Many-to-many information flow policies

Baldan, Paolo; Lluch Lafuente, Alberto

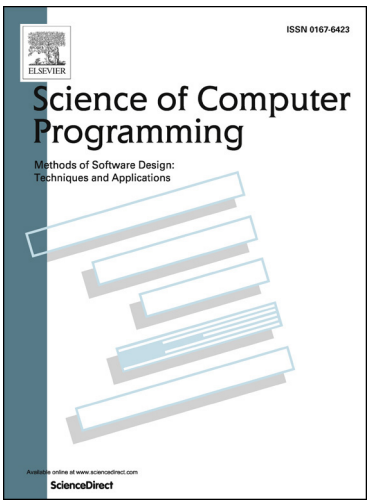
Published in:
Science of Computer Programming

Link to article, DOI:
10.1016/j.scico.2018.08.003

Publication date:
2018

Document Version
Peer reviewed version

Accepted Manuscript



Many-to-many information flow policies

Paolo Baldan, Alberto Lluch Lafuente

PII: S0167-6423(18)30326-5
DOI: https://doi.org/10.1016/j.scico.2018.08.003
Reference: SCICO 2230

To appear in: *Science of Computer Programming*

Received date: 13 November 2017
Revised date: 13 July 2018
Accepted date: 20 August 2018

Please cite this article in press as: P. Baldan, A. Lluch Lafuente, Many-to-many information flow policies, *Sci. Comput. Program.* (2018), https://doi.org/10.1016/j.scico.2018.08.003

Highlights

- A policy language for controlling flows of information in concurrent systems.
- Regulation of flows among sets of security levels instead of just individual levels.
- A causality-based semantics, defined on event structures.
- Expressivity and decidability properties of the semantics.
- A verification technique for safe Petri nets, based on finite net unfoldings.

Many-to-Many Information Flow Policies

Paolo Baldan[a], Alberto Lluch Lafuente[b]

[a] *Università di Padova, Dipartimento di Matematica*
baldan@math.unipd.it
[b] *Technical University of Denmark, DTU Compute*
albl@dtu.dk

Abstract

Information flow techniques typically classify information according to suitable security levels and enforce policies that are based on binary relations between individual levels, e.g., stating that information is allowed to flow from one level to another. We argue that some information flow properties of interest naturally require coordination patterns that involve *sets* of security levels rather than individual levels: some secret information could be safely disclosed to a set of confidential channels of incomparable security levels, with individual leaks considered instead illegal; a group of competing agencies might agree to disclose their secrets, with individual disclosures being undesired, etc. Motivated by this, we study a semantic foundation for such properties based on causal models of computation. We propose a simple language for expressing information flow policies where the usual admitted flow relation between individual security levels is replaced by a relation between sets of security levels, thus allowing to capture coordinated flows of information. The flow of information is expressed in terms of causal dependencies and the satisfaction of a policy is defined with respect to an event structure that is assumed to capture the causal structure of system computations. We also preliminarly explore possibilities for practical applicability of our approach by focusing on systems specified as safe Petri nets, a formalism with a well-established causal semantics. We show how unfolding-based verification techniques for Petri nets can be adopted for solving the problem of checking policy satisfaction.

Keywords: Information Flow, Coordination, Declassification, Non-Interference, Causality.

1. Introduction

As the number of interconnected devices increases, the focus on security-related aspects of coordinated computations gains more and more relevance and appeal. Techniques for controlling and enforcing the flow of information need to be applied, and possibly extended to deal with coordination aspects. Typically, the entities of a system are assigned a security level, and information flow policies

prescribe which interactions are legal and which are forbidden. This is normally expressed via a relation that models the admitted flows between security levels.

Motivation and Problem Statement. The information flow relations used in the literature to model policies are almost invariably binary relations between individual security levels. This paper is motivated by the observation that some desired information flow properties naturally involve suitable coordinated *sets* of security levels rather than mere individual levels.

For example, some secret information (say, owned by a government agency E, cf. Figure 1) could be safely disclosed to a set of confidential channels of incomparable security levels (say, corresponding to competing investors C and D) *simultaneously*, with individual leaks considered instead illegal or unfair. This is for instance, the spirit of U.S. security and exchange commission's *regulation fair disclosure* [1]. Dually, a group of competing companies (say A and B in Figure 1) may agree to *collectively* disclose their secrets (say to the government agency E), with individual disclosures being undesired. This paper is motivated by such scenarios and, in general, by the following research questions:

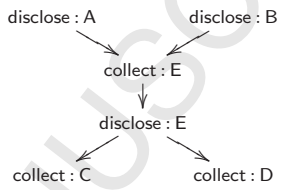


Figure 1: Information flow example.

Q1. *What could be a natural way of syntactically expressing information flow policies that regulate flows among sets of security levels?*

Q2. *What could be a suitable semantic foundation for such policies?*

Q3. *What are the possibilities for the development of effective analysis tools?*

Contributions. We address question Q1 by proposing a simple policy specification language that extends a well-studied and popular family of languages for information flow policies, namely *security diagrams*. The extension considers relations between *sets* of security levels instead of just *single* levels. The clauses in our policies are of the form $\mathsf{A}_1, \ldots, \mathsf{A}_m \rightsquigarrow \mathsf{B}_1, \ldots, \mathsf{B}_n$, intuitively meaning that the security levels $\mathsf{A}_1, \ldots, \mathsf{A}_m$ are allowed to coordinate in order to let information flow to the security levels $\mathsf{B}_1, \ldots, \mathsf{B}_n$.

As an answer to Q2 we propose to use a causality model rooted in well-studied models of concurrency, namely event structures [2, 3], as a reference semantic model. Under such model, the flow of information between entities is captured in terms of the existence of causal dependencies between events representing occurrences of actions of those entities. The idea is that causal dependencies between events represent the transfer of some information.Thus, causal dependencies are required to obey to coordination patterns as prescribed by the information flow policy. For traditional intransitive binary policies any flow of information, i.e., any (direct) causality $a < b$ between events a and b needs to be allowed by the policy, i.e., if the level of a is A and the level of b is B

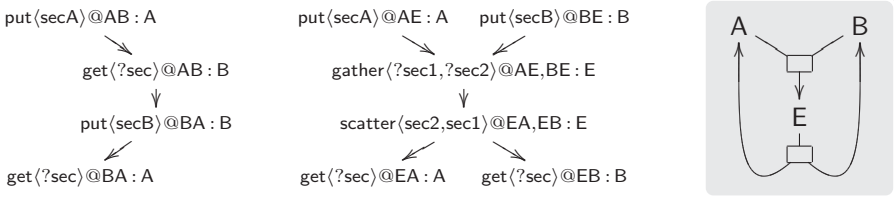
then the policy has to include a clause $\mathsf{A} \rightsquigarrow \mathsf{B}$. We generalise this to many-to-many policies by requiring that any direct causality $a < b$ is part of a possibly more complex interaction that conforms to a coordination pattern allowed by the policy, i.e., if A_1 and B_1 are the security levels of a and b, respectively, there must exist a clause $\mathsf{A}_1, \mathsf{A}_2 \ldots \mathsf{A}_n \rightsquigarrow \mathsf{B}_1, \mathsf{B}_2, \ldots, \mathsf{B}_m$ in the policy and events $a_1, a_2, \ldots, a_n$ (with a possibly but not necessarily equal to a_1), $b_1, b_2, \ldots, b_m$ (with b equal to b_1) such that each event a_i has level A_i, each event b_j has level B_j, and events $a_1, \ldots a_n$ are (suitably coordinated) causes of the events $b_1, \ldots, b_m$.

As an example, consider the diagram of Figure 1, where arrows represent direct causalities between events and the security levels coincide with the entities participating in the scenario. For events we use the notation $\mathsf{name} : \mathsf{level}$. The direct causality from event $\mathsf{disclose} : \mathsf{A}$ to event $\mathsf{collect} : \mathsf{E}$ is allowed by the policy $\mathsf{A}, \mathsf{B} \rightsquigarrow \mathsf{E}$ since $\mathsf{collect} : \mathsf{E}$ is also causally dependent on $\mathsf{disclose} : \mathsf{B}$, thus providing some guarantee of the fact that A and B disclose their secrets collectively. Analogously, the direct causality from $\mathsf{disclose} : \mathsf{E}$ to $\mathsf{collect} : \mathsf{C}$ is allowed by the policy $\mathsf{E} \rightsquigarrow \mathsf{C}, \mathsf{D}$ since there is a causality relation from $\mathsf{disclose} : \mathsf{E}$ to $\mathsf{collect} : \mathsf{D}$ as well, ensuring that E discloses the secrets to both C and D.

To answer Q3, we study several properties of our policy language. First, we discuss how different policies can be related in terms of the strictness of their requirements. We provide a sound and complete axiomatisation of the strictness relation that can be used to compare policies and decide whether one poses stricter constraints than the other. Moreover, we show how the framework developed abstractly over event structures can be instantiated on a concrete specification formalism for which the connection to event structures as semantic model has been extensively studied, namely finite safe Petri nets. We show how existing (meta)techniques for analysing Petri nets can be adopted to decide whether (the event structure semantics of) a Petri net satisfies a given policy. In particular, the technique is based on the generation of a finite prefix of the unfolding of the net, which reflects a prefix of the underlying event structure and is shown to be complete with respect to the policy satisfaction problem. This investigation gives some indications on the complexity of the policy satisfaction problem for specific classes of policies.

Previous work. This paper is a revised and extended version of [4]. The most significant revision regards the semantics of information flow policies, which has been refined and simplified. As a result the satisfaction of the two kinds of security clauses (unfair and fair) can be uniformly defined in terms of the possibility/necessity of certain computations.[1] With respect to [4] the verification technique for safe Petri nets (Section 5) is an entirely novel contribution and

[1] For the convenience of reviewers, section Appendix C provides some more details and examples where the semantics differ.



(a) Protocol P_1 (b) Protocol P_2 (c) Policy

Figure 2: Two secret exchange protocols and a security policy.

the proofs all formal statements have been added.[2]

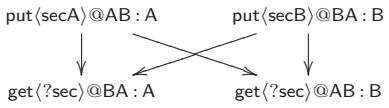
Synopsis. In Section 2 we introduce several motivating examples, including a running example inspired by secret exchange protocols, and we suggest applications to program security. Section 3 provides some technical background on event structures. Section 4 presents the policy language and the notion of policy satisfaction. Section 5 presents the verification techniques for safe Petri nets. Section 6 discusses the relation with notions of information flow based on interleaving semantics, in particular with classical trace- and bisimulation-based non-interference. Section 6 discusses other related works. Section 7 concludes our paper and outlines future research. Appendix A contains formal proofs of all statements in the paper. Appendix B proves the decidability of policy satisfaction over a subclass of event structures, namely regular trace event structures [5]. The proof is based on the fact that policy satisfaction is expressible as a monadic trace logic property and the corresponding model checking problem is decidable [6]. Since regular trace event structures are exactly the event structures associated with finite safe Petri nets, this result is implied by the material in Section 5. Still, the result has a value since it shows that policy satisfaction can be expressed as a monadic second order property, in a way that the proof would be easily adaptable for mild changes of the policy language. Finally, Appendix C includes a comparison with the semantics of information flow policies in the conference version of the paper.

2. Motivating Examples

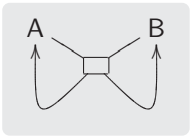
We introduce here some examples that have motivated our work and that we envisage as application domains for many-to-many information flow analysis.

Simultaneous Secret Exchange. Consider the problem of exchanging a secret between two parties, say Alice and Bob. We shall use this as a running example

[2]These include, beside the novel results related to the verification technique, the revised proofs of the formal statements of [4], which were used for reviewing purposes but not published in [4].



(a) Protocol P_3



(b) Policy

Figure 3: A secret exchange protocol satisfying a policy without intermediary.

throughout the paper. A way to solve the problem would be to proceed according to the protocol in Figure 2a, where Alice starts the protocol by sending her secret to Bob, which then replies with his own secret. Technically this is a causal security model (see Definition 6) that will be explained later. Here it suffices to understand that the figure models the structure of the communication in an execution, where event $\mathsf{put}\langle \mathsf{m} \rangle @ \mathsf{C} : \mathsf{P}$, represents party P sending message m on channel C, and event $\mathsf{get}\langle \mathsf{t} \rangle @ \mathsf{C} : \mathsf{P}$, represents party P receiving a message from channel C to be saved according to the pattern t, where message fields used to capture information (often called "formal fields" or "binders") are denoted with a leading question mark "?". Arrows represent (direct) causal dependencies between events. The protocol has essentially the same structure of classical key exchange protocols [7] and does not solve one of the main concerns of the so-called *simultaneous secret exchange problem*, which is to avoid or minimise competitive advantage among the parties exchanging the secrets (see e.g. [8]). If we assume to deal with information of two security levels (one for each party), a standard approach to information flow does not help much in this scenario, as we can just allow flows between Alice and Bob (and thus accept any protocol with no guarantee) or forbid them (and thus reject all protocols).

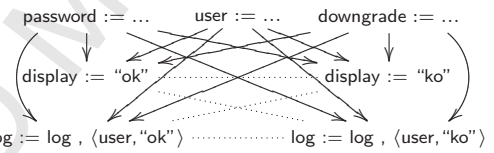
One standard solution to the simultaneous secret exchange problem is to use a trusted intermediary (say Eve). Many-to-many information flow policies can be used to specify some desired properties of intermediary-based protocols. For example, we may require that the intermediary forwards information to both Alice and Bob (denoted by an information flow policy $\mathsf{Eve} \rightsquigarrow \mathsf{Alice}, \mathsf{Bob}$), and that the intermediary accepts information from Alice and Bob only if collectively disclosed (denoted by an information flow policy by $\mathsf{Alice}, \mathsf{Bob} \rightsquigarrow \mathsf{Eve}$). A graphical representation for this security policy, that will be refined and explained in detail later, can be found in Figure 2c. The protocol sketched in Figure 2b, which uses multi-party interactions, satisfies the desired information flow properties. The protocol uses in particular a gather operation to piece-wise collect a list of messages from different sources, and a scatter operation to piece-wise broadcast the list of secrets.

The use of many-to-many information flow policies, as we will see, allows one to require stronger guarantees on secret exchange protocols, by specifying that only collective and simultaneous flow between Alice and Bob, without intermediaries, are admitted. This is expressed by the policy $\mathsf{Alice}, \mathsf{Bob} \rightsquigarrow \mathsf{Alice}, \mathsf{Bob}$ (see Figure 3b, for a graphical representation) and realised by the protocol in

Figure 3a, where vertical and cross dependencies are control and data dependencies, respectively. Note that Alice and Bob concurrently send their secrets first and then gather the secrets sent from the other party.

Language-Based Security and Declassification. We use a classic problem of declassification in program security to illustrate how our approach can be used to check information flow properties in the traditional sense [9], and how our policies can be used to specify several useful forms of declassification [10, 11]. The access control program below, on the left, written in a simple imperative programming language in the style of [9], is checking a password and preparing a reply to the user in a variable display. As already mentioned, a causal semantics has to be given to a program, suitably tracking the dependencies of interest. Here we proceed according to the following idea: (i) events correspond to variable updates or variable initialisations, (ii) an update $\mathsf{x} := \mathsf{e}$ causally depends on previous updates of all variables in e or their initialisation, (iii) if a control point is conditioned by y then all updates in all branches of the control causally depend on the latest update of y (or its initialisation), and (iv) conflict relations (represented as dotted lines) capture branching. For the program at hand, the resulting event structure can be found on the right.

```
if downgrade then
    if check(user,password) then
        display := "ok"
        log := log , ⟨user,"ok"⟩
    else
        display :="ko"
        log :=log , ⟨user,"ko"⟩
```



Disregarding whether the password is correct or not, there is a flow of information concerning the password to the user, represented as a causality relation from the latest update of the password to the updates of variable display.

For simplicity assume that each variable has its own security level, coinciding with the variable name. A standard security type system would consider the program to be insecure and our approach would agree on that in the absence of a policy password $\rightsquigarrow$ display allowing the leaks. Of course, such a policy is not desirable in general (e.g., the user may be malicious). However, the program provides some guarantees that make it acceptable from the security point of view. First, the reply is also influenced by a check on variable downgrade, which may be used to disable login (e.g., after several unsuccessful attempts). This provides a standard form of controlled declassification. Requiring that the declassification is controlled as desired, i.e. that the flow from password to display is only possible jointly with downgrade can be done by imposing a policy password, downgrade $\rightsquigarrow$ display. In addition, the program above is using a log to keep track of logging attempts. This provides the additional desirable property that password leaks are only possible when the information concerning the access attempt also flows to the log (denoted user, password $\rightsquigarrow$ display, log).

3. Event Structures

We model the causal behaviour of a system with (prime) event structures [2, 3], a well-studied semantic model of concurrency where causality is a primitive notion. The way in which an event structure is associated to a specific formalism will depend on the system features one intends to capture (e.g., data/control flow, etc.). In general terms, causality will be used to represent the flow of information of interest.

Definition 1 (event structure). An *event structure* $\mathcal{E} = \langle E, \leq, \# \rangle$ consists of a set E of events, a partial order relation $\leq \subseteq E \times E$ called *causality*, and an irreflexive symmetric relation $\# \subseteq E \times E$ called *conflict* such that:

(i) for all $e \in E$ the set of causes $[e] = \{e' \in E \mid e' \leq e\}$ is finite (finitariness);

(ii) for all $e, e', e'' \in E$, if $e \# e'$ and $e' \leq e''$ then $e \# e''$ (conflict inheritance).

For $e, e' \in E$ we write $e < e'$ (e is a proper cause of e') if $e \leq e'$ and $e \neq e'$. We write $e \lessdot e'$ and say that e is a direct cause of e' when $e < e'$ and for all $e'' \in E$ if $e \leq e'' \leq e'$ either $e'' = e$ or $e'' = e'$. Causality will be used on sets of events with a universal interpretation, i.e., for $X, Y \subseteq E$ we will write $X < Y$ (resp. $X \lessdot Y$) if $x < y$ (resp. $x \lessdot y$) for all $x \in X$ and $y \in Y$. The causes of a set of events $X \subseteq E$ are defined as $[X] = \bigcup_{e \in X} [e]$. For $e \in E$ we define the set of its *proper causes* as $[e) = [e] \setminus \{e\}$. We say that $e, e' \in E$ are in *direct conflict*, written $e \#_\mu e'$, when $e \# e'$ and for all $e'' \in [e)$ it holds $\neg(e'' \# e')$ and for all $e''' \in [e')$ it holds $\neg(e \# e''')$, i.e., the conflict is not inherited.

Figures 2 and 3a show three event structures corresponding to different protocols in our running example. The events correspond to communication operations and are annotated with the initials of the principal executing the actions (which is the security level assigned to the event, as we shall explain later). Causality is represented graphically by directed arrows, while conflict (appearing in later examples) is represented by dotted undirected lines. For the sake of a lighter notation, we follow the tradition of depicting direct causalities and conflicts only.

Event structures describe the possible events in computations and their mutual dependencies. When $e < e'$ then e must necessarily occur before e', while if $e \# e'$ then the occurrence of e excludes e' (and vice versa).

Computations are characterised as conflict-free sets of events, containing the causes of all events. These are formally introduced by the next definition.

Definition 2 (configuration). Let $\mathcal{E} = \langle E, \leq, \# \rangle$ be an event structure. A *configuration* of $\mathcal{E}$ is a subset $C \subseteq E$ such that $\neg(e \# e')$ for all $e, e' \in C$ and $[C] = C$. The set of configurations of $\mathcal{E}$ is denoted $\mathcal{C}(\mathcal{E})$.

For any event e, the causes $[e]$ and the proper causes $[e)$ are configurations.

Figure 4 depicts all the configurations of the event structure of Figure 3a. They are related by arrows that represent transitions from a configuration to a larger one by addition of an event.

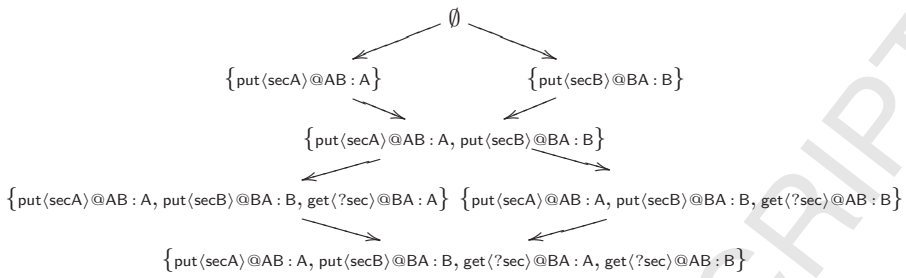


Figure 4: Configurations of the event structure in Figure 3a.

Definition 3 (extension). Let $\mathcal{E} = \langle E, \leq, \# \rangle$ be an event structure, and let $C \in \mathcal{C}(\mathcal{E})$. We say that the set of events X extends C when $C \cap X = \emptyset$ and $C \cup X \in \mathcal{C}(\mathcal{E})$. In this situation we write $C \oplus X$ for $C \cup X$. Whenever $X = \{e\}$ we often write $C \oplus e$ instead of $C \oplus \{e\}$. The set of possible singleton extensions of a configuration C is $\mathsf{PE}(C) = \{e \in E \mid C \oplus e \in \mathcal{C}(\mathcal{E})\}$. A configuration is *maximal* if it cannot be extended.

The transition system semantics of an event structure is obtained considering configurations as states and singleton extensions as transitions.

Definition 4 (transition system). The *transition system* of an event structure $\mathcal{E} = \langle E, \leq, \# \rangle$ is the tuple $TS(\mathcal{E}) = \langle \mathcal{C}(\mathcal{E}), \{C \to C \oplus e \mid C \in \mathcal{C}(\mathcal{E}) \wedge e \in E\} \rangle$.

The diagram of Figure 4 represents the transition system of the event structure in Figure 3a.

4. Many-to-Many Information Flow Policies

We introduce here our notion of many-to-many information flow policies that describe the legal interactions among sets of security levels. Section 4.1 presents the policy language, Section 4.2 defines the semantics and Section 4.3 studies some of its properties.

4.1. A Policy Language for Many-to-Many Information Flows

We first introduce the syntax of many-to-many information flow policies. Such policies specify relations among security levels taken from a finite set $\mathscr{L}$. The relations can be subject to *coordination constraints*, i.e. constraints on the directness and fairness of the interactions. Formally, a policy is defined as follows.

Definition 5 (many-to-many information flow policy). Let $\mathscr{L}$ be a finite set of security levels and $\mathscr{C} = \{d, f\}$ be a set of coordination constraints. A *many-to-many information flow policy* is a pair $\Pi = \langle \mathscr{L}, \rightsquigarrow \rangle$ where $\rightsquigarrow \subseteq 2^{\mathscr{L}} \times 2^{\mathscr{C}} \times 2^{\mathscr{L}}$ and $(\{\mathsf{A}\}, \emptyset, \{\mathsf{A}\}) \in \rightsquigarrow$ for all $\mathsf{A} \in \mathscr{L}$. We denote by $\mathcal{P}$ the set of all policies.

We write $\mathbb{A} \overset{\sigma}{\rightsquigarrow} \mathbb{B}$ for $(\mathbb{A}, \sigma, \mathbb{B}) \in \rightsquigarrow$, often dropping brackets from σ, $\mathbb{A}$ and $\mathbb{B}$. In particular, we will write $\mathbb{A} \rightsquigarrow \mathbb{B}$ instead of $\mathbb{A} \overset{\emptyset}{\rightsquigarrow} \mathbb{B}$. Security levels in $\mathscr{L}$ are ranged over by $\mathsf{A}, \mathsf{B}, \ldots$ and sets of security levels are ranged over by $\mathbb{A}, \mathbb{B}, \ldots$. The requirement that $\mathsf{A} \rightsquigarrow \mathsf{A}$ for all $\mathsf{A} \in \mathscr{L}$ is standard: information is always allowed to flow within the same level of security. Finiteness of the set of security levels $\mathscr{L}$ is a natural assumption. It could be meaningful to allow $\mathscr{L}$ to be infinite only when security levels can be generated dynamically. The theory in the paper could be adapted trivially apart from the algorithmic techniques in Section 5 (and the decidability result in Section Appendix B in the appendix).

Note that an information flow policy can be seen as a directed hyper-graph whose hyper-arcs are labelled by (possibly empty) subsets of $\mathscr{C}$. This analogy is exploited in the visual representation of security policies.

Informally, a clause $\mathbb{A} \overset{\sigma}{\rightsquigarrow} \mathbb{B}$ specifies that a group of entities whose set of security levels is $\mathbb{A}$ is allowed to *influence* a group of entities whose set of security levels is $\mathbb{B}$, subject to the coordination constraints in σ. Our notion of *influence* refers to the existence of causal dependencies, as it will be made more precise when formally defining the semantics (Section 4.2).

In order to explain the intended meaning of clauses, we first focus on clauses with an empty set of coordination constraints. Examples illustrating the use of constraints will be discussed later. As a first example, the policy

$$\mathsf{Alice}, \mathsf{Bob} \rightsquigarrow \mathsf{Eve} \tag{1}$$

allows Alice and Bob to jointly influence Eve. This imposes stricter requirements with respect to the policy below consisting of one-to-one clauses:

$$\begin{array}{rcl} \mathsf{Alice} & \rightsquigarrow & \mathsf{Eve} \\ \mathsf{Bob} & \rightsquigarrow & \mathsf{Eve} \end{array} \tag{2}$$

that allows both Alice and Bob to influence individually Eve.

Dually, the policy

$$\mathsf{Eve} \rightsquigarrow \mathsf{Alice}, \mathsf{Bob} \tag{3}$$

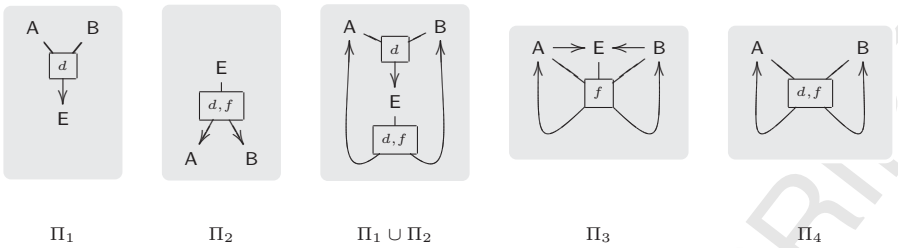
allows Eve to influence the set of entities composed by Alice and Bob. Again, this is stricter than the policy consisting of one-to-one clauses:

$$\begin{array}{rcl} \mathsf{Eve} & \rightsquigarrow & \mathsf{Alice} \\ \mathsf{Eve} & \rightsquigarrow & \mathsf{Bob} \end{array} \tag{4}$$

where Eve can influence both Alice and Bob individually.

The clauses in Equation 1 and 3 can be combined into a policy that allows Alice and Bob to exchange their secrets using Eve as an intermediary. An alternative, slightly different policy that allows a similar flow of information is

$$\begin{array}{rcl} \mathsf{Alice},\mathsf{Bob},\mathsf{Eve} & \rightsquigarrow & \mathsf{Alice},\mathsf{Bob} \\ \mathsf{Alice} & \rightsquigarrow & \mathsf{Eve} \\ \mathsf{Bob} & \rightsquigarrow & \mathsf{Eve} \end{array} \tag{5}$$



Π_1 Π_2 $\Pi_1 \cup \Pi_2$ Π_3 Π_4

Figure 5: Security policies, graphically.

In this case, Alice, Bob and Eve can influence Alice and Bob; Alice can influence Eve; and Bob can influence Eve. Intuitively, this can be used to specify that both Alice and Bob can talk individually to the intermediary Eve, which in turn can talk to Alice and Bob in coordination with them, thus desynchronising the first part of the communication.

As mentioned above, for a clause in $\mathbb{A} \overset{\sigma}{\rightsquigarrow} \mathbb{B}$ the superscript $\sigma \subseteq \mathscr{C}$ allows one to specify some additional coordination constraints on the interaction pattern among the levels in $\mathbb{A}$ and $\mathbb{B}$. The superscript d requires all entities in $\mathbb{A}$ to influence all the entities in $\mathbb{B}$ *directly*. For instance, for the policy (1) we might want Alice and Bob to influence Eve directly, with no delegation among them or to another intermediary. This leads to the policy $\Pi_1 \triangleq \mathsf{Alice}, \mathsf{Bob} \overset{d}{\rightsquigarrow} \mathsf{Eve}$ depicted in Figure 5.

By default the flow of information to all of the entities in $\mathbb{B}$ is required to be possible, i.e., if the information flows to some entity in $\mathbb{B}$ then it must be possible to continue the computation in a way that the information reaches all other levels in $\mathbb{B}$. The superscript f, that stands for "fair" disclosure, imposes a stricter requirement: whenever information flows to one of the entities in $\mathbb{B}$, then it must eventually flow to all other levels in $\mathbb{B}$ in all possible computations.

It is worth to remark that the information flowing to the entities in $\mathbb{B}$ need not be the same since causality in our approach represents in general the transfer of *some* information. Ensuring that the *same* information is being transferred depends on the actual causal semantics under consideration.

In our previous examples, for policies (3) and (5) it is natural to require fairness to forbid competition among Alice and Bob. This leads to the policies Π_2 and Π_3 in Figure 5. Observe that fairness constraints are automatically satisfied and thus superfluous when there is only one security level in the target, like Eve in Π_1. Notice that in policy Π_3, the absence of the "direct" constraint is essential to allow Eve to act as an intermediary. A variant of Π_3 without intermediary is $\Pi_4 \triangleq \mathsf{Alice}, \mathsf{Bob} \overset{d,f}{\rightsquigarrow} \mathsf{Alice}, \mathsf{Bob}$, which specifies a direct exchange of information between Alice and Bob.

4.2. Semantics of Many-to-Many Policies

In our setting, a *security model* is an event structure used to capture the structure of computations and flows. In such a structure, events intuitively correspond to actions of some security level, causal dependencies between events

may arise from data dependencies, control flow dependencies or communication dependencies (e.g. synchronisations). Formally, a causal security model is defined as follows.

Definition 6 (causal security model). Let $\mathscr{L}$ be a set of security levels. A *(causal) security model* in $\mathscr{L}$ is a pair $\langle \mathcal{E}, \lambda \rangle$ where $\mathcal{E} = \langle E, \leq, \# \rangle$ is an event structure and $\lambda : E \to \mathscr{L}$ is a *security assignment*.

The security assignment λ maps each event to a security level. For $X \subseteq E$ and $\mathbb{A} \subseteq \mathscr{L}$ we write $X : \mathbb{A}$ if $|X| = |\mathbb{A}|$ and $\{\lambda(e) \mid e \in X\} = \mathbb{A}$. We write $e : \mathsf{A}$ instead of $\{e\} : \{\mathsf{A}\}$. The event structures of Figures 1–3a are security models, where the security assignment corresponds to the principal annotations.

We can now define what it means for a security model to satisfy a policy. A policy does not prescribe any progress by the system, but when some progress occurs, the flows of information involved must be allowed by some clause. Clauses are hence used to justify information flows, which, in our approach, are represented as direct causalities. In order to justify a flow, a clause requires the existence of a computation context defined by two sets of events (X, Y) such that, roughly speaking, events in X are flow sources (causes) and events in Y are the flow targets (events being enabled by the sources). This is formalised in the following definition.

Definition 7 (conformance to a clause). Let $\langle \langle E, \leq, \# \rangle, \lambda \rangle$ be a security model in a set of security labels $\mathscr{L}$, $X, Y \subseteq E$ be two sets of events, and $\sigma \subseteq \{f\}$ be a set of coordination constraints, we say that (X, Y) *conforms* to the clause $\mathbb{A} \overset{\sigma}{\rightsquigarrow} \mathbb{B}$ (resp. $\mathbb{A} \overset{\sigma \cup \{d\}}{\rightsquigarrow} \mathbb{B}$) if $X : \mathbb{A}$, $Y : \mathbb{B}$ and $X < Y$ (resp. $X \lessdot Y$).

In words, (X, Y) conforms to a clause $\mathbb{A} \overset{\sigma}{\rightsquigarrow} \mathbb{B}$, where σ does not include the directness constraint, whenever X and Y contain events whose levels are exactly those in $\mathbb{A}$ and $\mathbb{B}$ and each event in X causes all events in Y. In presence of the directness constraint, the causal dependencies must be direct. The notion of conformance is instrumental to identify the context of execution that a clause can use to justify a direct causal dependency between two events. Justification of direct causal dependencies is formalised as follows.

Definition 8 (justification of a direct flow). Let $\langle \langle E, \leq, \# \rangle, \lambda \rangle$ be a security model in a set of security labels $\mathscr{L}$, $e, e' \in E$ be two events such that $e \lessdot e'$, and $\mathbb{A} \overset{\sigma}{\rightsquigarrow} \mathbb{B}$ be a clause. We say that $\mathbb{A} \overset{\sigma}{\rightsquigarrow} \mathbb{B}$ *justifies* $e \lessdot e'$, denoted $e \lessdot e' \models \mathbb{A} \overset{\sigma}{\rightsquigarrow} \mathbb{B}$, iff $\lambda(e) \in \mathbb{A}$, $\lambda(e') \in \mathbb{B}$ and the following holds:

- if $f \notin \sigma$ then for all configurations $C \in \mathcal{C}(\mathcal{E})$ such that $e' \in \mathsf{PE}(C)$ there is *some* configuration $C' \in \mathcal{C}(\mathcal{E})$ such that $C \oplus e' \subseteq C'$ and there are sets $X, Y \subseteq C'$ with $e' \in Y$ such that (X, Y) conforms to $\mathbb{A} \overset{\sigma}{\rightsquigarrow} \mathbb{B}$.
- if $f \in \sigma$ then for all configurations $C \in \mathcal{C}(\mathcal{E})$ such that $e' \in \mathsf{PE}(C)$ *for all maximal* configurations $C' \in \mathcal{C}(\mathcal{E})$ such that $C \oplus e' \subseteq C'$ there are set $X, Y \subseteq C'$ with $e' \in Y$ such that (X, Y) conforms to $\mathbb{A} \overset{\sigma}{\rightsquigarrow} \mathbb{B}$.

We now discuss this definition in detail as it is a crucial one to understand the semantics of our policies. The definition distinguishes two cases, respectively the absence and the presence of fairness requirements. Let us first focus on the case in which fairness is not required. Intuitively, a direct causality $e \lessdot e'$ is justified by a clause $\mathbb{A} \overset{\sigma}{\rightsquigarrow} \mathbb{B}$ if, whenever a computation enables e', we can execute e' and continue the computation in a way that the dependency $e \lessdot e'$ becomes part of a (possibly) larger interaction of sets of events X and Y that conforms to the clause, i.e., where all security levels in $\mathbb{A}$ are represented by an event in X, all security levels in $\mathbb{B}$ are represented by an event in Y and each event in Y depends from all events in X.

In the second case, when fairness is required, such a larger interaction made of the sets of events X and Y that conforms to the clause is not enough to occur in *some* computation. Instead it must occur in *every maximal* configuration executing the event e'. Note that restricting the attention to maximal configurations, i.e. configurations where enabled events are eventually executed, excludes violations of the policy determined by the fact that some flow does not occur despite being infinitely often enabled (e.g., due to unfairness of the scheduler).

Another issue worth discussing is whether e' and e are involved in the interaction used to justify the direct causality. We require $e' \in Y$ in order to ensure that the flow to e' is influenced by $X : \mathbb{A}$ (recall that $X < Y$) and thus by all levels in $\mathbb{A}$. On the other hand, event e is not necessarily in X since the information of level $\lambda(e)$ that is required to flow to all levels in $\mathbb{B}$ might have been provided by another cause of e'. Still, since e is a cause of e', it will be part of any computation including e', hence, intuitively, e will coordinate with the events in X to enable e'. Not requiring $e \in X$ is fundamental, e.g., to provide an asynchronous implementation of the simultaneous disclosure of a secret to several parties as we shall explain later.

Definition 9 (policy satisfaction). Let $\Pi = \langle \mathscr{L}, \rightsquigarrow \rangle$ be an information flow policy and let $\langle \mathcal{E}, \lambda \rangle$ be a security model in $\mathscr{L}$. We say that $\langle \mathcal{E}, \lambda \rangle$ satisfies Π or that $\langle \mathcal{E}, \lambda \rangle$ is Π-secure, denoted $\langle \mathcal{E}, \lambda \rangle \models \Pi$, if for all direct causalities $e \lessdot e'$ there exists a clause $\mathbb{A} \overset{\sigma}{\rightsquigarrow} \mathbb{B} \in \Pi$ such that $e \lessdot e' \models \mathbb{A} \overset{\sigma}{\rightsquigarrow} \mathbb{B}$. When λ is clear from the context we sometimes write $\mathcal{E} \models \Pi$.

Observe that only direct causalities need to be justified by the policy. This is because conceptually we are dealing with a form of intransitive information flow policy, hence only direct flows are of interest.

Table 1 summarises the satisfaction of policies by the protocols of our running example. The double horizontal lines separate protocols without intermediaries (P_1,P_3) from those with intermediaries (P_2,P_4,P_5,P_6), and the vertical double lines separates policies with intermediaries ($\Pi_1 \cup \Pi_2$ and Π_3) from policies without intermediaries (Π_4).

More in detail, let us start by discussing the scenarios with intermediaries. The security model of Figure 2b (Protocol P_2) satisfies the policies $\Pi_1 \cup \Pi_2$ and Π_3 of Figure 5. For example, the direct causality $\mathsf{put}\langle\mathsf{secA}\rangle@\mathsf{AE} : \mathsf{A} \lessdot$

	$\Pi_1 \cup \Pi_2$	Π_3	Π_4
P_1			
P_3			✓
P_2	✓	✓	
P_4			
P_5		✓	
P_6		✓	

Table 1: Models versus policies.

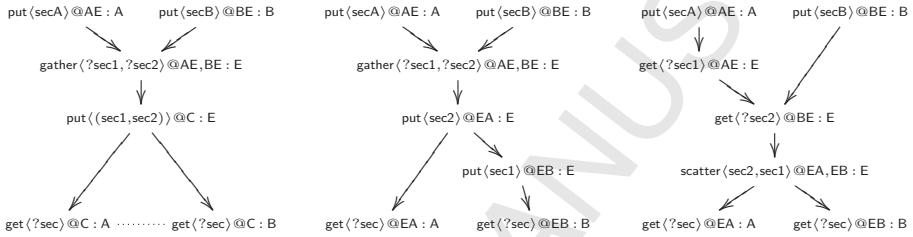


Figure 6: Secret exchange protocols P_4 (left), P_5 (center) and P_6 (right).

$\mathsf{gather}\langle ?\mathsf{sec1}, ?\mathsf{sec2}\rangle @\mathsf{AE}, \mathsf{BE} : \mathsf{E}$ can be justified by using clause $\mathsf{Alice}, \mathsf{Bob} \overset{d}{\rightsquigarrow} \mathsf{Eve}$ policy Π_1. Indeed, the only configuration that enables the flow (the causes of the gather operation), can be extended in a way that we can set X to be the set of events $\{\mathsf{put}\langle \mathsf{secA}\rangle @\mathsf{AE} : \mathsf{A}, \mathsf{put}\langle \mathsf{secB}\rangle @\mathsf{BE} : \mathsf{B}\}$ and Y to be the set of events $\{\mathsf{gather}\langle ?\mathsf{sec1}, ?\mathsf{sec2}\rangle @\mathsf{AE}, \mathsf{BE} : \mathsf{E}\}$. In other words, the disclosure to Eve is allowed since it is collectively made by Alice and Bob. Similarly, one can show that the direct causalities between the multicast of Eve and the receptions by Alice and Bob are justified by the clauses in policies Π_2 or Π_3. In order to see this for the only clause of Π_2, note that the all configurations that enable the broadcast (again, just one) extend in a way that we can set $X = \{\mathsf{scatter}\langle \mathsf{sec2}, \mathsf{sec1}\rangle @\mathsf{EA}, \mathsf{EB} : \mathsf{E}\}$ and $Y = \{\mathsf{get}\langle ?\mathsf{sec}\rangle @\mathsf{EA} : \mathsf{A}, \mathsf{get}\langle ?\mathsf{sec}\rangle @\mathsf{EB} : \mathsf{B}\}$. In other words, both message receptions of Alice and Bob depend on Eve sending the secrets, and they cannot be disabled, hence they necessarily appear in the (unique) maximal configuration. For the only clause of Π_3 the reasoning is similar but a bit more involved: the idea is to select Y as before and X to be the rest of the events. Intuitively, Eve acts as a delegated intermediary: the submissions of the secrets by Alice and Bob influence their final receptions, indirectly through the events of Eve.

Consider now the event structures in Figure 6, which represent variants of the protocols of our running example where Eve is used as an intermediary. They can be seen as alternatives to the protocol P_2 in Figure 2b. Protocol P_4 is like P_2 but the intermediary does not scatter the secrets; these are instead combined in a composite message and sent to a channel C thus creating a conflict between Alice and Bob since both expect to extract the (combined) secrets from

C. In protocol P_5, Eve sends the messages asynchronously with point-to-point operations, first to Alice and then to Bob. That is, the behaviour of Eve can be expressed by the program fragment below, whose control flow imposes control dependencies reflected as causal dependencies in P_5

$$\begin{array}{l}\mathsf{gather}\langle ?\mathsf{sec1}, ?\mathsf{sec2}\rangle @\mathsf{AE},\mathsf{BE} : \mathsf{E}\\ \mathsf{put}\langle \mathsf{sec2}\rangle @\mathsf{EA} : \mathsf{E}\\ \mathsf{put}\langle \mathsf{sec1}\rangle @\mathsf{EB} : \mathsf{E}\end{array}$$

Finally, in P_6, the reception of the secrets is asynchronous, with point-to-point operations. In this case the behaviour of Eve could be expressed as:

$$\begin{array}{l}\mathsf{get}\langle ?\mathsf{sec1}\rangle @\mathsf{AE} : \mathsf{E}\\ \mathsf{get}\langle ?\mathsf{sec2}\rangle @\mathsf{BE} : \mathsf{E}\\ \mathsf{scatter}\langle \mathsf{sec2},\mathsf{sec1}\rangle @\mathsf{EA},\mathsf{EB} : \mathsf{E}\end{array}$$

None of these variants satisfies policy $\Pi_1 \cup \Pi_2$. For instance, no clause in $\Pi_1 \cup \Pi_2$ justifies the direct causality $\mathsf{put}\langle(\mathsf{sec1}, \mathsf{sec2})\rangle @\mathsf{C} : \mathsf{E} \lessdot \mathsf{get}\langle ?\mathsf{sec}\rangle @\mathsf{C} : \mathsf{A}$ in P_4 due to the conflict between $\mathsf{get}\langle ?\mathsf{sec}\rangle @\mathsf{C} : \mathsf{A}$ and $\mathsf{get}\langle ?\mathsf{sec}\rangle @\mathsf{C} : \mathsf{B}$. In P_5 the direct causalities from Eve's put events to the get events of Alice and Bob cannot be justified since there is no Eve-labelled event that is a direct cause for both get events. Notice that such causalities could be justified if we drop the directness constraints from the clauses since the reception of the messages by Bob and Alice are causally dependent by the first put of Eve. Similarly, the direct causalities from Alice and Bob's put events to Eve's get events in P_6 cannot be justified.

The situation is different for Π_3. Indeed, both P_5 and P_6 satisfy the policy. Intuitively, the asynchronous collection of the secrets in P_6 that could not be justified in Π_2 can now be justified since Alice and Bob are allowed to talk to Eve independently. On the other hand, the asynchronous disclosure by Eve in P_5 is justified since it also depends on both Alice and Bob without directness constraint.

Similarly, it can be seen that among the protocols not using intermediaries, namely P_1 (Figure 2a) and P_3 (Figure 3a), only P_3 satisfies the policy Π_4. Protocol P_3 is indeed the only one that guarantees that Alice and Bob collectively make their secrets available to Alice and Bob simultaneously. Indeed, protocol P_3 has a unique advantage over P_1: when Alice (resp. Bob) gets the secrets, (s)he is not ensured to be in competitive advantage. Clearly, protocol P_1 does not have this property. Therefore, P_3 offers a solution to the simultaneous secret exchange problem with guarantees based on causality rather than on bounds on the amount of different information obtained by the parties (as e.g., in [12]).

4.3. Semantic properties

In order to clarify the semantics of many-to-many policies we present some properties that illustrate how different policies can be related in terms of the strictness of their requirements.

A first simple observation is that aggregating security levels preserves the satisfaction of policies. This result is expected and desired. It implies that, if during the design of the system, one realises that the set of security levels is too fine-grained, i.e., entities that were supposed to belong to different levels should actually be unified, this can be done without affecting the satisfaction of the policy.

Proposition 1 (soundness of level aggregation)**.** *Let $\langle \mathcal{E}, \lambda \rangle$ be a security model in $\mathscr{L}$ and $\rho : \mathscr{L} \to \mathscr{L}$, a total mapping between security levels (possibly non-injective). If $\langle \mathcal{E}, \lambda \rangle \models \Pi$ then $\langle \mathcal{E}, \rho \circ \lambda \rangle \models \rho(\Pi)$.*

In the above definition $\rho(\Pi)$ is the obvious lifting of ρ to policies, i.e., $\rho(\Pi \cup \Pi') = \rho(\Pi) \cup \rho(\Pi')$, $\rho(\mathbb{A} \overset{\sigma}{\rightsquigarrow} \mathbb{B}) = \rho(\mathbb{A}) \overset{\sigma}{\rightsquigarrow} \rho(\mathbb{B})$ and $\rho(\emptyset) = \emptyset$.

Secondly, we discuss how policies can be related according to the strictness of their requirements. Semantically, a policy Π is less strict than Π', written $\Pi \sqsubseteq \Pi'$, whenever all security models satisfying Π' satisfy also Π. This can be useful in practice when policies need to be compared and possibly simplified. For instance, if two policies have been defined by two different policy designers one would like to know if they are equivalent, or if one is stricter than the other or if they are incomparable. A similar situation can arise if a policy needs to be updated or corrected. In our running example, policy $\Pi_1 \cup \Pi_2$ is less restrictive than Π_1 and Π_2. On the other hand, $\Pi_1 \cup \Pi_2$ and Π_3 are incomparable.

We provide a sound and complete axiomatisation of the "less strict" relation by means of a rule system.

Definition 10 (strictness relation). The strictness relation $\sqsubseteq \subseteq \mathcal{P} \times \mathcal{P}$ is defined as the least transitive and reflexive relation among policies closed under

$$\frac{\Pi \sqsubseteq \Pi' \quad \Pi'_1 \subseteq \Pi_1}{\Pi \cup \Pi_1 \sqsubseteq \Pi' \cup \Pi'_1}\ (\textsc{Ctx}) \qquad \frac{\sigma' \subseteq \sigma}{\mathbb{A} \overset{\sigma'}{\rightsquigarrow} \mathbb{B} \sqsubseteq \mathbb{A} \overset{\sigma}{\rightsquigarrow} \mathbb{B}}\ (\textsc{Constr})$$

$$\frac{\bigcup_{i=1}^{n} (\mathbb{A}_i \times \mathbb{B}_i) = \mathbb{A} \times \mathbb{B}}{\{\mathbb{A}_i \overset{\sigma}{\rightsquigarrow} \mathbb{B}_i \mid i \in \{1, \ldots, n\}\} \sqsubseteq \mathbb{A} \overset{\sigma}{\rightsquigarrow} \mathbb{B}}\ (\textsc{Split})$$

$$\frac{|\mathbb{B}| = 1}{\mathbb{A} \overset{\sigma \cup \{f\}}{\rightsquigarrow} \mathbb{B} \sqsubseteq \mathbb{A} \overset{\sigma}{\rightsquigarrow} \mathbb{B}}\ (\textsc{ConstrF}) \qquad \frac{|\mathbb{A}| = |\mathbb{B}| = 1}{\mathbb{A} \overset{\sigma \cup \{d\}}{\rightsquigarrow} \mathbb{B} \sqsubseteq \mathbb{A} \overset{\sigma}{\rightsquigarrow} \mathbb{B}}\ (\textsc{ConstrD})$$

The intuition of the rules is the following. Rule Ctx says that the strictness relation is preserved under context closure (i.e., if $\Pi \sqsubseteq \Pi'$ then $\Pi \cup \Pi'' \sqsubseteq \Pi' \cup \Pi''$). In addition, the relation is preserved even if the weaker policy Π is embedded in a larger context since the addition of clauses to a policy makes it more permissive. By rule Split a clause $\mathbb{A} \overset{\sigma}{\rightsquigarrow} \mathbb{B}$ can be split into a set of clauses $\{\mathbb{A}_i \overset{\sigma}{\rightsquigarrow} B_i \mid i \in \{1, \ldots, n\}\}$ provided that $\bigcup_{i=1}^{n} (\mathbb{A}_i \times \mathbb{B}_i) = \mathbb{A} \times \mathbb{B}$. This means that for each $i \in \{1, \ldots, n\}$ it must be $\mathbb{A}_i \subseteq \mathbb{A}$ and $\mathbb{B}_i \subseteq \mathbb{B}$. Moreover, for each pair of levels $\mathsf{A} \in \mathbb{A}$ and $\mathsf{B} \in \mathbb{B}$ there must be a clause $\mathbb{A}_i \overset{\sigma}{\rightsquigarrow} \mathbb{B}_i$ such that $\mathsf{A} \in \mathbb{A}_i$ and $\mathsf{B} \in \mathbb{B}_i$. In fact, as shown in detail in the proof of Proposition 2, this ensures that all direct causalities justified by $\mathbb{A} \overset{\sigma}{\rightsquigarrow} \mathbb{B}$ will be justified by one

of the sub-clauses $\mathbb{A}_i \overset{\sigma}{\rightsquigarrow} \mathbb{B}_i$. Rule Constr says that a clause can be relaxed by removing coordination constraints in σ. The last two rules ConstrF and ConstrD capture the fact that some constraints are trivially satisfied when clauses have a special shape. More precisely, given a clause $\mathbb{A} \overset{\sigma}{\rightsquigarrow} \mathbb{B}$ with $|\mathbb{B}| = 1$ the fairness constraint is vacuously satisfied and this motivates rule ConstrF. If, additionally, $|\mathbb{A}| = 1$ the same applies to the directness constraint and this leads to rule ConstrD.

An interesting result is that the strictness relation $\sqsubseteq$ is sound and complete with respect to policy satisfaction.

Proposition 2 ($\sqsubseteq$ is sound and complete)**.** *Let* $\Pi = \langle \mathscr{L}, \rightsquigarrow \rangle, \Pi' = \langle \mathscr{L}, \rightsquigarrow' \rangle$ *be two policies. The following holds*

(i) if $\Pi' \sqsubseteq \Pi$ *then, for all models* $\langle \mathcal{E}, \lambda \rangle$ *in* $\mathscr{L}$, $\langle \mathcal{E}, \lambda \rangle \models \Pi$ *implies* $\langle \mathcal{E}, \lambda \rangle \models \Pi'$*;*

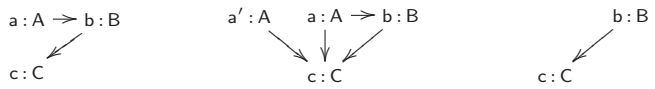
(ii) if for all models $\langle \mathcal{E}, \lambda \rangle$ *in* $\mathscr{L}$, $\langle \mathcal{E}, \lambda \rangle \models \Pi$ *implies* $\langle \mathcal{E}, \lambda \rangle \models \Pi'$*, then* $\Pi' \sqsubseteq \Pi$*.*

By soundness, whenever $\Pi \sqsubseteq \Pi' \sqsubseteq \Pi$, the policies Π and Π' are equivalent, i.e., they are satisfied by exactly the same models. Syntactically different policies can be equivalent because they have clauses that differ for constraints which are vacuously satisfied (see rules ConstrF and ConstrD). Moreover, equivalent policies can differ for the presence of clauses which are useless because they are subsumed by others (e.g., if a policy includes the clauses $\mathbb{A} \overset{\sigma}{\rightsquigarrow} \mathbb{B}$, and $\mathbb{A}' \overset{\sigma}{\rightsquigarrow} \mathbb{B}$, the addition of a clause $\mathbb{A} \cup \mathbb{A}' \overset{\sigma}{\rightsquigarrow} \mathbb{B}$ produces an equivalent policy).

It can be proved that the partial order induced by the preorder $(\mathcal{P}, \sqsubseteq)$ is a finite complete lattice. The (equivalence class of the) policy $\{(\mathsf{A}, \mathscr{C}, \mathsf{A}) \mid \mathsf{A} \in \mathscr{L}\}$ is the top element (the most restrictive policy, which admits only flows within individual security levels), and (the equivalence class of) $\mathscr{L} \times \emptyset \times \mathscr{L}$ is the bottom element (the most permissive policy, which accepts all flows).

The soundness and completeness $\sqsubseteq$ allows one to use the rule system of Def. 10 as the basis of an algorithm to decide whether a policy Π is stricter or equivalent to another policy Π'. Roughly, one just needs to consider a finite part of the lattice $(\mathcal{P}, \sqsubseteq)$ as a graph and check reachability between Π and Π'.

Another property of theoretical and practical interest is the fact that the most restrictive policy satisfied by a security model does not exist in general. In fact, in order to determine the most restrictive policy for a given model a possible algorithm could exploit the rule system of Def. 10 as follows. The algorithm would start with the most permissive policy, allowing for all binary flows, and would keep using the rules right-to-left (by joining or removing clauses, or adding coordination constraints) in order to restrict it until no more rules can be applied without introducing violations in the model. This procedure would eventually stop producing a policy that cannot be further restricted. However, this procedure is non-deterministic and it is not confluent in general. This means non-equivalent policies could be obtained depending on how the rules are chosen and applied. An example that shows that the most restrictive policy for a model is not unique is the following. Consider the security model $\mathcal{M}$ on the left of the figure below.



and the policies $\Pi_1 = (\mathsf{A}, \mathsf{B} \rightsquigarrow \mathsf{C}), (\mathsf{A} \rightsquigarrow \mathsf{B})$ and $\Pi_2 = (\mathsf{A} \rightsquigarrow \mathsf{B}), (\mathsf{B} \rightsquigarrow \mathsf{C})$. Clearly, $\mathcal{M}$ satisfies both Π_1 and Π_2. Such policies are incomparable with respect to $\sqsubseteq$ (none of them can be obtained from the other by removing or joining clauses), and indeed event structures can be found that satisfy one of them but not the other like those at the middle and the right of the figure above. In particular, the event structure in the middle does not satisfy Π_2 since the direct causal dependency $\mathsf{a}' \lessdot \mathsf{c}$ cannot be justified, but the structure does satisfy Π_1 since all direct causalities can be justified. For example, $\mathsf{a}' \lessdot \mathsf{c}$ would be justified by clause $\mathsf{A}, \mathsf{B} \rightsquigarrow \mathsf{C}$ by taking $X = \{\mathsf{a}', \mathsf{b}\}$ and $Y = \{\mathsf{c}\}$. On the other hand, the event structure on the right does clearly satisfy Π_2 but not Π_1, since Π_1 does not have any clause that would justify the only direct causality $\mathsf{b} \lessdot \mathsf{c}$. Hence Π_1 and Π_2 are distinct minimal policies satisfied by model $\mathcal{M}$: the most restrictive policy satisfied by $\mathcal{M}$ does not exist.

5. Verifying policy satisfaction on safe Petri nets

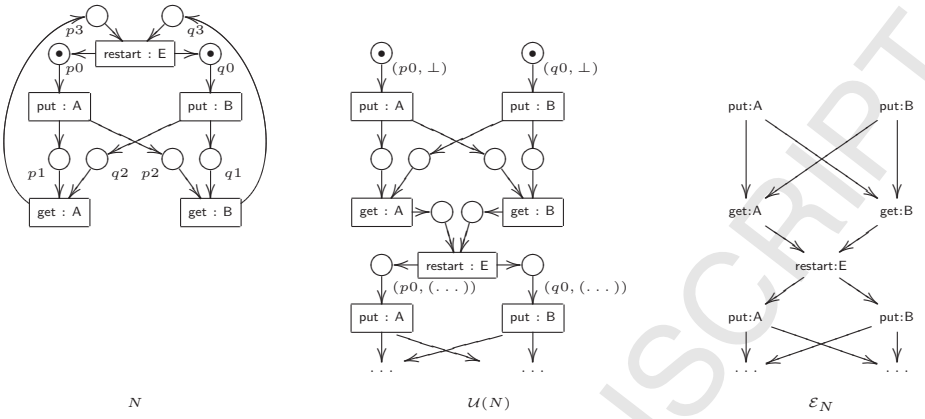
We show here how the abstract framework devised in the previous section can be put at work on a concrete specification formalism, namely Petri nets. Petri nets are a good example to explore applicability of our approach, since they can be endowed with a causal semantics by means of a so-called unfolding construction [2]. More precisely we focus on finite safe Petri nets, and we provide algorithms for deciding satisfaction of policies based on the construction of a suitable finite prefix of the unfolding. As a preparation for the algorithms we first recall the notions of Petri nets and their unfoldings in Section 5.1 and the prefix construction technique in Section 5.2. All in all, Sections 5.2, 5.3 and 5.4 provide an incremental presentation of techniques to solve the policy satisfaction problem.

5.1. Petri nets and unfoldings

We start by recalling Petri nets and their unfolding semantics.

Definition 11 (Petri Net). A *(Petri) net* is a tuple $N = (P, T, F, m_0)$ where P, T are disjoint sets of *places* and *transitions*, respectively, $F : (P \times T) \cup (T \times P) \to \{0, 1\}$ is the *flow function*, and $m_0 \subseteq P$ is the initial marking, i.e., the initial state of the net. A *marking* of N is a function $m : P \to \mathbb{N}$. For $x \in P \cup T$ the *pre-set* and *post-set* are defined ${}^\bullet x = \{y \in P \cup T : F(y, x) = 1\}$ and $x^\bullet = \{y \in P \cup T : F(x, y) = 1\}$, respectively.

In a Petri net, transitions represent events that can occur in the system, while places represent resources. A marking $m : P \to \mathbb{N}$ specifies for each resource $p \in P$, the number $m(p)$ of instances available.



N $\mathcal{U}(N)$ $\mathcal{E}_N$

Figure 7: A Petri net N (left), its unfolding $\mathcal{U}(N)$ and its event structure $\mathcal{E}_N$.

Since Petri nets will be used to represent security models, we assume that some set of security labels $\mathscr{L}$ is fixed and that for all Petri nets $N = (P, T, F, m_0)$, transitions are taken from some set $\mathbb{T}$ for which a labelling over $\mathscr{L}$, $\lambda : \mathbb{T} \to \mathscr{L}$ is fixed. In this way, each transition $t \in T$ will always have an associated security level $\lambda(t)$.

Figure 7 (left) provides a simple Petri net representing a cyclic version of protocol P_3 of our case study. We follow the standard notation, where places are represented by circles, transitions are represented by boxes, the flow function is represented by arrows, and a marking is denoted by bullets inside the places. A transition t such that $\lambda(t) = \mathsf{A}$ is labelled as $t : \mathsf{A}$.

In order to be executed a transition requires the presence of the resources in its pre-set. These are consumed and new instances of the resources in the post-set are produced. Formally, a transition $t \in T$ is *enabled* at a marking m if $m(p) \geq F(p, t)$ for all $p \in P$. An enabled transition t at marking m can *fire* leading to a new marking m' defined by $m'(p) = m(p) + F(t, p) - F(p, t)$ for all places $p \in P$. In this case we write $m[t\rangle m'$. A marking m is *reachable* in N if there is a firing sequence $m_0[t_0\rangle m_1[t_1\rangle \dots [t_n\rangle m$ in N.

A net N is *safe* if for every $p \in P$ and every reachable marking m we have $m(p) \leq 1$. Since all Petri nets considered will be safe, their markings m will be often identified with the corresponding subset of places $\{p \mid m(p) = 1\}$.

The behaviour of a Petri net can be represented by its unfolding $\mathcal{U}(N)$, defined below as an acyclic net constructed inductively starting from the initial marking of N and then adding, at each step, an occurrence of each enabled transition of N. In what follows we indicate by π_1 the projection over the first component of a pair, i.e., $\pi_1(a, b) = a$.

Definition 12 (unfolding). Let $N = (P, T, F, m_0)$ be a safe net. The unfolding is the least net $\mathcal{U}(\mathbf{N}) = (P^u, T^u, F^u, m_0^u)$ such that

- $m_0^u = \{(p, \perp) \mid p \in m_0\} \subseteq P^u$, where $\perp$ is a new element, not in P, T or F denoting that no transition has generated the token.

- if $t \in T$ and $X \subseteq P^u$ is coverable (i.e., some $X' \supseteq X$ is reachable) with $\pi_1(X) = {}^\bullet t$, then $(t, X) \in T^u$;
- for any $e = (t, X) \in T^u$, the set $Z = \{(p, e) : p \in t^\bullet\} \subseteq P^u$; moreover ${}^\bullet e = X$ and $e^\bullet = Z$.

Places and transitions in the unfolding represent tokens and firing of transitions, respectively, of the original net. The projection π_1 over the first component maps places and transitions of the unfolding to the corresponding items of the original net N. Transitions in the unfolding $\mathcal{U}(\mathbf{N})$ inherit the security level from the transition of N. Formally, if $e = (t, X)$ then $\lambda(e) = \lambda(t)$. The initial marking m_0^u consists of the set of minimal places. For historical reasons transitions and places in the unfolding are also called *events* and *conditions*, respectively.

Figure 7 provides the unfolding (center) of a net (left) and the corresponding event structure (right). Again, for a lighter notation, only some places and transitions are named, and only part of the names is included.

The events of the unfolding of a finite safe net, endowed with suitably defined relations of causality and conflict, form an event structure.

Lemma 1 (event structure for finite safe nets)**.** *Let N be a finite safe Petri net and let its unfolding be $\mathcal{U}(\mathbf{N}) = (P^u, T^u, F^u, m_0^u)$. Let causality $\leq$ be defined as the transitive and reflexive closure of the flow relation, i.e., the least reflexive and transitive relation such that $t \leq t'$ whenever $t^\bullet \cap {}^\bullet t' \neq \emptyset$. Let conflict be the least relation such that $e \# e'$ if ${}^\bullet e \cap {}^\bullet e' \neq \emptyset$ and inherited along causality, i.e., if $e \# e' \leq e''$ then $e \# e''$. Then $\mathcal{E}(N) = (T^u, \leq, \#)$ is an event structure.*

Abusing the notation, we will often use the unfolding in place of the underlying event structure. For instance, we will write $\mathcal{C}(\mathcal{U}(N))$ to refer to the configurations of $\mathcal{E}(N)$. Also observe, that $\mathcal{E}(N)$ endowed with the labelling λ of the events in t^u over the set of security levels $\mathscr{L}$ is a causal security model.

The transitions of a configuration $C \in \mathcal{C}(\mathcal{U}(N))$ can be fired in the unfolding $\mathcal{U}(N)$ in any order compatible with causality, producing a marking $C^\circ = (m_0^u \cup \bigcup_{t \in C} t^\bullet) \setminus (\bigcup_{t \in C} {}^\bullet t)$; in turn, this marking corresponds to a marking in the original net N given by $\mathsf{M}(C) = \pi_1(C^\circ) \subseteq P$.

5.2. Verification using finite prefixes

The unfolding of a net provides a concise and faithful representation of concurrency (as opposed to the marking graph, i.e., the transition system over markings). The main drawback is that, in the presence of cyclic behaviours, the unfolding is infinite. On the other hand, many properties of interest can be verified by exploring only a finite initial part (so called-prefix) of the unfolding.

For example, the finite prefix in Figure 8a is enough if we are just interested in the reachability of markings in the net. Indeed, the restart transition does not need to be included in the prefix as it just generates the initial marking, which is already present. Such a prefix can be obtained by applying the approach of [13] for the construction of a complete finite prefix of the unfolding. Roughly

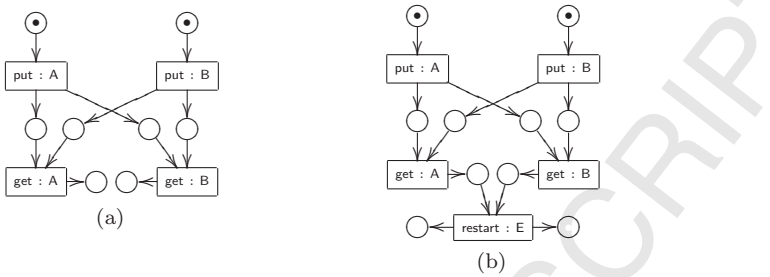


Figure 8: Finite prefixes for the unfolding of net N of Figure 7: (a) a generic prefix, (b) a prefix based on marking equality.

speaking, the main ingredient of that approach is an equivalence relation over configurations that equates configurations providing the same information with respect to the property of interest. In the case of marking reachability, we can just consider the equivalence induced by $\mathsf{M}(C) = \mathsf{M}(C')$, i.e. two configurations are equivalent if they produce the same markings on the original net. Then the prefix construction works by inductively generating the unfolding, adding at each step an enabled event e with minimal causal history and stopping at so-called cut-offs, i.e., events whose causal history is equivalent to that of an already added event. Note that in our example the only event that could be added is an instance of restart but this would result in a configuration identical to the initial one (the empty configuration). The results in [13] guarantee that such prefix construction process terminates if the equivalence is of finite index. Clearly, $\mathsf{M}(C) = \mathsf{M}(C')$ is of finite index since the set of markings of any safe Petri net is finite.

Of course, other properties of interest may require finer equivalence relations that allow to distinguish configurations not just on the marking generated but also, for example, on properties of their history. For example, if we are interested in tracing the dependencies between security levels we could define two configurations to be equivalent if they generate the same marking m and, in addition, the places in m have direct dependencies with the same security levels. In our example, such an equivalence relation would generate the finite prefix in Figure 8b, and it would allow us to decide properties of the kind "is it possible to fire transition t as a direct consequence of having fired a transition of level A?". Clearly, finer equivalences lead to larger prefixes. For instance, in this case we need to include an instance of the restart transition: even if it produces a marking already represented (the initial marking) this second instance of the marking has a direct dependency from level E while the first instance has none. However, as long as the equivalence is of finite index, the construction is guaranteed to terminate [13].

Our verification techniques will be based on the above sketched approach. In particular, we will define a suitable equivalence relation for the verification

problem at hand and we will show that such relation has the desired properties as required in [13], i.e., it is preserved by extension, of finite index and it allows to decide the property of interest.

5.3. Verification of many-to-one policies

We first consider the the verification of many-to-one policies, namely policies only containing clauses of the form $\mathbb{A} \overset{\sigma}{\rightsquigarrow} \mathsf{B}$. This restriction leads to a simpler and more efficient algorithm than that for general policies, discussed later in Section 5.4.

We consider an equivalence on configurations similar to the one informally discussed in the previous section. It is based on an enrichment of the markings with indications about the levels they directly and indirectly depend on.

For the example of this section and assuming a policy Π that regulates flows between A, B and E, the finite prefix built using such equivalence is depicted in Figure 9. Note that we need to unfold the net one "round" more. The reason is that the first instance of the initial marking has no dependencies, the second depends directly on levels A and B, while the third one depends directly on levels A and B and indirectly on level E. Further extensions are not needed since they would produce configurations equivalent to other already in the prefix.

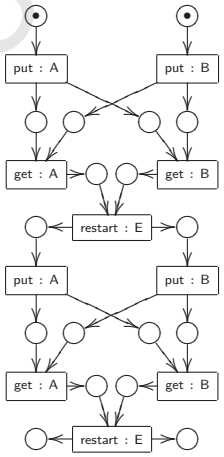


Figure 9: A finite prefix for the unfolding of net N of Figure 7 based on level markings.

In order to define such equivalence formally, we introduce some notation. We denote below by 2_1^X the set of subsets of X of cardinality less or equal 1, i.e., singletons and the empty set. Moreover, we assume that the components of a safe Petri net and of its unfolding are $N = (P, T, F, m_0)$ and $\mathcal{U}(N) = (P^u, T^u, F^u)$. Given a configuration $C \in \mathcal{C}(\mathcal{U}(N))$ and a place $p \in \mathsf{M}(C)$ of the net N, we denote by $\pi_C^{-1}(p)$ the unique condition such that $\pi_1(\pi_C^{-1}(p)) = p$.

Configurations are deemed as equivalent if they have the same level marking, which is a marking enriched with information regarding the history of tokens.

Definition 13 (level marking). Let N be a safe Petri net, $\mathcal{U}(N)$ be its unfolding and $C \in \mathcal{C}(\mathcal{U}(N))$. We define the *level marking* of C as the structure $\mathsf{M}^*(C) = (\mathsf{M}(C), \lambda_d^C, \lambda_i^C, \leq_C)$ where

- $\lambda_d^C : \mathsf{M}(C) \to 2_1^{\mathscr{L}}$ is a function defined by $\lambda_d^C(p) = \lambda(\pi_1({}^\bullet\pi_C^{-1}(p)))$.
- $\lambda_i^C : \mathsf{M}(C) \to 2^{\mathscr{L}}$ is a function defined by $\lambda_i^C(p) = \{\lambda(e') \mid e' \leq \pi_C^{-1}(p))\}$.
- $\leq_C$ is a partial order on $\mathsf{M}(C)$ defined by $p \leq_C p'$ if ${}^\bullet\pi_C^{-1}(p) = \{e\}$ and ${}^\bullet\pi_C^{-1}(p') = \{e'\}$ and $e \leq e'$.

We say that two configurations $C, C' \in \mathcal{C}(\mathcal{U}(N))$ are *level marking equivalent*, written $C \approx C'$, when $\mathsf{M}^*(C) = \mathsf{M}^*(C')$.

The level marking $\mathsf{M}^*(C)$ of a configuration C is the marking $\mathsf{M}(C)$ of C enriched with information concerning the direct and indirect causes of the tokens. More precisely, the function λ_d^C maps each place p of the marking $\mathsf{M}(C)$ to the security level of its generator (or to the empty set when p is in the initial marking). The function λ_i^C maps each p in the marking to the set of security levels of events in the history of p, hence of all direct or indirect causes. Finally, $\leq_C$ is the projection on the tokens of the causal order on the generator events: $p \leq_C p'$ when the event $\pi_C^{-1}(p)$ generating p is a cause of the event $\pi_C^{-1}(p')$generating p'. This information is used, when extending a configuration, in order to understand which are the direct causes of the added event.

As an example, Figure 10 depicts the level marking for the minimal configuration that contains one occurrence of $\mathsf{put} : \mathsf{A}$ (top), and any minimal configuration that contains two or more occurrences of $\mathsf{put} : \mathsf{A}$ (bottom). The fact that having two or more occurrences of $\mathsf{put} : \mathsf{A}$ does not alter the level marking explains why, under the equivalence $\approx$, the prefix in Figure 9 does not contain configurations with more than two occurrences of $\mathsf{put} : \mathsf{A}$.

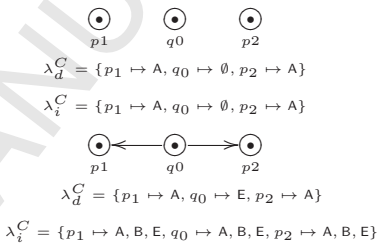


Figure 10: Two level markings for the unfolding of net N of Figure 7.

According to [13], in order to conclude that the generation of a prefix outlined above terminates and produces a complete prefix for $\approx$, we need to show that $\approx$ is of finite index and that it is preserved by extension, namely given $C, C' \in \mathcal{C}(E)$, $C \approx C'$ and $C \to C \oplus e$ then $C' \to C' \oplus e'$ for some $e' \in E$ and $C \oplus e \approx C' \oplus e'$.

Lemma 2 (Preservation and finiteness)**.** *Let N be a safe Petri net, $\mathcal{U}(N)$ be its unfolding. The equivalence $\approx$ is preserved by extension and is of finite index.*

In view of Lemma 2, according to [13], given a finite safe Petri net N we can construct a finite prefix U of the unfolding $\mathcal{U}(N)$, which is complete with respect to $\approx$, namely such that for every $C \in \mathcal{C}(\mathcal{U}(N))$ there exists a $\approx$-equivalent configuration $C' \in \mathcal{C}(U)$.

We conclude by showing that policy satisfaction can be checked on prefix U since for any witness of a violation in $\mathcal{U}(N)$ we can find an equivalent witness in U. Formally, given two events $e_0, e_1 \in T^u$ such that $e_0 \lessdot e_1$ we say that e_0 and e_1 witness a violation of the policy Π if all $\mathbb{A} \overset{\sigma}{\rightsquigarrow} \mathsf{B} \in \Pi$ and all $X \subseteq T^u$ it holds that $(X, \{e_1\})$ does not conform to $\mathbb{A} \overset{\sigma}{\rightsquigarrow} \mathsf{B}$.

We first observe that the level marking $\mathsf{M}^*(C)$ contains sufficient information to determine whether a direct causality for an event enabled by the marking $\mathsf{M}(C)$ is justified by some clause.

Lemma 3 (Justification)**.** *Let Π be a policy and $N = (P, T, F)$ be a safe Petri net. Let e_0, e_1 be events such that $e_0 \lessdot e_1$, with $\lambda(e_1) = \mathsf{B}$. Let $C \in \mathcal{C}(\mathcal{U}(N))$ be a configuration that can be extended with e_1, i.e., $C \oplus e_1 \in \mathcal{C}(\mathcal{U}(N))$. Then*

- $e_0 \lessdot e_1$ *is justified by a clause* $\mathbb{A} \rightsquigarrow \mathsf{B}$ *iff* $\mathbb{A} \subseteq \bigcup\{\lambda_i^C(p) \mid p \in {}^\bullet\pi_1(e_1)\}$*;*
- $e_0 \lessdot e_1$ *is justified by a clause* $\mathbb{A} \overset{d}{\rightsquigarrow} \mathsf{B}$ *iff* $\mathbb{A} \subseteq \bigcup\{\lambda_d^C(p) \mid p \in {}^\bullet\pi_1(e_1) \wedge p$ *is* $\leq_C$*-maximal* $\}$

By the lemma above we immediately deduce the completeness of the prefix.

Corollary 1 (Completeness)**.** *Let* Π *be a policy,* N *be a safe Petri net,* $\mathcal{U}(N)$ *be its unfolding,* $U = (P', T', F')$ *be the complete prefix with respect to* $\approx$*, and* $e_0, e_1 \in T^u$ *be events that witness a violation of policy* Π*. Then there are events* e_0', e_1' *in* T' *such that* $\lambda(e_0) = \lambda(e_0')$ *and* $\lambda(e_1) = \lambda(e_1')$ *that witness a violation of* Π*.*

The above results immediately suggest an algorithm for checking many-to-one policy satisfaction: (1) generate the finite prefix U according to the equivalence relation $\approx$, and (2) for each potential violation $e_0 \lessdot e_1$ in U check whether some clause $\mathbb{A} \overset{\sigma}{\rightsquigarrow} \mathsf{B}$ in Π justifies $e_0 \lessdot e_1$, using Lemma 3. The complexity of such procedure is polynomial in the size of the prefix U and the number of clauses in Π. The size of the prefix, in the worst case, is exponential in the number of markings of the original net N, since markings are "annotated" with subsets of levels. Concretely, thanks to the compact representation provided by the branching structure of the unfolding, it can be much smaller. The number of clauses in Π also introduces a factor, but in practical cases such number can be assumed to be asymptotically smaller than the system.

5.4. Extension to many-to-many policies

We now consider the general case of many-to-many policies. Recall that, in order to use a clause $\mathbb{A} \rightsquigarrow \mathbb{B}$ to justify a direct causal dependency, we need to find sets $X : \mathbb{A}$ and $Y : \mathbb{B}$ such that each element in X is a cause of each event in Y. The level marking $\mathsf{M}^*(C)$ that provides the security levels of the causes of each token is not sufficient for this purpose. E.g., if $Y = \{\mathsf{A}, \mathsf{B}\}$, given events $e : \mathsf{A}$ and $e' : \mathsf{B}$, relying on the level marking we can deduce that there are $X \subseteq E$ such that $X : \mathbb{A}$ and $X < e$ and $X' \subseteq E$ such that $X' : \mathbb{B}$ and $X' < e'$. However, since the level marking does not track the identity of the causes but only records their security level, there is no way of ensuring that $X = X'$. We need to consider a finer equivalence. The one we propose relies on what we call the *shadow* of a configuration.

Definition 14 (shadow equivalence). Let N be a safe Petri net, $\mathcal{U}(N)$ be its unfolding, $C \in \mathcal{C}(\mathcal{U}(N))$ and $X \subseteq C$. We define the *shadow* of X in C as the structure $sh(X, C) = \langle \mathsf{M}(C), \lambda_d^C, \lambda_i^C, \leq_C \rangle$ where

- $\lambda_d^C : \mathsf{M}(C) \to 2_1^{\mathscr{L}}$ is defined by $\lambda_d^C(p) = \lambda(\pi_1({}^\bullet\pi_C^{-1}(p) \cap X))$.
- $\lambda_i^C : \mathsf{M}(C) \to 2^{\mathscr{L}}$ is defined by $\lambda_i^C(p) = \{\lambda(e') \mid e' \in X \wedge e' \leq \pi_C^{-1}(p))\}$.
- $\leq_C$ is a partial order on $\mathsf{M}(C)$ defined by $p \leq_C p'$ if ${}^\bullet\pi_C^{-1}(p) = \{e\}$, ${}^\bullet\pi_C^{-1}(p') = \{e'\}$ and $e \leq e'$.

For a fixed policy Π, the Π-*shadow set* of C is defined as the set $SH_\Pi(C) = \{sh(X, C) \mid \exists \mathbb{A} \overset{\sigma}{\rightsquigarrow} \mathbb{B} \in \Pi.\ X : \mathbb{A}\}$. We say that C and C' are Π-*shadow equivalent*, denoted $C \approx_\Pi C'$, if $SH_\Pi(C) = SH_\Pi(C')$.

Intuitively, the shadow of $X \subseteq C$ is a specialised level marking where we focus on how the subset of events X influences the current marking. Then for a fixed policy Π, we consider two configurations equivalent whenever the set of shadows induced by subsets X that cover the left-hand side of some clause are identical.

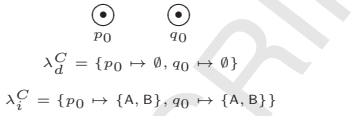


Figure 11: A shadow set for the unfolding of net N of Figure 7.

As an example, consider the the policy Π_4 of Figure 5, which just contains the clause $\{\mathsf{A}, \mathsf{B} \overset{d.f}{\rightsquigarrow} \mathsf{A}, \mathsf{B}\}$ and consider the running example of this section. Figure 11 illustrates the shadow set of all minimal configurations containing one or more instances of restart, which generates instances of the initial marking.

This explains the finite prefix illustrated in Figure 12, that we obtain with the shadow equivalence. Note that adding further instances of restart would not lead to newer configurations with respect to the shadow equivalence. Indeed all of them have the same shadow set illustrated in Figure 11.

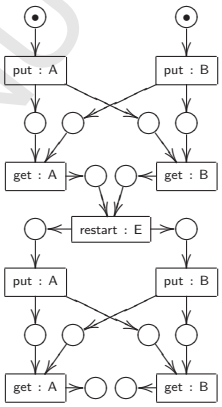


Figure 12: A finite prefix for the unfolding of net N of Figure 7 based on shadow markings.

As in the previous case, according to the theory in [13], in order to conclude that the generation of a prefix outlined above terminates and produces a complete prefix for $\approx_\Pi$, we need to show that $\approx_\Pi$ is of finite index and that it is preserved by extension.

We start with a simple but fundamental observation: shadow markings can be defined incrementally.

Lemma 4 (shadow transitions)**.** *Let $C \in \mathcal{C}(\mathcal{U}(N))$, and let e be an event such that $C \oplus e \in \mathcal{C}(\mathcal{U}(N))$. Then the k-shadow set $SH_k(C \oplus e)$ only depends on $SH_k(C)$ and $t = \pi_1(e)$. In fact, for any $X \subseteq C \oplus e$, $|X| \leq k$, with λ injective on X, we have that $sh(X, C \oplus e) = \langle \mathsf{M}(C \oplus e), \lambda_d^{C\oplus e}, \lambda_i^{C\oplus e}, \leq_{C\oplus e} \rangle$ where*

- $\mathsf{M}(C \oplus e) = (\mathsf{M}(C) \setminus {}^\bullet t) \cup t^\bullet$;
- $\lambda_d^{C\oplus e}(p) = \begin{cases} \lambda_d^C(p) & \text{if } p \notin t^\bullet \\ \lambda(t) & \text{if } p \in t^\bullet \end{cases}$
- $\lambda_i^{C\oplus e}(p) = \begin{cases} \lambda_i^C(p) & \text{if } p \notin t^\bullet \\ \{\lambda(x) \in X \mid x \leq e\} & \text{if } p \in t^\bullet \end{cases}$

- *for* $p, p' \in \mathsf{M}(C \oplus e)$, *we have* $p \leq_{C\oplus e} p'$ *if* $\begin{cases} p, p' \notin t^\bullet \text{ and } p \leq_C p', \text{or} \\ p' \in t^\bullet \text{ and } \exists p'' \in {}^\bullet t.\ p \leq_C p'' \end{cases}$

The observation of Lemma 4 is crucial to show preservation by extension of the equivalence relation $\approx_\Pi$. For finiteness, observe that the number of different shadows is finite as it follows from finiteness of the the number of markings finite (the net is safe) and of the set of security levels $\mathscr{L}$. Then one can conclude by using that fact that the policy Π consists of a finite number of clauses.

Lemma 5 (Preservation and finiteness)**.** *Let N be a safe Petri net, $\mathcal{U}(N)$ be its unfolding, and Π be a policy. The equivalence $\approx_\Pi$ is preserved by extension and is of finite index.*

In view of Lemma 5, according to [13], given a finite safe Petri net N we can construct a finite prefix U of the unfolding $\mathcal{U}(N) = (P^u, T^u, F^u)$, which is complete with respect to $\approx_\Pi$, namely for every $C \in \mathcal{C}(\mathcal{U}(N))$ there exists a $\approx_\Pi$-equivalent configuration $C' \in \mathcal{C}(U)$. We can conclude that the prefix U contains a witness of any violation of the policy.

Lemma 6 (Completeness)**.** *Let Π be a policy, N be a safe Petri net, $\mathcal{U}(N)$ be its unfolding, $U = (P', T', F')$ be the finite complete prefix respect to $\approx_\Pi$, and $e_0, e_1 \in T^u$ be events that witness a violation of the policy Π. Then there are events $e_0', e_1' \in T'$ such that $\lambda(e_0) = \lambda(e_0')$ and $\lambda(e_1) = \lambda(e_1')$ that witness a violation of Π.*

Once a finite complete prefix is available, we are still left with the problem of checking whether the direct causalities in the prefix are justified or not. This is considerably more difficult than in the restricted case of many-to-one policies.

Given a direct causality $e_0 \lessdot e_1$ and a clause $c = \mathbb{A} \overset{\sigma}{\rightsquigarrow} \mathbb{B}$, whether $e_0 \lessdot e_1 \models c$ can be checked as follows. Consider the configuration $[e_1]$ and define a notion of enriched marking that records (i) the shadows of subsets X such that $X : \mathbb{A}$; (ii) for each such X the elements in $\mathbb{B}$ which have been caused, directly and indirectly, by X (note that $\lambda(e_1)$ will be always present). More precisely, we map each configuration $C \supseteq [e_1]$ to a modified shadow set $SH(C)^{e_1,\mathbb{B}} = \{\langle sh(X,C), D(X,C), I(X,C)\rangle \mid X \subseteq [e_1] \wedge X : \mathbb{A}\}$ where $D(X,C) = \mathbb{B} \cap \{\lambda(y) \mid y \in C \ \wedge\ X \lessdot y\}$ and $I(X,C) = \mathbb{B} \cap \{\lambda(y) \mid y \in C \ \wedge\ X < y\}$ are the sets of levels in $\mathbb{B}$ caused directly and indirectly by X. Lemma 4 can be easily adapted to show that also the evolution of this modified shadow set only depends on the current shadow set and on the enabled transitions. Therefore the problem of verifying whether $e_0 \lessdot e_1$ is justified by the clause $c = \mathbb{A} \overset{\sigma}{\rightsquigarrow} \mathbb{B}$ reduces to the success (for at least one of the candidate sets X) of the following CTL model-checking problem on the finite state transition system over such modified shadow sets:

- if $\sigma = \emptyset$, the *possibility* ($\exists\Diamond$) of reaching C such that $I(X,C) : \mathbb{B}$;
- if $\sigma = \{d\}$, the *possibility* ($\exists\Diamond$) of reaching C such that $D(X,C) : \mathbb{B}$;
- if $\sigma = \{f\}$, the *necessity* ($\forall\Diamond$)of reaching C such that $I(X,C) : \mathbb{B}$ under the fairness constraint that every enabled transition is eventually executed;

- if $\sigma = \{d, f\}$, the *necessity* ($\forall\Diamond$) of reaching C such that $D(X, C) : \mathbb{B}$ under the fairness constraint that every enabled transition is eventually executed.

The complexity of the verification procedure for many-to-many policies is higher than in the case of many-to-one policies, mainly due to two factors. First, the size of the finite prefix needed to spot potential flows may be larger since we need to consider all shadow sets for all possible candidate X and not just the direct and indirect causal dependencies on levels. Fortunately, the requirement on X to be well-typed can mitigate the problem in practical cases, where the number of different sets of levels used in the left-hand side of clauses can be expected to be low or, at least, asymptotically smaller than the net under study. Second, we need to perform nested explorations for every potential flow discovered. Such nested explorations require a number of fair CTL model checking problems.

6. Related Work

Non-interference. Our causality-based notion of security is related to information flow security based on non-interference for concurrent systems (see e.g. [14, 15, 16]). The key differentiating factor resides in the semantic model: non-interference is usually based on interleaving semantics of systems which is normally coarser than true concurrent semantics. The paradigmatic example is given by processes $a \mid b$ and $a.b + b.a$ which are equated in an interleaving world but are different from a causal point of view. Hence one would expect our notion of security, when confined to binary policies, to be more restrictive than those based on interleaving semantics. However, this is not always the case mainly because observational non-interference approaches capture flows of information due to conflicts (i.e., branching), which are instead not considered in our notion of security. A generalisation of our work dealing with conflicts could be shown to produce a refinement of observational intransitive non-interference [17], also in the multi-level case [18]. There are also works that consider causality-based notions of non-interference. For instance, the authors of [19] study decidability of several non-interference properties for message-sequence charts, a true concurrent model. For a deeper discussion on non-interference we refer to [4].

Declassification. The intransitive nature of our policies allows us to specify certain forms of declassification. In terms of classical taxonomies of declassification in [10, 11], our approach can be related to the *what* dimension in [11] (i.e. the objects/events being declassified; sets of objects/events in our case) and the *where* (in the policy) declassification occurs in [10].

In the setting of Petri nets some works have been done in so-called *selective declassification* [17], where a downgrading action can declassify a subset of the high-level actions. The work in [20] shows that checking intransitive non-interference (namely, INI) under such selective declassifications is decidable even

for infinite state systems. Our approach could be used to model similar declassification policies by considering the set of security levels $\mathcal{L}$ to be $2^{\mathbb{H}} \cup \mathbb{D} \cup \mathsf{L}$, where $\mathbb{H}$ and $\mathbb{D}$ are respectively sets of high and downgrading actions, and L is just the low level. In other words, the levels under consideration are all sets of high level actions, single downgrading actions and the set of all low level actions. A selective downgrading policy in our approach would then have the shape $\mathsf{H}_1, \ldots, \mathsf{H}_n \rightsquigarrow \mathsf{D}_i \rightsquigarrow \mathsf{L}$, where d_i declassifies just a subset of the high level actions.

Another related model is that of *Disjunction Category Labels* (DLC) [21], an approach that allows to label data with confidentiality and integrity policies represented as pairs $\langle C, I\rangle$ of boolean proposition over a set of system entities, essentially representing valid subsets of entities. Information flow policies in DLC are of the form $\langle C, I\rangle \rightsquigarrow \langle C', I'\rangle$, some of which are always granted (roughly, removing readers / adding writers) while other flows are considered as downgradings. Controlled forms of downgrading are based on privileges, possibly involving the context of execution as in the bounded and robust privileges of [22]. In our setting, a clause $\mathsf{A}_1 \ldots \mathsf{A}_n \rightsquigarrow \mathsf{B}_1 \ldots \mathsf{B}_m$ is analogous to a DLC downgrading clause $\langle \mathsf{A}_1 \vee \cdots \vee \mathsf{A}_n, \mathsf{A}_1 \wedge \cdots \wedge \mathsf{A}_n\rangle \rightsquigarrow \langle \mathsf{B}_1 \vee \cdots \vee \mathsf{B}_m, \mathsf{B}_1 \vee \cdots \vee \mathsf{B}_m\rangle$, i.e. data can be read by any of A_i and written by all of A_i can flow into data that can be read and written by any of B_i.

Logic-based Policy Languages. Logic-based languages have been investigated as flexible formalisms to express security properties. We mention here, e.g., *hyperproperties* [23] and their logics (e.g. [24, 25]). Our policies can be easily integrated in logic-based languages to obtain a richer policy language. For instance, the simplest solution is to combine our policies by using standard propositional operators (at present, conceptually, our policies are disjunction of clauses, whose left- and right-hand sides are interpreted as conjunctions).

Another example is Paralocks [26], a language for specifying role-based information flow policies. In Paralocks a data item x can be annotated with clauses of the form $\Sigma \to a$, specifying the conditions Σ under which x can flow into actor a. Such policies can be related to many-to-one policies in our approach, i.e., of the form $\mathsf{A}_1 \ldots \mathsf{A}_m \rightsquigarrow \mathsf{B}_1$. It is less clear how many-to-many policies, i.e., policies stating that data can flow to an agent a if it also flows to an agent b, could be expressed in Paralocks.

In addition to MSO for event structures [6], our work can be related to other logics motivated by causality. An example is *Event Order Logic* (EOL) of [27], a logic based language used to specify boolean conditions on the occurrence of events, with applications to safety analysis. EOL is inspired by Linear-time Temporal Logic (LTL) and introduces ad-hoc operators to specify the order of events. The more relevant operator is $\psi_1 \mathbin{\backslash\!\!\backslash} \psi_2$ which allows one to express that the events described by ψ_1 occur before those described by ψ_2. A key difference with our approach is that EOL focuses on (counterfactual) causality between instances of events that are observationally equivalent instead of the traditional notion of causality between individual events in a concurrent system.

7. Conclusion

We have presented a novel approach to many-to-many information flow policies that allows one to specify patterns of coordination among several security levels. In particular, each clause in a policy can specify that a certain set of levels is allowed to cooperate to provide the flow *collectively*, and possibly *simultaneously* to another set of security levels. We believe that the approach can turn useful in several scenarios including secret exchange protocols, security architectures and systems with controlled forms of declassification. We have investigated decidability results and verification techniques for safe Petri nets. Moreover, we have discussed the relation to some traditional notions of security, including observational non-interference and language-based security. We are currently investigating extensions of our work in several directions, including how to suitably deal with indirect flows of information due to conflicts, finer notions of flow that would distinguish implicit flows (e.g. due to data dependencies) from explicit flows (e.g. due to synchronisations or control), and how to deal with the transfer of specific values.

References

[1] Selective disclosure and insider trading, Tech. rep., U.S. Securities and Exchange Commission (SEC) (August 2000).
URL `https://www.sec.gov/rules/final/33-7881.htm`

[2] M. Nielsen, G. D. Plotkin, G. Winskel, Petri nets, event structures and domains, in: G. Kahn (Ed.), Proceedings of the International Symposium on the Semantics of Concurrent Computation, Vol. 70 of LNCS, Springer, 1979, pp. 266–284.

[3] G. Winskel, Event structures, in: W. Brauer, W. Reisig, G. Rozenberg (Eds.), Petri Nets: Central Models and Their Properties, Advances in Petri Nets 1986, Part II, Proceedings of an Advanced Course, Vol. 255 of LNCS, Springer, 1986, pp. 325–392.

[4] P. Baldan, A. Beggiato, A. Lluch-Lafuente, Many-to-many information flow policies, in: J. Jacquet, M. Massink (Eds.), 19th IFIP WG 6.1 International Conference on Coordination Models and Languages, Vol. 10319 of Lecture Notes in Computer Science, Springer, 2017, pp. 159–177.

[5] P. S. Thiagarajan, Regular event structures and finite Petri nets: A conjecture, in: Formal and Natural Computing, Vol. 2300 of LNCS, Springer, 2002, pp. 244–256.

[6] P. Madhusudan, Model-checking trace event structures, in: Proceedings of LICS 2013, IEEE Computer Society, 2003, pp. 371–380.

[7] R. C. Merkle, Secure communications over insecure channels, Commun. ACM 21 (4) (1978) 294–299.

[8] T. Okamoto, K. Ohta, How to simultaneously exchange secrets by general assumptions, in: D. E. Denning, R. Pyle, R. Ganesan, R. S. Sandhu (Eds.), CCS '94, Proceedings of the 2nd ACM Conference on Computer and Communications Security, ACM, 1994, pp. 184–192.

[9] A. Sabelfeld, A. C. Myers, Language-based information-flow security, IEEE Journal on Selected Areas in Communications 21 (1) (2003) 5–19.

[10] H. Mantel, D. Sands, Controlled declassification based on intransitive noninterference, in: W. Chin (Ed.), Proceedings of APLAS 2004, Vol. 3302 of LNCS, Springer, 2004, pp. 129–145.

[11] A. Sabelfeld, D. Sands, Declassification: Dimensions and principles, Journal of Computer Security 17 (5) (2009) 517–548.

[12] E. Fujisaki, T. Okamoto, How to enhance the security of public-key encryption at minimum cost, in: H. Imai, Y. Zheng (Eds.), Proceedings of PKC 1999, Vol. 1560 of LNCS, Springer, 1999, pp. 53–68.

[13] V. Khomenko, M. Koutny, W. Vogler, Canonical prefixes of Petri net unfoldings, Acta Informatica 40 (2003) 95–118.

[14] R. Focardi, R. Gorrieri, A taxonomy of security properties for process algebras, Journal of Computer Security 3 (1) (1995) 5–34.

[15] R. Focardi, R. Gorrieri, Classification of security properties (part I: information flow), in: R. Focardi, R. Gorrieri (Eds.), Proceedings of FOSAD 2000, Vol. 2171 of LNCS, Springer, 2000, pp. 331–396.

[16] N. Busi, R. Gorrieri, Structural non-interference in elementary and trace nets, Mathematical Structures in Computer Science 19 (6) (2009) 1065–1090.

[17] R. Gorrieri, M. Vernali, On intransitive non-interference in some models of concurrency, in: A. Aldini, R. Gorrieri (Eds.), Proceedings of FOSAD 2010, Vol. 6858 of LNCS, Springer, 2011, pp. 125–151.

[18] P. Baldan, A. Beggiato, Multilevel transitive and intransitive noninterference, causally, in: A. Lluch Lafuente, J. Proença (Eds.), Proceedings of COORDINATION 2016, Vol. 9686 of LNCS, Springer, 2016, pp. 1–17.

[19] B. Bérard, L. Hélouët, J. Mullins, Non-interference in partial order models, ACM Trans. Embedded Comput. Syst. 16 (2) (2017) 44:1–44:34.

[20] E. Best, P. Darondeau, Deciding selective declassification of Petri nets, in: P. Degano, J. D. Guttman (Eds.), Proceedings of POST 2012, Vol. 7215 of LNCS, Springer, 2012, pp. 290–308.

[21] D. Stefan, A. Russo, D. Mazières, J. C. Mitchell, Disjunction category labels, in: P. Laud (Ed.), Proceedings of NordSec 2011, Vol. 7161 of LNCS, Springer, 2011, pp. 223–239.

[22] L. Waye, P. Buiras, D. King, S. Chong, A. Russo, It's my privilege: Controlling downgrading in dc-labels, in: S. Foresti (Ed.), Proceedings of STM 2015, Vol. 9331 of LNCS, Springer, 2015, pp. 203–219.

[23] M. R. Clarkson, F. B. Schneider, Hyperproperties, Journal of Computer Security 18 (6) (2010) 1157–1210.

[24] M. R. Clarkson, B. Finkbeiner, M. Koleini, K. K. Micinski, M. N. Rabe, C. Sánchez, Temporal logics for hyperproperties, in: M. Abadi, S. Kremer (Eds.), Proceedings of POST 2014, Vol. 8414 of LNCS, Springer, 2014, pp. 265–284.

[25] D. Milushev, D. Clarke, Towards incrementalization of holistic hyperproperties, in: P. Degano, J. D. Guttman (Eds.), Proceedings of POST 2012, Vol. 7215 of LNCS, Springer, 2012, pp. 329–348.

[26] N. Broberg, D. Sands, Paralocks: role-based information flow control and beyond, in: M. V. Hermenegildo, J. Palsberg (Eds.), Proceedings of POPL 2010, ACM, 2010, pp. 431–444.

[27] F. Leitner-Fischer, S. Leue, Probabilistic fault tree synthesis using causality computation, IJCCBS 4 (2) (2013) 119–143.

Appendix A. Proofs

Proposition 1 (soundness of level aggregation)**.** *Let $\langle \mathcal{E}, \lambda\rangle$ be a security model in $\mathscr{L}$ and $\rho : \mathscr{L} \to \mathscr{L}$, a total mapping between security levels (possibly non-injective). If $\langle \mathcal{E}, \lambda\rangle \models \Pi$ then $\langle \mathcal{E}, \rho \circ \lambda\rangle \models \rho(\Pi)$.*

Proof. The key observation is that given two sets of events $X, Y \subseteq E$ such that (X, Y) conforms to $\mathbb{A} \overset{\sigma}{\rightsquigarrow} \mathbb{B}$ in $\langle \mathcal{E}, \rho\rangle$ then (X, Y) conforms to the translated clause $\rho(\mathbb{A}) \overset{\sigma}{\rightsquigarrow} \rho(\mathbb{B}) \in \rho(\Pi)$ in $\langle \mathcal{E}, \rho \circ \lambda\rangle$.

Now, if $\langle \mathcal{E}, \lambda\rangle \models \Pi$ then we know that for all $e, e' \in E$ such that $e \lessdot e'$ there exists some clause $\mathbb{A} \overset{\sigma}{\rightsquigarrow} \mathbb{B} \in \Pi$ such that $e \lessdot e' \models \mathbb{A} \overset{\sigma}{\rightsquigarrow} \mathbb{B}$. By the first observation, the pairs (X, Y) used to show that $e \lessdot e' \models \mathbb{A} \overset{\sigma}{\rightsquigarrow} \mathbb{B}$ in $\langle \mathcal{E}, \rho\rangle$ can be used to show that $e \lessdot e' \models \rho(\mathbb{A}) \overset{\sigma}{\rightsquigarrow} \rho(\mathbb{B})$ in $\langle \mathcal{E}, \rho \circ \lambda\rangle$. □

Proposition 2 ($\sqsubseteq$ is sound and complete)**.** *Let $\Pi = \langle \mathscr{L}, \rightsquigarrow\rangle, \Pi' = \langle \mathscr{L}, \rightsquigarrow'\rangle$ be two policies. The following holds*

(i) if $\Pi' \sqsubseteq \Pi$ then, for all models $\langle \mathcal{E}, \lambda\rangle$ in $\mathscr{L}$, $\langle \mathcal{E}, \lambda\rangle \models \Pi$ implies $\langle \mathcal{E}, \lambda\rangle \models \Pi'$;

(ii) if for all models $\langle \mathcal{E}, \lambda\rangle$ in $\mathscr{L}$, $\langle \mathcal{E}, \lambda\rangle \models \Pi$ implies $\langle \mathcal{E}, \lambda\rangle \models \Pi'$, then $\Pi' \sqsubseteq \Pi$.

Proof. We first observe that considering the least reflexive and transitive relation among policies satisfying rules Ctx-ConstrD, amount to add to the set of rules the following:

$$\frac{}{\Pi \sqsubseteq \Pi}\ (\textsc{Refl}) \qquad \frac{\Pi \sqsubseteq \Pi' \quad \Pi' \sqsubseteq \Pi''}{\Pi \sqsubseteq \Pi''}\ (\textsc{Trans})$$

Now, soundness (i) can be proved by showing that all rules are sound, i.e., if their premises are satisfied then also their conclusion is.

Soundness is trivial for rules Refl and Trans.

Also rule Ctx is clearly sound since adding clauses allows for the same or more justifications for direct causalities.

Consider rule Split. The clause $\mathbb{A} \overset{\sigma}{\rightsquigarrow} \mathbb{B}$ is "split" into $\mathbb{A}_i \overset{\sigma}{\rightsquigarrow} \mathbb{B}_i$, for $i \in \{1, \ldots, n\}$.

We show that each direct causality justified by $\mathbb{A} \overset{\sigma}{\rightsquigarrow} \mathbb{B}$ is justified by $\mathbb{A}_i \overset{\sigma}{\rightsquigarrow} \mathbb{B}_i$ for some $i \in \{1, \ldots, n\}$. In fact, let $e \lessdot e'$ be a direct causality justified by $\mathbb{A} \overset{\sigma}{\rightsquigarrow} \mathbb{B}$, and let $e : \mathsf{A}$ and $e' : \mathsf{B}$. By the premise of rule Split, we know that there is $i \in \{1, \ldots, n\}$ such that $\mathsf{A} \in \mathbb{A}_i$ and $\mathsf{B} \in \mathbb{B}_i$. We claim that $e \lessdot e'$ is justified by $\mathbb{A}_i \overset{\sigma}{\rightsquigarrow} \mathbb{B}_i$. The key observation is the following. Let $C, C' \in \mathcal{C}(\mathcal{E})$, be configurations such that $C \subseteq C'$, $e' \in \mathsf{PE}(C)$ and $e' \in C'$. Assume that there is a pair (X, Y), $X, Y \subseteq C'$ with $e' \in Y$ that conforms to $\mathbb{A} \overset{\sigma}{\rightsquigarrow} \mathbb{B}$, i.e., $X : \mathbb{A}$ and $Y : \mathbb{B}$ and $X < Y$ (resp. $X \lessdot Y$) if $d \notin \sigma$ (resp. $d \in \sigma$). Then, if we consider $X' \subseteq X$ such that $X' : \mathbb{A}_i$ and $Y' \subseteq Y$ such that $Y' : \mathbb{B}_i$, it is clearly the case that (X', Y') conforms to $\mathbb{A}_i \overset{\sigma}{\rightsquigarrow} \mathbb{B}_i$.

This allows us to conclude that $e \lessdot e' \models \mathbb{A}_i \overset{\sigma}{\rightsquigarrow} \mathbb{B}_i$.

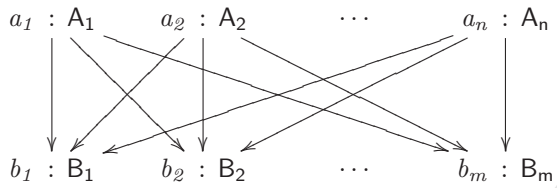
Soundness of Constr follows by the observation that removing a constraint on σ yields the same or more justifications for direct causalities. Concerning ConstrF and ConstrD, soundness follows by observing that, when the clauses have the special shape indicated by the premises, the constraint added is vacuously satisfied.

We now move on to completeness (ii). Assume that for all security models $\langle \mathcal{E}, \lambda \rangle \models \Pi$ implies $\langle \mathcal{E}, \lambda \rangle \models \Pi'$. We have to prove that $\Pi' \sqsubseteq \Pi$.

We proceed by proving that for each clause $\mathbb{A} \overset{\sigma}{\rightsquigarrow} \mathbb{B}$ in Π we have $\Pi' \sqsubseteq \mathbb{A} \overset{\sigma}{\rightsquigarrow} \mathbb{B}$. Then the fact that $\Pi' \sqsubseteq \Pi$ follows by rule Ctx and transitivity. More in detail, this can be proved by induction on the number of clauses in Π. The base case $|\Pi| = 1$ is trivial. If $|\Pi| > 1$, let $\mathbb{A} \overset{\sigma}{\rightsquigarrow} \mathbb{B} \in \Pi$ and let $\Pi'' = \Pi' \setminus \{\mathbb{A} \overset{\sigma}{\rightsquigarrow} \mathbb{B}\}$. By inductive hypothesis, $\Pi' \sqsubseteq \Pi''$. Moreover, since $\Pi' \sqsubseteq \mathbb{A} \overset{\sigma}{\rightsquigarrow} \mathbb{B}$, by using rule Ctx, we can obtain $\Pi' \sqsubseteq \Pi' \cup \{\mathbb{A} \overset{\sigma}{\rightsquigarrow} \mathbb{B}\}$. From $\Pi' \sqsubseteq \Pi''$, again by rule Ctx, we get $\Pi' \cup \{\mathbb{A} \overset{\sigma}{\rightsquigarrow} \mathbb{B}\} \sqsubseteq \Pi'' \cup \{\mathbb{A} \overset{\sigma}{\rightsquigarrow} \mathbb{B}\} = \Pi$. By transitivity thus we conclude $\Pi' \sqsubseteq \Pi$.

Thus, let $\mathbb{A} \overset{\sigma}{\rightsquigarrow} \mathbb{B}$ be a fixed clause in Π and let us prove that $\Pi' \sqsubseteq \mathbb{A} \overset{\sigma}{\rightsquigarrow} \mathbb{B}$. Let $\mathbb{A} = \{\mathsf{A}_1, \mathsf{A}_2, \ldots \mathsf{A}_n\}$ and $\mathbb{B} = \{\mathsf{B}_1, \mathsf{B}_2, \ldots \mathsf{B}_m\}$. We distinguish one case for each possible σ.

1. If $\sigma = \{d, f\}$ consider a security model $E_{\mathbb{A} \overset{\sigma}{\rightsquigarrow} \mathbb{B}}$ as illustrated below.



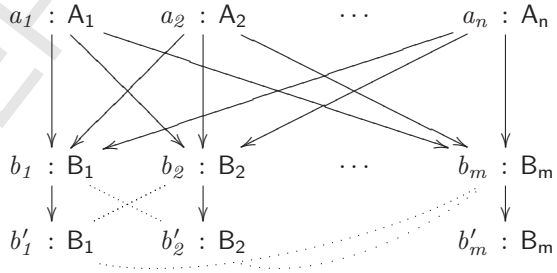
The clause $\mathbb{A} \overset{\sigma}{\rightsquigarrow} \mathbb{B}$ justifies all direct causalities in $E_{\mathbb{A} \overset{\sigma}{\rightsquigarrow} \mathbb{B}}$. Thus the security model $E_{\mathbb{A} \overset{\sigma}{\rightsquigarrow} \mathbb{B}}$ satisfies Π and hence, by our assumption, it also satisfies Π'. Therefore, for each direct causality $a_i : \mathsf{A_i} \lessdot b_j : \mathsf{B_j}$ the policy Π' must include a justifying clause. This must have the shape $\mathbb{A}'_{ij} \overset{\sigma'_{ij}}{\rightsquigarrow} \mathbb{B}'_{ij}$ with $\mathsf{A}_i \in \mathbb{A}'_{ij} \subseteq \mathbb{A}$, $\mathsf{B}_j \in \mathbb{B}'_{ij} \subseteq \mathbb{B}$ and trivially $\sigma'_{ij} \subseteq \sigma$ since $\sigma = \{d, f\}$. By using rule Constr, we can show that $\mathbb{A}'_{ij} \overset{\sigma'_{ij}}{\rightsquigarrow} \mathbb{B}'_{ij} \sqsubseteq \mathbb{A}'_{ij} \overset{\sigma}{\rightsquigarrow} \mathbb{B}'_{ij}$. Therefore, by using rules Ctx and transitivity we can show $\{\mathbb{A}'_{ij} \overset{\sigma'_{i,j}}{\rightsquigarrow} \mathbb{B}'_{i,j} \mid i \in \{1, \ldots, n\}, j \in \{1, \ldots, m\}\} \sqsubseteq \{\mathbb{A}'_{ij} \overset{\sigma}{\rightsquigarrow} \mathbb{B}'_{i,j} \mid i \in \{1, \ldots, n\}, j \in \{1, \ldots, m\}\}$

In turn, by rule Split, $\{\mathbb{A}'_{ij} \overset{\sigma'_{i,j}}{\rightsquigarrow} \mathbb{B}'_{i,j} \mid i \in \{1, \ldots, n\}, j \in \{1, \ldots, m\}\} \sqsubseteq \mathbb{A} \overset{\sigma}{\rightsquigarrow} \mathbb{B}$, hence by transitivity

$$\{\mathbb{A}'_{ij} \overset{\sigma'_{ij}}{\rightsquigarrow} \mathbb{B}'_{ij} \mid i \in \{1, \ldots, n\}, j \in \{1, \ldots, m\}\} \sqsubseteq \mathbb{A} \overset{\sigma}{\rightsquigarrow} \mathbb{B}$$

Hence, again using rule Ctx, we conclude $\Pi' \sqsubseteq \mathbb{A} \overset{\sigma}{\rightsquigarrow} \mathbb{B}$.

2. If $\sigma = \{d\}$ we can proceed along the lines of case (1), but constructing a slightly different security model $E_{\mathbb{A} \overset{\sigma}{\rightsquigarrow} \mathbb{B}}$ where each of the events b_i causes an event b'_i of the same level that is in conflict with all other b_j's, $j \neq i$

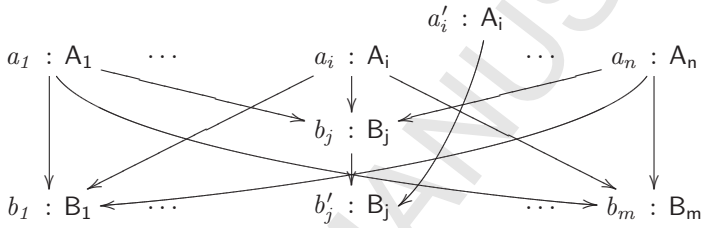


As in the previous case, clause $\mathbb{A} \overset{\sigma}{\rightsquigarrow} \mathbb{B}$ (together with the reflexive clauses $\mathsf{B}_i \rightsquigarrow \mathsf{B}_i$ which are present by default in all policies) justifies all direct causalities in $E_{\mathbb{A} \overset{\sigma}{\rightsquigarrow} \mathbb{B}}$. Hence $E_{\mathbb{A} \overset{\sigma}{\rightsquigarrow} \mathbb{B}}$ satisfies Π and thus Π'. We deduce that Π' includes a clause $\mathbb{A}'_{ij} \overset{\sigma'_{ij}}{\rightsquigarrow} \mathbb{B}'_{ij}$ justifying the direct causality $a_i : \mathsf{A_i} \lessdot b_j : \mathsf{B_j}$ with $\mathsf{A}_i \in \mathbb{A}'_{ij} \subseteq \mathbb{A}$, $\mathsf{B}_j \in \mathbb{B}'_{ij} \subseteq \mathbb{B}$. The only delicate point is the inclusion $\sigma'_{ij} \subseteq \sigma$. This clearly holds when $|\mathbb{B}'_{i,j}| > 1$. In fact, if $f \in \sigma'_{ij}$,

then the clause would not justify the corresponding causality since there is a maximal configuration $\{a_1, \ldots, a_n, b_1, b'_1\}$ which does not admit any pair (X, Y) with $b_1 \in Y$ conforming to the clause. If instead $|\mathbb{B}'_{i,j}| = 1$ we could have $f \in \sigma'_{ij}$. If this happens we can safely add the f constraint by using rule ConstrF.

Therefore, as in (1) we conclude that $\Pi' \sqsubseteq \mathbb{A} \overset{\sigma}{\rightsquigarrow} \mathbb{B}$.

3. If $\sigma = \{f\}$ we construct $n \times m$ security models $E^{i,j}_{\mathbb{A} \overset{\sigma}{\rightsquigarrow} \mathbb{B}}$ for $i \in \{1, \ldots, n\}$ and $j \in \{1, \ldots, m\}$ as follows



All models $E^{i,j}_{\mathbb{A} \overset{\sigma}{\rightsquigarrow} \mathbb{B}}$ satisfy the policy $\mathbb{A} \overset{\sigma}{\rightsquigarrow} \mathbb{B}$. Hence they also satisfy Π and thus Π'.

In particular, for all $i \in \{1, \ldots n\}$ and $j \in \{1, \ldots, m\}$ the policy Π' will include a clause that justifies the direct causality $a'_i : \mathsf{A_i} \lessdot b'_j : \mathsf{B_j}$ in the model $E^{i,j}_{\mathbb{A} \overset{\sigma}{\rightsquigarrow} \mathbb{B}}$. This must be of the kind $\mathbb{A}'_{ij} \overset{\sigma'_{ij}}{\rightsquigarrow} \mathbb{B}'_{ij}$ with $\mathsf{A}_i \in \mathbb{A}'_{ij} \subseteq \mathbb{A}$, $\mathsf{B}_j \in \mathbb{B}'_{ij} \subseteq \mathbb{B}$. Note that necessarily $d \notin \sigma'_{ij}$ and thus $\sigma'_{ij} \subseteq \sigma$ whenever $|\mathbb{A}'_{i,j}| > 1$ or $|\mathbb{B}'_{i,j}|$. In fact, otherwise the clause would not justify the corresponding causality since b'_j is caused directly only by a'_i (and b_j that here does not play a role) and a'_i causes directly only b'_j. If instead $|\mathbb{A}'_{i,j}| = |\mathbb{B}'_{i,j}| = 1$ we could have $d \in \sigma'_{ij}$. If this happens we can safely add the d constraint by using rule ConstrF.

Therefore, as in the previous cases (1) we conclude that $\Pi' \sqsubseteq \mathbb{A} \overset{\sigma}{\rightsquigarrow} \mathbb{B}$.

4. Finally, if $\sigma = \emptyset$ we can construct $n \times m$ security models $E^{i,j}_{\mathbb{A} \overset{\sigma}{\rightsquigarrow} \mathbb{B}}$ for $i \in \{1, \ldots, n\}$ and $j \in \{1, \ldots, m\}$ as in case (3), but where each of the events b_i causes a b'_i in conflict with b_j for $j \neq i$ (as in (2)).

□

Lemma 2 (Preservation and finiteness)**.** *Let N be a safe Petri net, $\mathcal{U}(N)$ be its unfolding. The equivalence $\approx$ is preserved by extension and is of finite index.*

Proof. Concerning the first part, we need to show that if $C \approx C'$, with $C, C' \in \mathcal{C}(\mathcal{U}(N))$, and there is e such that $C \oplus e \in \mathcal{C}(\mathcal{U}(N))$, then there exists e' such that $C' \oplus e' \in \mathcal{C}(\mathcal{U}(N))$ and $C \oplus e \approx C' \oplus e'$. Assume that here is $e \in \mathsf{PE}(C)$

such that $C \oplus e \in \mathcal{C}(\mathcal{U}(N))$. This means that there is a transition t in the net N such that $\pi_1(e) = t$ and t is enabled by $\mathsf{M}(C) = \pi_1(C^\circ)$. Since $\mathsf{M}(C) = \mathsf{M}(C')$ it follows that there is also an event $e' \in \mathsf{PE}(C')$ such that $\pi_1(e') = t$. Hence $C' \oplus e' \in \mathcal{C}(\mathcal{U}(N))$. Clearly $\mathsf{M}(C \oplus e) = \mathsf{M}(C' \oplus e')$. Moreover, we can easily prove that $\lambda_d^{C\oplus e} = \lambda_d^{C'\oplus e'}$ and $\lambda_i^{C\oplus e} = \lambda_i^{C'\oplus e'}$. In fact, let $p \in \mathsf{M}(e \oplus C)$ and let $b \in (e \oplus C)^\circ$ be the corresponding condition. If $b \notin e^\bullet$ then clearly $\lambda_d^{C\oplus e}(p) = \lambda_d^{C'\oplus e'}(p)$ and $\lambda_i^{C\oplus e}(p) = \lambda_i^{C'\oplus e'}(p)$. If instead $b \in e^\bullet$, we have that $\lambda_d^{C\oplus e}(p) = \{\lambda(t)\} = \lambda_d^{C'\oplus e'}(p)$. Moreover,

$$\begin{aligned}\lambda_i^{C\oplus e}(p) &= \{\lambda(t)\} \cup \textstyle\bigcup_{p' \in {}^\bullet t} \lambda_i^{C}(p') \\ &= \{\lambda(t)\} \cup \textstyle\bigcup_{p' \in {}^\bullet t} \lambda_i^{C'}(p') \quad [\text{since } \mathsf{M}^*(C) = \mathsf{M}^*(C')] \\ &= \lambda_i^{C'\oplus e'}(p)\end{aligned}$$

We can also prove that $\leq_{C\oplus e} = <_{C'\oplus e'}$. In fact, assume that $p_1 \leq_{C\oplus e} p_2$. If $p_1, p_2 \notin t^\bullet$ then we conclude by the fact that $C \approx C'$. Otherwise, it is easy to see that only one of the two places must be in $t^\bullet$. Assume, without loss of generality that $p_1 \notin t^\bullet$ and $p_2 \in t^\bullet$. Let $b_1 \in C \oplus e^\circ$, $b_1' \in C' \oplus e'^\circ$ be the conditions corresponding to p_1 in the two configurations and let e_1, e_1' be the generating events. The fact that $p_1 \leq_{C\oplus e} p_2$ means that $e_1 \leq e$ and thus $e_1 \leq b_3$ for some $b_3 \in C^\circ \cap {}^\bullet e$. If we let $p_3 = \pi_1(b_3)$ this implies that $p_1 <_C p_3$. Since $\mathsf{M}^*(C) = \mathsf{M}^*(C')$ we deduce that also $p_1 <_{C'} p_3$, from which we immediately get $p_1 <_{e'\oplus C'} p_2$.

Therefore we deduce that $\mathsf{M}^*(C \oplus e) = \mathsf{M}^*(e' \oplus C')$, and thus $C \oplus e \approx C' \oplus e'$, as desired.

For finiteness, observe that because N is safe and the set of security levels is finite the number of different level marking for a given net is clearly finite. □

Lemma 3 (Justification)**.** *Let* Π *be a policy and* $N = (P, T, F)$ *be a safe Petri net. Let* e_0, e_1 *be events such that* $e_0 \lessdot e_1$*, with* $\lambda(e_1) = \mathsf{B}$*. Let* $C \in \mathcal{C}(\mathcal{U}(N))$ *be a configuration that can be extended with* e_1*, i.e.,* $C \oplus e_1 \in \mathcal{C}(\mathcal{U}(N))$*. Then*

- $e_0 \lessdot e_1$ *is justified by a clause* $\mathbb{A} \rightsquigarrow \mathsf{B}$ *iff* $\mathbb{A} \subseteq \bigcup\{\lambda_i^C(p) \mid p \in {}^\bullet\pi_1(e_1)\}$*;*
- $e_0 \lessdot e_1$ *is justified by a clause* $\mathbb{A} \overset{d}{\rightsquigarrow} \mathsf{B}$ *iff* $\mathbb{A} \subseteq \bigcup\{\lambda_d^C(p) \mid p \in {}^\bullet\pi_1(e_1) \ \wedge$ p *is* $\leq_C$*-maximal* $\}$

Proof. This is trivial since, by construction, the level marking of a configuration precisely captures for every event, the set of levels it (directly and indirectly) depends on. □

Corollary 1 (Completeness)**.** *Let* Π *be a policy,* N *be a safe Petri net,* $\mathcal{U}(N)$ *be its unfolding,* $U = (P', T', F')$ *be the complete prefix with respect to* $\approx$*, and* $e_0, e_1 \in T^u$ *be events that witness a violation of policy* Π*. Then there are events* e_0', e_1' *in* T' *such that* $\lambda(e_0) = \lambda(e_0')$ *and* $\lambda(e_1) = \lambda(e_1')$ *that witness a violation of* Π*.*

Proof. The result follows immediately from Lemma 3 □

Lemma 4 (shadow transitions)**.** *Let $C \in \mathcal{C}(\mathcal{U}(N))$, and let e be an event such that $C \oplus e \in \mathcal{C}(\mathcal{U}(N))$. Then the k-shadow set $SH_k(C \oplus e)$ only depends on $SH_k(C)$ and $t = \pi_1(e)$. In fact, for any $X \subseteq C \oplus e$, $|X| \leq k$, with λ injective on X, we have that $sh(X, C \oplus e) = \langle \mathsf{M}(C \oplus e), \lambda_d^{C \oplus e}, \lambda_i^{C \oplus e}, \leq_{C \oplus e} \rangle$ where*

- $\mathsf{M}(C \oplus e) = (\mathsf{M}(C) \setminus {}^\bullet t) \cup t^\bullet$;
- $\lambda_d^{C \oplus e}(p) = \begin{cases} \lambda_d^C(p) & \text{if } p \notin t^\bullet \\ \lambda(t) & \text{if } p \in t^\bullet \end{cases}$
- $\lambda_i^{C \oplus e}(p) = \begin{cases} \lambda_i^C(p) & \text{if } p \notin t^\bullet \\ \{\lambda(x) \in X \mid x \leq e\} & \text{if } p \in t^\bullet \end{cases}$
- *for $p, p' \in \mathsf{M}(C \oplus e)$, we have $p \leq_{C \oplus e} p'$ if* $\begin{cases} p, p' \notin t^\bullet \text{ and } p \leq_C p', \text{or} \\ p' \in t^\bullet \text{ and } \exists p'' \in {}^\bullet t.\ p \leq_C p'' \end{cases}$

Proof. Immediate from the definitions. □

Lemma 5 (Preservation and finiteness)**.** *Let N be a safe Petri net, $\mathcal{U}(N)$ be its unfolding, and Π be a policy. The equivalence $\approx_\Pi$ is preserved by extension and is of finite index.*

Proof. Concerning the first part, we need to show that if $C \approx_\Pi C'$, with $C, C' \in \mathcal{C}(\mathcal{U}(N))$, and there is e such that $C \oplus e \in \mathcal{C}(\mathcal{U}(N))$, then there exists e' such that $C' \oplus e' \in \mathcal{C}(\mathcal{U}(N))$ and $C \oplus e \approx_\Pi C' \oplus e'$.

Assume that here is $e \in \mathsf{PE}(C)$ such that $C_1 = C \oplus e \in \mathcal{C}(\mathcal{U}(N))$ and let $t = \pi_1(e)$. Since t is enabled by $\mathsf{M}(C) = \mathsf{M}(C')$ it follows that there is also an event $e' \in \mathsf{PE}(C')$ such that $\pi_1(e') = t$. Hence $C_1' = C' \oplus e' \in \mathcal{C}(\mathcal{U}(N))$. By Lemma 4, it immediately follows that $C_1 \approx_\Pi C_1'$, as desired.

For finiteness, just observe that because N is safe, it has at most $2^{|P|} = n$ distinct markings. Together with the fact that the set of security levels $\mathscr{L}$ is finite, this allows us to deduce that there is a finite number of different shadows. Since a shadow set collects the shadows of sets X such that $X : \mathbb{A}$ for some clause $\mathbb{A} \overset{\sigma}{\rightsquigarrow} \mathbb{B}$ and Π consists of a finite number of clauses, we conclude. □

Lemma 6 (Completeness)**.** *Let Π be a policy, N be a safe Petri net, $\mathcal{U}(N)$ be its unfolding, $U = (P', T', F')$ be the finite complete prefix respect to $\approx_\Pi$, and $e_0, e_1 \in T^u$ be events that witness a violation of the policy Π. Then there are events $e_0', e_1' \in T'$ such that $\lambda(e_0) = \lambda(e_0')$ and $\lambda(e_1) = \lambda(e_1')$ that witness a violation of Π.*

Proof. First observe that given two equivalent configurations $C_1 \approx_\Pi C_2$ and a subset $D_1 \subseteq E^u$ which extends C_1, by Lemma 5, there exists $D_2 \subseteq E^u$ which extends C_2 such that $C_1 \oplus D_1 \approx_k C_2 \oplus D_2$. It is also easy to see that there is an isomorphism $\iota : D_1 \to D_2$ (with D_1 and D_2 endowed with the security labelling and order inherited from $\mathcal{U}(\mathbf{N})$). This implies that, given a clause $\mathbb{A} \overset{\sigma}{\rightsquigarrow} \mathbb{B}$, $X_1 \subseteq C_1$ and $Y_1 \subseteq D_1$ such that (X_1, Y_1) conforms to the clause then there is $X_2 \subseteq C_2$ such that $(X_2, \iota(Y_1))$ conforms to the same clause.

Now, let $e_0, e_1 \in E^u$ be such that $e_0 \lessdot e_1$. By construction of the finite prefix (see [13, Theorem 6]) there exists $e_1' \in \mathcal{C}(U)$ such that $[e_1') \approx_\Pi [e_1)$.

Fix any clause $c = \mathbb{A} \overset{\sigma}{\rightsquigarrow} \mathbb{B} \in \Pi$. Observe that each configuration C_1 such that $e_1 \in \mathsf{PE}(C_1)$ is an extension of C_1, say $C_1 = [e_1) \oplus D_1$. For any extension of C_1, say $C_1 \oplus E_1 = [e_1) \oplus (D_1 \cup E_1)$, if there are $X_1 \subseteq [e_1)$ and $Y_1 \subseteq D_1 \cup E_1$ such that $e_1 \in Y_1$ and (X_1, Y_1) conforms to c then, by applying twice the above observation, we get an extension D_2 of $[e_2)$ such that $C_1 \approx_\Pi C_2 = [e_2) \oplus D_2$ and $e_2 \in \mathsf{PE}(C_2)$, and an extension extension E_2 of C_2 such that $[e_1) \oplus (D_1 \cup E_1) = E_1 \oplus C_1 \approx_\Pi C_2 \oplus E_2 = [e_2) \oplus (D_2 \cup E_2)$. Moreover there is an isomorphism $\iota : D_1 \cup E_1 \to D_2 \cup E_2$, with $\iota(e_1) = e_1'$, and we can find $X_2 \subseteq [e_1')$ such that $(X_2, \iota(Y_1))$ conforms to c and $\iota(e_1) = e_1 \in \iota(Y_1)$.

The above implies that if $e_0 \lessdot e_1 \models c$ then $e_0 \lessdot e_1 \models c$. The reasoning is perfectly symmetric and can be reversed, i.e., $e_0 \lessdot e_1 \models c$ iff $e_0 \lessdot e_1 \models c$. From this, the thesis immediately follows. □

Appendix B. Decidability on Regular Trace Event Structures via MSO

The event structure associated with a system exhibiting a cyclic behaviour is infinite, since events represent "occurrences" of computational steps. As shown in Section 5, policy satisfaction is decidable once we focus on event structures generated by finite safe Petri nets. They have been characterised abstractly as the class of regular trace event structures [5]. Roughly, one can think that they are the kind of event structures that arise from finite state systems (finite safe Petri nets, finite state processes in process calculi, suitable program abstractions, etc.).

In this appendix we provide an alternative proof of decidability of policy satisfaction on regular trace event structures relying on the observation that it can be expressed as a formula in MTL (monadic trace logic, a restriction of monadic second order logic where quantification is restricted to conflict-free sets of events) that is known to be decidable on regular trace event structures [6].

The interest for the alternative proof is twofold. First, it shows that policy satisfaction can be expressed as a monadic second order property. Second, this fact makes the proof easily adaptable for mild changes of the policy language.

We first instantiate the notion of regularity to the setting of security models.

Definition 15 (regular trace model). A security model $\langle \mathcal{E}, \lambda \rangle$ in $\mathscr{L}$ is *regular trace* if there exists a safe Petri net N such that $\mathcal{E} = \mathcal{E}(N)$ and λ is the security assignement to events in the unfolding.

As mentioned above, policy satisfaction can be encoded as a sentence in MTL. The details can be found in the proof of Proposition 3. Intuitively, in order to check whether a causal dependency $a \lessdot b$ is justified by some clause $\mathbb{A} \overset{\sigma}{\rightsquigarrow} \mathbb{B}$, we need check the existence of consistent subsets of events $X : \mathbb{A}$, $Y : \mathbb{B}$ satisfying some certain properties ($X < Y$, etc.). Then decidability follows from decidability of MTL model checking over regular event structures [6].

Proposition 3 (decidability on regular trace security models)**.** *Let* $\Pi = \langle \mathscr{L}, \rightsquigarrow \rangle$ *be an information flow policy and let* $\langle \mathcal{E}, \lambda \rangle$ *be a regular trace security model in* $\mathscr{L}$*. The decision problem* $\langle \mathcal{E}, \lambda \rangle \models \Pi$ *is decidable.*

Proof. The proof just consists in showing $\langle \mathcal{E}, \lambda \rangle \models \Pi$ is expressible in MTL on M-labelled trace event structures. The syntax of MTL on $\mathscr{L}$-labelled trace event structures as considered and shown to be decidable on regular trace event structures in [6] is as follows

$$\varphi \ ::= \ x = x' \mid x \leq x' \mid x \# x' \mid x : \mathsf{A} \mid \neg\varphi \mid \varphi \wedge \varphi \mid \exists x.\, \varphi \mid \exists X.\, \varphi \mid x \in X$$

where x, x' range in a set of event variables, X ranges over a set of event set variables, and $\alpha \in \Sigma$. The predicate $x : \mathsf{A}$ means that the λ-labelling of x is A. Second order quantification is done over conflict-free sets of events only. For the rest of the constructs, the semantics is the expected one.

We start by defining some basic predicates:

$$\begin{array}{rcl} x < y & \triangleq & x \leq y \ \wedge \ \neg(x = y) \\ x \lessdot y & \triangleq & x < y \ \wedge \ \forall z.((x \leq z \ \wedge \ z \leq y) \ \rightarrow \ (z = x \ \vee \ z = y)) \\ \mathit{conf}(X) & \triangleq & \forall x \in X.\, \forall y \leq x.\, y \in X \\ \mathit{maxConf}(X) & \triangleq & \mathit{conf}(X) \ \wedge \ (\forall C''.\mathit{conf}(C'') \ \wedge \ C' \subseteq \ C'' \rightarrow C' = C'') \end{array}$$

Note that in the $\mathit{conf}(X)$ predicate we do not need to ask for absence of conflicts since second order quantification is limited, by the semantics, to consistent sets.

The causality and immediate causality predicates can be also extended to sets of fixed size:

$$\begin{array}{rcl} \{x_1, \ldots, x_n\} < \{y_1, \ldots, y_m\} & \triangleq & \bigwedge_{i=1}^{n} \bigwedge_{j=1}^{m} x_i < y_j \\ \{x_1, \ldots, x_n\} \lessdot \{y_1, \ldots, y_m\} & \triangleq & \bigwedge_{i=1}^{n} \bigwedge_{j=1}^{m} x_i \lessdot y_j \end{array}$$

Then we can define predicates stating that the dependencies between the sets of events $X = \{x_1, \ldots, x_n\}$ and $Y = \{y_1, \ldots, y_m\}$ conform to some clause $\mathbb{A} \overset{\sigma}{\rightsquigarrow} \mathbb{B}$, with $\mathbb{A} = \{\mathsf{A}_1, \ldots, \mathsf{A}_n\}$ and $\mathbb{B} = \{\mathsf{B}_1, \ldots, \mathsf{B}_m\}$ as follows

$$\begin{array}{rcl} \mathit{conform}(X, Y, \mathbb{A} \rightsquigarrow \mathbb{B}) & \triangleq & (\bigwedge_{i=1}^{n} x_i : \mathsf{A}_i) \ \wedge \ (\bigwedge_{j=1}^{m} y_j : \mathsf{B}_j) \ \wedge \\ & & (\{x_1, \ldots, x_n\} < \{y_1, \ldots, y_n\}) \\ \mathit{conform}(X, Y, \mathbb{A} \overset{d}{\rightsquigarrow} \mathbb{B}) & \triangleq & (\bigwedge_{i=1}^{n} x_i : \mathsf{A}_i) \ \wedge \ (\bigwedge_{j=1}^{m} y_j : \mathsf{B}_j) \ \wedge \\ & & (\{x_1, \ldots, x_n\} \lessdot \{y_1, \ldots, y_n\}) \\ \mathit{conform}(X, Y, \mathbb{A} \overset{\sigma \cup \{f\}}{\rightsquigarrow} \mathbb{B}) & \triangleq & \mathit{conform}(X, Y, \mathbb{A} \overset{\sigma}{\rightsquigarrow} \mathbb{B}) \end{array}$$

In turn, this can be used to express the fact that some clause $\mathbb{A} \overset{\sigma}{\rightsquigarrow} \mathbb{B}$, with $\mathbb{A} = \{\mathsf{A}_1, \ldots, \mathsf{A}_n\}$ and $\mathbb{B} = \{\mathsf{B}_1, \ldots, \mathsf{B}_m\}$ can be used to justify a causal dependency $x \lessdot y$, as follows. For $f \notin \sigma$:

$$\begin{aligned} \mathit{justify}(x, y, \mathbb{A} \overset{\sigma}{\rightsquigarrow} \mathbb{B}) \quad \triangleq \quad & \forall C.\,(\mathit{conf}(C) \ \wedge \ y \in PE(C)) \rightarrow \\ & \exists C'.(\mathit{conf}(C') \ \wedge \ C \subseteq C' \ \wedge \ y \in C' \ \wedge \\ & \exists x_1, \ldots, x_n.\, \exists y_1, \ldots, y_m. \\ & (\textstyle\bigvee_{i=1}^{n} \lambda(x) = \lambda(x_i)) \ \wedge \ (\bigvee_{j=1}^{m} y = y_j) \wedge \\ & \mathit{conform}(x_1, \ldots, x_n, y_1, \ldots, y_n, \mathbb{A} \overset{\sigma}{\rightsquigarrow} \mathbb{B}) \end{aligned}$$

where $n = |\mathbb{A}|$ and $m = |\mathbb{B}|$ since, by definition we must consider sets of events x_i, y_i of the same cardinality as the source $\mathbb{A}$ and target $\mathbb{B}$ of the clause.

If instead, $f \in \sigma$

$$\begin{aligned} \mathit{justify}(x, y, \mathbb{A} \overset{\sigma}{\rightsquigarrow} \mathbb{B}) \quad \triangleq \quad & \forall C.\,(\mathit{conf}(C) \ \wedge \ y \in PE(C)) \rightarrow \\ & \forall C'.(\mathit{maxConf}(C') \ \wedge \ C \subseteq C' \ \wedge \ y \in C') \\ & \exists x_1, \ldots, x_n.\, \exists y_1, \ldots, y_m. \\ & (\textstyle\bigvee_{i=1}^{n} \lambda(x) = \lambda(x_i)) \ \wedge \ (\bigvee_{j=1}^{m} y = y_j) \wedge \\ & \mathit{conform}(x_1, \ldots, x_n, y_1, \ldots, y_n, \mathbb{A} \overset{\sigma}{\rightsquigarrow} \mathbb{B}) \end{aligned}$$

Finally, $\langle \mathcal{E}, \lambda \rangle \models \Pi$ can be expressed as follows

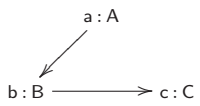
$$\langle \mathcal{E}, \lambda \rangle \models \Pi \ \triangleq \ \forall x, y.(x \lessdot y \ \rightarrow \bigvee_{\mathbb{A} \overset{\sigma}{\rightsquigarrow} \mathbb{B} \in \Pi} \mathit{justify}(x, y, \mathbb{A} \overset{\sigma}{\rightsquigarrow} \mathbb{B}))$$

Since it is expressed as a MSO formula, with second order quantification restricted to consistent sets, decidability on regular trace security models immediately follows from decidability of MTL on regular trace event structures proved in [6, Theorem 4.1]. □

Appendix C. Differences in the semantics of policies respect to the conference version

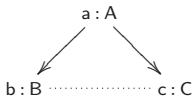
Our recent investigations have revealed examples that suggest that the original notion of presented in [4] was too demanding in some cases (especially when requiring fairness), while the notion of unfair flows was too liberal in some cases. We have hence respectively relaxed and strengthened the required (un)fairness constraints, and we dropped a constraint that forbade flow recipients to cooperate.

One major difference in the semantics is that the set Y is not required to be flat anymore. This means that events in Y can have causal dependencies. For example, the security model



satisfies the policy $(\mathsf{A} \rightsquigarrow \mathsf{B}, \mathsf{C}), (\mathsf{B} \rightsquigarrow \mathsf{C})$ with the new semantics, but it did not in the old one. We believe that forbidding flows between recipients in Y is too restrictive.

Another major difference is that the requirement that the flow should go to all recipients has been made stricter in unfair policies. For example, the security model



was conformant to the policy $(\mathsf{A} \rightsquigarrow \mathsf{B}, \mathsf{C})$ in the old semantics, but it is not anymore. We believe that allowing competition without any possibility for co-operation is not appropriate.

Finally, we have relaxed the notion of fairness by dropping one constraint related to conflicts in the set Y used to justify the flow. In the original semantics, any arbitrary pair of events in Y was required to be in conflict with the same sets of events. This condition has now been dropped so that we just require that the flow to all recipients can happen one way or the other. An example is the following security model

which satisfies the policy $(\mathsf{A} \overset{f}{\rightsquigarrow} \mathsf{B}, \mathsf{C})$ with the new semantics, but it did not in the old one. Since, the flow from A to B necessarily goes to C through one of the two (conflicting) events c or c' we believe it would be too restrictive not to accept it.