrences and is denoted by $m(a)$. For example, let $A = \{a, b, c\}$ be a finite alphabet, and $L = \{a, ab, abc, abca\}$ be a language over A. Each word can be represented as a multiset whose underlying set is A. Similarly the word *abca* can be represented as $m(a) = 2,\ m(b) = 1,\ m(c) = 1$. An ordinary set can be seen as the special case where the multiplicity of each element of the underlying set is 1. In the following we consider the languages with infinite words, hence we define an *extended multiset* on N as a map $\mu : A \to \mathbb{N} \cup \{\infty\}$.

Given an alphabet A and a word $\alpha \in A^\infty$, we associate to α an extended multiset $\mu_\alpha : A \to \mathbb{N} \cup \{\infty\}$ where for each x in A, $\mu_\alpha(x)$ is the number of occurrences of x in α if it is finite, ∞ otherwise.

Let A be an alphabet, $\alpha \in A^\infty$ a word, $\mu_\alpha : A \to \mathbb{N} \cup \{\infty\}$ its associated extended multiset, $x \in A$ and $W \subseteq A$. Then,

$$x \in \alpha \text{ iff } \mu_\alpha(x) > 0 \ \wedge \ \mu_\alpha(x) = \infty.$$

$$W \subseteq \alpha \text{ iff } \forall x \in W : \ \ x \in \alpha.$$

$$W \cap \alpha = \{x \in W : \ x \in \alpha\}.$$

4.8.1 Reveals, extended-reveals and repeated reveals relations on languages

Reveals and extended reveals relations were formerly defined for events of occurrence nets in [1, 39]. In Section 4.3 (see also [9]) we have redefined them for transitions of Petri nets. Here we redefine them between letters in a language.

Intuitively, a letter reveals another letter in a language if whenever the first letter appears in a maximal word, the second also appears in that word. Hence, it is not possible to see the first one in a maximal word without the second one.

Definition 4.8.1. Let $a, b \in A$. We say a *reveals* b, denoted $a \rhd_l b$, in language L, iff $\forall \alpha \in L_M : \ \ a \in \alpha \implies b \in \alpha$.

In some cases, a set of letters together give information about another letter or again a set of letters, whereas appearance of only one of them may not give any information.

Definition 4.8.2. Let $W, Z \subseteq A$. We say W *extended reveals* Z, denoted $W \twoheadrightarrow_l Z$, iff $\forall \alpha \in L_M : \ W \subseteq \alpha \implies Z \cap \alpha \neq \emptyset$. Intuitively, W extended-reveals Z in language L, if and only if each maximal word in L which includes all letters in W also includes at least one letter in Z.

For example, let $a, b, c, d \in A$. $\{a, b\} \twoheadrightarrow_l \{c, d\}$ in L means that if a maximal word includes both a and b in it, it must also include either c or d. In Section 4.3, we also introduce a new relation called repeated reveals between transitions of a Petri net. This new relation focuses on the information about a transition which is given by the number of occurrences of another transition. Here we redefine it for the letters in a language. Intuitively, when a letter n-repeated reveals another letter, it means that n occurrences of the first letter in a maximal word imply that the second letter must also appear in that maximal word.

Definition 4.8.3. Let $a, b \in A$ and $n > 0$. Let $R_a^n = \{w \in L$ such that w includes exactly $n\ a's\}$ and Ω_a^n denote the set of maximal words in R_a^n with respect to set inclusion (i.e., $\Omega_a^n \subseteq R_a^n$ such that if $u, v \in \Omega_a^n \ \wedge \ u$is not a proper prefix of$v$ then $u = v$).

If $\Omega_a^n \neq \emptyset$ then a *n-repeated reveals* b, denoted $a \ \rhd_l^n b$, iff $\forall w \in \Omega_a^n \ b \in w$.

If $\Omega_a^n = \emptyset$ then $a \ \rhd_l^n b$ is not defined.

Notation. $a \ \not\rhd_l^n \ b$ will denote that there is at least one word in Ω_a^n such that a appears n times and b does not appear. $\neg(t_1 \ \rhd_l^n \ t_2)$ will denote that either $a \ \rhd_l^n b$ is not defined, or $a \ \not\rhd_l^n \ b$.

Example 4.8.1. Let $A = \{a, b, c\}$ and L be such that $L_M = \{a^\omega, b^\omega, (abc)^\omega\}$. In this language, it is easy to see that $a \ \not\rhd_l \ b$ since there is a maximal word in which only a appears but b never appears. Similarly, $b \ \not\rhd_l \ a$ because there is the maximal word consisting of only b's. However, appearance of a and b together implies the appearance of c, so $\{a, b\} \twoheadrightarrow_l \{c\}$. On the other hand, $c \ \rhd_l \ a$ and $c \ \rhd_l \ b$ since c never appears without an a and a b. In this language repeated appearance of none of the letters gives any additional information. □

Example 4.8.2. Let $A = \{a, b, c, d\}$ and let the set of all maximal words of language L be $L_M = \{a^\omega, c^\omega, ad, abcc, babd, bbac, bbbd, bbbbd, bababd, abcd^\omega\}$. The language consists of both finite and infinite words. Let us consider the relations between the letters. It is easy to see that neither a nor c reveals any other letter since there are two infinite words in L_M consisting respectively of only a's and only c's. Let us consider b and d. Appearance of b once or twice does not give any information about d whereas b cannot appear in a maximal word more than two times without a d, i.e., $b \ \rhd_l^3 d$. We can also see that neither a nor b reveals c or d whereas $\{a, b\} \twoheadrightarrow_l \{c, d\}$. There is no maximal word in which a and b appear together without any c or d. □

4.8.2 Excludes relation on languages

In Section 4.4 we introduce excludes relation on Petri nets. The intuition behind introducing this relation is to express negative information flow between two transitions. If two transitions exclude each other, then they can never appear in the same maximal run. Thus, by observing an occurrence of one of them, an attacker can deduce that the other transition did not occur and will not occur.

Here we redefine this relation for the letters in a language.

Definition 4.8.4. Let $a, b \in A$. We say a *excludes* b, denoted $a\ \underline{ex}_l\ b$, in language L, iff $\forall \alpha \in L_M : \quad a \in \alpha \implies b \notin \alpha$, i.e., they never appear together in the same maximal word.

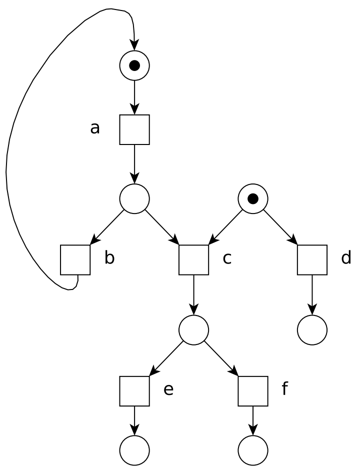

FIGURE 4.26

Example 4.8.3. Let us consider the Petri net in Figure 4.26. Let $A = \{a, b, c, d, e, f\}$ be the alphabet which consists of labels of transitions. Let L be the language generated by the Petri net. The set of maximal words of this language is $L_M = \{(ab)^*d(ab)^\omega$, $a(ba)^*ce$, $a(ba)^*cf$, $a(ba)^*d(ba)^\omega$. In this Petri net, there are some transitions that can never occur in the same maximal run together, indeed in the language formed by this Petri net we see that the corresponding letters never appear in the same maximal word together. For example, there is no maximal word in L_M that includes both c and d, i.e., $c\ \underline{ex}_l\ d$. Similarly, $e\ \underline{ex}_l\ f$, $e\ \underline{ex}_l\ d$ and $f\ \underline{ex}_l\ d$. □

4.8.3 Non-interference notions based on reveals and excludes relations with languages

Let $A = S \cup V$ be a finite alphabet such that $S \cap V = \emptyset$ and $S, V \neq \emptyset$ where S denotes *silent/hidden letters* and V denotes *visible/observable letters*. The language L and the set of maximal words of L, L_M, are defined as above. In the sequel, we will define non-interference notions over an alphabet A and a language L, analogous to those in Section 4.5. With these notions we aim to tell if an attacker, who knows how the language is generated (knows the rules/structure of the language), is able to deduce information about silent letters in maximal words by looking at the visible ones.

The first non-interference notion is based on the reveals relation. It aims to catch positive information flow.

Definition 4.8.5. Language L is *Reveals-based Non-Interference (RNI)* secure iff $\forall v \in V,\ \forall s \in S: \quad v \not\triangleright_l s$.

In other words, L is secure with respect to *RNI* if it is not possible to deduce information about silent letters in a maximal word by looking at the visible ones.

The second non-interference notion for catching positive information flow (called *k-ERNI)*) considers combinations of visible letters and checks if they together give information about some silent letters.

Definition 4.8.6. Let $|V| \geqslant k \geqslant 1$. The language L is *k-Extended-Reveals based Non-Interference (k-ERNI)* secure iff $\forall s \in S$, $\forall V' \subseteq V$ such that $|V'| \leqslant k \ \wedge \ \exists \alpha \in L_M$: $\bigwedge_{v \in V'} (\alpha \cap \{v\} \neq \emptyset)$, it holds $V' \not\twoheadrightarrow_l \{s\}$.

Another non-interference notion which aims to catch positive information flow is *n-ReRNI*. This parametric notion considers the number of appearances of a visible letter and checks if the repeated appearance of a visible letter reveals a silent letter.

Definition 4.8.7. Let $n > 0$. Language L is *n-Repeated-Reveals based Non-Interference (n-ReRNI)* secure iff $\forall v \in V,\ \forall s \in S,\ \forall m \geqslant n$, if $\Omega_v^m \neq \emptyset$, then $\neg(v \ \triangleright_l^m \ s)$.

L is *ReRNI*, if it is *n-ReRNI* for all $n > 0$.

The next non-interference notion not only catches positive information flow but it also catches negative information flow. It considers both reveals relation and excludes relation between visible letters and silent letters.

Definition 4.8.8. Language L is *Positive/Negative Non-Interference (PNNI)* secure iff $\forall v \in V,\ \forall s \in S: \ v \not\triangleright_l s \ \wedge \ \neg(v \ \underline{ex}_l \ s)$.

Example 4.8.4. Let us consider N_{13} in Figure 4.16 on page 72. Let the alphabet be T, the net generates a language, L, whose set of maximal words is $L_M = \{l_1h_1, l_1h_2, l_2l_3\}$. Let $T = S \cup V$ be such that $S \cap V = \emptyset$ and $S, V \neq \emptyset$ where S denotes silent letters and V denotes visible ones.

We see that $l_1 \not\rhd_l h_1$ and $l_1 \not\rhd_l h_2$; similarly, neither l_2 nor l_3 reveals any silent letter. Moreover also the combination of visible letters does not give information about silent letters: the only combination we observe is $\{l_2, l_3\}$ and $\{l_2, l_3\} \not\rightarrowtriangle_l \{h_1\}$, $\{l_2, l_3\} \not\rightarrowtriangle_l \{h_2\}$. We do not have any repeated appearances of visible letters hence we cannot see any repeated reveals relation between visible and silent letters. Consequently language L which is generated by N_{13} is secure with respect to *RNI*, *ERNI* and *ReRNI*. Indeed the net itself is secure with respect to these notions as examined in Example 4.5.4.

However L is not secure with respect to *PNNI* since there is a negative information flow, i.e., there exist visible letters which never appear together with any silent letter. For example $l_2 \ \underline{ex}_l \ h_1$ and $l_2 \ \underline{ex}_l \ h_2$. Thus, the attacker can tell that a maximal word does not include any h_1 or h_2 if l_2 appears. Similarly l_3 excludes both h_1 and h_2. Thus L is not a secure language with respect to *PNNI*. Indeed, in Example 4.5.4, it is explained that N_{13} is not secure with respect to *PNNI*. $\square$

Chapter 5

Conclusions and Future Work

This thesis provides a formal framework for liveness and security of distributed systems. We propose new approaches for defining secure and serviceable systems, and discuss associated model checking methods. We examine distributed systems in which there are observable and unobservable actions that can be performed. An entity of such a distributed system can be a service provider or a user. A service provider is responsible for providing a reliable service to the users and protection of sensitive information. In general, maximizing the security results in minimizing the serviceability and vice versa. We study the two properties separately yet in a common formal framework. In the proposed framework the two notions can be put together in order to achieve a compromise.

For the sake of serviceability, we examine distributed systems in which a user can behave as an attacker and can try to break down the system. The thesis provides a novel notion of liveness called *observable liveness* which guarantees that the service provider will continue to give the requested services to the users. In the observable liveness setting, we give the control of some observable actions to the user. Intuitively, a distributed system is observably live if, whatever state is reached, the user can force the system to get the requested service by using the controllable actions. Hence, if a system satisfies observable liveness, it means that, if the model is in accordance with the real system, even if an attacker tries to break down the system, the users will continue to get the requested service. We formalize this notion with 1-safe Petri nets and examine its properties. We also compare it with the classical liveness in Petri nets theory. In general, observable liveness does not imply the classical liveness as well as the classical liveness does not imply observable liveness. However, if the net model behaves deterministically in its uncontrollable part, we prove that liveness implies observable liveness. Some results on observable liveness are published in [25, 26].

Observable liveness of an observable transition t requires that at each reachable marking, a user must be able to force the system to run in a way that eventually t fires. However, in some cases this requirement can be considered too strong. For example, assume that a user decides at the beginning which observable transition he/she targets to fire eventually. If there is a strategy at the initial marking and the user follows it from the beginning, the system will never reach a state which is not in accordance with the strategy. Thus, instead of considering each reachable marking, we can consider the ones that are reached by the transition sequences which respect the strategy. For these cases, we define *weak observable liveness*. If an observable transition t is weakly observably live, it means that a user has a strategy at the initial marking to force the system to fire t, i.e., if the user respects the strategy starting from the initial marking, t will eventually fire.

For checking weak observable liveness, we discuss a possible application of Streett games which are two-player infinite games on finite graphs. We translate the problem of checking whether an observable transition t is weakly observably live to a Streett game between the user and the system. If the user has a winning strategy it means that t is weakly observably live. However, the proposed approach deals with a restricted class of 1-safe Petri nets in which, at each reachable marking, enabled transitions are either all controllable or all uncontrollable. Currently we are working on exploring games and their relations with Petri nets and finding a game theoretic method for checking weak observable liveness of general 1-safe Petri nets. We also consider to translate observable liveness and weak observable liveness as LTL properties in order to explore suitable methods based on LTL model checking.

For the security part, we consider information flow and non-interference. We provide several new notions of non-interference for Petri nets, and compare them with notions already proposed in the literature. In the considered setting, the transitions of a Petri net are partitioned into two disjoint sets: the low (observable) and the high (unobservable/hidden) transitions. The attacker knows the structure of the system and tries to deduce information about the high transitions by observing the low transitions.

A Petri net is considered *secure*, or *free from interference*, if, from the observation of the occurrence of a low transition, or a set of low transitions, it is not possible to infer information on the occurrence of a high transition. Our new non-interference notions rely on net unfolding and on two new relations among transitions with their variants. The first relation is called *reveals* and is an adaptation to Petri nets from occurrence nets. The original reveals notion has been used in fault diagnosis, but had not been considered in the field of non-interference. In this thesis, we introduce reveals relation for transitions of Petri nets and two parametrized versions of it, *extended-reveals* and *repeated reveals*, which are then used for catching positive information flow, i.e., the possibility for an attacker to deduce the occurrence of a high transition by observing the low transitions. The second relation is called *excludes* and it is introduced here with two variants, *past excludes* and *future excludes*, for the aim of capturing negative information flow, i.e., the possibility of an attacker to deduce the non-occurrence of a high transition by observing the low transitions.

On the basis of the reveals and the excludes relations with their variants, we introduce in this thesis several non-interference notions for Petri nets. These new notions deal with information flow about both past occurrences and inevitable future occurrences of high transitions. The notion of *RNI* states that a net is secure if no low transition reveals any high transition. We show that this notion captures some situations which are not captured by the existing notions, e.g., a low transition reveals inevitable future occurrence of a high transition. We also propose more restrictive notions: *k-ERNI* and *n-ReRNI.* In some cases occurrences of different low transitions together can give information about the occurrence of a high transition. With *k-ERNI*, which uses the extended-reveals relation, we are able to catch this kind of information flow. Another case can be that repeated occurrences of a low transition can give information about a high transition. *n-ReRNI* deals with this kind of information flow on the basis of the repeated reveals relation between high and low transitions.

By adding the *excludes* relation to the picture, we are able to capture also negative information flow. By considering reveals and excludes together we define *PNNI* and *I-PNNI* notions that are able to catch both positive and negative information flow about both past and future occurrences of high transitions. The thesis includes a comparison between the notions introduced here and those found in the literature on the subject.

Some results on reveals and excludes based non-interference have been published in [9]. The notions proposed in this thesis, and further variants of them, should now be tested on more realistic cases. We have now a collection of different non-interference properties, so that a system designer, or a system analyzer, can choose those more appropriate to a specific case. A generalization could be a non-interference notion based on a parametric reveals relation between multisets of transitions. The thesis includes a generalization of the reveals and the excludes relations with the non-interference notions on the basis of them to formal languages.

We have discussed two methods for checking non-interference. The first method is based on translating the underlying relations, reveals and excludes, to LTL and then expressing the reveals and excludes based non-interference notions as LTL properties. With this proposed method, we are able to use LTL model checking methods for checking non-interference. The second method we provide is based on computing the reveals and the excludes relations on finite prefixes of unfoldings. The method adapts the diagnosis algorithm given in [40] to the problem of checking the reveals and the excludes relations between transitions of 1-safe Petri nets. With this approach, non-interference notions can be checked on finite prefixes by checking the reveals and the excludes relations between high and low transitions. We are interested in considering other methods such as game theoretic methods that are explored in [60]. We also consider a different class of games, namely infinite games on finite graphs, which we also explore for checking observable liveness.

We plan to explore the possible application of the new transition based reveals relation in fault diagnosis. Moreover, we are interested in further investigating the excludes relation and the possibility to apply it in different contexts, e.g., systems biology.

Checking a system for some properties is helpful to make sure that some properties are satisfied by the modeled system. An alternative approach can be to design the system from the beginning in a way that the system will satisfy those properties. We consider to use the proposed notions and methods for designing secure and serviceable systems from scratch. Compositional approach is one way to design a distributed system [10, 11]. We aim to develop a compositional method for designing a secure and serviceable system that will preserve observable liveness and non-interference properties. In [7, 8], we provide a case study for modeling a cryptographic protocol by using compositional approach, and we plan to investigate this approach for preserving observable liveness and non-interference in the future.

Bibliography

[1] Sandie Balaguer, Thomas Chatain, and Stefan Haar. Building tight occurrence nets from reveals relations. In Benoît Caillaud, Josep Carmona, and Kunihiko Hiraishi, editors, *11th International Conference on Application of Concurrency to System Design, ACSD 2011, Newcastle Upon Tyne, UK, 20-24 June, 2011*, pages 44–53. IEEE, 2011.

[2] Sandie Balaguer, Thomas Chatain, and Stefan Haar. Building occurrence nets from reveals relations. *Fundam. Inform.*, 123(3):245–272, 2013.

[3] Paolo Baldan and Alberto Carraro. Non-interference by unfolding. In Gianfranco Ciardo and Ekkart Kindler, editors, *Application and Theory of Petri Nets and Concurrency - 35th International Conference, PETRI NETS 2014, Tunis, Tunisia, June 23-27, 2014. Proceedings*, volume 8489 of *Lecture Notes in Computer Science*, pages 190–209. Springer, 2014.

[4] Paolo Baldan and Alberto Carraro. A causal view on non-interference. *Fundam. Inform.*, 140(1):1–38, 2015.

[5] J. Barwise and J. Seligman. *Information Flow: The Logic of Distributed Systems.* Cambridge Tracts in Theoretical Computer Science. Cambridge University Press, 1997.

[6] Albert Benveniste, Eric Fabre, Stefan Haar, and Claude Jard. Diagnosis of asynchronous discrete-event systems: a net unfolding approach. *IEEE Trans. Automat. Contr.*, 48(5):714–727, 2003.

[7] Luca Bernardinello, Görkem Kılınç, Elisabetta Mangioni, and Lucia Pomello. Modeling distributed private key generation by composing Petri nets. In Daniel Moldt, editor, *Joint Proceedings of the International Workshop on Petri Nets and Software Engineering (PNSE'13) and the International Workshop on Modeling and Business Environments (ModBE'13), Milano, Italy, June 24 - 25, 2013*, volume 989 of *CEUR Workshop Proceedings*, pages 77–96. CEUR-WS.org, 2013.

[8] Luca Bernardinello, Görkem Kılınç, Elisabetta Mangioni, and Lucia Pomello. Modeling distributed private key generation by composing Petri nets. *T. Petri Nets and Other Models of Concurrency*, 9:19–40, 2014.

[9] Luca Bernardinello, Görkem Kılınç, and Lucia Pomello. Non-interference notions based on reveals and excludes relations for Petri nets. In Daniel Moldt, Heiko Rölke, and Harald Störrle, editors, *Proceedings of the International Workshop on Petri Nets and Software Engineering (PNSE'15), including the International Workshop on Petri Nets for Adaptive Discrete Event Control Systems (ADECS 2015) A satellite event of the conferences: 36th International Conference on Application and Theory of Petri Nets and Concurrency Petri Nets 2015 and 15th International Conference on Application of Concurrency to System Design ACSD 2015, Brussels, Belgium, June 22-23, 2015.*, volume 1372 of *CEUR Workshop Proceedings*, pages 59–78. CEUR-WS.org, 2015.

[10] Luca Bernardinello, Elisabetta Mangioni, and Lucia Pomello. Local state refinement and composition of elementary net systems: An approach based on morphisms. *T. Petri Nets and Other Models of Concurrency*, 8:48–70, 2013.

[11] Luca Bernardinello, Elena Monticelli, and Lucia Pomello. On preserving structural and behavioural properties by composing net systems on interfaces. *Fundam. Inform.*, 80(1-3):31–47, 2007.

[12] Gérard Berthelot. Transformations and decompositions of nets. In Wilfried Brauer, Wolfgang Reisig, and Grzegorz Rozenberg, editors, *Advances in Petri Nets*, volume 254 of *Lecture Notes in Computer Science*, pages 359–376. Springer, 1986.

[13] Eike Best and Philippe Darondeau. Deciding selective declassification of Petri nets. In *Proceedings of the First International Conference on Principles of Security and Trust*, POST'12, pages 290–308, Berlin, Heidelberg, 2012. Springer-Verlag.

[14] Eike Best, Philippe Darondeau, and Roberto Gorrieri. On the decidability of non interference over unbounded Petri nets. In Konstantinos Chatzikokolakis and Véronique Cortier, editors, *Proceedings 8th International Workshop on Security Issues in Concurrency, SecCo 2010, Paris, France, 30th August 2010.*, volume 51 of *EPTCS*, pages 16–33, 2010.

[15] Jeremy Bryans, Maciej Koutny, Laurent Mazaré, and Peter Y. A. Ryan. Opacity generalised to transition systems. *Int. J. Inf. Sec.*, 7(6):421–435, 2008.

[16] Jeremy Bryans, Maciej Koutny, and Peter Y. A. Ryan. Modelling opacity using Petri nets. *Electr. Notes Theor. Comput. Sci.*, 121:101–115, 2005.

[17] J. Richard Büchi and Lawrence H. Landweber. Solving sequential conditions by finite-state strategies. *Transactions of the American Mathematical Society*, 138:pp. 295–311, 1969.

[18] Nils Buhrke, Helmut Lescow, and Jens Vöge. Strategy construction in infinite games with Streett and Rabin chain winning conditions. In *In TACAS 96, volume 1055 of Lect. Notes in Comp. Sci*, pages 207–225. Springer, 1996.

[19] Nadia Busi and Roberto Gorrieri. A survey on non-interference with Petri nets. In Jörg Desel, Wolfgang Reisig, and Grzegorz Rozenberg, editors, *Lectures on Concurrency and Petri Nets*, volume 3098 of *Lecture Notes in Computer Science*, pages 328–344. Springer, 2003.

[20] Nadia Busi and Roberto Gorrieri. Positive non-interference in elementary and trace nets. In Jordi Cortadella and Wolfgang Reisig, editors, *Applications and Theory of Petri Nets 2004, 25th International Conference, ICATPN 2004, Bologna, Italy, June 21-25, 2004, Proceedings*, volume 3099 of *Lecture Notes in Computer Science*, pages 1–16. Springer, 2004.

[21] Nadia Busi and Roberto Gorrieri. Structural non-interference in elementary and trace nets. *Mathematical Structures in Computer Science*, 19(6):1065–1090, 2009.

[22] Christos G. Cassandras and Stephane Lafortune. *Introduction to Discrete Event Systems*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006.

[23] Jörg Desel and Javier Esparza. *Free Choice Petri Nets*. Cambridge tracts in theoretical computer science. Cambridge University Press, 1995.

[24] Jörg Desel, Hans-Michael Hanisch, Gabriel Juhás, Robert Lorenz, and Christian Neumair. A guide to modelling and control with modules of signal nets. In Hartmut Ehrig, Werner Damm, Jörg Desel, Martin Große-Rhode, Wolfgang Reif, Eckehard Schnieder, and Engelbert Westkämper, editors, *SoftSpez Final Report*, volume 3147 of *Lecture Notes in Computer Science*, pages 270–300. Springer, 2004.

[25] Jörg Desel and Görkem Kılınç. Observable liveness. In Daniel Moldt and Heiko Rölke, editors, *International Workshop on Petri Nets and Software Engineering (PNSE'14)*, number 1160 in CEUR Workshop Proceedings, pages 143–163, Aachen, 2014. CEUR-WS.org. http://ceur-ws.org/Vol-1160/.

[26] Jörg Desel and Görkem Kılınç. Observable liveness of Petri nets. *Acta Inf.*, 52(2-3):153–174, 2015.

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

[28] Javier Esparza and Keijo Heljanko. A new unfolding approach to LTL model checking. In Ugo Montanari, José D. P. Rolim, and Emo Welzl, editors, *Automata, Languages and Programming*, volume 1853 of *Lecture Notes in Computer Science*, pages 475–486. Springer Berlin Heidelberg, 2000.

[29] Javier Esparza and Keijo Heljanko. *Unfoldings - A Partial-Order Approach to Model Checking*. Monographs in Theoretical Computer Science. An EATCS Series. Springer, 2008.

[30] Javier Esparza and Christian Kern. Reactive and proactive diagnosis of distributed systems using net unfoldings. In Jens Brandt and Keijo Heljanko, editors, *12th International Conference on Application of Concurrency to System Design, ACSD 2012, Hamburg, Germany, June 27-29, 2012*, pages 154–163. IEEE Computer Society, 2012.

[31] Javier Esparza, Stefan Römer, and Walter Vogler. An improvement of McMillan's unfolding algorithm. *Formal Methods in System Design*, 20(3):285–310, 2002.

[32] Riccardo Focardi and Roberto Gorrieri. A classification of security properties for process algebras. *Journal of Computer Security*, 3(1):5–34, 1995.

[33] Riccardo Focardi and Roberto Gorrieri. Classification of security properties (Part I: Information flow). In *FOSAD* [34], pages 331–396.

[34] Riccardo Focardi and Roberto Gorrieri, editors. *Foundations of Security Analysis and Design, Tutorial Lectures [revised versions of lectures given during the IFIP WG 1.7 International School on Foundations of Security Analysis and Design, FOSAD 2000, Bertinoro, Italy, September 2000]*, volume 2171 of *Lecture Notes in Computer Science*. Springer, 2001.

[35] Rob Gerth, Doron Peled, Moshe Y. Vardi, and Pierre Wolper. Simple on-the-fly automatic verification of linear temporal logic. In Piotr Dembinski and Marek Sredniawa, editors, *Protocol Specification, Testing and Verification XV, Proceedings of the Fifteenth IFIP WG6.1 International Symposium on Protocol Specification, Testing and Verification, Warsaw, Poland, June 1995*, volume 38 of *IFIP Conference Proceedings*, pages 3–18. Chapman & Hall, 1995.

[36] Joseph A. Goguen and José Meseguer. Security policies and security models. In *IEEE Symposium on Security and Privacy*, pages 11–20, 1982.

[37] Roberto Gorrieri and Matteo Vernali. On intransitive non-interference in some models of concurrency. *Foundations of Security Analysis and Design VI*, page 125.

[38] Erich Grädel, Wolfgang Thomas, and Thomas Wilke, editors. *Automata Logics, and Infinite Games: A Guide to Current Research.* Springer-Verlag New York, Inc., New York, NY, USA, 2002.

[39] Stefan Haar. Unfold and cover: Qualitative diagnosability for Petri nets. In *Proc. 46th IEEE Conference on Decision and Control*, 2007.

[40] Stefan Haar, César Rodríguez, and Stefan Schwoon. Reveal your faults: It's only fair! In Josep Carmona, Mihai T. Lazarescu, and Marta Pietkiewicz-Koutny, editors, *13th International Conference on Application of Concurrency to System Design, ACSD 2013, Barcelona, Spain, 8-10 July, 2013*, pages 120–129. IEEE, 2013.

[41] M. Hack. Petri net languages. Technical report, Cambridge, MA, USA, 1976.

[42] Serge Haddad. A reduction theory for coloured nets. In Grzegorz Rozenberg, editor, *European Workshop on Applications and Theory in Petri Nets*, volume 424 of *Lecture Notes in Computer Science*, pages 209–235. Springer, 1988.

[43] Lawrence E. Holloway, Bruce H. Krogh, and Alessandro Giua. A survey of Petri net methods for controlled discrete event systems. *Discrete Event Dynamic Systems*, 7(2):151–190, 1997.

[44] Florian Horn. Streett games on finite graphs. In *In GDV'05*, 2005.

[45] Marian V. Iordache and Panos J. Antsaklis. Design of t-liveness enforcing supervisors in Petri nets. *IEEE Trans. Automat. Contr.*, 48(11):1962–1974, 2003.

[46] Victor Khomenko, Maciej Koutny, and Walter Vogler. Canonical prefixes of Petri net unfoldings. *Acta Inf.*, 40(2):95–118, 2003.

[47] Lawrence E. Landweber and Edward Lowell Robertson. Properties of conflict-free and persistent Petri nets. *J. ACM*, 25(3):352–364, July 1978.

[48] Laurent Mazaré. Using unification for opacity properties. In *Proc. of WITS*, pages 165–176, 2004.

[49] Kenneth L. McMillan. *Symbolic model checking.* Kluwer, 1993.

[50] Robert McNaughton. Infinite games played on finite graphs. *Annals of Pure and Applied Logic*, 65(2):149 – 184, 1993.

[51] T. Murata. Petri nets: Properties, analysis and applications. In *Proceedings of the IEEE*, volume 77, pages 541–580, April 1989.

[52] James L. Peterson. Petri nets. *ACM Comput. Surv.*, 9(3):223–252, September 1977.

[53] N. Piterman and A. Pnueli. Faster solutions of Rabin and Streett games. In *21st Symposium on Logic in Computer Science*, pages 275–284, 2006.

[54] Amir Pnueli. The temporal logic of programs. In *Foundations of Computer Science, 1977., 18th Annual Symposium on*, pages 46–57, Oct 1977.

[55] Lucia Pomello. Some equivalence notions for concurrent systems. an overview. In Grzegorz Rozenberg, editor, *Advances in Petri Nets 1985, covers the 6th European Workshop on Applications and Theory in Petri Nets, Espoo, Finland in June 1985, selected papers*, volume 222 of *Lecture Notes in Computer Science*, pages 381–400. Springer, 1985.

[56] Peter J. Ramadge and W. Murray Wonham. The control of discrete event systems. In *IEEE*, volume 77, pages 81–98, 1989.

[57] Wolfgang Reisig. Partial order semantics versus interleaving semantics for CSP-like languages and its impact on fairness. In *Proceedings of the 11th Colloquium on Automata, Languages and Programming*, pages 403–413, London, UK, UK, 1984. Springer-Verlag.

[58] Wolfgang Reisig. *Elements of distributed algorithms: modeling and analysis with Petri nets*. Springer, 1998.

[59] A. W. Roscoe. CSP and determinism in security modelling. In *IEEE Symposium on Security and Privacy*, pages 114–127. IEEE Computer Society, 1995.

[60] Sankardas Roy, Charles Ellis, Sajjan Shiva, Dipankar Dasgupta, Vivek Shandilya, and Qishi Wu. A survey of game theory as applied to network security. *2014 47th Hawaii International Conference on System Sciences*, 0:1–10, 2010.

[61] John Rushby. Noninterference, transitivity, and channel-control security policies. Technical report, dec 1992.

[62] Peter Y. A. Ryan. Mathematical models of computer security. In Focardi and Gorrieri [34], pages 1–62.

[63] Peter Y. A. Ryan and Steve A. Schneider. Process algebra and non-interference. In *CSFW*, pages 214–227. IEEE Computer Society, 1999.

[64] Meera Sampath, Raja Sengupta, Stephane Lafortune, KAsim Sinnamohideen, and Demosthenis Teneketzis. Diagnosability of discrete-event systems. *IEEE Trans. Automat. Contr.*, 40(9):1555–1557, September 1995.

[65] Steve Schneider and Abraham Sidiropoulos. CSP and anonymity. In *In European Symposium on Research in Computer Security*, pages 198–218. Springer-Verlag, 1996.

[66] Manuel Silva. Half a century after Carl Adam Petri's Ph.D. thesis: A perspective on the field. *Annual Reviews in Control*, 37(2):191 – 219, 2013.

[67] Roberto Gorrieri Simone Frau and Carlo Ferigato. Petri net security checker: Structural non-interference at work. In Pierpaolo Degano, Joshua D. Guttman, and Fabio Martinelli, editors, *Formal Aspects in Security and Trust*, volume 5491 of *Lecture Notes in Computer Science*, pages 210–225. Springer, 2008.

[68] David Sutherland. A model of information. In *Proc. of National Computer Security Conference*, pages 175–183, 1986.

[69] Wil M. P. van der Aalst. The application of Petri nets to workflow management. *Journal of Circuits, Systems, and Computers*, 8(1):21–66, 1998.

[70] Rob J. van Glabbeek, Ursula Goltz, and Jens-Wolfhard Schicke. On causal semantics of Petri nets. In Joost-Pieter Katoen and Barbara König, editors, *CONCUR*, volume 6901 of *Lecture Notes in Computer Science*, pages 43–59. Springer, 2011.

[71] Moshe Y. Vardi. An automata-theoretic approach to linear temporal logic. In Faron Moller and Graham Birtwistle, editors, *Logics for Concurrency*, volume 1043 of *Lecture Notes in Computer Science*, pages 238–266. Springer Berlin Heidelberg, 1996.

[72] Walter Vogler. Failures semantics and deadlocking of modular Petri nets. *Acta Inf.*, 26(4):333–348, 1989.

[73] Walter Vogler, Christian Stahl, and Richard Müller. Trace- and failure-based semantics for bounded responsiveness. In Carlos Canal and Massimo Villari, editors, *ESOCC Workshops*, volume 393 of *Communications in Computer and Information Science*, pages 129–143. Springer, 2013.

[74] Frank Wallner. Model checking LTL using net unforldings. In Alan J. Hu and Moshe Y. Vardi, editors, *Computer Aided Verification, 10th International Conference, CAV '98, Vancouver, BC, Canada, June 28 - July 2, 1998, Proceedings*, volume 1427 of *Lecture Notes in Computer Science*, pages 207–218. Springer, 1998.