Possibilistic Information Flow Control in MAKS and Action Refinement*

Dieter Hutter

German Research Center for Artificial Intelligence (DFKI GmbH)
Stuhlsatzenhausweg 3, 66123 Saarbrücken, Germany
hutter@dfki.de

Abstract. Formal methods emphasizes the need for a top-down approach when developing large reliable software systems. Refinements are used to map step by step abstract algebraic specifications to executable specifications. Action refinements are used to add detailed design information to abstract actions. Information flow control is used to specify and verify the admissible flow of confidential information in a complex system. However, it is well-known that in general action refinement will not preserve information flow properties which have been proved on an abstract level. In this paper we develop criteria ensuring that these properties are inherited during action refinement. We adopt Mantel's MAKS framework on possibilistic information flow control to formulate security predicates but advance to configuration structures instead of trace event systems to cope with necessary modeling of concurrency.

1 Introduction

In order to deal with the complexity of the development of reliable software systems, formal methods propose the use of a top-down approach. Starting with an abstract specification, step by step more implementation details are added to subsequent specification layers. Various verification methods have been developed to support this stepwise refinement of specifications in order to guarantee that subsequent refined specifications satisfy the requirements of previous layers. While this approach guarantees that, for instance, an implementation level satisfies the logical requirements of the abstract level, it is well known that information flow properties are typically incompatible with refinement (e.g. [12]). Since security orderings are in general neither monotonic nor anti-monotonic with respect to safety orderings, information flow properties are in general not preserved under refinement.

Information flow control (e.g. [16, 22, 13]) relies on the idea of modeling confidentiality (and dually: privacy) of data as restrictions on the flow of information between different domains of a system. Starting with the work of Goguen and Meseguer [9, 10], the restrictions on information flow for deterministic systems

* This work was supported by the German Federal Ministry of Education and Research (BMBF) and the German Research Foundation (DFG).

G. Müller (Ed.): ETRICS 2006, LNCS 3995, pp. 268–281, 2006.

have been formalized as independence properties between actions and observations of domains: Alice's actions are confidential wrt. Charly if his observations are independent of her actions, i.e. if Alice changes her actions this does not cause different observations for Charly. In this case Alice is said to be non-interfering with Charly. For non-deterministic systems, the intuition works backwards: Alice is possibilistically non-interfering with Charly if the observations of Charly can be explained by several, different behaviors of Alice. Thus, Charly's observation does not reveal which actions Alice has chosen.

In the area of information flow control, security predicates are typically closure properties: while an adversary may observe visible parts of a system behaviour he must not be able to predict or deduce the non-visible parts. Thus the set of system traces causing a specific visible behaviour has to contain sufficiently many traces that significantly vary in their confidential behaviour.

Technically, security predicates basically enforce that the occurrences of confidential events in a system trace are independent of the occurrences of visible events, which can be observed by an adversary. As a simple example, suppose $\langle v\rangle$ is a system trace and c a confidential event. A security predicate like the so-called *BSIA* (which we will inspect in more detail later on) demands that observing the visible part v of a system run does not imply that c has not happened at some point. That means, that besides $\langle v\rangle$ also $\langle c, v\rangle$ and $\langle v, c\rangle$ have to be possible system traces. Once we refine v to a sequence v_1, v_2, the refinement of the security property would demand $\langle c, v_1, v_2\rangle$ and $\langle v_1, v_2, c\rangle$ to be system traces of the refined system. However, if we apply the closure property to the refined system, we additionally have to show that $\langle v_1, c, v_2\rangle$ is a possible system trace. This phenomenon closely relates to the problem of differentiating the situations $v||c$ (executing v and c independently) and $v; c, c; v$ (executing v and c in any sequel) which cause different perceptions of possible refinements. While $v||c$ and $v; c, c; v$ imply the same set of traces, namely $\{\langle v, c\rangle, \langle c, v\rangle\}$ their refinements $\{\langle c, v_1, v_2\rangle, \langle v_1, c, v_2\rangle, \langle v_1, v_2, c\rangle\}$ and $\{\langle c, v_1, v_2\rangle\langle v_1, v_2, c\rangle\}$, respectively, do not. In general, interleaving trace equivalence or interleaving bisimulation equivalence are not preserved under action refinement (see [4]). As a consequence, trace-based systems as they are used in MAKS are not appropriate when considering non-atomic events that can be refined later on. Given a trace based specification of an abstract system, we are not able to distinguish whether confidential and visible events run in parallel or in any arbitrary sequel. However, this difference becomes apparent if we refine the system (cf. the example above) and thus can be also observed by an adversary watching the refined system.

In this paper we transfer basic parts of MAKS to so-called configuration structures [6]. Configuration structures are known to preserve bisimulation equivalences during action refinement if some preconditions are met. We base our techniques on the notions developed for the framework MAKS [15] to specify and verify possibilistic information flow policies. We present the translation of the main basic security predicates *BSD* and *BSIA* of MAKS in terms of configuration structures and illustrate under which conditions both are preserved under action refinement.

We start with a brief introduction to the framework MAKS for possibilistic information flow in Section 2 and continue with another introduction to configuration structures in Section 3. In section 4 we introduce the basic concepts of transferring possibilistic information flow control to configuration structures and translate the most prominent security predicates of MAKS into the notion of configuration structures in 5. Finally we compare our approach with related work in 6.

2 MAKS

In this section we will shortly discuss concepts and notation and briefly present the parts of MAKS [15] that we use as a starting point of our paper. Systems are described by an *event system* $ES = (E, I, O, Tr)$, which consists of a set E of events, two sets $I, O \subseteq E$ of input and output events, respectively, and the set $Tr \subseteq 2^{E^*}$ of possible system traces. The set Tr of finite sequences of events is required to be closed under prefixes, i.e. $\alpha.\beta \in Tr$ implies $\alpha \in Tr$, where we write $\alpha.\beta$ for the sequence resulting from concatenating the sequences α and β. We write $\langle e_1, \ldots, e_n \rangle$ for the sequence consisting of the events $e_1, \ldots, e_n$.

In MAKS a security predicate Θ is defined as a conjunction of closure properties on sets of traces. The idea behind using closure properties is the following. Suppose an attacker observes the visible events of a system run (while the confidential ones are invisible). We assume that attackers know all possible system runs, thus they know the set of all possible system runs which might have caused the observed behavior. In particular, an attacker knows the confidential events occurring in these possible runs, and can try to deduce constraints on the confidential events that must have occurred in the observed run. Information flow happens if the attacker is able to deduce knowledge about the occurrence or non-occurrence of confidential events beyond the knowledge already deducible from knowing the system specification, by inspecting the set of runs that are consistent with the observed behavior. A system is secure if this set of runs contains a *sufficient* variety of different possible sequences of confidential events. Closure properties are used to describe this variety because, intuitively, they demand that if there is a possible system run τ satisfying some precondition, then there is also another possible system run τ' such that the attacker cannot distinguish both. Suppose τ' in turn satisfies the precondition. Then we can inductively deduce the existence of another trace τ'' and so on. To assess the security of a system satisfying some basic security predicates we need to understand the guaranteed variance of traces wrt. confidential events being in the transitive closure $\{\tau, \tau', \tau'', \ldots\}$ of an observed system run τ.

The closure properties of sets of possible system traces (parametrized over an arbitrary set of events E) are described by a conjunction of *basic security predicates* (BSPs) and a *view*. A view $\mathcal{V} = (V, N, C)$ for E is a disjoint, exhaustive partition of E and formalises an observer or attacker: C comprises those events whose occurrence or non-occurrence should be confidential for the observer, V represents those events that are directly visible for the observer, and N are all other events. An event system satisfies a security property if each BSP holds for the view and the set of possible system traces. BSPs that we will be using as examples in this paper are *BSD* and *BSIA*[1] defined as

$$\begin{aligned} BSD_{\mathcal{V}}(Tr) \iff & [\forall \alpha, \beta \in E^*, c \in C.\ (\beta.\langle c\rangle.\alpha \in Tr \wedge \alpha|_C = \langle\rangle \\ & \implies \exists \alpha' \in E^*, \tau' \in Tr.\ (\beta.\alpha' = \tau' \wedge \alpha'|_V = \alpha|_V \wedge \alpha'|_C = \langle\rangle))] \end{aligned} \tag{1}$$

$$\begin{aligned} BSIA^{\rho}_{\mathcal{V}}(Tr) \iff & [\forall \alpha, \beta \in E^*, c \in C.\ (\beta.\alpha \in Tr \wedge \alpha|_C = \langle\rangle \wedge Adm^{\rho}_{\mathcal{V}}(Tr, \beta, c) \\ & \implies \exists \alpha' \in E^*, \tau' \in Tr.\ (\beta.\langle c\rangle.\alpha' = \tau' \wedge \alpha'|_V = \alpha|_V \wedge \alpha'|_C = \langle\rangle))] \end{aligned} \tag{2}$$

where $\tau|_D$ is the projection of τ to the events in $D \subseteq E$. $Adm^{\rho}_{\mathcal{V}}(Tr, \beta, c)$ holds if the confidential event c is admissible after the trace β, when only events in the set $\rho(\mathcal{V})$ are considered, i.e. for all functions ρ from views over E to sets of events, we have $\forall \beta \in E^*, c \in C.\ Adm^{\rho}_{\mathcal{V}}(Tr, \beta, c) \iff \exists \gamma \in E^*.\ \gamma.\langle c\rangle \in Tr \wedge \gamma|_{\rho(\mathcal{V})} = \beta|_{\rho(\mathcal{V})}$.

3 Configuration Structures

In this section we summarize the concept of configuration structures and their essential properties. Nevertheless, the reader is referred to the literature, for instance [6, 7], for further particulars. Configuration structures provide a general model to formalize concurrent systems in a modular way while allowing for a stepwise refinement. They have been also used as semantic models for CCS-like [18] languages.

Configuration structures are based on a set of events $\mathcal{E}$ that denote *occurrences* of actions. Thus, each event e is labeled by an action $l(e)$. A concurrent system is described as a configuration structure by defining the possible states $\mathcal{S}$, so called *c*onfigurations, it can reach. Each configuration is a finite set of events. The intuition behind is that this set of events represent the set of actions the system had to perform to reach this particular state. Thus, the configuration of a successor state will always contain the configuration of the original state as a subset. State transitions are implicitly defined by the subset relation of configurations. A subset $\mathcal{T}$ of the configurations is considered as terminating configurations, i.e. these configurations are maximal in the set of all configurations.

Definition 1 (Configuration Structure). *A* configuration structure *(over an alphabet Σ) is a triple $\mathcal{CS} = (\mathcal{S}, \mathcal{T}, l)$ where $\mathcal{S}$ is a family of finite sets (configurations), $\mathcal{T} \subset \mathcal{S}$ a termination predicate satisfying $X \in \mathcal{T} \wedge X \subseteq Y \in \mathcal{S} \implies X = Y$ and $l : \bigcup_{X\in\mathcal{S}} X \to \Sigma$ is a labellings function. $\mathcal{ACS}$ denotes the domain of all configuration structures and $\mathcal{E}_{\mathcal{CS}} = \bigcup_{X\in\mathcal{S}} X$ is the set of events of $\mathcal{CS}$.*

Given a configuration structure $\mathcal{CS} = (\mathcal{S}, \mathcal{T}, l)$ we use $\mathcal{S}_{\mathcal{CS}}$, $\mathcal{T}_{\mathcal{CS}}$, and $l_{\mathcal{CS}}$ to select the individual elements of the tuple $\mathcal{CS}$.

[1] *BSD* stands for backwards-strict deletion and *BSIA* for backwards-strict insertion of admissible events.

Definition 2. *Let $\mathcal{CS} = (\mathcal{S}, \mathcal{T}, l)$ be a configuration structure. The* step transition relation $\rightarrow$ *of $\mathcal{CS}$ is defined by $\forall X, Y \in \mathcal{S} : X \rightarrow Y$ iff $X \subset Y$, and $\forall Z : X \subset Z \subset Y \implies Z \in \mathcal{S}$.*

For our purposes we restrict ourselves to so-called *stable* configuration structures that are closely associated to stable event structures (see [21]). Stable configuration structures have the property that causal dependencies in configurations can faithfully be represented by partial orders.

Definition 3 (Stable Configuration Structures). *A configuration structure $\mathcal{CS} = (\mathcal{S}, \mathcal{T}, l)$ is*

- rooted *iff $\emptyset \in \mathcal{S}$,*
- connected *iff $\emptyset \neq X \in \mathcal{S} \implies \exists e \in X : X \setminus \{e\} \in \mathcal{S}$,*
- closed under bounded unions *iff $X, Y, Z \in \mathcal{S}, X \cup Y \subseteq Z \implies X \cup Y \in \mathcal{S}$,*
- closed under bounded intersection *iff $X, Y, Z \in \mathcal{S}, X \cap Y \subseteq Z \implies X \cap Y \in \mathcal{S}$.*

$\mathcal{CS}$ is stable *iff it is rooted, connected, closed under bounded union and closed under bounded intersection.*

To refine a configuration structure $\mathcal{CS}$, each action $a \in \Sigma_{\mathcal{CS}}$ is associated to an individual configuration structure $\mathcal{CS}_a$ that represents the refinement of this particular action. Given a configuration $X \in \mathcal{CS}$, its refinement $\tilde{X}$ combines each event $e \in X$ with a non-empty configuration X_e in its refinement $\mathcal{CS}_{l(e)}$. Given a configuration $\tilde{X}$ in the refinement we can compute a set $busy(\tilde{X})$ of events e for which X_e is not a terminating configuration. These events $busy(\tilde{X})$ are performed in parallel since the execution of their refinements is done more or less "interleaved". Formally we define:

Definition 4 (Refinement). *A function $ref : \Sigma \rightarrow \mathcal{ACS} \setminus \{\epsilon\}$ is called a* refinement function. *Let $\mathcal{CS} = (\mathcal{S}, \mathcal{T}, l) \in \mathcal{ACS}$ and let ref be a refinement function. Then $\tilde{X}$ is a refinement of a configuration $X \in \mathcal{S}$ by ref iff*

- *$\tilde{X} = \bigcup_{e \in X} \{e\} \times X_e$ where $\forall e \in X : \; X_e \in \mathcal{S}_{ref(l(e))} \setminus \{\emptyset\}$,*
- *$\forall Y \subseteq busy(\tilde{X}) : X - Y \in \mathcal{S}$ with $busy(\tilde{X}) := \{e \in X \mid X_e \notin \mathcal{T}_{ref(l(e))}\}$*

A refinement is terminated *iff $busy(\tilde{X}) = \emptyset$.*

The refinement $ref(\mathcal{CS}) = (\mathcal{S}_{ref(\mathcal{CS})}, \mathcal{T}_{ref(\mathcal{CS})}, l_{ref(\mathcal{CS})})$ of a configuration structure $\mathcal{CS}$ by a refinement function ref is defined by

- *$\mathcal{S}_{ref(\mathcal{CS})} = \{\tilde{X} \mid \tilde{X}$ is a refinement of some $X \in \mathcal{S}$ by $ref\}$,*
- *$\mathcal{T}_{ref(\mathcal{CS})} = \{\tilde{X} \mid \tilde{X}$ is a terminated refinement for some $X \in \mathcal{T}$ by $ref\}$, and*
- *$l_{ref(\mathcal{CS})}(e, e') = l_{ref(l(e))}(e')$ for all $(e, e') \in E_{ref(\mathcal{CS})}$.*

Refinements are well-defined operations on configurations structures, i.e. $ref(\mathcal{CS}) \in \mathcal{ACS}$ if $\mathcal{CS} \in \mathcal{ACS}$ and ref is a refinement function. Also $ref(\mathcal{CS})$ is stable if $\mathcal{CS}$ and all configuration structures $\mathcal{CS}_{ref(l(e))}$ for the refinements of all actions $l(e)$ are stable.

Given a stable configuration structure $\mathcal{CS} = (\mathcal{S}, \mathcal{T}, l)$, we are able to formalize the causal dependencies in a configuration by a partial order. We define $d \leq_X e$ iff $\forall Y \in \mathcal{S} : Y \subseteq X \wedge e \in Y \implies d \in Y$. The *causality relation* on $X \in \mathcal{S}$ is given by $d <_X e$ iff $d \leq_X e \wedge d \neq e$.

As a consequence of stableness, causality relations on refined configuration structures are completely determined by the causality relations on the original configuration structure and the ones associated to the actions by the refinement function:

Lemma 1. *Let $\tilde{X}$ be a refinement of $X \in \mathcal{S}$ by a refinement function ref, i.e. $\tilde{X} = \bigcup_{e \in X} \{e\} \times X_e$. Then, $(d_1, d_1') <_{\tilde{X}} (d_2, d_2')$ iff $(d_1 <_X d_2) \vee (d_1 = d_2 \wedge d_1' <_{X_{d_1}} d_2')$.*

Proof. A proof of this lemma can be found in [7].

This allows us to establish a partial order on the event of a configuration. In particular, $\mathcal{X} = (X, <_X, l_X)$ represents a partial order which is labeled over Σ. Let $\mathcal{Y} = (Y, <_Y, l_Y)$ then $\mathcal{X}$ and $\mathcal{Y}$ are isomorphic iff there is a bijection between $\mathcal{X}$ and $\mathcal{Y}$ respecting ordering and labellings.

In the following we will make use of the following property.

Lemma 2. *Given two configurations $X, X' \in \mathcal{S}$ of a stable configuration structure with $X \subset X'$ there are always configurations $X_0, \ldots, X_n$ and actions $a_1, \ldots, a_n$ with $X = X_0 \rightarrow_{a_1} X_1 \rightarrow \ldots \rightarrow_{a_n} X_n = X'$.*

Proof. A proof of this lemma can be found in [7].

4 Security in Configuration Structures

In this section we will translate the ideas of MAKS to configuration structures. We introduce the notion of a view for configuration structures which classify their actions into visible, non-visible or confidential actions. Notice, that an event in a trace-based system corresponds to an action in a configuration structure. Events in a configuration structure relate to *occurrences* of events in trace-based systems. Therefore we define:

Definition 5 (View). *Let $\mathcal{CS}$ be configuration structure over an alphabet Σ. A* view *$\mathcal{V} = (V, N, C)$ for $\mathcal{CS}$ is a triple such that V, N, C forms a disjoint partition of Σ.*

In the following, we use the notation $U_{V|N|C} = \{e \in U \mid \ l(e) \in V \mid N \mid C\}$ to refer to the visible, non visible, or confidential parts of U.

The following definition formalizes possible refinements of visible, non-visible, and confidential actions. Intuitively, the refinement of non-visible actions consists again of non-visible actions only. Visible actions can be refined by using visible and non-visible actions but obviously they must not contain confidential actions. The refinement of confidential actions is more delicate. Similar to visible actions, the refinement of confidential actions can only contain confidential and non-visible actions but must not contain any visible actions. Otherwise, an adversary could easily deduce the occurrence of confidential actions by looking at its visible actions in the refinement.

However, in order to guarantee that action refinements will preserve security predicates like *BSD* or *BSIA* we have to go one step further: if an action refinement would translate a confidential action c into a sequel of confidential actions, say c_1, c_2, and suppose that both actions would occur only inside this refinement, then the refinement would introduce a dependency between confidential actions (which can be utilized by an adversary). Notice that confidential events in MAKS are closely related to high-input (rather than high-) events in other approaches. Therefore confidential events are typically used to model the *introduction* of a secret into a system rather than the *processing* of a secret, which will be modeled by non-visible events. In the following we demand that the refinement of a confidential event always results in a sequel of events in which only the first event can be confidential and all others are non-visible. Thus, roughly speaking, we assume that a secret introduced to a system is always atomic.

Definition 6 (View Refinement). *Let $\mathcal{CS} = (\mathcal{S}, \mathcal{T}, l)$ be configuration structure and $\mathcal{V} = (V, N, C)$ be a view for $\mathcal{CS}$. Given a refinement function ref, a view $\tilde{\mathcal{V}} = (\tilde{V}, \tilde{N}, \tilde{C})$ for $ref(\mathcal{CS})$ is called a* view refinement *of $\mathcal{V}$ wrt. ref iff*

- $\forall a \in N : \ \Sigma_{ref(a)} \subseteq \tilde{N}$
- $\forall a \in V : \ \Sigma_{ref(a)} \subseteq \tilde{V} \cup \tilde{N}$
- $\forall a \in C : \ \Sigma_{ref(a)} \subseteq \tilde{C} \cup \tilde{N}$
- $\forall a \in C : \ \forall \{e\}, X \in \mathcal{S}_{ref(a)} : \ e \in X \implies l(X \setminus \{e\}) \subseteq \tilde{N}$

Typically basic security predicates in MAKS represent closure properties demanding that observing the visible events of a trace does not reveal any information about the confidential events of this trace. Given an admissible system trace, there must be another trace with different confidential events that cause the same visible behavior, i.e. both traces are equivalent with respect to their visible behavior. To translate this idea into configuration structures, we have to formalize the notion of visible behavior. In contrast to trace based system this includes also the branching behavior of a particular configuration. In the following we introduce the notion of V-simulation between configurations. A configuration X V-simulates another configuration Y if X behaves(with respect to successor configurations and branching behavior) as Y on the low-level. The problem of formalizing such a property that is also preserved under action refinement is closely related to the general problem of defining equivalence relations invariant under refinement. For our purposes we adopt the notion of history preserving bisimulations [5] relating two configurations with same causal history which are known to be preserved under action refinement. However, in our setting we are only interested in visible parts of the history and in simulation (instead of bisimulation):

Definition 7 ($\mathcal{V}$-simulation). *Let $\mathcal{CS} = (\mathcal{S}, \mathcal{T}, l)$ be configuration structure with events $\mathcal{E}$, $\mathcal{V} = (V, N, C)$ be a view for $\mathcal{CS}$, and $X, Y \in \mathcal{S}$. Y $\mathcal{V}$-*simulates *X iff there is relation $R \subseteq (\mathcal{S}, \mathcal{S}, \mathcal{P}(\mathcal{E}_V, \mathcal{E}_V))$ such that $(X, Y, id) \in R$ and whenever $(U, W, f) \in R$ then*

- *f is an isomorphism between $(U_V, <_{U_V}, l\mid_{U_V})$ and $(W_V, <_{W_V}, l\mid_{W_V})$, and*
- *for all $U' \in \mathcal{S}$ with $U \subset U'$ and $(U' - U) \cap \mathcal{E}_C = \emptyset$*
  there is $Y' \in \mathcal{S}$ and $f' \in \mathcal{P}(\mathcal{E}_V, \mathcal{E}_V)$ such that:
  $W \subseteq Y'$, $(W' - W) \cap \mathcal{E}_C = \emptyset$, $f'\mid_W = f$, and $(U', W', f') \in R$.

The following lemma guarantees that $\mathcal{V}$-simulation is preserved under action refinement if we refine $\mathcal{V}$ appropriately (cf. Def. 6).

Lemma 3. *Let $\mathcal{CS} = (\mathcal{S}, \mathcal{T}, l)$ be configuration structure (with events $\mathcal{E}$) together with a view $\mathcal{V} = (V, N, C)$, ref be a refinement function for $\mathcal{CS}$, and $\tilde{\mathcal{V}}$ be a view refinement of $\mathcal{V}$ wrt. $\mathcal{CS}$.*

Let $X, Y \in \mathcal{S}$ and $\tilde{X}, \tilde{Y} \in \mathcal{S}_{ref(\mathcal{CS})}$ such that $\forall e \in \mathcal{E}_V$: $X_e = Y_e$ holds. Then, $\tilde{Y}$ $\tilde{\mathcal{V}}$-simulates $\tilde{X}$ if Y $\mathcal{V}$-simulates X.

Proof. Let $\tilde{R}$ be a relation with $(\tilde{U}, \tilde{W}, \tilde{f}) \in \tilde{R}$ iff there is a $(U, W, f) \in R$ such that

$$\tilde{U} = \bigcup_{e \in U} e \times U_e \text{ with } U_e \neq \emptyset \tag{3}$$

$$\tilde{W} = \bigcup_{e \in W} e \times W_e \text{ with } W_e \neq \emptyset \tag{4}$$

$$\forall e \in \mathcal{E}_V : U_e = W_{f(e)} \tag{5}$$

$$\forall (e, e') \in \tilde{\mathcal{E}}_V : \; \tilde{f}(e, e') = (f(e), e') \tag{6}$$

First, we have to prove that $(\tilde{X}, \tilde{Y}, id) \in \tilde{R}$ holds. Since $R(X, Y, id)$ holds, $f = id$ obviously implies $\forall e \in V$: $X_e = Y_{f(e)}$ and $\tilde{f}(e, e') = (e, e') = (f(e), e')$.

Second, we have to prove that $\tilde{f}$ is an isomorphism between $(\tilde{U}_{\tilde{V}}, <_{\tilde{U}_{\tilde{V}}}, l\mid_{\tilde{U}_{\tilde{V}}})$ and $(\tilde{W}_{\tilde{V}}, <_{\tilde{W}_{\tilde{V}}}, l\mid_{\tilde{W}_{\tilde{V}}})$ Therefore, we have to prove that $(d, d') <_{\tilde{U}} (e, e') \leftrightarrow \tilde{f}(d, d') <_{\tilde{W}} \tilde{f}(e, e')$ and $l(\tilde{f}(e, e')) = l((e, e')$:

$$\begin{aligned}
&(d, d') <_{\tilde{U}} (e, e') \leftrightarrow (d <_U e) \vee ((d = e) \wedge (d' <_{ref(l(d))} e')) \\
\leftrightarrow\ & (d <_U e) \vee ((d = e) \wedge (d' <_{ref(l(f(d)))} e') \\
\leftrightarrow\ & (f(d) <_W f(e)) \vee ((f(d) = f(e)) \wedge (d' <_{ref(l(f(d)))} e') \\
\leftrightarrow\ & \tilde{f}(d, d') <_{\tilde{W}} \tilde{f}(e, e')
\end{aligned}$$

$$l(\tilde{f}(e, e')) = l((f(e), e')) = l_{ref(l(f(e)))}(e') = l_{ref(l(e))}(e') = l((e, e'))$$

Third, let $(\tilde{U}, \tilde{W}, \tilde{f}) \in \tilde{R}$ then we have to prove that for all $\tilde{U}' \in \mathcal{S}_{ref(\mathcal{CS})}$ with $\tilde{U} \subset \tilde{U}'$ and $(\tilde{U}' - \tilde{U}) \cap \tilde{\mathcal{E}}_{\tilde{C}} = \emptyset$ there is $\tilde{W}' \in \mathcal{S}_{ref(\mathcal{CS})}$ and $\tilde{f}' \in \mathcal{P}(\tilde{\mathcal{E}}_{\tilde{V}}, \tilde{\mathcal{E}}_{\tilde{V}})$ such that: $\tilde{W} \subseteq \tilde{W}'$, $(\tilde{W}' - \tilde{W}) \cap \tilde{\mathcal{E}}_{\tilde{C}} = \emptyset$, $\tilde{f}'\mid_{\tilde{W}} = \tilde{f}$, and $(\tilde{U}', \tilde{W}', \tilde{f}') \in \tilde{R}$.

Let $(\tilde{U}, \tilde{W}, \tilde{f}) \in \tilde{R}$ and $\tilde{U}' \in \mathcal{S}_{ref(\mathcal{CS})}$ with $\tilde{U} \subset \tilde{U}'$ and $(\tilde{U}' - \tilde{U}) \cap \tilde{\mathcal{E}}_{\tilde{C}} = \emptyset$. Since $(\tilde{U}, \tilde{W}, \tilde{f}) \in \tilde{R}$, let (U, W, f) be the corresponding element in R as required by the construction of $\tilde{R}$. $\tilde{U}' \in \mathcal{S}_{ref(\mathcal{CS})}$ implies $U' \in \mathcal{S}_{\mathcal{CS}}$. Further, obviously $U \subseteq U'$, and $(U' - U) \cap \mathcal{E}_C = \emptyset$ because otherwise, there would be a confidential event e in $U - U'$ and thus the refinement $e \times U'_e$ would include (by the definition of 7) at least one confidential event (e, e'). Thus, there is a $(U', W', f') \in R$ with $W' \in \mathcal{S}_{\mathcal{CS}}$, $W \subseteq W'$, $(W' - W) \cap \mathcal{E}_C = \emptyset$, and $f' \mid_W = f$.

Consider $\tilde{W}' = \bigcup_{e \in W'} e \times W'_e$ with $W'_e = U'_{f^{-1}(e)}$ if $e \in \mathcal{E}_V$ and $W'_e \in \mathcal{T}_{ref(l(e))}$ with $W_e \subseteq W_e$ otherwise. Obviously, $(e, e') \in \tilde{\mathcal{E}}_{\tilde{N}}$ for $e \notin \mathcal{E}_V$ and $e' \in W'_e - W_e$ because the refinement of non-visible events introduces only non-visible events while the refinement of confidential events only causes one confidential event at the start (i.e. is already included in W_e) followed by non-visible events.

We know that $busy(\tilde{W}') \subseteq \mathcal{E}_V$. Since $\tilde{U}' \in \mathcal{S}_{ref(\mathcal{CS})}$ we know also that $\forall Y \subseteq busy(\tilde{U}) : U - Y \in \mathcal{S}_{\mathcal{CS}}$ and thus $\forall Y \subseteq busy(\tilde{U}) \cap \mathcal{E}_V : U - Y \in \mathcal{S}_{\mathcal{CS}}$. Since f is an isomorphism on the pomsets, $\forall Y \subseteq f(busy(\tilde{U}) \cap \mathcal{E}_V) : W - Y \in \mathcal{S}_{\mathcal{CS}}$ holds. Thus, $\forall Y \subseteq busy(\tilde{W}') : W - Y \in \mathcal{S}_{\mathcal{CS}}$ and $\tilde{W}' \in \mathcal{S}_{\mathcal{CS}}$ □

5 Basic Security Predicates

In the following subsections we will translate the two most prominent basic security predicates of MAKS, *BSD* and *BSIA*$_\rho$, into our framework based on configuration structures and prove that both notions are (under some preconditions) preserved under action refinement.

5.1 Backward Strict Deletion

Enforcing the Backward Strict Deletion property in a trace-based system guarantees that an adversary cannot deduce that a specific confidential event has happened when monitoring the visible behavior of the system. Technically, this property ensures that for each (finite) trace *tr* we can take the prefix of this trace up to the last confidential event and then simulate the rest of *tr* without confidential events (see Section 2). The translation of this property to configuration structures in straight forward. If we are in a particular configuration $X \in \mathcal{S}$ and have the possibility to perform an confidential action, i.e. $X \cup \{e\} \in \mathcal{S}$ with $e \in \mathcal{E}_C$, then X should cause the same visible behavior as $X \cup \{e\}$ would do. Formally we define:

Definition 8. *Let $\mathcal{CS} = (\mathcal{S}, \mathcal{T}, l)$ be a configuration structure together with a view $\mathcal{V} = (V, N, C)$. $\mathcal{CS}$* satisfies Backward Strict Deletion (or *BSD* for short) *iff for all $X \in \mathcal{S}$ and $e \in \mathcal{E}_C$: $X \cup \{e\} \in \mathcal{S}$ implies that X $\mathcal{V}$-simulates $X \cup \{e\}$.*

The following theorem guarantees that the basic security predicate *BSD* is always preserved under action refinement as long as we use a view refinement as specified in Definition 6. Since secrets are considered as atomic we are able to remove the complete refinement of a confidential event since *BSD* on the abstract level guarantees that we can remove this confidential event already on the abstract level.

Theorem 1. *Let $\mathcal{CS} = (\mathcal{S}, \mathcal{T}, l)$ be a configuration structure together with a view $\mathcal{V} = (V, N, C)$ that satisfies BSD wrt. $\mathcal{V}$. Let $ref(\mathcal{CS})$ be a refinement of $\mathcal{CS}$ and $\tilde{\mathcal{V}}$ be view refinement of $\mathcal{V}$ wrt. ref and $\mathcal{CS}$. Then, $\tilde{\mathcal{CS}}$ satisfies BSD wrt. $\tilde{\mathcal{V}}$.*

Proof. Let $ref(\mathcal{CS}) = (\tilde{\mathcal{S}}, \tilde{\mathcal{T}}, \tilde{l})$ and $\tilde{\mathcal{V}} = (\tilde{V}, \tilde{N}, \tilde{C})$. Suppose, there is an $(e, e') \in \tilde{\mathcal{E}}_{\tilde{C}}$ and $\tilde{X} \in \tilde{\mathcal{S}}$ such that $\tilde{X} \cup \{(e, e')\} \in \tilde{\mathcal{S}}$. Let $\tilde{X}$ be the refinement of some $X \in \mathcal{CS}$. Since $(e, e') \in \tilde{\mathcal{E}}_C$ we know that $e \notin X$ because confidential events (e, e') can only occur as a first step in the refinement of a confidential event e. Thus, $\tilde{X} \cup \{(e, e')\}$ is a refinement of a configuration $X' = X \cup \{e\}$. Furthermore, since $\mathcal{CS}$ satisfies BSD, we know that X $\mathcal{V}$-simulates X'. Then, Lemma 3 ensures that $\tilde{X}$ $\tilde{\mathcal{V}}$-simulates $\tilde{X} \cup \{(e, e')\}$, since $X_d = X'_d$ holds for all $d \in \mathcal{E}_V$ trivially. □

5.2 Backward Strict Insertion

While *BSD* is concerned with the non-deducability of occurrences of actions, enforcing Backward Strict Insertion will guarantee that an adversary cannot deduce that a confidential action has *not* occurred. Technically we have to guarantee that for any possible system trace *tr*: if we take any prefix of *tr* containing in particular all its confidential events and append another confidential event to the end of prefix then we can expand this trace to a system trace that causes the same visible behavior as *tr*. We can easily translate this property to configuration structures as follows. If we are in a particular configuration $X \in \mathcal{S}$ then we must be able to perform any confidential action, i.e. $X \cup \{e\} \in \mathcal{S}$ with $e \in \mathcal{E}_C$ and $X \cup \{e\}$ must cause the same visible behavior as X would do.

It is obvious that a system satisfying *BSIA* behaves totally randomly on confidential events since they can occur in a random sequel and are also randomly interleaved with the sequel of visible events. However, any intrinsic dependencies between (confidential) events are known to an adversary since he can inspect the admissible system traces. Since there is no general solution to the problem of how much system information should be leaked to an adversary, Mantel allows one to restrict the enforcement of the *BSIA* predicate only to specific situations. He introduces an admissibility predicate ρ on traces in order to specify those situations in which we have to guarantee that *BSIA* holds (see Section 2.)

We translate this admissibility restriction into the notion of configuration structures as follows:

Definition 9. *Let $\mathcal{CS} = (\mathcal{S}, \mathcal{T}, l)$ be a configuration structure with events E. A set $\rho \subseteq \mathcal{E}$ is called an* admissibility restriction. *A configuration X is ρ-admissible iff there is a configuration $X' \in \mathcal{S}$ such that $l(X \cap \rho) = l(X' \cap \rho)$.*

Definition 10. *Let $\mathcal{CS} = (\mathcal{S}, \mathcal{T}, l)$ be a configuration structure with events E and $\rho \subseteq E_{\mathcal{S}}$. $\mathcal{CS}$ satisfies $BSIA_\rho$ iff for all $X \in \mathcal{S}$ and all $e \in C_{\mathcal{S}}$: if $X \cup \{e\}$ is ρ-admissible then $X \cup \{e\} \in \mathcal{CS}$ and $X \cup \{e\}$ $\mathcal{V}$-simulates X.*

In order to translate a security predicate $BSIA_\rho$ that is satisfied by a configuration structure $\mathcal{CS}$ to its refinement $ref(\mathcal{CS})$ we have to provide an appropriate set $\tilde{\rho}$ such that on the one hand $ref(\mathcal{CS})$ satisfies $BSIA_{\tilde{\rho}}$ but on the other hand $\tilde{\rho}$ lacks only that degree of information about dependencies of confidential events that we are willing to provide to the adversary. Thus, we do not provide a unique translation of ρ to some $\tilde{\rho}$ but provide sufficient conditions of $\tilde{\rho}$ to guarantee that

$BSIA_\rho$ will be preserved under refinement. In particular, a refinement has to preserve admissibility: if a configuration $\tilde{X}$ of the refined configuration structure is admissible wrt. $\tilde{\rho}$ then it abstract configuration X should be also admissible wrt. ρ. Furthermore, we have to guarantee that in all admissible situations the inserted confidential event can be executed in parallel with non-atomic previous events, the refinements of which have not been finished yet.

Definition 11. *Let $\mathcal{CS} = (\mathcal{S}, \mathcal{T}, l)$ be a configuration structure and ref be a refinement function. An admissibility restriction $\tilde{\rho} \subseteq \tilde{\mathcal{E}}$ is a* refinement *of an admissibility restriction $\rho \subseteq \mathcal{E}$ wrt. ref and $\mathcal{CS}$ iff*
for all $\tilde{X} \in \tilde{\mathcal{S}}$ and all $(e, e') \in \tilde{\mathcal{E}}_{\tilde{C}}$ holds

- *$\tilde{X} \cup \{(e, e')\}$ is $\tilde{\rho}$-admissible implies $X \cup \{e\}$ is ρ-admissible, and*
- *$\forall Y \subset busy(\tilde{X}) : \; X \cup \{e\} - Y \in \mathcal{S}$*

Given this definition of refining ρ-admissibility, we are now able to formulate the preconditions under which $BSIA_\rho$ is preserved under action refinement:

Theorem 2. *Let $\mathcal{CS} = (\mathcal{S}, \mathcal{T}, l)$ be a configuration structure together with a view $\mathcal{V} = (V, N, C)$ that satisfies $BSIA_\rho$ wrt. $\mathcal{V}$. Let ref$(\mathcal{CS})$ be a refinement of $\mathcal{CS}$, $\tilde{\mathcal{V}}$ be view refinement of $\mathcal{V}$ wrt. ref and $\mathcal{CS}$, and $\tilde{\rho}$ is a refinement of ρ wrt. ref and $\mathcal{CS}$. Then, $\tilde{\mathcal{CS}}$ satisfies $BSIA_{\tilde{\rho}}$ wrt. $\tilde{\mathcal{V}}$.*

Proof. Suppose, $\tilde{X} \in \tilde{\mathcal{S}}$ and $\tilde{X} \cup \{e, e'\}$ is $\tilde{\rho}$-admissible. Therefore, $X' = X \cup \{e\}$ is ρ-admissible and $X' \in \mathcal{S}$. Since $\forall Y \subset busy(\tilde{X}) : \; X \cup \{e\} - Y = X' - Y \in \mathcal{S}$ and also $\forall Y \subset busy(\tilde{X}) : \; X - Y \in \mathcal{S}$ we know that $\forall Y \subset busy(\tilde{X} \cup \{e, e'\}) : X' - Y \in \mathcal{S}$ and thus $\tilde{X} \cup \{e, e'\} \in \tilde{S}$.

Since X' is ρ-admissible, $X' \in \mathcal{CS}$, and $\mathcal{CS}$ satisfies $BSIA_\rho$ we know that X' $\mathcal{V}$-simulates X. Thus, lemma 3 ensures that $\tilde{X} \cup \{(e, e')\}$ $\tilde{\mathcal{V}}$-simulates $\tilde{X}$, since $X_d = X'_d$ holds for all $d \in \mathcal{E}_V$ trivially. □

6 Related Work

Action refinement has been the subject of intensive studies in between 1985 and 1995. We refer to [8] for an overview and classification of the different syntactic and semantic based interpretations of action refinement. Configuration structures are closely related to event structures which have been introduced by Winskel [21]. We refer the reader to [7] for a discussion of the various approaches, the corresponding notions of action refinements and the problems of finding appropriate bisimulation equivalences that are preserved under refinement.

Starting with the work of Goguen and Meseguer, information flow control has been subject of a large variety of different approaches introducing different formal notions of independence. Most prominent, McLean [17], Zakinthinos and Lee [22] and Mantel [13] proposed frameworks to embed these different notions in a uniform framework.

There is a large number of work that is concerned with the problem of combining information flow control and refinement. This work can be divided into different categories according to the different versions of refinements considered.

Jacob [12] as well as Mantel [14] proposed approaches for secure refinement considering refinement as a process to eliminate indeterminism. In terms of trace-based systems such a refinement reduces the set of admissible system traces while actions (or events) are considered as atomic. Mantel introduces a collection of refinement operations for specific information flow properties that ensure that these properties are preserved under refinement (i.e. reduction of the set of admissible systems traces). In contrast, Jacob allows for an uncontrolled refinement but provides measures to translate the obtained refined system into a secure one. [1] also proposes a notion of refinement of states for processes described in terms of a Security Process Algebra (SPA).

Our approach is based on action refinement in which actions are considered as non-atomic. This allows one to model procedures as actions on the abstract level and use action refinements to implementation them on an implementation level. Investigating information flow properties in the presence of action refinement has been done previously by [3]. This approach is more related to ours, since they use a CCS-like process algebra SPA as an underlying specification language. Flow event structures as a special form of event structures are particular suited for giving semantic to languages like CCS. The approach in [3] uses a bisimulation-based information flow property named P_{BNDC} and provide preconditions under which this property is preserved under refinement in SPA. Their notion of *weak bisimulation on low action* is related to our notion of $\mathcal{V}$-simulation; both are used to formalize the corresponding security predicates (see also [2]. However, both approaches strongly differ in the preconditions they impose on systems in order to guarantee that the information flow properties are preserved under refinement.

7 Conclusion

Based on Mantel's framework MAKS, we presented a framework for possibilistic information flow in configuration structures. We transfered the most prominent basic security predicates BSD and $BSIA_\rho$ in terms of configuration structures and elaborated the situations in which these properties are preserved under action refinement. The work was motivated by developing a framework to investigate the security of multi-agent systems with the help of possibilistic information flow [11]. In this work we used a scenario of comparison shopping agents as a case study. It turned out that the verification of the security properties of individual agents (and in particular the formulation of the unwinding conditions) was hindered by the large number of N-events used to formalize the internal processing of incoming messages. In the approach presented in this paper this internal processing could be easily modeled as a refinement of various N-events which allows us to abstract away from a large part of the internal computation. However, a precondition of doing such an approach is the existence of appropriate unwinding theorems to verify BSD and $BSIA_\rho$ on configuration structures.

This development is still work in progress. Future work will be concerned with weakening the restrictive definition of a view refinement which now restricts the refinement of a confidential event to a single confidential event followed by non-visible events.

References

1. A. Bossi, R. Focardi, C. Piazza, and S. Rossi. Refinement Operators and Information Flow Security In: *Proceedings of the 1st International Conference on Software Engineering and Formal Methods (SEFM'03)*, IEEE Computer Science, 2001
2. A. Bossi, R. Focardi, C. Piazza, and S. Rossi. Bisimulation and Unwinding for Verifying Possibilistic Security Properties Proceedings of the 4th International Conference on Verification, Model Checking, and Abstract Interpretation (VMCAI 2003), Springer LNCS 2575, 2003
3. A. Bossi, D. Macedono, C. Piazza, and S. Rossi. Compositional Action Refinement and Information Flow Security. Technical Report CS-2003-13. Dipartimento di Informatica, Univerista Ca Foscari di Venezia, 2003
4. L. Castellano, G. de Michelis, and L. Pomello. Concurrency vs. interleaving: an instructive example. Bulletin of the EATCS 31, pp. 12–15, 1987.
5. P. Degano, R. de Nicola, and U. Montanari. Observational equivalences for concurrency models. In: *Proceedings of the 3rd IFIP WG 2.2 working conference: Formal description of programming concepts III*, Ebberup, North-Holland, 1987
6. R.J. Van Glabbeek and G.D. Plotkin. Configuration structures. In: *Proceedings of the 10th Annual IEEE Symposium on Logic in Computer Science*, IEEE Computer Society, 1995.
7. R.J. Van Glabbeek and U. Goltz. Refinement of actions and equivalence notions for concurrent systems. Acta Informatica, Vol. 37(4-5), pp. 229–327, 2001.
8. R. Gorrieri and A. Rensink. Action Refinement. Technical report UBLCS-99-09, University of Bologna, 1999.
9. J. A. Goguen and J. Meseguer. Security policies and security models. In *Proceedings of the IEEE Symposium on Security and Privacy.* IEEE Computer Society, 1982.
10. J. A. Goguen and J. Meseguer. Inference control and unwinding. In *Proceedings of the IEEE Symposium on Security and Privacy.* IEEE Computer Society, 1984.
11. D. Hutter, H. Mantel, A.Schairer, and I. Schaefer Security in Multiagent Systems – A Case Study on Comparison Shopping. Journal of Applied Logic. Special Issue: Logic-based Verification of Multiagent Systems, Elsevier, Article in press, doi:10.1016/j.jal.2005.12.015, 2006
12. J. Jacob. On the derivation of secure components. In: *Proceedings of the 1989 IEEE Symposium on Security and Privacy*, IEEE Computer Society, 1989.
13. H. Mantel. Possibilistic definitions of security – an assembly kit. In *Proceedings of the IEEE Computer Security Foundations Workshop.* IEEE Computer Society, 2000.
14. H. Mantel. Preserving Information Flow Properties under Refinement. In: *Proceedings of the 2001 IEEE Symposium on Security and Privacy*, IEEE Computer Society, 2001.
15. H. Mantel. *A Uniform Framework for the Formal Specification and Verification of Information Flow Security.* PhD thesis, Universität des Saarlandes, 2003. Published as a manuscript.
16. J. D. McLean. Proving Noninterference and Functional Correctness using Traces. *Journal of Computer Security*, 1(1):37–57, 1992.

17. J.D. McLean. A general theory of composition for trace sets closed under selective interleaving functions. In *Proceedings of IEEE Symposium on Security and Privacy.* IEEE Computer Society, 1994.
18. R. Milner A Calculus of Communicating Systems. Springer, LNCS 92, 1980
19. J. Rushby. Noninterference, transitivity, and channel-control security policies. Technical Report CSL-92-02, SRI International, Menlo Park, CA, 1992.
20. P.Y.A. Ryan and S.A Schneider. Process algebra and non-interference. *Journal of Computer Security*, 9(1/2):75–103, 2001.
21. G. Winskel. Event structures. In: *Petri Nets: Applications and Relationships of other models of concurrency*, Advances in Petri Nets. Springer, LNCS 255, 1986
22. A. Zakinthinos and E. S. Lee. A general theory of security properties. In *Proceedings of the IEEE Symposium on Security and Privacy.* IEEE Computer Society, 1997.