Verifying Information Flow Security in Visibly Pushdown Automata[⋆]

Barbara Sprick[1⋆⋆] and Daniel Staesche[2]

[1] SRH University of Applied Science Heidelberg
Faculty of Informatics,
barbara.sprick@hshd.de
[2] Technische Universität Darmstadt, Department of Computer Science,
Modeling and Analysis of Information Systems
staesche@mais.informatik.tu-darmstadt.de

Abstract. Information flow control is an important area in computer security. It aims at restricting what a low level observer can deduce from his observations about high level system behavior. We investigate the problem of verifying possibilistic information flow properties for visibly pushdown automata. Such properties can be expressed by Basic Security Predicates (BSPs) in the Modular Assembly Kit for Security [15]. They are, however, known to be undecidable for context free languages [8].
The class of visibly pushdown languages (VPLs) [2] lies between regular and deterministic context free languages. Still, VPLs share many desirable properties with the class of regular languages. In addition, they allow for model checking non-regular properties like stack inspection or pre- and post-conditions. Assuming a low level observable stack, we show in this paper how BSPs can be automatically verified for VPLs. In a second step, we extend this result to intransitive information flow properties.

1 Introduction

Controlling the flow of information is a central concern in computer security. Access control policies specify, which accesses to which data are allowed for a system. Usage control restricts the ways in which systems are allowed to make use of data. However, these methods can only control direct access and usage of data (over overt channels), leakage of information over covert channels like, e.g. trojan horses or observable system behavior, are not controllable by such methods. Information flow security approaches this problem by formally specifying and restricting the information that may flow between security domains. The original notion of non-interference, introduced by Goguen and Meseguer [GM82], defines the restriction of information flow as a lack of dependency. Two users/processes of different security domains *high* (H) and *low* (L) are

⋆ This work was funded by the DFG (German research foundation) under the project FM-SecEng in the Computer Science Action Program (MA 3326/1-3). The work is published as Technical Report at TU Darmstadt under TR-number TUD-CS-2013-0275.

⋆⋆ Work was done while author was affiliated with TU Darmstadt.

considered, and the property states that H is non-interfering with L if L's observations of the system do not depend on H's inputs. The original definition was based on deterministic state machines and appropriate only for deterministic systems. Furthermore, it was too strict for practical purposes. Since then, many similar information flow properties were introduced that relaxed this restriction by requiring that every low level behavior an attacker can observe must also be *possible* in the context of several (unobservable) high level behaviors. These properties are called *possibilistic information flow properties*. Noninference [22,19,25], generalized non-interference [18,19,25], forward correctability [14], nondeducibility for outputs [13] and separability [19] are examples of such properties. Comparing these information flow properties was difficult as they are based on different system models like, e.g., state event systems, trace-based systems or process algebras. Several frameworks targeting a uniform description of information flow properties have been developed, e.g. by McLean [19], Focardi and Gorrieri [11], Zakinthinos and Lee [25], and Mantel [15]. For our approach, we choose the *Modular Assembly Kit for Security* (MAKS) [15]. The MAKS framework is the most comprehensive among them and provides a uniform representation of most possibilistic information flow properties known from the literature, including the ones mentioned above. This framework is based on traces as system model. It contains a set of basic information flow properties called *Basic Security Predicates* (BSPs) that serve as building blocks for the known non-interference-like information flow properties.

D'Souza et al have shown in [9,7,8] that the Basic Security Predicates in the MAKS framework are decidable for finite state systems but undecidable for pushdown systems. In [9], the authors have introduced a set of language-theoretic operations and have shown, that the question of whether a particular BSD is satisfied for a system Tr boils down to the language inclusion problem $op_1(Tr) \subseteq op_2(Tr)$ for some language-theoretic operations op_1 and op_2.

In this paper we will identify a larger class of systems, namely Visibly Pushdown Systems, and show that all BSPs and, consequently, all information flow properties that can be expressed in MAKS are decidable for this class of systems. Finite state systems are an interesting system class as it is expressive enough for a wide range of system properties. However, there are some properties like, e.g. pre- and postconditions or nested structures, that are very desirable for system specifications but cannot be expressed in a finite state system.

Visibly Pushdown Automata (VPA) [2] allow for the specification of many non-regular properties and are yet tractable like finite state systems.

They are pushdown automata which require an explicit partition of the input alphabet into three distinct subsets: call events, return events and internal events where the type of the input symbol determines the stack operation.

The resulting languages, Visibly Pushdown Languages (VPLs), allow for representation of data with both a linear as well as a hierarchical ordering as e.g. the execution of HTML/XML documents or structured programs. The class of VPLs is a strict subclass of context-free languages and properly includes the regular languages. VPLs have the same desirable closure properties as regular languages: they are closed under union, intersection, complement and Kleene-*. Furthermore, many problems like the word problem and the inclusion problem are decidable. Deterministic VPAs are as expressive as their nondeterministic counterparts. Since their introduction by Alur and Madhusudan [2], several issues have been studied. The word problem was shown to be decidable in linear time, if the VPL is specified by a visibly pushdown grammar [24]. The minimization problem has been investigated for several subclasses of automata by [5] and [1]. Visibly Pushdown Languages are related to regular languages of nested words [3], which can be used for software verification or XML stream processing.

After showing, that properties of the MAKS framework are decidable for Visibly Pushdown Automata, we will extend our decidability result in another direction. The above information flow properties assume a transitive flow policy, stating that if information may flow from A to B and from B to C, then it may also flow from A to C directly. However, areas like channel control in distributed systems, explicit declassification and information filters require intransitive flow policies where information may flow from A to C only via B. This line of research was started by Rushby [23]. In [16], Mantel developed new properties in the style of BSPs that can cope with intransitive flow policies.

In this paper we will also obtain a decidability result for intransitive information flow properties in the style of [16]. We will show how the language-theoretic operations used for deciding BSPs can be adapted such that we also get decision procedures for intransitive information flow properties.

Turning to related work, several decidability results for information flow properties for finite state systems were established. Focardi and Gorrieri presented decision procedures for information flow properties that can be defined in the framework presented in [11]. Closely related is the work by D'Souza et al. [9], who have provided decision procedures for all security properties in the more expressive MAKS framework defined by [15]. Also, van der Meyden and Zhang [20] provided algorithmic verification techniques for a subset of these information flow properties.

In our work, we go beyond finite state systems and show decidability of information flow properties for visibly pushdown systems. In the context of infinite state systems, several undecidability results were established, e.g. [7] [6]. However, Dam [6] has also given decision procedures for *strong low bisimulation* for simple parallel while programs. Another work considering infinite state systems was done by Best et al. in [4], where they showed the intransitive information flow property *INI* defined on elementary net systems to be decidable in an extended framework of unbounded Petri nets. We provide decision procedures for VPLs for all security properties that are expressible in MAKS, as well as for some intransitive predicates.

Contributions. The contribution of this work is threefold: First, we prove the classical information flow property *noninference* to be decidable for VPAs. Second, we prove that all BSPs in MAKS and, hence, a wide range of information flow properties known from the literature is decidable for Visibly Pushdown Languages. Third, we extend the result to intransitive information flow properties.

The remainder of the paper is organized as follows. In Section 4 we introduce a variant of Visibly Pushdown Automata (VPAs) and Visibly Pushdown LanguagesVPL. In Section 3 we introduce the classical information flow property *noninference* and show, that noninference is decidable for VPLs. Section 5 introduces the MAKS framework and shows that all BSPs in MAKS are decidable for VPLs. In Section 6 we extend the decidability result to intransitive information flow properties that are defined in the style of BSPs. We do this by appropriately adapting the language-theoretic operations and showing that the class of VPAs is closed under these modified operations. We finally conclude in Section 7.

2 Visibly Pushdown Systems

Looking for a more expressive class of languages for which noninference is decidable, we investigate visibly pushdown languages (VPLs, [2]). They constitute a strict subclass of context-free languages and properly include regular languages. Hence, VPLs are a promising candidate for new decidability results. In contrast to regular languages, they allow for representation of data with both a linear as well as a hierarchical ordering as e.g. structured programs. Despite their expressiveness, VPLs share desirable closure properties with regular languages: they are closed under union, intersection, complement and Kleene-*. Furthermore, many problems like the word problem and the inclusion problem are decidable. VPLs are characterized by visibly pushdown automata (VPA, [2]). For our purpose of showing decidability of noninference for VPA we require a slightly modified definition, which we call VPA^{sec}. Other than the original version of VPA, VPA^{sec} allow a restricted use of ϵ-transitions. Though this does not add any expressive power (as we will briefly show at the end of this section), it will, however, come in handy for the proofs and constructions in section 3.

2.1 Visibly Pushdown Automata for Security

Visibly pushdown automata are similar to pushdown systems but operate on a partitioned alphabet $\widetilde{\Sigma} = (\Sigma_c, \Sigma_r, \Sigma_{int})$ that determines which events modify the stack in which way: call events (Σ_c) push symbols onto the stack, return events (Σ_r) pop symbols from it and internal events (Σ_{int}) leave the stack unmodified.

We define visibly pushdown automata for security (VPA^{sec}) as follows:

Definition 21 (VPA^{sec}) *A VPA^{sec} A on finite words over $\widetilde{\Sigma} = (\Sigma_{int}, \Sigma_c, \Sigma_r)$ is a tuple $A = (Q, q_{in}, \widetilde{\Sigma}, \Gamma, \delta^{sec}, Q_F)$ where Q is a finite set of states, $q_{in} \in Q$ is an initial state, Γ is a finite stack alphabet that contains $\perp$ as a special bottom element, and $\delta^{sec} \subseteq (Q \times \Sigma_c \times Q \times (\Gamma \backslash \{\perp\})) \cup (Q \times \Sigma_r \times \Gamma \times Q) \cup (Q \times \Sigma_{int} \cup \{\epsilon\} \times Q)$ such that $Q_F \subseteq Q$ is a set of final states.*

The relation $reach_A : Q \to 2^Q$ determines the ϵ-closure of states in A. It is the smallest set satisfying $s \in reach_A(s)$ and $\forall q' \in Q. \forall q \in reach_A(s) : (q, \epsilon, q') \in \delta^{sec} \Rightarrow q' \in reach_A(s)$.

The automaton A accepts a word $w \in \Sigma^*$ (with length $|w| = k + 1$) iff there is a sequence $\rho = (q_0, \sigma_0) \dots (q_k, \sigma_k)$ with $q_0 = reach_A(q_{in})$, $\sigma_0 = \perp$ and $q_k \in Q_F$ such that for all $0 \leq i < k$ the following holds:

Push: If $w_i \in \Sigma_c$ then there is a $\gamma \in \Gamma$ and $q' \in Q$ such that $(q_i, w_i, q', \gamma) \in \delta^{sec}$ and $q_{i+1} \in reach_A(q')$ and $\sigma_{i+1} = \gamma.\sigma_i$

Pop: If $w_i \in \Sigma_r$ then there is a $\gamma \in \Gamma$ and $q' \in Q$ such that $(q_i, w_i, \gamma, q') \in \delta^{sec}$ and $q_{i+1} \in reach_A(q')$ and either $\sigma_i = \gamma.\sigma_{i+1}$ or $\gamma = \sigma_{i+1} = \sigma_i = \perp$.

Internal: If $w_i \in \Sigma_{int}$ then there is $q' \in Q$ such that $(q_i, w_i, q') \in \delta^{sec}$ and $q_{i+1} \in reach_A(q')$ and $\sigma_{i+1} = \sigma_i$

The language $\mathcal{L}(A)$ recognized by A, is the set of all words $w \in \Sigma^*$ accepted by A.

The difference between the original definition of VPA given in [2] and the above definition of VPA^{sec} is that we allow the occurrence of ϵ-transitions as internal events. Like in the original definition, call and return transitions, that modify the stack, are not possible on ϵ-events. Interestingly, we can show that our definition of VPA^{sec} has the same expressiveness as the original definition:

Theorem 1. *Each VPA^{sec} A over $\widetilde{\Sigma}$ can be transformed into an (original) VPA A' over $\widetilde{\Sigma}$ with $\mathcal{L}(A) = \mathcal{L}(A')$.*

Proof sketch: We construct A' using a power set construction. The set of states of A' is the power set of the states of A. A transition from one set of states to another corresponds to the set of possible moves in A, which is determined by the ϵ-closure of a transition in A. As a result, we have a VPA with a finite set of states and a transition function without ϵ-transitions.

The new VPA accepts the same language as the VPA^{sec}. One can show by induction over finite words w that for each A-run over w there is a corresponding A'-run and vice versa. Since ϵ-transitions do not alter the stack, the nondeterminism only affects internal transitions. Consequently, the power set construction is similar to that for finite automata.

3 Noninference

In this section, we focus on a simple information flow property called noninference [22] and show that noninference is decidable for Visibly Pushdown Languages. In section 5 we will generalize our result for a framework for non-interference properties.

3.1 Security by Noninference

Noninference [22] is equivalent to the original definition of non-interference [12] for deterministic systems but also suitable for nondeterministic systems. Intuitively, noninference assumes two security domains, *high* and *low*, and requires that any observable low-level behavior is also possible in absence of any high level behavior. Hence, any user who observes low level behavior is unable to detect whether any high level behavior has taken place at all.

Technically, it means that when the system behavior is described as a set of event traces (consisting of high- as well as low-level events), then for every trace its projection to low level events must also be a possible trace.

An alphabet E denotes a finite set of *events* of a system. We assume a partition on E into *high* (H) and *low* (L) events. The behavior of a system is specified by a set $Tr \subseteq E^*$ of event-traces. Noninference is formally defined by

$$NF(Tr) \equiv \forall \tau \in Tr. \tau \restriction_L \in Tr$$

where $\tau \restriction_L$ denotes the trace resulting from τ with all *high* events being removed.

3.2 Decidability of Noninference

Noninference is a closure property. It requires the set of system behaviors to be closed under purging of high level events. Despite its simplicity it has been shown to be undecidable for pushdown systems.

Theorem 2. *The problem of model checking noninference for deterministic pushdown systems is undecidable.* *[7]*

However, when the system is specified as finite state system, then model checking noninference becomes decidable:

Theorem 3. *The problem of model checking noninference for finite state systems is decidable.* *[9]*

In [9] the authors developed an automata based decision procedure that can automatically verify for a finite state system whether it satisfies noninference or not. We will now extend this result and show, that noninference is also decidable for VPLs when the stack is observable. We first assign every event in $\widetilde{\Sigma}$ a security domain *L* or *H*. Assuming, that the stack and hence all call and return events, which modify the stack, are low level observable, we require $\Sigma_c \cup \Sigma_r \subseteq L$.

We show, that for VPLs with such a domain assignment the question of whether a system satisfies noninference is decidable:

Theorem 4. *Let Tr be a VPL over a partitioned alphabet $(\Sigma_c, \Sigma_r, \Sigma_{int})$. Let $\mathrm{H} \subseteq \Sigma_{int} \setminus X$ and $\mathrm{L} = \Sigma_c \cup \Sigma_r \cup X$ with $X \subseteq \Sigma_{int}$, i.e. all call and return events are low level.*

Then the question whether $NF(Tr)$ holds is decidable.

This theorem can be shown by reduction of satisfiability of NF to language inclusion and the following two lemmas.

Lemma 1. *Let Tr be a VPL over a partitioned alphabet $(\Sigma_c, \Sigma_r, \Sigma_{int})$. Let $\mathrm{L} = \Sigma_c \cup \Sigma_r \cup X$ with $X \subseteq \Sigma_{int}$. Tr satisfies $NF(Tr)$ with respect to L iff $Tr \restriction_{\mathrm{L}} \subseteq Tr$, where $Tr \restriction_{\mathrm{L}} = \{\tau \restriction_{\mathrm{L}} \mid \tau \in Tr\}$ is the projection of Tr upon L.*

This lemma reduces the problem of decidability of NF to language inclusion. A more general version of it has already been shown in [9]. Since VPLs are closed under language inclusion [2], we only need to show that VPLs are also closed under projection to *L*, with $L = \Sigma_c \cup \Sigma_r \cup X$ and $X \subseteq \Sigma_{int}$.

Lemma 2. *If Tr is a VPL on $(\Sigma_c, \Sigma_r, \Sigma_{int})$ then also $Tr\restriction_{\mathrm{L}}$ with $\mathrm{L} = \Sigma_c \cup \Sigma_r \cup X$ and $X \subseteq \Sigma_{int}$ is a VPL.*

Proof. Let $L = \Sigma_c \cup \Sigma_r \cup X$ and $X \subseteq \Sigma_{int}$.

The projection of Tr upon L results from eliminating every symbol which is not in L from all words in Tr. Thus, for all $a \in \Sigma_{int} \setminus X$, all transitions $(q, a, q') \in \delta$ are replaced by ϵ-transitions (q, ϵ, q'). The resulting automaton is a VPA^{sec} accepting $Tr\restriction_L$, which hence is a VPL.

Now theorem 6, i.e. the decidability of $NF(Tr)$ for a VPL Tr, follows directly from lemmas 3 and 4.

4 Decidability of Noninference for a Larger Class of Systems

Looking for a more expressive class of languages for which noninference is decidable, we investigate visibly pushdown languages (VPLs, [2]). They constitute a strict subclass of context-free languages and properly include regular languages. Hence, VPLs are a promising candidate for new decidability results. In contrast to regular languages, they allow for representation of data with both a linear as well as a hierarchical ordering as e.g. structured programs. Despite their expressiveness, VPLs share desirable closure properties with regular languages: they are closed under union, intersection, complement and Kleene-*. Furthermore, many problems like the word problem and the inclusion problem are decidable. VPLs are characterized by visibly pushdown automata (VPA, [2]). For our purpose of showing decidability of noninference for VPA we require a slightly modified definition, which we call VPA^{sec}.

4.1 Visibly Pushdown Automata for Security

Visibly pushdown automata are similar to pushdown systems but operate on a partitioned alphabet $\widetilde{\Sigma} = (\Sigma_c, \Sigma_r, \Sigma_{int})$ that determines which events modify the stack in which way: call events (Σ_c) push symbols onto the stack, return events (Σ_r) pop symbols from it and internal events (Σ_{int}) leave the stack unmodified.

We define visibly pushdown automata for security (VPA^{sec}) as follows:

Definition 41 (VPA^{sec}) *A VPA^{sec} A on finite words over $\widetilde{\Sigma} = (\Sigma_{int}, \Sigma_c, \Sigma_r)$ is a tuple $A = (Q, q_{in}, \widetilde{\Sigma}, \Gamma, \delta^{sec}, Q_F)$ where Q is a finite set of states, $q_{in} \in Q$ is an initial state, Γ is a finite stack alphabet that contains $\perp$ as a special bottom element, and $\delta^{sec} \subseteq (Q \times \Sigma_c \times Q \times (\Gamma \setminus \{\perp\})) \cup (Q \times \Sigma_r \times \Gamma \times Q) \cup (Q \times \Sigma_{int} \cup \{\epsilon\} \times Q)$ such that $Q_F \subseteq Q$ is a set of final states.*

The relation $reach_A : Q \to 2^Q$ determines the ϵ-closure of states in A. It is the smallest set satisfying $s \in reach_A(s)$ and $\forall q' \in Q. \forall q \in reach_A(s) : (q, \epsilon, q') \in \delta^{sec} \Rightarrow q' \in reach_A(s)$.

The automaton A accepts a word $w \in \Sigma^*$ (with length $|w| = k + 1$) iff there is a sequence $\rho = (q_0, \sigma_0) \ldots (q_k, \sigma_k)$ with $q_0 = reach_A(q_{in})$, $\sigma_0 = \perp$ and $q_k \in Q_F$ such that for all $0 \leq i < k$ the following holds:

Push: If $w_i \in \Sigma_c$ then there is a $\gamma \in \Gamma$ and $q' \in Q$ such that $(q_i, w_i, q', \gamma) \in \delta^{sec}$ and $q_{i+1} \in reach_A(q')$ and $\sigma_{i+1} = \gamma.\sigma_i$

Pop: If $w_i \in \Sigma_r$ then there is a $\gamma \in \Gamma$ and $q' \in Q$ such that $(q_i, w_i, \gamma, q') \in \delta^{sec}$ and $q_{i+1} \in reach_A(q')$ and either $\sigma_i = \gamma.\sigma_{i+1}$ or $\gamma = \sigma_{i+1} = \sigma_i = \bot$.

Internal: If $w_i \in \Sigma_{int}$ then there is $q' \in Q$ such that $(q_i, w_i, q') \in \delta^{sec}$ and $q_{i+1} \in reach_A(q')$ and $\sigma_{i+1} = \sigma_i$

The language $\mathcal{L}(A)$ recognized by A, is the set of all words $w \in \Sigma^*$ accepted by A.

The difference between the original definition of VPA given in [2] and the above definition of VPA^{sec} is that we allow the occurrence of ϵ-transitions as internal events. Like in the original definition, call and return transitions, that modify the stack, are not possible on ϵ-events. Interestingly, we can show that our definition of VPA^{sec} has the same expressiveness as the original definition:

Theorem 5. *Each VPA^{sec} A over $\widetilde{\Sigma}$ can be transformed into an (original) VPA A' over $\widetilde{\Sigma}$ with $\mathcal{L}(A) = \mathcal{L}(A')$.*

Proof sketch: We construct A' using a power set construction. The set of states of A' is the power set of the states of A. A transition from one set of states to another corresponds to the set of possible moves in A, which is determined by the ϵ-closure of a transition in A. As a result, we have a VPA with a finite set of states and a transition function without ϵ-transitions.

The new VPA accepts the same language as the VPA^{sec}. One can show by induction over finite words w that for each A-run over w there is a corresponding A'-run and vice versa. Since ϵ-transitions do not alter the stack, the nondeterminism only affects internal transitions. Consequently, the power set construction is similar to that for finite automata.

4.2 Decidability of Noninference for VPLs

We are now ready to identify VPLs as a class of systems that is more expressive than finite state systems but for which noninference is still decidable. To show this, we first assign every event in $\widetilde{\Sigma}$ a security domain *L* or *H*. Assuming, that the stack and hence all call and return events, which modify the stack, are low level observable, we require $\Sigma_c \cup \Sigma_r \subseteq L$.

We show, that for VPLs with such a domain assignment the question of whether a system satisfies noninference is decidable:

Theorem 6. *Let Tr be a VPL over a partitioned alphabet $(\Sigma_c, \Sigma_r, \Sigma_{int})$. Let $\mathrm{H} \subseteq \Sigma_{int} \setminus X$ and $\mathrm{L} = \Sigma_c \cup \Sigma_r \cup X$ with $X \subseteq \Sigma_{int}$, i.e. all call and return events are low level.*

Then the question whether $NF(Tr)$ holds is decidable.

This theorem can be shown by reduction of satisfiability of NF to language inclusion and the following two lemmas.

Lemma 3. *Let Tr be a VPL over a partitioned alphabet $(\Sigma_c, \Sigma_r, \Sigma_{int})$. Let $\mathrm{L} = \Sigma_c \cup \Sigma_r \cup X$ with $X \subseteq \Sigma_{int}$. Tr satisfies $NF(Tr)$ with respect to L iff $Tr \upharpoonright_{\mathrm{L}} \subseteq Tr$, where $Tr \upharpoonright_{\mathrm{L}} = \{\tau \upharpoonright_{\mathrm{L}} \mid \tau \in Tr\}$ is the projection of Tr upon L.*

This lemma reduces the problem of decidability of NF to language inclusion. A more general version of it has already been shown in [9]. Since VPLs are closed under language inclusion [2], we only need to show that VPLs are also closed under projection to L, with $L = \Sigma_c \cup \Sigma_r \cup X$ and $X \subseteq \Sigma_{int}$.

Lemma 4. *If Tr is a VPL on $(\Sigma_c, \Sigma_r, \Sigma_{int})$ then also $Tr \upharpoonright_{\mathrm{L}}$ with $\mathrm{L} = \Sigma_c \cup \Sigma_r \cup X$ and $X \subseteq \Sigma_{int}$ is a VPL.*

Proof. Let $L = \Sigma_c \cup \Sigma_r \cup X$ and $X \subseteq \Sigma_{int}$.

The projection of Tr upon L results from eliminating every symbol which is not in L from all words in Tr. Thus, for all $a \in \Sigma_{int} \setminus X$, all transitions $(q, a, q') \in \delta$ are replaced by ϵ-transitions (q, ϵ, q'). The resulting automaton is a VPA^{sec} accepting $Tr \upharpoonright_L$, which hence is a VPL.

Now theorem 6, i.e. the decidability of $NF(Tr)$ for a VPL Tr, follows directly from lemmas 3 and 4.

5 Decidability Results for the Modular Assembly Kit for Security

Besides noninference, many more non-interference-like information flow properties have been defined in the literature. The Modular Assembly Kit for Security (MAKS) [15] provides a uniform representation of a wide range of these properties. The aim of this section is to generalize our decidability result from the previous section and show that the Basic Security Predicates (BSPs) in MAKS and, hence, a large set of the classical information flow properties are decidable for Visibly Pushdown Systems.

In subsection 5.1 we introduce the MAKS framework. In section 5.2 we present the decidability of all MAKS predicates. Section 5.3 give a detailed proof for two of the BSPs and shows how the proof leads to an automatic decision procedure for these properties.

5.1 Modular Assembly Kit for Security (MAKS)

MAKS is an approach to uniformly formalize most possibilistic information flow properties known from the literature. Based on sets of traces as the system model, Mantel has defined a set of *Basic Security Predicates (BSPs)* that serve as building blocks for these information flow properties. Before recalling the definitions of BSPs, we first introduce some notational conventions.

BSPs are defined over sets of traces of events from an alphabet E. A trace is a finite sequence of events modeling a particular system behavior. A set of traces is a prefix-closed language $Tr \subseteq E^*$. It includes the empty string denoted by ϵ. We range over elements of Tr by $\tau, \tau', \tau_1, \tau_2, \ldots$ and over sequences in E^* by $\alpha, \alpha', \beta, \beta', \ldots$. We represent the concatenation of α and β by $\alpha\beta$. For a sequence $\alpha \in E^*$ and a set $X \subseteq E$ the term $\alpha \upharpoonright_X$ denotes the sequence obtained by omitting all elements in α that are in $\overline{X}$, with $\overline{X} = E \setminus X$. Considering two sequences α and β, they are equal *with disregard for corrections on events in* X iff $\alpha \upharpoonright_{\overline{X}} = \beta \upharpoonright_{\overline{X}}$. We denote this equivalence with $\alpha =_X \beta$

and extend this notation to languages. Consequently, for two languages $Tr_1, Tr_2 \subseteq E^*$ we write $Tr_1 \subseteq_X Tr_2$ iff $Tr_1 \upharpoonright_{\overline{X}} \subseteq Tr_2 \upharpoonright_{\overline{X}}$.

In MAKS, each basic security predicate is defined with respect to a view $\mathcal{V} = (V, N, C)$, which constitutes a partition of the events in E. The set V of *visible* events denotes the set of events that are directly observable by an adversary. The set C of events is the set of events that can not be observed but may not be deducible from observations either. Finally the set N is the set of events that cannot be directly observed but are not confidential either.

The BSPs require, that the occurrence of confidential events may neither increase nor reduce the possible observable behaviors, i.e. the BSPs express the kind of independence of visible and confidential events.

Technically, each trace that is perturbed by either deletion of the last confidential event or insertion of a last confidential event must be correctable by deletion or insertion of N-events such that the resulting trace is again a possible system trace.

Four different types of BSPs can be distinguished in MAKS, depending on the corrections they allow: non-strict (allowing corrections anywhere in the trace), strict (allowing no corrections at all), backward strict (allowing corrections only after the event where the perturbation happened), and forward correctable.

In Figure 1, we recall the definitions of BSPs from [15]. Let $A \subseteq E$ be a set of events admissible for correction, and let $C' \subseteq C$ and $V' \subseteq V$.

5.2 Decidability Results in MAKS

Like *noninference*, the BSPs from MAKS are undecidable for context-free languages [7] but decidable for regular languages [9]. We will now generalize Theorem 6 from section 4 and show all BSPs in MAKS also to be decidable for VPLs. For this, we adapt the assumptions of Theorem 6 and assume all call and return events to be visible (V) in terms of a MAKS-view $\mathcal{V}$. The proof of the generalization provides an automatic decision procedure based on VPAs.

Theorem 7 (BSPs decidable for VPLs). *Let Tr be a VPL over $\widetilde{\Sigma}$ and $\mathcal{V} = (V, N, C)$ a view with $C, N \subseteq \Sigma_{int}$. Let $A = \Sigma_c \cup \Sigma_r \cup X$ be the set of admissible events for some $X \subseteq \Sigma_{int}$ and let $C' \subseteq C$ and $V' \subseteq V$.*

Then all BSPs in MAKS are decidable for Tr with respect to view $\mathcal{V}$ and event sets A, C' and V'.

Proof sketch: We reduce the problem of verifying BSPs to a language inclusion problem, where the language Tr gets transformed by BSP-specific language-theoretic operations. Deciding a strict BSP P for Tr involves applying language-theoretic operations f and f' to Tr and checking whether $f(Tr) \subseteq f'(Tr)$. Non-strict BSPs are decided by checking whether $f(Tr) \subseteq_N f'(Tr)$. The latter can be expressed as a language inclusion having projected both languages onto $\overline{N}$ first. Thus, deciding a BSP is reduced to a language inclusion problem. Recall that language inclusion is decidable for VPLs. Furthermore, due to Lemma 4 and our restriction on the range of N, VPLs are closed under projection to $\overline{N}$.

1. Tr satisfies R (Removal of Events) iff for all $\tau \in Tr$ there exists $\tau' \in Tr$ such that $\tau' \restriction_C = \epsilon$ and $\tau' \restriction_V = \tau \restriction_V$).
2. Tr satisfies D (Stepwise Deletion of events) iff for all $\alpha c \beta \in Tr$, $c \in C$ such that $\beta \restriction_C = \epsilon$, we have $\alpha' \beta' \in Tr$ with $\alpha' \restriction_{V \cup C} = \alpha \restriction_{V \cup C}$ and $\beta' \restriction_{V \cup C} = \beta \restriction_{V \cup C}$.
3. Tr satisfies I (Insertion of events) iff for all $\alpha\beta \in Tr$ such that $\beta \restriction_C = \epsilon$, we have $\alpha' c \beta' \in Tr$, for every $c \in C$ with $\beta' \restriction_{V \cup C} = \beta \restriction_{V \cup C}$, $\alpha' \restriction_{V \cup C} = \alpha \restriction_{V \cup C}$.
4. Tr satisfies IA^A (Insertion of A-admissible events) iff for all $\alpha\beta \in Tr$ with $\beta \restriction_C = \epsilon$ and there exists $\gamma c \in Tr$, $c \in C$ with $\gamma \restriction_A = \alpha \restriction_A$, we have $\alpha' c \beta' \in Tr$ with $\beta' \restriction_{V \cup C} = \beta \restriction_{V \cup C}$, $\alpha' \restriction_{V \cup C} = \alpha \restriction_{V \cup C}$).
5. Tr satisfies BSD (Backwards Strict Deletion) iff for all $\alpha c \beta \in Tr$, $c \in C$ such that $\beta \restriction_C = \epsilon$, we have $\alpha \beta' \in Tr$ with $\beta' \restriction_{V \cup C} = \beta \restriction_{V \cup C}$.
6. Tr satisfies BSI (Backwards Strict Insertion) iff for all $\alpha\beta \in Tr$ such that $\beta \restriction_C = \epsilon$, we have $\alpha c \beta' \in Tr$, for every $c \in C$ with $\beta' \restriction_{V \cup C} = \beta \restriction_{V \cup C}$.
7. Tr satisfies $BSIA^A$ (Backwards Strict Insertion of A-admissible events) iff for all $\alpha\beta \in Tr$ with $\beta \restriction_C = \epsilon$ and there exists $\gamma c \in Tr$, $c \in C$ with $\gamma \restriction_A = \alpha \restriction_A$, we have $\alpha c \beta' \in Tr$ with $\beta' \restriction_{V \cup C} = \beta \restriction_{V \cup C}$.
8. Tr satisfies FCD (Forward Correctable Deletion) iff for all $\alpha c v \beta \in Tr$, $c \in C'$, $v \in V'$ with $\beta \restriction_C = \epsilon$ we have $\alpha \delta v \beta' \in Tr$ where $\delta \in (N')^*$ and $\beta' \restriction_{V \cup C} = \beta \restriction_{V \cup C}$.
9. Tr satisfies FCI (Forward Correctable Insertion) iff for all $\alpha v \beta \in Tr$, $v \in V'$ such that $\beta \restriction_C = \epsilon$, we have $\alpha c \delta v \beta' \in Tr$, for every $c \in C'$ with $\delta \in (N')^*$ and $\beta' \restriction_{V \cup C} = \beta \restriction_{V \cup C}$.
10. Tr satisfies $FCIA^A$(Forward Correctable Insertion of A-admissible events) iff for all $\alpha v \beta \in Tr$, $v \in V'$ with $\beta \restriction_C = \epsilon$ and there exists $\gamma c \in Tr$, $c \in C'$ with $\gamma \restriction_A = \alpha \restriction_A$, we have $\alpha c \delta v \beta' \in Tr$ with $\delta \in (N')^*$ and $\beta' \restriction_{V \cup C} = \beta \restriction_{V \cup C}$.
11. Tr satisfies SD (Strict Deletion) iff for all $\alpha c \beta \in Tr$, $c \in C$ such that $\beta \restriction_C = \epsilon$, we have $\alpha\beta \in Tr$.
12. Tr satisfies SI (Strict Insertion) iff for all $\alpha\beta \in Tr$ such that $\beta \restriction_C = \epsilon$, we have $\alpha c \beta \in Tr$, for every $c \in C$.
13. Tr satisfies SIA^A (Strict Insertion of A-admissible events) iff for all $\alpha\beta \in Tr$ such that $\beta \restriction_C = \epsilon$ and there exists $\gamma c \in Tr$, $c \in C$ with $\gamma \restriction_A = \alpha \restriction_A$, we have $\alpha c \beta \in Tr$.

Fig. 1. Basic Security Predicates (BSPs) in MAKS

The required language-theoretic operations and their correspondence to BSPs have already been established in [9]. It remains to be shown that all language-theoretic operations f and f' preserve the language class of VPLs. This can be shown by constructing a VPA for each of the functions. We present the constructions for all operations in subsection 5.3 and appendix C. The constructions are such that we receive an automatic decision procedure. This concludes our proof sketch. □

Many known non-interference-like security properties from the literature can be expressed in the MAKS framework. Hence, we immediately get the following result:

Corollary 1. *The problem of verifying information flow properties that are expressible as conjunctions of BSPs w.r.t. a view $\mathcal{V}$ is decidable for VPLs if all call and return events are visible w.r.t. view $\mathcal{V}$.*

1. *l-del-mark*(Tr). For every trace τ in Tr, the last symbol of C is replaced by $\natural$.

$$\textit{l-del-mark}(Tr) := \{\alpha\natural\beta \mid \alpha c\beta \in Tr,\ c \in C,\ \beta\upharpoonright_C = \epsilon\}$$

2. *l-ins-adm-mark*$^A(Tr)$. For every trace τ in Tr, a symbol c from C is inserted, followed by $\natural$, at a position where no more symbols of C are seen, however only if there is a trace τ' in Tr that ends in c and is similar (in the A-events) to the prefix of τ upto the position where c was inserted in τ. It is defined as

$$\textit{l-ins-adm-mark}^A(Tr) := \{\alpha c\natural\beta \mid \alpha\beta \in Tr,\ \beta\upharpoonright_C = \epsilon, \text{ and there exists } \gamma c \in Tr \text{ with } \gamma\upharpoonright_A = \alpha\upharpoonright_A,\ c \in C\}$$

3. $mark(Tr)$. Yields a marked language, resulting from inserting a single occurrence of $\natural$ at any position of all possible traces.

$$mark(Tr) := \{\alpha\natural\beta \mid \alpha\beta \in Tr\}$$

4. *Marked Projection*. This operates on a marked language Tr^M over $\widetilde{\Sigma}^M$ where each trace in Tr^M contains exactly one $\natural$. Every trace in Tr^M is projected to Y after the mark.

$$Tr^M\upharpoonright^m_Y := \{\alpha\natural\beta' \mid \alpha\natural\beta \in Tr^M,\ \beta' = \beta\upharpoonright_Y\}$$

Fig. 2. Language-theoretic operations

For instance, generalized non-interference (GNI) [18] and its modification GNI^* (a property introduced in [17] that is less restrictive and more appropriate than GNI in the sense that not every sequence of confidential events is required to be possible), can both be expressed in the MAKS as conjunctions of BSPs. While GNI is equivalent to the conjunction of BSD and BSI, the property GNI^* is defined as the conjunction of BSD and $BSIA^C$ with respect to a view $\mathcal{V} = (V, N, C)$. Thus, if we can verify BSD and $BSIA^X$, we can also verify GNI^*.

We will now detail the proof of Theorem 7 exemplarily for the BSPs BSD and $BSIA^C$. As was shown in [9], the problem of verifying these BSPs can be characterized as the following language inclusion problems.

1. Tr satisfies $BSD_{\mathcal{V}}(Tr)$ iff *l-del-mark*$(Tr)\upharpoonright^m_N \subseteq mark(Tr)\upharpoonright^m_N$
2. Tr satisfies $BSIA^A_{\mathcal{V}}(Tr)$ iff *l-ins-adm-mark*$^A(Tr)\upharpoonright^m_N \subseteq mark(Tr)\upharpoonright^m_N$

These characterizations use the operations *l-del-mark*, *l-ins-adm-mark*^{A}, $mark$, and *marked projection* ($\upharpoonright^m_N$). We recall their definitions from [9] in Figure 2. Some operations add a special mark $\natural$ to the traces, which is not part of the original alphabet. In that case, we write $\widetilde{\Sigma}^M$ to mean the partitioned alphabet including the mark, $(\Sigma_c, \Sigma_r, \Sigma_{int} \cup \{\natural\})$.

A complete list of operations required for characterizing BSPs is provided in appendix A. A list of all relations between BSPs and the according language-theoretic operations can be found in appendix B.

5.3 VPA constructions for BSD and $BSIA$

We now complete the proof of Theorem 7 for the properties BSD and $BSIA$. For this, we show that the class of visibly pushdown languages is closed under the language-theoretic operations defined in Figure 2 of subsection 5.2. As a result, we have decidability of BSPs for VPLs along with an automatic decision procedure. The schema for constructing the automata is inspired by [9], however the automata are now VPAs rather than finite-state machines.

Lemma 5. *Let Tr be a VPL over $\widetilde{\Sigma} = (\Sigma_{int}, \Sigma_c, \Sigma_r)$ with $N, C \subseteq \Sigma_{int}$ for a given view $\mathcal{V} = (V, N, C)$. In addition, let $A = \Sigma_c \cup \Sigma_r \cup X$ for some $X \subseteq \Sigma_{int}$. Then also l-del-mark(Tr) and l-ins-mark(Tr) are visibly pushdown languages.*

Proof. Let $\mathcal{A} = (Q, q_{in}, \widetilde{\Sigma}, \Gamma, \delta, Q_F)$ be a VPA that accepts Tr. Without loss of generality we assume $\mathcal{A}$ to be deterministic.

We prove that applying each of the language-theoretic operations to a VPL again yields a VPL, by showing that there is a VPA $\mathcal{A}' = (Q', q'_{in}, \widetilde{\Sigma}', \Gamma', \delta', Q'_F)$ that accepts the language returned by the operation:

1. *l-del-mark*(Tr). We construct a VPA^{sec} $\mathcal{A}'$ accepting *l-del-mark*(Tr). As in [9], $\mathcal{A}'$ consists of two copies of the set of states of A, while leaving the set of input and stack symbols unchanged. The initial states are within the first set of states; the final states are within the second set. The idea is to allow to replace the last occurrence of $c \in C$ non-deterministically by adding $\natural$-transitions on top of every c-transition, but whose target now is in the second set of states, where all c-transitions are removed in order to ensure the c replaced by $\natural$ was the last one that would have been read. The construction is sketched in Figure 3.

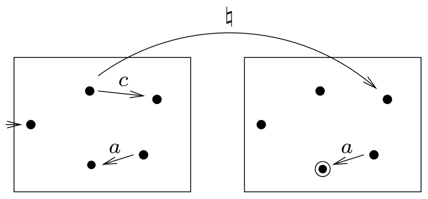

Fig. 3. *l-del-mark*(Tr) with $c \in C \subseteq \Sigma_{int}$ and a representing all other transitions, including calls and returns for a $\gamma \in \Gamma$.

Formally, the new VPA^{sec} $\mathcal{A}'$ is defined by $Q' = Q \times \{1,2\}$, $q'_{in} = (q_{in}, 1)$, $\widetilde{\Sigma}' = \widetilde{\Sigma}^M$, $\Gamma' = \Gamma$, $Q'_F = Q_F \times \{2\}$ and δ' with the following transitions:
For all $q, q' \in Q$, $a_{int} \in \Sigma_{int}$, $a_{call} \in \Sigma_c$, $a_{ret} \in \Sigma_r$:

Internal:	$((q,1), a_{int}, (q',1)) \in \delta' \Leftrightarrow (q, a_{int}, q') \in \delta$
	$((q,2), a_{int}, (q',2)) \in \delta' \Leftrightarrow (q, a_{int}, q') \in \delta \wedge a_{int} \notin C$
	$((q,1), \natural, (q',2)) \in \delta' \Leftrightarrow (q, a_{int}, q') \in \delta \wedge a_{int} \in C$
Call:	$((q,i), a_{call}, (q',i), \gamma) \in \delta' \Leftrightarrow (q, a_{call}, q', \gamma) \in \delta, \quad i \in \{1,2\}$
Return:	$((q,i), a_{ret}, \gamma, (q',i)) \in \delta' \Leftrightarrow (q, a_{ret}, \gamma, q') \in \delta, \quad i \in \{1,2\}$

Hence, $\mathcal{A}'$ accepts *l-del-mark*(Tr), which therefore is a visibly pushdown language over $\widetilde{\Sigma}^M$.

2. *l-ins-adm-mark*$^A(Tr)$. First, we construct a VPA $\mathcal{B}$ for $Tr(\mathcal{A}) \upharpoonright_A$ by replacing all transitions (q, a, q') where $a \notin A$ with (q, ϵ, q'). Then we construct $\mathcal{A}'$ as follows. Let $\mathcal{A}'$ be given by $Q' = (Q \times 2^Q) \cup (Q \times \{1, 2\})$, $q'_{in} = (q_{in}, reach_{\mathcal{B}}(q_{in}))$, $\widetilde{\Sigma}' = \widetilde{\Sigma}^M$, $\Gamma' = \Gamma$, $Q'_F = Q_F \times \{2\}$, and by δ', which is the smallest set satisfying:
   - If $a \in \Sigma_{int}$ and $a \notin A$, then for each transition $(p, a, q) \in \delta$ of $\mathcal{A}$ we have $((p, T), a, (q, T)) \in \delta_{\mathcal{A}'}$ for all $T \in 2^Q$
   - For each transition of $\mathcal{A}$ that reads an $a \in A$ and goes from p to q, maybe depending on a $\gamma \in \Gamma$ (there is at most one such γ for each a, p and q, because $\mathcal{A}$ is deterministic) we have a similar transition reading a but going from (p, T) to (q, U) for all $T \in 2^Q$ where U is the set of all states that are reachable from a state $t \in T$ in $\mathcal{B}$ with a corresponding a-transition and maybe ϵ-transitions.
   - For each transition $(t, c, q) \in \delta$ of $\mathcal{A}$ with $t \in Q$, $q \in Q_F$ and $c \in C$, then for all $T \in 2^Q$ with $t \in T$ and for all $p \in Q$ we have $((p, T), c, (p, 1)) \in \delta'$.
   - For each $q \in Q$ of $\mathcal{A}$ we have $((q, 1), \natural, (q, 2)) \in \delta'$.
   - For each transition in δ that reads an $a \notin C$ and that goes from p to q, there is a similar transition in δ' reading a going from $(p, 2)$ to $(q, 2)$.

   That is, in the first part of the automaton we keep track of the word being read and simultaneously of all states that are A-admissible (they are similar to the current word's prefix in terms of projection onto A). If from that set of states we could reach a final state with a c-transition, we additionally allow a $c \in C$ that leads to the second part of $\mathcal{A}'$, modeling the A-admissible insertion of c. Consecutively, we enforce reading a $\natural$, before allowing to proceed, however without reading any further $c \in C$. Hence, $\mathcal{A}'$ recognizes *l-ins-adm-mark*$^A(Tr)$, which thus is a VPL over $\widetilde{\Sigma}^M$.
3. $mark(Tr)$. We construct VPA^{sec} $\mathcal{A}'$ consisting of two copies of the sets of state in $\mathcal{A}$. The initial states are in the first copy and the final states are in the second copy. In both copies, we leave all transitions as they were but add an additional $\natural$-transition from each state in the first set of states to its corresponding state in the second set of states. Thus, we allow only sequences where exactly one $\natural$ has occurred at an arbitrary position compared to the original trace. Hence, $mark(Tr)$ is a VPL over $\widetilde{\Sigma}^M$.
4. $Tr^M \upharpoonright^m_Y$. We construct VPA^{sec} $\mathcal{A}'$ consisting of two copies of the sets of states in $\mathcal{A}$. The initial states are in the first copy and the final states are in the second copy. The transitions in the first set of states are left as they were, in the second set of states we remove all transitions for symbols not in Y. We add a $\natural$-transition from each state in the first set of states to its corresponding state in the second set of states. By this means, each trace in Tr^M is projected to Y only after the single occurrence of $\natural$.

As a result, *BSD* and *BSIA* are decidable for BSPs. A similar lemma can be shown for all other language-theoretic operations provided in appendix A. This completes our proof of Theorem 7.

6 Deciding Intransitive Information Flow Properties

In this chapter, we present an approach to verifying intransitive information flow properties with visibly pushdown automata. Intransitive policies allow information flow across intermediate domains while prohibiting immediate flow between certain domains. For example, consider a scenario where messages may be sent from component A to a component B only if they are sent via an encryption component C, while direct communication between A and B is forbidden. Then the information flow policy is intransitive, whereas transitivity would imply direct flow from A to B to be legitimate.

Intransitive properties have been developed and analyzed [23] [21] [10] [4], however for varying system models.

In [16], Mantel proposed *extended views* in order to allow for intransitive security policies to be expressed in a similar notion as the BSPs in MAKS. As a result, new security properties were introduced, prominently $IBSD$ and $IBSIA$ (*intransitive backwards strict deletion of confidential events* and *intransitive backwards strict insertion of admissible confidential events*). They not only depend on a view $\mathcal{V}$ but also on an extension set $X \subseteq V$ of events. The extension set X extends the view of domain V in the sense that visible events occurring after an event in X may depend on confidential events preceding this event. Intuitively, the events in X can be seen as downgraders for confidential events.

In this section, we extend the result of the previous section and provide a decision procedure for $IBSD$. To this end, we define a new language-theoretic operation *il-del-mark*^{X} and prove that deciding $IBSD$ can be reduced to language inclusion using *il-del-mark*^{X}. We then show that VPLs are closed under *il-del-mark*^{X}. We can also get a similar result for $IBSIA$, however, the proof is structurally similar and is hence not included in this paper.

Definition 61 (IBSD and IBSIA) *Let $\mathcal{V} = (V, N, C)$ and $X \subseteq V$.*

1. *Tr satisfies IBSD$_{\mathcal{V}}^X$ iff for all $\alpha c \beta \in Tr$, $c \in C$ such that $\beta \restriction_{C \cup X} = \epsilon$, we have $\alpha\beta' \in Tr$ with $\beta \restriction_V = \beta' \restriction_V$ and $\beta' \restriction_{C\cup X} = \epsilon$.*
2. *Tr satisfies IBSIA$_{\mathcal{V}}^X$ iff for all $\alpha\beta \in Tr$, $c \in C$ such that $\beta \restriction_{C\cup X} = \epsilon$ and $\alpha c \in Tr$, we have $\alpha c \beta' \in Tr$ with $\beta \restriction_V = \beta' \restriction_V$ and $\beta' \restriction_{C \cup X} = \epsilon$.*

[16]

We can adapt the language-theoretic operations such that the new BSPs can be decided with LTOs. The new operations can be represented by VPAs, resulting in the decidability of intransitive BSPs for VPLs. Here, we only present results for $IBSD$. However, *l-ins-adm-mark*^{A} can be extended likewise, giving us a similar decidability result for $IBSIA$.

Definition 62 (*il-del-mark*^{X}) *For a language Tr and an extension set $X \subseteq V$ we define il-del-mark$^X(Tr) := \{\alpha\natural\beta \mid \alpha c\beta \in Tr, c \in C, \beta \restriction_{C\cup X} = \epsilon\}$*

Theorem 8. *Tr satisfies IBSD if and only if*

$$\textit{il-del-mark}^X(Tr) \restriction^m_N \subseteq_N mark(Tr) \restriction^m_N$$

Proof.

- $\Rightarrow$ Suppose Tr satisfies $IBSD$. Any trace $\tau \in$ *il-del-mark*$^X(Tr) \restriction^m_{\overline{N}}$ has the form $\alpha\natural\beta'$ with $\beta \restriction_{C \cup X} = \epsilon$ and there is an $\alpha\natural\beta \in$ *il-del-mark*$^X(Tr)$ with $\beta \restriction_{\overline{N}} = \beta'$. Then there must be a $c \in C$ with $\alpha c\beta \in Tr$ and $\beta \restriction_{C \cup X} = \epsilon$. Tr satisfying $IBSD$ implies that there is $\alpha\beta'' \in Tr$ with $\beta'' =_N \beta$. Then $\alpha\natural\beta'' \in mark(Tr)$ holds by definition and, since $\beta'' =_N \beta =_N \beta'$, implies $\alpha\natural\beta' \in mark(Tr) \restriction^m_{\overline{N}}$. Hence *il-del-mark*$^X(Tr) \restriction^m_{\overline{N}} \subseteq_N mark(Tr) \restriction^m_{\overline{N}}$.
- $\Leftarrow$ Suppose *il-del-mark*$^X(Tr) \restriction^m_{\overline{N}} \subseteq_N mark(Tr) \restriction^m_{\overline{N}}$. For any trace $\alpha c\beta \in Tr$ with $c \in C$ and $\beta \restriction_{C \cup X} = \epsilon$ we have $\alpha\natural\beta \in$ *il-del-mark*$^X(Tr)$ and thus $\alpha\natural\beta' \in$ *il-del-mark*$^X(Tr) \restriction^m_{\overline{N}}$ for a β' with $\beta \restriction_{\overline{N}} = \beta'$. Due to the initial assumption, $\alpha\natural\beta' \in mark(Tr) \restriction^m_{\overline{N}}$. Thus, there is an $\alpha\natural\beta'' \in mark(Tr)$ with $\beta'' \restriction_{\overline{N}} = \beta'$ and consequently $\beta'' =_N \beta$. Then $\alpha\beta'' \in Tr$, hence Tr satisfies $IBSD$.

Theorem 9 (IBSD Decidable for VPLs). *Let Tr be a VPL over $\widetilde{\Sigma} = (\Sigma_{int}, \Sigma_c, \Sigma_r)$ with $N, C \subseteq \Sigma_{int}$ for a given extended view $((V, N, C), X)$ with extension set $X \subseteq V$. Then also il-del-mark$^X(Tr)$ is a visibly pushdown language.*

Proof sketch: We have already shown how $IBSD$ can be expressed language-theoretically using *il-del-mark*^{X}. What remains to be shown in order to get a decision procedure for $IBSD$ is that VPLs are closed under *il-del-mark*^{X}.

For a VPA A accepting the language Tr we can construct a VPA A' recognizing *il-del-mark*$^X(Tr)$ similar to the VPA recognizing *l-del-mark*. Again, A' consists of three copies of A, however in the third copy we remove all transitions labeled with $e \in (C \cup X)$. The rest of the construction is as before.

7 Conclusion

In this paper we presented new decidability results for a range of information flow properties. We first proved noninference, a classical non-interference-like information flow property, to be decidable for visibly pushdown systems. Then a generalization of this result for the MAKS framework for information flow properties followed. We chose this framework as it can represent a large class of properties and is more expressive than earlier frameworks. By defining a new language-theoretic operation, we also proved decidability for an intransitive information flow property for VPLs. Though we showed this extension only for one property, we claim that it can be easily extended for other intransitive MAKS-like properties as well. For all decidability results in this paper we assumed call and return events of VPAs to be observable by any low level entity. Without this restriction, VPLs would not be closed under the given language-theoretic operations used in this paper. This does not mean that non-interference properties are not decidable for VPLs in general. Investigating whether such security properties can be decided over VPLs in general, i.e. with confidential call and returns, remains future work.

Acknowledgements. This work was funded by the DFG (German research foundation) under the project FM-SecEng in the Computer Science Action Program (MA 3326/1-3).

References

1. Alur, R., Kumar, V., Madhusudan, P., Viswanathan, M.: Congruences for Visibly Pushdown Languages. In: Proc. of ICALP. pp. 1102–1114 (2005)
2. Alur, R., Madhusudan, P.: Visibly pushdown languages. In: Proc. of the Annual ACM Symposium on Theory of Computing (STOC). pp. 202–211. ACM (2004)
3. Alur, R., Madhusudan, P.: Adding Nesting Structure to Words. In: Proc. of Developments in Language Theory. LNCS, vol. 4036, pp. 1–13. Springer (2006)
4. Best, E., Darondeau, P., Gorrieri, R.: On the Decidability of Non Interference over Unbounded Petri Nets. In: Proc. of International Workshop on Security Issues in Concurrency (SecCo). pp. 16–33 (2010)
5. Chervet, P., Walukiewicz, I.: Minimizing Variants of Visibly Pushdown Automata. In: Kucera, L., Kucera, A. (eds.) Proc. of Mathematical Foundations of Computer Science (MFCS), Lecture Notes in Computer Science, vol. 4708, pp. 135–146. Springer (2007)
6. Dam, M.: Decidability and proof systems for language-based noninterference relations. SIGPLAN 41, 67–78 (January 2006)
7. D'Souza, D., Holla, R., Kulkarni, J., Raghavendra, K., Sprick, B.: On The Decidability of Model-Checking Information Flow Properties. In: Proc. of International Conference on Information Systems Security (ICISS). LNCS, vol. 5352, pp. 26–40. Springer (2008)
8. D'Souza, D., Holla, R., Raghavendra, K., Sprick, B.: Model Checking trace-based information flow properties. Journal of Computer Security 19(1), 101–138 (2011)
9. D'Souza, D., Raghavendra, K., Sprick, B.: An Automata Based Approach for Verifying Information Flow Properties. In: Proc. of Workshop on Automated Reasoning for Security Protocol Analysis (ARSPA). ENTCS, vol. 135, pp. 39–58 (2005)
10. Eggert, S., Van Der Meyden, R., Schnoor, H., Wilke, T.: The complexity of intransitive noninterference. 2011 IEEE Symposium on Security and Privacy pp. 196–211 (2011), `http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5958030`
11. Focardi, R., Gorrieri, R.: A Classification of Security Properties for Process Algebras. Journal of Computer Security 3, 5–33 (1995)
12. Goguen, J.A., Meseguer, J.: Security Policies and Security Models. In: Proc. IEEE Symp. on Security and Privacy. pp. 11–20 (1982)
13. Guttman, J.D., Nadel, M.E.: "What Needs Securing?". In: Proceedings of the IEEE Symposium on security and Privacy. pp. 34–57 (1988)
14. Johnson, D.M., Thayer, F.: Security and the Composition of Machines. In: Proceedings of the Computer Security Foundations Workshop. pp. 72 – 89 (1988)
15. Mantel, H.: Possibilistic Definitions of Security – An Assembly Kit. In: Proc. of IEEE Computer Security Foundations Workshop (CSFW). pp. 185–199. IEEE (2000)
16. Mantel, H.: Information flow control and applications - bridging a gap. In: Proc. of Formal Methods Europe (FME). LNCS, vol. 2021, pp. 153–172. Springer-Verlag (2001)
17. Mantel, H.: A Uniform Framework for the Formal Specification and Verification of Information Flow Security. Ph.D. thesis, Universität des Saarlandes (2003)
18. McCullough, D.: Specifications for multilevel security and a hookup property. In: Proc. of IEEE Symp. Security and Privacy (SP&P) (1987)
19. McLean, J.: A General Theory of Composition for Trace Sets Closed Under Selective Interleaving Functions. In: Proc. of IEEE Symposium on Research in Security and Privacy (SP&P). pp. 79 – 93. IEEE Computer Society Press (1994)
20. van der Meyden, R., Zhang, C.: Algorithmic verification of noninterference properties. Electron. Notes Theor. Comput. Sci. 168, 61–75 (February 2007)

21. Meyden, R.V.D.: What, indeed, is intransitive noninterference? Discovery 4734, 235–250 (2007)
22. O'Halloran, C.: A Calculus of Information Flow. In: Proc. of European Symposium on Research in Computer Security (ESORICS) (1990)
23. Rushby, J.: Noninterference, transitivity and channel-control security policies. Tech. rep., SRI International (1992)
24. Torre, S., Napoli, M., Parente, M.: The word problem for visibly pushdown languages described by grammars. Formal Methods in System Design 31, 265–279 (December 2007)
25. Zakinthinos, A., Lee, E.S.: A general theory of security properties. In: Proc. of the 1997 IEEE Symposium on Security and Privacy (SP '97). p. 94. IEEE Computer Society (1997)

A Language-theoretic operations

Let Tr be language over Σ and let Tr^M be a language over $\widetilde{\Sigma}^M$ with exactly one occurrence of $\natural$ in each word. Σ is partitioned into the three sets V, N, C. Let A and Y denote arbitrary subsets of Σ. V', N' and C' are subsets of V, N and C, respectively.

1. *Projection*. The *Projection* of Tr with respect to Y is defined as
$Tr \upharpoonright_Y := \{\tau \upharpoonright_Y \mid \tau \in Tr\}$ where $\tau \upharpoonright_Y$ is obtained by deleting all symbols from τ that are not in Y
2. *l-del*(Tr). For every string τ in Tr, the last occurring C-symbol is deleted.
l-del$(Tr) := \{\alpha\beta \mid \alpha c\beta \in Tr,\ \beta \upharpoonright_C = \epsilon\}$
3. *l-ins*(Tr). For every string τ in Tr, a symbol from C is inserted at a position where no more symbols of C are seen.
l-ins$(Tr) := \{\alpha c\beta \mid \alpha\beta \in Tr,\ \beta \upharpoonright_C = \epsilon,\ c \in C\}$
4. *l-ins-adm*$^A(Tr)$ is defined as *l-ins-adm*$^A(Tr) :=$
$\{\alpha c\beta \mid \alpha\beta \in Tr,\ \beta \upharpoonright_C = \epsilon,$ there exists $\gamma c \in Tr,\ \gamma \upharpoonright_A = \alpha \upharpoonright_A,\ c \in C\}$
5. *l-del-mark*(Tr). In each string τ in Tr, the last C-symbol is replaced by a mark $\natural$.
l-del-mark$(Tr) := \{\alpha\natural\beta \mid \alpha c\beta \in Tr,\ \beta \upharpoonright_C = \epsilon\}$
6. *l-ins-mark*(Tr). A modified version of *l-ins*(Tr), where a mark is introduced after the new symbol in the string.
l-ins-mark$(Tr) := \{\alpha c\natural\beta \mid \alpha\beta \in Tr,\ \beta \upharpoonright_C = \epsilon,\ c \in C\}$
7. *l-ins-adm-mark*$^A(Tr)$. Similar to *l-ins-mark*(Tr), it is defined as
l-ins-adm-mark$^A(Tr) :=$
$\{\alpha c\natural\beta \mid \alpha\beta \in Tr,\ \beta \upharpoonright_C = \epsilon,$ there exists $\gamma c \in Tr,\ \gamma \upharpoonright_A = \alpha \upharpoonright_A,\ c \in C\}$
8. $mark(Tr)$ is defined as $mark(Tr) := \{\alpha\natural\beta \mid \alpha\beta \in Tr\}$
9. *Marked Projection*. This operates on a marked language. Every string in Tr^M is projected to Y after the mark.
$Tr^M \upharpoonright_Y^m := \{\alpha\natural\beta' \mid \alpha\natural\beta \in Tr^M,\ \beta' = \beta \upharpoonright_Y\}$
10. *l-del-con-mark*$_{C',V'}(Tr)$ is defined as
l-del-con-mark$_{C',V'}(Tr) := \{\alpha v\natural\beta \mid \alpha cv\beta \in Tr,\ \beta \upharpoonright_C = \epsilon,\ c \in C',\ v \in V'\}$
11. *l-ins-con-mark*$_{C',V'}(Tr)$ is defined as
l-ins-con-mark$_{C',V'}(Tr) := \{\alpha cv\natural\beta \mid \alpha v\beta \in Tr,\ \beta \upharpoonright_C = \epsilon,\ c \in C',\ v \in V'\}$
12. *l-ins-adm-con-mark*$^A_{C',V'}(Tr)$ is defined as
l-ins-adm-con-mark$^A_{C',V'}(Tr) :=$
$\{\alpha cv\natural\beta \mid \alpha v\beta \in Tr,\ \beta \upharpoonright_C = \epsilon,$ there exists $\gamma c \in Tr,\ \gamma \upharpoonright_A = \alpha \upharpoonright_A,\ c \in C',\ v \in V'\}$
13. *erase-con-mark*$_{N',V'}(Tr)$ is defined as
erase-con-mark$_{N',V'}(Tr) := \{\alpha v\natural\beta \mid \alpha\delta v\beta \in Tr,\ \delta \in (N')^*,\ v \in V'\}$

B Deciding BSPs by language inclusion

For $Y \subseteq \Sigma$ and languages Tr_1 and Tr_2 we define $\overline{Y} := (\Sigma \backslash Y)$ and $Tr_1 \subseteq_Y Tr_2 := Tr_1 \upharpoonright_{\overline{Y}} \subseteq Tr_2 \upharpoonright_{\overline{Y}}$. Then the BSPs of Mantel can be characterized in terms of the language theoretic operations as follows. Let Tr be a language over Σ and $\mathcal{V} = (V, N, C)$ be a view over Σ. In the following statements we assume the properties are w.r.t. the view $\mathcal{V}$.

1. Tr satisfies R iff $Tr \upharpoonright_V \subseteq_N Tr$
2. Tr satisfies D iff *l-del*$(Tr) \subseteq_N Tr$
3. Tr satisfies I iff *l-ins*$(Tr) \subseteq_N Tr$
4. Tr satisfies IA w.r.t. A iff *l-ins-adm*$^A(Tr) \subseteq_N Tr$

5. Tr satisfies *BSD* iff *l-del-mark*$(Tr)\upharpoonright^m_{\overline{N}} \subseteq_N mark(Tr)\upharpoonright^m_{\overline{N}}$
6. Tr satisfies *BSI* iff *l-ins-mark*$(Tr)\upharpoonright^m_{\overline{N}} \subseteq_N mark(Tr)\upharpoonright^m_{\overline{N}}$
7. Tr satisfies *BSIA* w.r.t. A iff *l-ins-adm-mark*$^A(Tr)\upharpoonright^m_{\overline{N}} \subseteq_N mark(Tr)\upharpoonright^m_{\overline{N}}$
8. Tr satisfies *FCD* w.r.t. V',C',N' iff *l-del-con-mark*$_{C',V'}(Tr)\upharpoonright^m_{\overline{N}} \subseteq_N$ *erase-con-mark*$_{N',V'}(Tr)\upharpoonright^m_{\overline{N}}$
9. Tr satisfies *FCI* w.r.t. V',C',N' iff *l-ins-con-mark*$_{C',V'}(Tr)\upharpoonright^m_{\overline{N}} \subseteq_N$ *erase-con-mark*$_{N',V'}(Tr)\upharpoonright^m_{\overline{N}}$
10. Tr satisfies *FCIA* w.r.t. A, V',C',N' iff *l-ins-adm-con-mark*$^A_{C',V'}(Tr)\upharpoonright^m_{\overline{N}} \subseteq_N$ *erase-con-mark*$_{N',V'}(Tr)\upharpoonright^m_{\overline{N}}$
11. Tr satisfies *SR* iff $Tr\upharpoonright_{\overline{C}} \subseteq Tr$
12. Tr satisfies *SD* iff *l-del*$(Tr) \subseteq Tr$
13. Tr satisfies *SI* iff *l-ins*$(Tr) \subseteq Tr$
14. Tr satisfies *SIA* iff *l-ins-adm*$^A(Tr) \subseteq Tr$

C Proof: VPLs closed under language-theoretic operations

We continue our proof of Theorem 7.

1. *l-del*(L). The construction of $\mathcal{A}'$ for *l-del* is similar to that of *l-del-mark*, however no additional $\natural$ is read. Therefore, ϵ-transitions allow for omitting the last occurrence of a $c \in \Sigma_{int}$. Then $\mathcal{A}'$ accepts *l-del*(L). According to Theorem 5, *l-del*(L) is a visibly pushdown language over $\widetilde{\Sigma}$.
2. *l-ins*(L). Construct $\mathcal{A}'$ such that $\mathcal{A}' = (Q \times \{1,2\}, \{(q_{in},1)\}, \widetilde{\Sigma}, \Gamma, \delta', Q_F \times \{2\})$, where δ' is the smallest set satisfying:
   - For each transition in $\mathcal{A}$ from q to q' with $q, q' \in Q$ there is a similar transition in $\mathcal{A}'$ that now goes from $(q,1)$ to $(q',1)$
   - For each transition from q to q' in $\mathcal{A}$, except for c-transitions with $c \in C$, there is a similar transition in δ', now going from $(q,2)$ to $(q',2)$
   - For all $c \in C$ and for all $q \in Q$ there is a $((q,1),c,(q,2)) \in \delta'$

   As a consequence, $\mathcal{A}'$ is a VPA that allows non-deterministic reads of an additional $c \in C$ with no following $c' \in C$. Hence, $\mathcal{A}'$ accepts *l-ins*(L), thus *l-ins*(L) is a VPL over $\widetilde{\Sigma}$.
3. *l-ins-adm*$^A(L)$. The construction of $\mathcal{A}'$ for *l-ins-adm*$^A(L)$ is similar to that of *l-ins-adm-mark*$^A(L)$, however no additional $\natural$ needs to be read, thus only two copies of the sets of states of $\mathcal{A}$ are necessary.
4. *l-ins-mark*(L). The construction of $\mathcal{A}'$ for *l-ins-mark*(L) is similar to that of *l-ins*(L), however an additional $\natural \in \Sigma_{int}$ must be read after the inserted $c \in C$, which is enforced by a third copy of $\mathcal{A}$.
5. $mark(L)$. Let $\mathcal{A}' = (Q \times \{1,2\}, (q_{in},1), \widetilde{\Sigma}^M, \Gamma, \delta', Q_F \times \{2\})$, where δ' is the smallest set satisfying:
   - For each transition in $\mathcal{A}$ from q to q' with $q, q' \in Q$, there is is similar transition in δ' going from $(q,1)$ to $(q',1)$ and another one going from $(q,2)$ to $(q',2)$.
   - For all $q \in Q$: $((q,1),\natural,(q,2))$ is in δ'

   By this means, a single $\natural$ can be inserted at every position of the originally accepted word. $\mathcal{A}'$ is a VPA, hence $mark(L)$ is a VPL over $\widetilde{\Sigma}^M$.
6. $M\upharpoonright^m_X$. Let $\mathcal{A}' = (Q \times \{1,2\}, (q_{in},1), \widetilde{\Sigma}^M, \Gamma, \delta_{\mathcal{A}'}, Q_F \times \{2\})$, where $\delta_{\mathcal{A}'}$ is the smallest set satisfying:
   - For each transition in $\mathcal{A}$ from q to q' with $q, q' \in Q$ that does not read a $\natural$ there is a similar transition from $(q,1)$ to $(q',1)$ in $\mathcal{A}'$.
   - For each $(q,\natural,q') \in \delta$ there is a $((q,1),\natural,(q',2)) \in \delta_{\mathcal{A}'}$
   - For each transition from q to q' in $\mathcal{A}$ that reads an $a \in X$, there is a similar transition in $\mathcal{A}'$ from $(q,2)$ to $(q',2)$.
   - For each $(q,a,q') \in \delta$ with $a \notin X$ there is a transition $((q,2),\epsilon,(q',2)) \in \delta_{\mathcal{A}'}$

With $\mathcal{A}'$ accepting $M\restriction_X^m$, this language is a VPL over $\widetilde{\Sigma}^M$.

7. *l-del-con-mark*$_{C',V'}(L)$. In order to accept *l-del-con-mark*$_{C',V'}(L)$ by a VPA with $C' \subseteq C$ and $V' \subseteq V$, we a VPA^{sec} $\mathcal{A}'$ that allows non-deterministic skips of $c' \in C'$, then immediately enforces the original successive v'-transition with $v' \in V'$ and an additional transition reading a $\natural$. We achieve this by using four copies of $\mathcal{A}$ where the first copy contains all of the transition in $\mathcal{A}$ and additional ϵ-transitions $((q,1),\epsilon,(q',2))$ leading into the second copy for all $(q,c,q') \in \delta$. From the second copy only v'-transitions exist for each original v'-transition in $\mathcal{A}$, yet leading into the third copy, where a $\natural$-transition leads into the fourth copy from $(q,3)$ to $(q,4)$ for all $q \in Q$. The fourth copy contains all transitions of $\mathcal{A}$ except for c'-transitions with $c' \in C'$.
   $\mathcal{A}'$ accepts *l-del-con-mark*$_{C',V'}(L)$, which therefore is a VPL over $\widetilde{\Sigma}^M$.
8. *l-ins-con-mark*$_{C',V'}(L)$. We construct a VPA^{sec} $\mathcal{A}'$ by adding c'-transitions on top of v' transitions with $c' \in C'$ and $v' \in V'$ and then enforce the actual v'-transition followed by a $\natural$-transition. As a result, *l-ins-con-mark*$_{C',V'}(L)$ is a VPL over $\widetilde{\Sigma}^M$.
9. *l-ins-adm-con-mark*$^A_{C',V'}(L)$. We construct $\mathcal{A}'$ that is similar to the VPA^{sec} recognizing *l-ins-adm*^{A} but with two more subautomata. The states of the first and fourth part correspond to the first and second of the automaton for *l-ins-adm*^{A}, whereas we add additional c'-transitions from the first to the second subautomaton on top of v'-transitions within the first part, followed by the actual v'-transition going from the second to the third part, hereby recognizing c'-transitions followed by v'-transitions. We enforce a subsequent $\natural$-transition leading from the third to the fourth subautomaton, containing the final states and all transitions except those reading any $c \in C$. Consequently, *l-ins-adm-con-mark*$^A_{C',V'}(L)$ is a VPL over $\widetilde{\Sigma}^M$.
10. *erase-con-mark*$_{N',V'}(L)$. We construct $\mathcal{A}'$ by copying $\mathcal{A}$ four times, leaving the transitions of the first and the fourth copy as they were. All transitions of the second copy reading $a \in N'$ are replaced by ϵ-transitions, all v'-transitions from p to q are bent such that they go from p in the second copy to q in the third copy; the remaining transitions within the second copy are deleted. Within the third copy there are no transitions at all. For each state p in the first copy we add ϵ-transitions to p in the second copy. From the third copy to the fourth, we add $\natural$-transitions between corresponding states.
    Therefore *erase-con-mark*$_{N',V'}(L)$ is a VPL over $\widetilde{\Sigma}^M$.