# Information Flow Control and Applications – Bridging a Gap –

Heiko Mantel

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

**Abstract.** The development of formal security models is a difficult, time consuming, and expensive task. This development burden can be considerably reduced by using generic security models. In a security model, confidentiality as well as integrity requirements can be expressed by restrictions on the information flow. Generic models for controling information flow in distributed systems have been thoroughly investigated. Nevertheless, the known approaches cannot cope with common features of secure distributed systems like channel control, information filters, or explicit downgrading. This limitation caused a major gap which has prevented the migration of a large body of research into practice. To bridge this gap is the main goal of this article.

## 1 Introduction

With the growing popularity of e-commerce the security of networked information systems becomes an increasingly important issue. Since such distributed systems are usually quite complex, the application of formal methods in their development appears to be most appropriate in order to ensure security. In this process, the desired security properties are specified in a formal security model. This becomes a necessary task if the system shall be evaluated according to criteria like ITSEC or CC (level E4/EAL5 or higher). However, the development of security models is a difficult, time consuming, and expensive task. Therefore it is highly desirable to have *generic security models* which are well suited for certain application domains and which only need to be instantiated (rather than being constructed from scratch) for each application. In a security model, confidentiality as well as integrity requirements can be expressed by restrictions on the information flow. Generic security models for information flow control like [GM82,Sut86,McL96] are well-known. However, the use of such models for distributed systems has been quite limited in practice. The main reason is that the known models cannot cope with intransitive flow policies which are necessary in order to express common features like channel control, information filters, or explicit downgrading. In this article, we propose a solution to this problem.

In information flow control one first identifies different domains within a system and then decides if information may flow between these domains or not. This results in a *flow policy*. Next, a *definition of information flow* must be

J.N. Oliveira and P. Zave (Eds.): FME 2001, LNCS 2021, pp. 153–172, 2001.

chosen. The common intuition underlying such definitions is that information flows from a domain $D_1$ to a domain $D_2$ if the behaviour of $D_2$ can be affected by actions of $D_1$. However, this intuition can be formalized in different ways and at least for non-deterministic systems no agreement on an optimal definition of information flow has been reached. Rather a collection of definitions co-exist. Frameworks like [McL96,ZL97,Man00a] provide a suitable basis for choosing an appropriate definition for a given application since they allow one to investigate the various definitions in a uniform way and to compare them to each other.

To achieve confidentiality or integrity by restricting the flow of information within a system is a very elegant and thus appealing approach. However, the assumptions underlying the existing approaches for information flow control are often too restrictive for real applications. Even though information flow shall be restricted in such applications, it must be possible to allow for *exceptions* to these restrictions. Typical examples for such exceptions are that two domains *should not* communicate with each other *unless* they use a particular communication channel which contains an information filter, that a domain which has access to sensitive data *should not* communicate with an open network *unless* the data has been properly encrypted, or that data *should not* be publicly accessible *unless* the data has been downgraded because a certain period of time has passed or a particular event has occurred. In information flow control, such exceptions can be expressed by *intransitive flow policies*. Intransitive policies indeed are necessary for real applications as can be seen at case studies like [SRS$^+$00]. However, all known approaches (e.g. [Rus92,Pin95,RG99]) which are compatible with intransitive flow are limited to deterministic systems ([RG99] can deal with some, but severely limited non-determinism). Hence, they are not applicable to distributed systems which are certainly the most interesting ones in the presence of the Internet. The unsolved problem of how to cope with intransitive policies created a major gap which has prevented the application of a large body of work on information flow control in practice. To *bridge this gap* is the main goal of this article in which we extend our previously proposed framework [Man00a] to cope also with intransitive policies. We are confident that this is a major step for bringing information flow control into practice.

The overall structure of a security model based on information flow control is depicted in Figure 1. As usual, such a model consists of three main components: a formal specification of the system under consideration, a specification of one or more security properties, and a proof that the system satisfies these security properties. In information flow control, a security property again consists of two parts: a flow policy which defines where information flow is permissible or restricted, and a formal definition of what information flow means.

In this article, we focus on definitions of information flow. Our main contributions are novel definitions which can cope with a class of flow policies, namely intransitive policies, which, for non-deterministic systems, has been outside the scope of the existing approaches (cf. Section 3). Moreover, we present an unwinding theorem (cf. Section 4) which simplifies the proof that a system satisfies a security property. How to develop system specifications, however, is *not* dis-

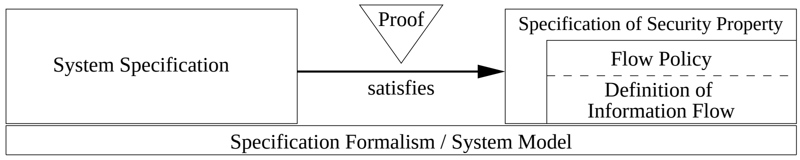



**Fig. 1.** Structure of a security model based on information flow control

cussed in this article. Nevertheless, we have to choose a specification formalism in order to refer to the underlying concepts in the specification of security properties. In Section 2 we introduce such a formalism and also give an introduction to security properties. We conclude this article by discussing related work in Section 5 and summarizing our results in Section 6.

# 2 Information Flow Control

In this section, we give an introduction to the basic concepts of information flow control before we turn our attention to intransitive information flow in subsequent sections. In Section 2.1 we define a specification formalism, or more precisely a system model on which such a formalism can be based. In Section 2.2 we introduce flow policies and provide various examples. Existing definitions of information flow are investigated in Section 2.3.

## 2.1 Specification Formalism / System Model

For the formal specification of distributed systems one has a choice among many different formalisms, like process algebras, temporal logics, or non-deterministic state machines. Rather than choosing a specific syntactic formalism we use a system model which is semantically motivated. This trace based model has already a tradition in the context of information flow control [McC87,JT88,ZL97, Man00a].

An *event* is an atomic action with no duration. Examples are sending or receiving a message on a communication channel, or writing data into a file. We distinguish *input events* which cannot be enforced by the system from *internal* and *output events* which are controlled by the system. However, we do *not* make the restricting assumption that input events are always enabled. At the interface, input as well as output events can be observed while internal events cannot. The possible behaviours of a system are modeled as sequences of events.

**Definition 1.** *An* event system *$ES$ is a tuple $(E, I, O, Tr)$ where $E$ is a set of events, $I, O \subseteq E$ respectively are the input and output events, and $Tr \subseteq E^*$ is the set of traces, i.e. finite sequences over $E$. $Tr$ must be closed under prefixes.*

Although event systems are used as system model throughout this article, our results are not limited to systems which are specified using event systems. To apply our results, it is sufficient that there exists a translation from the particular specification formalism into event systems. We illustrate such a translation by the example of state-event systems which will also be used in Section 4 where we present an unwinding theorem. State-event systems can be regarded as event systems which have been enriched by states. With this enrichment the pre-condition of an event $e$ is the set of states in which $e$ possibly can occur. The post-condition is a function from states to the set of possible states after the event has occurred in the respective state. The notion of state is transparent. Note that the occurrence of events can be observed while states are not observable.

**Definition 2.** *A* state-event system *$SES$ is a tuple $(S, S_I, E, I, O, T)$ where $S$ is a set of states, $S_I \subseteq S$ are the initial states, $E$ is a set of events, $I, O \subseteq E$ are the input and output events, and $T \subseteq S \times E \times S$ is a transition relation.*

A *history* of a state-event system $SES$ is a sequence $s_1.e_1.s_2 \ldots s_n$ of states and events. The set of histories $\mathit{Hist}(SES) \subseteq S \times (E \times S)^*$ for $SES$ is defined inductively. If $s \in S_I$ then $s \in \mathit{Hist}(SES)$. If $s_1.e_1.s_2 \ldots s_n \in \mathit{Hist}(SES)$ and $(s_n, e_n, s_{n+1}) \in T$ then $s_1.e_1.s_2 \ldots s_n.e_n.s_{n+1} \in \mathit{Hist}(SES)$. Each state-event system $SES = (S, S_I, E, I, O, T)$ can be translated into an event system $ES_{SES} = (E, I, O, \mathit{Tr}_{SES})$ where the set of traces $\mathit{Tr}_{SES} \subseteq E^*$ results from $\mathit{Hist}(SES)$ by deleting states from the histories.

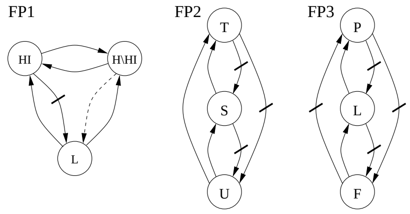
### 2.2 Flow Policies

*Flow policies* specify restrictions on the information flow within a system. They are defined with the help of a set $\mathcal{D}$ of *security domains*. Typical domains are e.g. groups of users, collections of files, or memory sections. We associate such a security domain $dom(e) \in \mathcal{D}$ to each event $e \in E$.

**Definition 3.** *A* flow policy *$FP$ is a tuple $(\mathcal{D}, \rightsquigarrow_V, \rightsquigarrow_N, \not\rightsquigarrow)$ where $\rightsquigarrow_V, \rightsquigarrow_N, \not\rightsquigarrow \subseteq \mathcal{D} \times \mathcal{D}$ form a disjoint partition of $\mathcal{D} \times \mathcal{D}$ and $\rightsquigarrow_V$ is reflexive. $FP$ is called* transitive *if $\rightsquigarrow_V$ is transitive and, otherwise,* intransitive.

$\not\rightsquigarrow$ is the *non-interference relation* of $FP$ and $D_1 \not\rightsquigarrow D_2$ expresses that there must be no information flow from $D_1$ to $D_2$. Rather than having only a single interference relation $\rightsquigarrow$ to specify allowed information flow we distinguish two relations $\rightsquigarrow_V$ and $\rightsquigarrow_N$. While $D_1 \rightsquigarrow_V D_2$ expresses that events in $D_1$ are visible for $D_2$, $D_1 \rightsquigarrow_N D_2$ expresses that events from $D_1$ may be deducible for $D_2$ but must not reveal any information about other domains.

We depict flow policies as graphs where each node corresponds to a security domain. The relations $\rightsquigarrow_V$, $\rightsquigarrow_N$, and $\not\rightsquigarrow$ are respectively depicted as solid, dashed, and crossed arrows. For the sake of readability, the reflexive subrelation of $\rightsquigarrow_V$ is usually omitted. This graphical representation is shown on the left hand side of Figure 2 for the flow policy $FP1$ which consists of three domains $HI$ (high-level input events), $L$ (low-level events), and $H \backslash HI$ (high-level internal and output events). According to $FP1$, low-level events are visible for both high-level domains ($L \rightsquigarrow_V HI$, $L \rightsquigarrow_V H \backslash HI$). High-level inputs must not be




| | $\mathcal{V}_{HI}$ | $\mathcal{V}_{L}$ | $\mathcal{V}_{H\setminus HI}$ |
|---|---|---|---|
| $FP1$ | $(H\cup L,\emptyset,\emptyset)$ | $(L,H\setminus HI,HI)$ | $(H\cup L,\emptyset,\emptyset)$ |
| | $\mathcal{V}_{U}$ | $\mathcal{V}_{S}$ | $\mathcal{V}_{T}$ |
| $FP2$ | $(U,\emptyset,S\cup T)$ | $(U\cup S,\emptyset,T)$ | $(U\cup S\cup T,\emptyset,\emptyset)$ |
| | $\mathcal{V}_{F}$ | $\mathcal{V}_{L}$ | $\mathcal{V}_{P}$ |
| $FP3$ | $(F,\emptyset,L\cup P)$ | $(L\cup F,\emptyset,P)$ | $(P\cup L,\emptyset,F)$ |

**Fig. 2.** Example flow policies and corresponding basic scenes

deducible for the low-level ($HI \not\rightsquigarrow L$). Other high-level events may be deduced (due to $H\setminus HI \rightsquigarrow_N L$). However, such deductions must not reveal any information about (confidential) high-level inputs. E.g. if each occurrence of an event $ho \in H\setminus HI$ is directly preceeded by a high-level input $hi \in HI$ then an adversary should not learn that $ho$ has occurred because, otherwise, he could deduce that $hi$ has occurred. Thus, if an event $e \in H\setminus HI$ closely depends on events in $HI$ then nothing about $e$ must be deducible for $L$. However, if $e$ does not depend on confidential events from $HI$ then everything about $e$ may be deducible.

Traditionally, $FP1$ would be defined as a policy with two domains $L, H$ and the policy $H \not\rightsquigarrow L$, $L \rightsquigarrow H$. This leaves it implicit that high-level internal and output events may be deducible for the low-level. Our novel distinction between $\rightsquigarrow_V$ and $\rightsquigarrow_N$ allows one to make such assumptions explicit in the flow policy.

$I, O$ specifies the interface of a system when it is used in a non-malicious environment. This intended interface should be used when properties apart from security are specified. However, in the context of security other interfaces must be considered as well since usually not all internal events are protected against malicious access. Making a worst case assumption, we assume that internal events are observable. The view of a given domain expresses which events are visible or confidential for that domain. Formally, a *view* $\mathcal{V}$ is a triple $(V, N, C)$ of sets of events such that the sets $V$, $N$, $C$ form a disjoint partition of $E$.

**Definition 4.** *The* view $\mathcal{V}_D = (V, N, C)$ for a domain $D \in \mathcal{D}$ *in FP is defined by* $V = \bigcup\{D' \in \mathcal{D} \mid D' \rightsquigarrow_V D\}$, $N = \bigcup\{D' \in \mathcal{D} \mid D' \rightsquigarrow_N D\}$, *and* $C = \bigcup\{D' \in \mathcal{D} \mid D' \not\rightsquigarrow D\}$. *The* basic scene $\mathcal{BS} = \{\mathcal{V}_D \mid D \in \mathcal{D}\}$ *for FP contains views for all domains in* $\mathcal{D}$.

We call $V$ the *visible*, $C$ the *confidential*, and $N$ the *non-confidential events* of $\mathcal{V}_D$. Only events in $V$ are directly observable from a given view. Among the non-observable events we distinguish events in $C$ which must be kept confidential and events in $N$ which need not. While events in $C$ must not be deducible, events in $N$ may be deducible, however, such deductions must not reveal any information about confidential events in $C$. Note that in Definition 4 each of the sets $V$, $N$, $C$ is constructed using one of $\rightsquigarrow_V$, $\rightsquigarrow_N$, $\not\rightsquigarrow$ (hence the indices).

*Example 1.* The basic scene for flow policy $FP1$ is depicted in the table on the right hand side of Figure 2. Most interesting is the view of domain $L$. For this domain, events in $L$ are visible, events in $HI$ confidential, and events in $H\setminus HI$

may be deduced (but must not reveal information about events in $HI$). The flow policy $FP2$ defines a multi-level security policy. $FP2$ could be used, for example, to express the security requirements of a file system with files of three different classifications: $T$ (top secret), $S$ (secret), and $U$ (unclassified). While events which involve files must not be deducible for domains which have a lower classification than these files, there is no such requirement for higher classifications. E.g. a user with clearance secret must not be able to learn anything about events on top secret files but may learn about unclassified files.

**Notational Conventions.** Throughout this article we assume that $ES$ denotes the event system $(E, I, O, Tr)$, that $SES$ denotes the state-event system $(S, S_I, E, I, O, T)$, and that $FP$ denotes the flow policy $(\mathcal{D}, \rightsquigarrow_V, \rightsquigarrow_N, \not\rightsquigarrow)$. The projection $\alpha|_{E'}$ of a sequence $\alpha \in E^*$ to the events in $E' \subseteq E$ results from $\alpha$ by deleting all events *not* in $E'$. We denote the set of all events in a given domain $D$ also by the name $D$ of the security domain and use that name in lower case, possibly with indices or primes, e.g. $d, d_1, \ldots$, to denote events in the domain. For a given view $\mathcal{V}$, we denote the components by $V_{\mathcal{V}}$, $N_{\mathcal{V}}$, and $C_{\mathcal{V}}$.

### 2.3 Formal Definitions of Information Flow

Various formal definitions of information flow have been proposed in the literature. Such a definition should accept a system as secure if and only if intuitively there is *no* information flow which violates the flow policy under consideration. The definitions of information flow which we investigate in this article follow the *possibilistic* approach. This is already implied by our choice of a system model in which only the possibility of behaviours is specified (in contrast to more complicated *probabilistic* models, e.g. [WJ90]). The possibilistic approach is compatible with non-determinism and allows us to abstract from probabilities and time.

When defining what information flow from a domain $D_1$ to a domain $D_2$ means, it is helpful to distinguish direct flow from indirect flow. *Direct flow* results from the observability of occurrences of events in $D_1$ from the perspective of $D_2$. For a given view $\mathcal{V} = (V, N, C)$ all occurrences of events in $V$ are directly observable, i.e. for a given behaviour $\tau \in E^*$, the projection $\tau|_V$ of $\tau$ to the visible events, is observed. *Indirect information flow* results from deductions about given observations. We assume that an adversary has complete knowledge of the static system, i.e. knows the possible behaviours in $Tr$. This is a worst case assumption which follows the 'no security by obscurity paradigm'. From this knowledge an adversary can deduce the set $\{\tau \in Tr \mid \tau|_V = \overline{\tau}\}$ of all traces which might have caused a given observation $\overline{\tau} \in V^*$. Confidentiality can be expressed as the requirement that this *equivalence set* is big enough in order to avoid leakage of confidential information. However, the various definitions of information flow formalize this requirement by different closure conditions.

*Non-inference* [O'H90], for example, demands that for any trace $\tau$ the sequence $\tau|_V$ must also be a trace, i.e. $\forall \tau \in Tr. \tau|_V \in Tr$. Thus, for non-inference, all equivalence sets must be closed under projections to events in $V$. For a system which fulfills non-inference, an adversary cannot deduce that confidential events have occurred because every observation could have been generated by a trace in which no such events have occurred. Another possibilistic definition is *separability* [McL96]. For any two traces $\tau_1, \tau_2$ it requires that any interleaving of the confidential subsequence of $\tau_1$ with the visible subsequence of $\tau_2$ must, again, be a trace. Thus, every confidential behaviour is compatible with every observation. Besides non-inference and separability, many other possibilistic definitions of information flow have been proposed (e.g. [Sut86,McC87,JT88,ZL97,FM99]) which correspond to different closure conditions on the equivalence sets. In order to simplify the investigation and comparison of such definitions, uniform frameworks have been developed [McL96,ZL97,Man00a].

Our assembly kit [Man00a] allows for the uniform and modular representation of possibilistic definitions of information flow. Each such definition is expressed as a *security predicate* which is assembled from *basic security predicates* (abbreviated by BSP in the sequel) by conjunction. BSPs can be classified in two dimensions. In the first dimension, it is required that the possible observations for a given view are *not increased* by the occurrence of confidential events. Otherwise, additional observations would be possible and one could deduce from such an observation that these confidential events must have occurred. In the second dimension the occurrence of confidential events must *not decrease* the possible observations. Otherwise, any of the observations which become impossible after these events, would lead to the conclusion that the confidential events have not occurred. In applications it can be sensible to emphasize one of these dimensions more than the other one. E.g. if a system is equipped with an alarm system then taking the alarm system off line must be kept confidential for possible intruders. However, it might be less important to keep it confidential that the alarm system has not been taken off-line because this is the default situation.

For the purposes of this paper it suffices to investigate two specific BSPs, one for each dimension. *B̲ackwards s̲trict d̲eletion of confidential events* ($BSD_{\mathcal{V}}$) demands for a given view $\mathcal{V} = (V, N, C)$ that the occurrence of an event from $C$ does *not add* possible observations. Considering the system after a trace $\beta$ has occurred, any observation $\overline{\alpha} \in V^*$ which is possible after $c \in C$ must also be possible if $c$ has not occurred. If the observation $\overline{\alpha}$ results from $\alpha \in (V \cup N)^*$, i.e. $\alpha|_V = \overline{\alpha}$, after $c$ has occurred then some $\alpha' \in (V \cup N)^*$ must be possible after $c$ has not occurred where $\alpha'$ may differ from $\alpha$ only in events from $N$. For a given view $\mathcal{V} = (V, N, C)$, $BSD_{\mathcal{V}}$ is formally defined as follows:

$$\begin{aligned} BSD_{V,N,C}(Tr) \equiv{} & \forall \alpha, \beta \in E^*. \forall c \in C.((\beta.c.\alpha \in Tr \wedge \alpha|_C = \langle\rangle) \\ & \Rightarrow \exists \alpha' \in E^*.(\alpha'|_V = \alpha|_V \wedge \alpha'|_C = \langle\rangle \wedge \beta.\alpha' \in Tr)) \ . \end{aligned}$$

Note that the definition of $BSD_{\mathcal{V}}$ becomes much simpler if $N = \emptyset$, i.e.

$$BSD_{V,\emptyset,C}(Tr) \equiv \forall \alpha, \beta \in E^*. \forall c \in C.((\beta.c.\alpha \in Tr \wedge \alpha|_C = \langle\rangle) \Rightarrow \beta.\alpha \in Tr) \, .$$

If $N$ is non-empty then the general definition of *BSD* is required for a correct handling of events in $N$. To allow such events in $\alpha$ and to allow their adaption in $\alpha'$ opens the spectrum from being deducible (but independent from confidential events) to being closely dependent on confidential events (but not deducible).

*Backwards strict insertion of admissible confidential events* ($BSIA_{\mathcal{V}}$) requires that the occurrence of an event from $C$ does *not remove* possible low-level observations. $\alpha$ and $\alpha'$ are related like in $BSD$. The additional premise $\beta.c \in Tr$ ensures that the event $c$ is admissible after $\beta$ which is a necessary condition for dependencies of confidential events on visible events [ZL97,Man00a].

$$\begin{aligned} BSIA_{V,N,C}(Tr) \equiv \forall \alpha, \beta \in E^*.\forall c \in C.((\beta.\alpha \in Tr \wedge \alpha|_C = \langle\rangle \wedge \beta.c \in Tr) \\ \Rightarrow \exists \alpha' \in E^*.(\alpha'|_V = \alpha|_V \wedge \alpha'|_C = \langle\rangle \wedge \beta.c.\alpha' \in Tr)) \end{aligned}$$

Inductive definitions of BSPs like $BSD$ and $BSIA$ were helpful to identify the two dimensions and simplified the development of unwinding conditions [Man00b]. They also provide a basis for handling intransitive policies in Section 3.

Recall that security predicates are constructed by conjoining BSPs. Each security predicate SP is a conjunction of BSPs. Often, one BSP from each dimension is taken. For example constructions of security predicates we refer to [Man00a]. Security predicates are parametric in the event system and in the flow policy. Fixing the flow policy yields a *security property* $\mathcal{SP} = (SP, FP)$.

**Definition 5.** *Let* $SP_{\mathcal{V}} \equiv BSP^1_{\mathcal{V}} \wedge \ldots \wedge BSP^n_{\mathcal{V}}$ *be a security predicate and FP be a transitive flow policy. The event system* $ES$ satisfies $(SP, FP)$ *iff* $BSP^i_{\mathcal{V}}(Tr)$ *holds for each* $i \in \{1, \ldots, n\}$ *and for each view* $\mathcal{V}$ *in the basic scene* $\mathcal{BS}_{FP}$.

*Example 2.* Let $ES = (E, I, O, Tr)$ be an event system which specifies a three-level file system and $FP2$ (cf. Figure 2) be the flow policy for $ES$. Assume that information flow is defined by the security predicate $SP_{\mathcal{V}} \equiv BSD_{\mathcal{V}} \wedge BSIA_{\mathcal{V}}$. This, together with $\mathcal{D} = \{U, S, T\}$ implies that the following theorems must be proved: $BSD_{\mathcal{V}_U}(Tr)$, $BSD_{\mathcal{V}_S}(Tr)$, $BSD_{\mathcal{V}_T}(Tr)$, $BSIA_{\mathcal{V}_U}(Tr)$, $BSIA_{\mathcal{V}_S}(Tr)$, and $BSIA_{\mathcal{V}_T}(Tr)$. The indices can be instantiated according to the table in Figure 2.

## 3 Intransitive Information Flow

Transitive flow policies, like the ones discussed in Example 1, are very restrictive. If $F \not\rightsquigarrow P$ is required for two domains $F$ and $P$ then *absolutely no* information must flow from $F$ to $P$. However, in practical applications it is often necessary to allow exceptions to such restrictions. Exceptions can be described by intransitive flow policies like $FP3$ (cf. Figure 2). In $FP3$, $F \not\rightsquigarrow P$ only requires that there is no information flow from $F$ *directly* to $P$. Although direct information flow is forbidden, information flow via the domain $L$ is permitted. Thus, $F \rightsquigarrow_V L$ and $L \rightsquigarrow_V P$ provide an exception to the requirement $F \not\rightsquigarrow P$. Events in $F$ may become deducible for $P$ if they are followed by events in $L$. An application for $FP3$ could be a system which consists of a printer (domain $P$), a labeller ($L$) and a file system ($F$). In $FP3$, $F \not\rightsquigarrow P$, $F \rightsquigarrow_V L$, and $L \rightsquigarrow_V P$ ensure that all files must be labelled before being printed. Note, that such a requirement could not be properly formalized with a transitive flow policy.

Unfortunately, intransitive flow policies have been outside the scope of definitions of information flow for non-deterministic systems. This includes the definitions investigated in [Sut86,McC87,JT88,O'H90,WJ90,McL96,ZL97] and also the BSPs which we discussed in Section 2.3 of this article. To the best of our knowledge, intransitive flow policies are outside the scope of all definitions of information flow which have been previously proposed for non-deterministic systems. The underlying problem is that these definitions cannot deal with exceptions. If a flow policy (like $FP3$) requires $F \not\rightsquigarrow P$ then these definitions require that there is *no* information flow from $F$ to $P$ (without exceptions).

In Section 3.1 we present further applications in which intransitive information flow is required. We illustrate the problems of previously proposed definitions of information flow with intransitive flow policies in Section 3.2 at the example of $BSD$. For one application we derive a specialized solution in Section 3.3 and then integrate a generalized solution into our assembly kit in Section 3.4. This allows us to represent BSPs which can cope with intransitive flow in the same uniform way as other BSPs. We evaluate our approach in Section 3.5.

### 3.1 More Applications of Intransitive Information Flow

Before we discuss the existing problems with intransitive flow policies we want to emphasize their practical importance by presenting typical applications for which intransitive information flow is necessary. The example of the printer/labeller/file system has already been investigated at the beginning of this section.

Another application is a communication component for connecting a system which contains classified data to an open network. In the component, a red side which has direct access to classified data and a black side which is connected to an open network are distinguished. Before a message which contains classified data may be passed from the red to the black side, the message body must be encrypted. The message header, however, may be transmitted in plaintext. This is expressed by the flow policy $FP4$ (cf. Figure 3). Events which involve the protected system are assigned domain $R$ (red), events which model encryption domain $CR$ (crypto), events which involve passing the header information domain $BP$ (bypass), and events which involve the open network domain $B$ (black). $FP4$ is an intransitive flow policy because the domains $BP$ and $CR$ provide exceptions to the requirement $R \not\rightsquigarrow B$.

Another application which requires an intransitive flow policy results from a modification of the three-level file system in Example 1. According to policy $FP2$ the classification of data cannot be lowered. However, the need to protect the confidentiality of data may disappear over time. For example, in order to execute

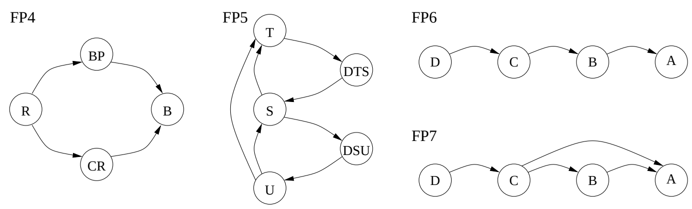



**Fig. 3.** More example flow policies ($\not\rightsquigarrow$ and reflexive subrelation of $\rightsquigarrow_V$ omitted)

a plan for a top secret mission, usually orders must be passed to people with lower clearance. Each of these orders reveals information about the mission plan. However, until the decision to execute the mission has been made no information about the corresponding mission plan must be revealed. This can be regarded as an example of downgrading. Policy $FP5$ in Figure 3 extends the policy $FP2$ for a three-level file system by two additional domains $DTS$ and $DSU$. These domains allow for the downgrading of information from top secret to secret (domain $DTS$) and from secret to unclassified ($DSU$).

### 3.2 The Problem

In order to illustrate the problems which are caused by intransitive policies, we use the printer/labeller/file system as a running example. Let $ES = (E, I, O, Tr)$ be the specification of such a system and $FP3$ (cf. Figure 2) be the flow policy which shall be enforced. Hence, files may only be passed to the printer if they have been labelled before. As definition of information flow we investigate *BSD*.

If we pretend that intransitive flow policies could be handled like transitive ones then we had to prove $BSD_{\mathcal{V}_P}(Tr)$, $BSD_{\mathcal{V}_L}(Tr)$, and $BSD_{\mathcal{V}_F}(Tr)$ (according to Definition 5). The view of the printer illustrates the problems with intransitivity. For this view we have to prove $BSD_{\mathcal{V}_P}(Tr)$, i.e.

$$\forall \alpha, \beta \in E^*. \forall f \in F.((\beta.f.\alpha \in Tr \wedge \alpha|_F = \langle\rangle) \Rightarrow \beta.\alpha \in Tr) \ . \qquad (1)$$

This requirement is too strong as the following example illustrates. Let $write(f, d)$ denote an event in which the contents of file $f$ is replaced by data $d$, $label(f, d, ld)$ denote an event in which the contents $d$ of file $f$ is labelled with result $ld$, and $print(ld)$ denote an event in which the data $ld$ is sent to the printer. Then

$$write(f_1, d_1).write(f_1, d_2).label(f_1, d_2, lab(d_2)).print(lab(d_2)) \qquad (2)$$

is a possible trace of the system. We assign domains by $dom(write(\_,\_)) = F$, $dom(label(\_,\_,\_)) = L$, and $dom(print(\_)) = P$. Thus, $BSD_{\mathcal{V}_P}(Tr)$ requires

$$write(f_1, d_1).label(f_1, d_2, lab(d_2)).print(lab(d_2)) \in Tr \, . \qquad (3)$$

The conclusion is (with $d_1 \neq d_2$) that the labeller must not depend on any changes to the contents of files but rather has to invent the data which it labels. This restriction is caused by the use of *BSD* and not by the flow policy according to which $F \rightsquigarrow_V L$ holds. In any sensible implementation of such a system the labeller would depend on the file system and, thus, the implementation would be rejected by *BSD* as being insecure, even if it intuitively respects $FP3$. Hence *BSD* is incompatible with the intransitive flow in policy $FP3$.

This example points to a general problem which is neither a peculiarity of *BSD* nor of this particular example. All previously proposed definitions of information flow for non-deterministic systems exclude intransitive information flow. Any system with intransitive flow would be rejected by these definitions as being insecure, even if it intuitively complies with the respective (intransitive) flow policy. This incompatibility has made it impossible to apply information flow control to non-deterministic systems when intransitive policies shall be enforced. However, intransitive flow is required by many applications (cf. the examples in Section 3.1). Thus, a limitation to transitive flow policies would be rather severe.

### 3.3 Towards a Solution

What is the reason for this problem? Let us revisit the printer/labeller/file system in which, according to $FP3$, events from domain $F$ *may* become deducible through events from domain $L$. However, $BSD_{\mathcal{V}_P}$ (cf. formula (1)) requires that deleting the last event with domain $F$ from a trace *must* again yield a trace, no matter whether an event with domain $L$ occurs or not. This is the reason why *BSD* is too restrictive for intransitive flow policies. Formally this problem is caused by the assumption $\alpha|_F = \langle\rangle$ in formula (1). Thus, the first step towards a solution is to replace it by the stronger assumption $\alpha|_{F \cup L} = \langle\rangle$. This results in

$$\forall \alpha, \beta \in E^*. \forall f \in F.((\beta.f.\alpha \in Tr \wedge \alpha|_{F \cup L} = \langle\rangle) \Rightarrow \beta.\alpha \in Tr) \ . \tag{4}$$

This modification of $BSD_{\mathcal{V}_P}$ requires that deleting events with domain $F$ must yield a trace only if these events are not followed by any events with domain $L$, e.g. deleting $write(f_1, d_2)$ from trace (2) need not yield a trace. This precisely reflects the requirements of the flow policy $FP3$. According to $FP3$, events in domain $F$ may be deduced by domain $P$ if they are followed by events in domain $L$. Thus, events in $L$ *extend* the view of $P$.

We now generalize this idea to arbitrary flow policies and define the notion of an *extension set*. For a given domain $D$, the extension set $X_D$ contains all events which are visible to $D$ and which possibly extend the view of $D$. Formally, $X_D$ is defined by $X_D = \bigcup\{D' \in \mathcal{D} \mid D' \rightsquigarrow_{V_D} D \wedge D' \neq D\}$. Generalizing formula (4) to an arbitrary view $\mathcal{V} = (V, N, C)$ and extension set $X$ results in

$$\begin{aligned} &\forall \alpha, \beta \in E^*. \forall c \in C.((\beta.c.\alpha \in Tr \wedge \alpha|_{C \cup X} = \langle\rangle) \\ &\Rightarrow \exists \alpha' \in E^*.(\alpha'|_V = \alpha|_V \wedge \alpha'|_{C \cup X} = \langle\rangle \wedge \beta.\alpha' \in Tr)) \, . \end{aligned} \tag{5}$$

If $X \cap N = \emptyset$ (which will hold in this article) then formula (5) is weaker than $BSD_{\mathcal{V}}(Tr)$. In fact, it is too weak as we will now illustrate at the flow policy $FP6$ in Figure 3 (The problem does not occur with $FP3$.). Let $a$, $b$, $c$, $d$ be events respectively with domain $A$, $B$, $C$, $D$, and $Tr = \{\langle\rangle, d, b, d.b, d.b.a\}$ be a set of traces. According to $Tr$, $a$ is only enabled if $d.b$ has previously occurred. Thus, an observer with view $A$ can conclude from the observation $a$ that $d$ has occurred. Such deductions result in information flow from $D$ to $A$ through $B$ which does not comply with the policy ($D \not\rightsquigarrow A$, $D \not\rightsquigarrow B$). Intuitively, $Tr$ violates $FP6$. Nevertheless, formula (5) is fulfilled for each of the views $\mathcal{V}_A$, $\mathcal{V}_B$, $\mathcal{V}_C$, $\mathcal{V}_D$ and the extension sets $X_A$, $X_B$, $X_C$, and $X_D$. The reason is that the assumptions of formula (5) are not fulfilled for a trace which contains an event $x \in X$ which is not followed by any events from $C$. Thus, formula (5) enforces no restrictions for such a trace. However, rather than making no restrictions, it should enforce $FP7$ (cf. Fig. 3) for such traces which compared to $FP6$ additionally permits information flow from $C$ to $A$. Note that information flow from $D$ to $A$ is *not* permitted by $FP7$. $FP7$ results from $FP6$ by combining the views of $A$ and $B$.

Consequently, BSPs should be enforced for a larger set of views. Additional views result from the combination of domains. Such combinations are constructed along $\rightsquigarrow_V$, e.g. $AB$ denotes the combination of $A$ and $B$ in $FP6$ which we discussed above. Other combinations are $BC$, $CD$, $ABC$, $BCD$, and $ABCD$. The resulting views which must be investigated for $FP6$ are depicted in Figure 4. Note that there are six additional views, $\mathcal{V}_{AB}$, $\mathcal{V}_{BC}$, $\mathcal{V}_{CD}$, $\mathcal{V}_{ABC}$, $\mathcal{V}_{BCD}$, and $\mathcal{V}_{ABCD}$ which are not contained in the basic scene. We will refer to the extension of basic scenes by these views as *scene*.

### 3.4 A Solution

The solution for intransitive information flow which we have derived for our running example can now be generalized to arbitrary systems and flow policies. The example showed that definitions of information flow like the basic security predicate *BSD* rule out intransitive flow. In order to be able to cope with intransitive policies in formula (5) we had to introduce the extension set $X$ as additional parameter. We now present two novel BSPs: *IBSD* (*intransitive backwards strict deletion of confidential events*) and *IBSIA* (*intransitive backwards strict insertion of admissible confidential events*) which are respectively derived from *BSD* and *BSIA* but which are compatible with intransitive flow. Let $\mathcal{V} = (V, N, C)$.

$$\begin{aligned} IBSD^X_{\mathcal{V}}(Tr) \equiv\ & \forall \alpha, \beta \in E^*. \forall c \in C.((\beta.c.\alpha \in Tr \wedge \alpha|_{C\cup X} = \langle\rangle) \\ & \Rightarrow \exists \alpha' \in E^*.(\alpha'|_V = \alpha|_V \wedge \alpha'|_{C\cup X} = \langle\rangle \wedge \beta.\alpha' \in Tr)) \end{aligned}$$

$$\begin{aligned} IBSIA^X_{\mathcal{V}}(Tr) \equiv\ & \forall \alpha, \beta \in E^*. \forall c \in C.((\beta.\alpha \in Tr \wedge \alpha|_{C\cup X} = \langle\rangle \wedge \beta.c \in Tr) \\ & \Rightarrow \exists \alpha' \in E^*.(\alpha'|_V = \alpha|_V \wedge \alpha'|_{C\cup X} = \langle\rangle \wedge \beta.c.\alpha' \in Tr)) \end{aligned}$$

Apparently, *IBSD* and *IBSIA* are very similar respectively to *BSD* and *BSIA*. The only differences are that $\alpha|_C = \langle\rangle$ is replaced by $\alpha|_{C\cup X} = \langle\rangle$ and $\alpha'|_C = \langle\rangle$ by $\alpha'|_{C\cup X} = \langle\rangle$. We now state some simple facts about the validity of these new BSPs and relate them to the existing ones.

**Fact 1.** *Let $\mathcal{V} = (V, N, C)$ be a view and $X \subseteq E$.*

1. $IBSD^{\emptyset}_{\mathcal{V}}(Tr)$ [$IBSIA^{\emptyset}_{\mathcal{V}}(Tr)$] *if and only if* $BSD_{\mathcal{V}}(Tr)$ [$BSIA_{\mathcal{V}}(Tr)$]
2. *If* $IBSD^{\emptyset}_{\mathcal{V}}(Tr)$ [$IBSIA^{\emptyset}_{\mathcal{V}}(Tr)$] *and* $X \subseteq V$ *then* $IBSD^X_{\mathcal{V}}(Tr)$ [$IBSIA^X_{\mathcal{V}}(Tr)$].
3. $IBSD^X_{V,N,\emptyset}(Tr)$ *and* $IBSIA^X_{V,N,\emptyset}(Tr)$ *hold.*

For intransitive policies it does not suffice to investigate the views of single domains. Rather the views of combinations of domains along $\rightsquigarrow_V$ must be considered as well. The need for the investigation of such views arises from the fact that events which are not deducible for a given domain can become deducible if they are followed by certain other events. E.g. in $FP6$, which we discussed at the

| | $A$ | $B$ | $C$ | $D$ | $AB$ | $BC$ | $CD$ | $ABC$ | $BCD$ | $ABCD$ |
|---|---|---|---|---|---|---|---|---|---|---|
| $V$ | $A\cup B$ | $B\cup C$ | $C\cup D$ | $D$ | $A\cup B\cup C$ | $B\cup C\cup D$ | $C\cup D$ | $A\cup B\cup C$ | $B\cup C\cup D$ | $A\cup B\cup C\cup D$ |
| $N$ | $\emptyset$ | $\emptyset$ | $\emptyset$ | $\emptyset$ | $\emptyset$ | $\emptyset$ | $\emptyset$ | $\emptyset$ | $\emptyset$ | $\emptyset$ |
| $C$ | $C\cup D$ | $A\cup D$ | $A\cup B$ | $A\cup B\cup C$ | $D$ | $A$ | $A\cup B$ | $\emptyset$ | $A$ | $\emptyset$ |
| $X$ | $B$ | $C$ | $D$ | $\emptyset$ | $C$ | $D$ | $\emptyset$ | $D$ | $\emptyset$ | $\emptyset$ |

**Fig. 4.** Basic scene and scene for $FP6$

end of the previous subsection, events in $C$ may become deducible for $A$ if they are followed by events in $B$. Events in $D$ may also become deducible for $A$ but this requires that they are followed by events in $C$ *and* events in $B$. The following definition expresses which combinations of domains must be considered.

**Definition 6.** *Let $FP = (\mathcal{D}, \rightsquigarrow_V, \rightsquigarrow_N, \not\rightsquigarrow)$ be a flow policy. The set of* combined domains *$\mathcal{C}_{FP} \subseteq \mathcal{P}(\mathcal{D})$ for FP is the minimal set which is closed under*

1. *If $D \in \mathcal{D}$ then $\{D\} \in \mathcal{C}_{FP}$ and*
2. *if $\mathcal{D}' \in \mathcal{C}_{FP}$, $D \in \mathcal{D}$, and $\exists D' \in \mathcal{D}'.D \rightsquigarrow_V D'$ then $\mathcal{D}' \cup \{D\} \in \mathcal{C}_{FP}$.*

For intransitive policies, extended views must be considered. An *extended view* $\mathcal{X}$ is a pair $(\mathcal{V}, X)$ consisting of a view $\mathcal{V}$ and a set $X$ of events, the *extension set*. Since extended views for combined domains must be considered, we define the extended view for sets $\mathcal{D}'$ of domains rather than for single domains.

**Definition 7.** *The* extended view *$\mathcal{X}_{\mathcal{D}'} = ((V, N, C), X)$ for $\mathcal{D}' \subseteq \mathcal{D}$ is defined by*

$$\begin{aligned}
V &= \{e \in E \mid \exists D' \in \mathcal{D}'.dom(e) \rightsquigarrow_V D'\} \\
N &= \{e \in E \mid \nexists D' \in \mathcal{D}'.dom(e) \rightsquigarrow_V D' \wedge \exists D' \in \mathcal{D}'.dom(e) \rightsquigarrow_N D'\} \\
C &= \{e \in E \mid \forall D' \in \mathcal{D}'.dom(e) \not\rightsquigarrow D'\} \\
X &= \bigcup \{D \in \mathcal{D} \mid \exists D' \in \mathcal{D}'.D \rightsquigarrow_V D' \wedge D \notin \mathcal{D}'\}
\end{aligned}$$

*The* scene *$\mathcal{S}_{FP}$ for FP contains the extended view $\mathcal{X}$ for each $\mathcal{D}' \in \mathcal{C}_{FP}$.*

We now state some facts about scenes which directly follow from Definition 7.

**Fact 2.** *Let $\mathcal{V} = (V, N, C)$ be a view and $X \subseteq E$.*

1. *If $(\mathcal{V}, X) \in \mathcal{S}_{FP}$ then $X \subseteq V$.*
2. *If FP is transitive then $(\mathcal{V}, X) \in \mathcal{S}_{FP} \Rightarrow (\mathcal{V}, \emptyset) \in \mathcal{S}_{FP}$.*

We now define when a security property with an arbitrary flow policy is satisfied.

**Definition 8.** *Let $ISP_{\mathcal{V}}^{X} \equiv IBSP_{\mathcal{V}}^{X,1} \wedge \ldots \wedge IBSP_{\mathcal{V}}^{X,n}$ be an intransitive security predicate and FP be a flow policy. An event system ES* satisfies *$(ISP, FP)$ iff $IBSP_{\mathcal{V}}^{X,i}(Tr)$ holds for each $i \in \{1, \ldots, n\}$ and for each $\mathcal{X}$ in the scene $\mathcal{S}_{FP}$.*

Definition 5 stated when an event system satisfies a given security property with a transitive flow policy. Clearly, Definition 5 and Definition 8 should be equivalent for the special case of transitive flow policies. The following theorem ensures that this, indeed, holds for *IBSD* and *IBSIA*.

**Theorem 1.** *Let FP be a transitive flow policy.*

1. *$BSD_{\mathcal{V}}(Tr)$ holds for each view $\mathcal{V}$ in the basic scene $\mathcal{BS}_{FP}$ if and only if $IBSD_{\mathcal{V}}^{X}(Tr)$ holds for each extended view $(\mathcal{V}, X)$ in the scene $\mathcal{S}_{FP}$.*
2. *$BSIA_{\mathcal{V}}(Tr)$ holds for each view $\mathcal{V}$ in the basic scene $\mathcal{BS}_{FP}$ if and only if $IBSIA_{\mathcal{V}}^{X}(Tr)$ holds for each extended view $(\mathcal{V}, X)$ in the scene $\mathcal{S}_{FP}$.*

*Proof.* We prove the first proposition. The second can be proved analogously.

$\Rightarrow$) Assume that $BSD_{\mathcal{V}}(Tr)$ holds for each $\mathcal{V} \in \mathcal{BS}_{FP}$. With Fact 1.1 we receive $IBSD_{\mathcal{V}}^{\emptyset}(Tr)$. Fact 1.2 implies $IBSD_{\mathcal{V}}^{X}(Tr)$ for all $X \subseteq V_{\mathcal{V}}$. From Fact 2.1 we conclude that $IBSD_{\mathcal{V}}^{X}(Tr)$ holds for all $(\mathcal{V}, X) \in \mathcal{S}_{FP}$.

$\Leftarrow$) Assume that $IBSD_{\mathcal{V}}^{X}(Tr)$ holds for each $(\mathcal{V}, X) \in \mathcal{S}_{FP}$. Fact 2.2 implies that if $IBSD_{\mathcal{V}}^{X}(Tr)$ then $IBSD_{\mathcal{V}}^{\emptyset}(Tr)$. From Fact 1.1 we conclude that $BSD_{\mathcal{V}}(Tr)$ holds for each $\mathcal{V} \in \mathcal{BS}_{FP}$. □

### 3.5 The Solution Revisited

Theorem 1 demonstrates that *IBSD* and *IBSIA* are, respectively, extensions of *BSD* and *BSIA* to the intransitive case. For the special case of transitive flow policies the corresponding BSPs are equivalent. However, in the intransitive case the new BSPs are less restrictive. E.g. *IBSD* accepts systems with intransitive flow as secure wrt. a given (intransitive) flow policy if they intuitively comply with this policy while *BSD* rejects any system with intransitive flow as insecure. It remains to be shown that *IBSD* (and *IBSIA*) rejects systems with intransitive flows as insecure if they intuitively violate the (intransitive) policy under consideration. We demonstrate this by several examples. Note that a formal proof of such a statement is impossible since the point of reference is our intuition.

We use the 3-level file system with 2 downgraders from Section 3.1 and flow policy $FP5$ from Figure 3 as running example. For each case, which we investigate, we assume that the system is intuitively insecure in a certain sense and then argue that *IBSD* indeed rejects the system as insecure.

*Example 3.* Let us first assume that downgrading events never occur. Thus there should not be any intransitive information flow in the system even though the flow policy is intransitive. Moreover, assume that domain $U$ can deduce that events from $T$ have occurred. Thus, the system is intuitively insecure. However, since events from $DSU \cup DTS$ do not occur in traces, *IBSD* enforces the same restrictions (for the extended views $\mathcal{X}_{TSU}$, $\mathcal{X}_{SU}$, $\mathcal{X}_{U}$ in $\mathcal{S}_{FP5}$) as *BSD* does for the views $\mathcal{V}_T$, $\mathcal{V}_S$, $\mathcal{V}_U$ in the basic scene of $FP2$. Thus, *IBSD* rejects such a system as insecure wrt. $FP5$ because *BSD* would reject the system for $FP2$.

Let us next assume that only downgrading events in $DSU$ never occur. Thus, there should not be any information flow from $T$ or $S$ to $U$ and information may flow from $T$ to $S$ only via $DTS$. Firstly, assume that the system is intuitively insecure because $U$ can deduce the occurrence of events from $T \cup S$. *IBSD* rejects such a system as insecure. The reason is that *BSD* would reject such a system for the (transitive) flow policy which is defined by $U \rightsquigarrow_V S$, $U \rightsquigarrow_V DTS$, $U \rightsquigarrow_V T$, $S \rightsquigarrow_V DTS$, $S \rightsquigarrow_V T$, $DTS \rightsquigarrow_V S$, $DTS \rightsquigarrow_V T$, $T \rightsquigarrow_V S$, $T \rightsquigarrow_V DTS$, $S \not\rightsquigarrow U$, $DTS \not\rightsquigarrow U$, and $T \not\rightsquigarrow U$. Secondly, assume that the system is intuitively insecure because $S$ can deduce the occurrence of events from $T$ which are not followed by any events from $DTS$. Thus, there must be a sequence $\beta.t.\alpha \in Tr$ with $t \in T$, $\alpha|_{DTS \cup T} = \langle\rangle$ such that $\beta.\alpha \notin Tr$. However, this would violate $IBSD_{\mathcal{X}}$ for the extended view $\mathcal{X} = ((DTS \cup S \cup DSU \cup U, \emptyset, T), DTS)$.

Let us now assume that no downgrading events in $DTS$ occur. If such a system is intuitively insecure then it is rejected by *IBSD* as insecure. The argument can be carried out along the same lines as in the previously discussed case where no events in $DSU$ occurred. The case which remains to be discussed is the general case in which events from all domains may occur. In this case more kinds of (intuitive) insecurity must be investigated. However, for each of these insecurities one can argue along the same lines as before that *IBSD* correctly rejects any corresponding (intuitively insecure) system.

## 4 Verification Conditions

*Unwinding conditions* simplify the proof that a system satisfies a given security property. While BSPs like *IBSD* or *IBSIA* are expressed in terms of *sequences of events*, unwinding conditions are stated in terms of the pre- and postconditions of *single events*. In [Man00b], we have presented such unwinding conditions for a large class of BSPs which can be applied for transitive flow policies. By an *unwinding theorem* we have guaranteed that these unwinding conditions are correct. The development of unwinding conditions for *IBSD* and *IBSIA* along the same lines is a straightforward task. However, in general, the unwinding conditions must be proved for all combinations of domains (cf. Definition 6) rather than only for single domains (as in [Man00b]). Interestingly, this can be optimized for the special case of flow policies with $\rightsquigarrow_N = \emptyset$. In this section we demonstrate that it suffices for such policies to prove the unwinding conditions for single domains only, thus, reducing the verification burden considerably.

In order to express unwinding conditions we use state-event systems (cf. Definition 2) and make the same assumptions as in [Man00b], i.e. there is only one initial state $s_I$ and the effect of events is deterministic (the transition relation $T$ is functional). However, state-event systems are still non-deterministic because of the choice between different events and since internal events may cause effects.

The *successor set* for $s_1 \in S$ and $e \in E$ is $succ(s_1, e) = \{s_2 \in S \mid (s_1, e, s_2) \in T\}$. According to our simplification, $succ(s_1, e)$ has at most one element. We extend $succ$ to sets $S_1 \subseteq S$ of states and sequences $\alpha \in E^*$ of events:

$$succ(S_1, \alpha) \equiv \mathit{if}\ \alpha = \langle\rangle\ \mathit{then}\ S_1\, \mathit{else\ let}\ e.\alpha' = \alpha\ \mathit{in}\ succ(\textstyle\bigcup_{s \in S_1} succ(s, e), \alpha')\ .$$

A sequence $\alpha$ of events is *enabled*, denoted by $enabled(\alpha, s)$, in a state $s$ if and only if $succ(s, \alpha) \neq \emptyset$. A state $s$ is *reachable*, denoted by $reachable(s)$, if and only if there is a sequence $\alpha$ of events such that $s \in succ(s_I, \alpha)$.

Our unwinding conditions are based on preorders (unlike most other approaches which are based on equivalence relations, e.g. [RS99]). For a discussion of the advantage of using preorders we refer to [Man00b]. A *domain possibility preorder* for a domain $D \in \mathcal{D}$ is a reflexive and transitive relation $\ltimes_D \subseteq S \times S$. Our intuition is that $s_1 \ltimes_D s_2$ should imply that every $D$-observation which is possible in $s_1$ should also be possible in $s_2$. We now construct a relation $\sqsubseteq_{\mathcal{D}'}$ with a corresponding idea for combined domains, i.e. $s_1 \sqsubseteq_{\mathcal{D}'} s_2$ should imply that every $\mathcal{D}'$-observation which is possible in $s_1$ should also be possible in $s_2$.

**Definition 9.** *Let $\mathcal{D}' \subseteq \mathcal{D}$ be a set of domains and $(\ltimes_D)_{D \in \mathcal{D}}$ be a family of domain possibility preorders on $S$. We define a relation $\sqsubseteq_{\mathcal{D}'} \subseteq S \times S$ by*

$$s_1 \sqsubseteq_{\mathcal{D}'} s_2 \equiv \forall D \in \mathcal{D}'.s_1 \ltimes_D s_2\ .$$

Each of our unwinding conditions $wosc_D$, $lrf_D$, and $lrb_D$ is defined in terms of single events. $wosc_D$ (weak output step consistency) demands that $s_1 \ltimes_D s'_1$ implies that any event $e' \in D$ which is enabled in $s_1$ is also enabled in $s'_1$. Moreover, if $s_1 \ltimes_D s'_1$, $s_1 \ltimes_{dom(e)} s'_1$, and an event $e \in E$ is enabled in $s_1$ then $e$ is also enabled in $s'_1$ and the preorder is preserved after the occurrence of $e$, i.e. $s_2 \ltimes_D s'_2$ holds for the successor states. If an event $e \in E$ is enabled in a state $s$ with resulting state $s'$ then $s' \ltimes_D s$ is required for all domains $D$ with $dom(e) \not\rightsquigarrow D$ by $lrf_D$ (locally respects forwards). Similarly, $lrb_D$ (locally respects backwards) requires $s \ltimes_D s'$.

$$
\begin{aligned}
wosc_D &: \forall s_1, s_2, s'_1 \in S. \forall e \in E.((s_1 \ltimes_D s'_1 \wedge (s_1, e, s_2) \in T \wedge s_1 \ltimes_{dom(e)} s'_1) \\
&\qquad \Rightarrow \exists s'_2 \in S.(s'_2 \in succ(s'_1, e) \wedge s_2 \ltimes_D s'_2) \\
lrf_D &: \forall s, s' \in S. \forall e \in E.((dom(e) \not\rightsquigarrow D \wedge reachable(s) \wedge (s, e, s') \in T) \Rightarrow s' \ltimes_D s) \\
lrb_D &: \forall s \in S. \forall e \in E.((dom(e) \not\rightsquigarrow D \wedge reachable(s) \wedge enabled(e, s)) \\
&\qquad \Rightarrow (\exists s' \in S.(s, e, s') \in T \wedge s \ltimes_D s'))
\end{aligned}
$$

The following lemma shows that $\ltimes_D$ and $\sqsubseteq_{\mathcal{D}'}$ respectively are orderings on $D$- and $\mathcal{D}'$-observations of arbitrary length when $wosc_D$ holds for all $D \in \mathcal{D}$.

**Lemma 1.** *If $SES$ fulfills $wosc_D$ for $\ltimes_D$ for all $D \in \mathcal{D}$ then*

$$
\begin{aligned}
&\forall s_1, s'_1 \in S. \forall \mathcal{D}' \subseteq \mathcal{D}. \forall \alpha \in (\textstyle\bigcup_{D \in \mathcal{D}'} D)^*.((s_1 \sqsubseteq_{\mathcal{D}'} s'_1 \wedge enabled(\alpha, s_1)) \\
&\Rightarrow \exists s_n \in succ(s_1, \alpha), s'_n \in succ(s'_1, \alpha). s_n \sqsubseteq_{\mathcal{D}'} s'_n) .
\end{aligned}
$$

*Proof.* We prove the lemma by induction on the length of $\alpha$. In the base case, i.e. for $\alpha = \langle\rangle$, the proposition holds trivially. In the step case, i.e. for $\alpha = e_1.\alpha'$, we assume $s_1 \sqsubseteq_{\mathcal{D}'} s'_1$ and $enabled(\alpha, s_1)$. Thus, there is a state $s_2 \in succ(s_1, e_1)$ with $enabled(\alpha', s_2)$. Let $D \in \mathcal{D}'$ be arbitrary. $s_1 \ltimes_D s'_1$, $(s_1, e, s_2) \in T$, $s_1 \ltimes_{dom(e)} s'_1$, and $wosc_D$ imply that there is a $s'_2 \in S$ with $(s'_1, e, s'_2) \in T$ and $s_2 \ltimes_D s'_2$. Since $D$ is arbitrary, we receive $s_2 \sqsubseteq_{\mathcal{D}'} s'_2$. $s_2 \sqsubseteq_{\mathcal{D}'} s'_2$, $enabled(\alpha', s_2)$, and the induction hypothesis imply the lemma. □

**Theorem 2 (Unwinding Theorem).** *Let $FP$ be a security policy with a finite set $\mathcal{D}$ of disjoint domains and $\rightsquigarrow_N = \emptyset$.*

1. $\forall D \in \mathcal{D}.(wosc_D \wedge lrf_D) \Rightarrow \forall \mathcal{D}' \in \mathcal{C}_{FP}.IBSD^{\mathcal{X}_{\mathcal{D}'}}_{\mathcal{V}_{\mathcal{D}'}}(Tr)$
2. $\forall D \in \mathcal{D}.(wosc_D \wedge lrb_D) \Rightarrow \forall \mathcal{D}' \in \mathcal{C}_{FP}.IBSIA^{\mathcal{X}_{\mathcal{D}'}}_{\mathcal{V}_{\mathcal{D}'}}(Tr)$

*Proof.* We prove the first proposition. The second can be proved analogously.

Let $\mathcal{D}' \in \mathcal{C}_{FP}$ and $\mathcal{X}_{\mathcal{D}'} = ((V, N, C), X)$. Let $\beta.c.\alpha \in Tr$ be arbitrary with $c \in C$ and $\alpha|_{C \cup X} = \langle\rangle$. We have to show that $\beta.\alpha \in Tr$ holds. $\beta.c.\alpha \in Tr$ implies that there are states $s_1, s_2 \in S$ with $s_1 \in succ(s_I, \beta)$, $(s_1, c, s_2) \in T$, and $enabled(\alpha, s_2)$. We choose $D' \in \mathcal{D}'$ arbitrarily. $dom(c) \not\rightsquigarrow D'$ because of $c \in C$ (cf. Definition 7). From $lrf_{D'}$ we conclude $s_2 \ltimes_{D'} s_1$. This implies $s_2 \sqsubseteq_{\mathcal{D}'} s_1$ because $D'$ was chosen arbitrarily. Finally, Lemma 1 implies that $enabled(\alpha, s_1)$, i.e. $\beta.\alpha \in Tr$. □

The unwinding theorem ensures that a proof of the unwinding conditions implies that the flow policy is respected, i.e. the unwinding conditions are correct. Interestingly, it suffices to prove the unwinding conditions *for single domains* (rather than for combined domains) for policies with $\leadsto_N = \emptyset$. In order to show that the unwinding conditions are not too restrictive a completeness result would be desirable. In the transitive case such a completeness result can be achieved if $\leadsto_N = \emptyset$ holds (cf. [Man00b]). For the intransitive case no general completeness result holds unless one makes the additional (quite artificial) assumption that different sequences of events always result in different states. However, we plan to investigate these issues more closely in future research.

## 5 Related Work

The approach to information flow control in non-deterministic systems which we have proposed in this article is compatible with intransitive information flow. All previously proposed approaches are either restricted to deterministic systems or cannot cope with intransitive information flow.

Information flow control based on non-interference was first introduced by Goguen and Meseguer in [GM82]. This original version of non-interference was incompatible with intransitive information flow. In order to overcome this shortcoming for channel control policies, a special case of intransitive flow policies, an *unless* construct was introduced in [GM84]. However, this *unless* construct did not capture the intuition of intransitive flow. It accepted many intuitively insecure systems as being secure. This weakness of the *unless* construct has some similarities to the weakness which would result from using basic scenes (rather than scenes) together with *IBSD* and *IBSIA* in our approach (cf. Section 3.3). The first satisfactory formal account of intransitive information flow was proposed by Rushby [Rus92]. The key for the compatibility with intransitive flow in his solution was the use of an *ipurge* function instead of the traditional *purge* function. A similar notion of non-interference was proposed by Pinsky [Pin95]. All work discussed so far in this section uses deterministic state machines as system model and, thus, is not directly applicable to non-deterministic systems. Another approach (based on determinism in CSP) which is more restrictive than Rushby's approach has been proposed by Roscoe and Goldsmith [RG99]. It detects some insecurities which are not detected by Rushby's approach but, since it is based on determinism, an extension to distributed systems will be difficult.

The first generalization of non-interference to non-deterministic systems was non-deducibility as proposed by Sutherland [Sut86]. Subsequently, various other generalizations (e.g. [McC87,O'H90,McL96,ZL97]) have been proposed and there seems not to be one single optimal generalization of non-interference for non-deterministic systems. To our knowledge, none of these generalizations can cope with intransitive information flow. The system models underlying the different approaches are either state based, like non-deterministic state machines, or event based, like event systems or the process algebras CSP or CCS. The various event based models differ in which specifications they consider as semantically equivalent. While event systems use trace semantics, i.e. specifications are equivalent if they describe the same set of traces, CSP uses failure divergence semantics (early versions used trace semantics), and CCS uses weak bisimulation. Trace semantics identify more specifications than failure divergence or weak bisimulation semantics, however, none of these semantics is in general superior to one of the others. For an overview on these and other semantics we refer to [vG90, Sch00].

Today, Rushby's approach to information flow control with intransitive policies seems to be the most popular one for deterministic systems. It is feasible for real applications as has been demonstrated by case studies like [SRS+00]. However, Roscoe and Goldsmith [RG99] recently identified a shortcoming of Rushby's solution which we explain at the example of the flow policy $FP5$ (cf. Figure 3). Let us assume that the file system has two files $f_{t1}$ and $f_{t2}$ which are both assigned the security domain $T$ and that there are two downgrading events $dts_1$ and $dts_2$ with domain $DTS$ which should respectively downgrade information only about either $f_{t1}$ or $f_{t2}$. Note, however, that no such requirement is expressed in $FP5$. Consequently, certain insecurities, e.g that $dts_1$ downgrades information about $f_{t1}$ as well as $f_{t2}$, cannot be detected by applying Rushby's intransitive non-interference. Roscoe and Goldsmith argued that this would be a shortcoming of Rushby's definition of information flow. However, we do not fully agree with this critique (although it points to an important problem) because it does not identify a problem which is specific to this definition of information flow. Either a security requirement can be expressed by a flow policy (e.g. by assigning different domains $T1$ and $T2$ respectively to $f_{t1}$ and $f_{t2}$) or the concept of flow policies alone is not adequate and, hence should be combined with some other concept which further restricts possible downgrading of information. In the first case, Rushby's intransitive non-interference can be applied but, in the second case, flow policies are insufficient, in general. Intransitive flow policies restrict *where* downgrading can occur but do not allow further restrictions on *what* may be downgraded. How to specify *good downgrading* is an important question which, however, is unresolved for deterministic as well as for non-deterministic systems. In our opinion the contribution of [Rus92] has been to allow information flow control to be applied for restricting *where* downgrading can occur.

Unwinding conditions for information flow control (in the intransitive case) have been proposed by Rushby [Rus92] and Pinsky [Pin95]. While Rushby's unwinding conditions are based on equivalence relations, Pinsky's unwinding conditions are based on equivalence classes ($\beta$-families in his terminology). Both authors proved unwinding theorems which ensure the correctness of their unwinding conditions and also present completeness results. However, the completeness results are limited to the special case of transitive policies (in [Pin95] this restriction results from the assumption $\mathrm{SA}(\mathrm{basis}_\pi(z), \alpha) \subseteq \mathrm{view}(\mathrm{state\_action}(z, \alpha))$ in the proof of the corollary on page 110).

## 6 Conclusion

When using information flow control in real applications it is often necessary to allow for certain exceptions to the restrictions of information flow. Such exception can be expressed by intransitive flow policies. The incompatibility of all previously proposed approaches for information flow control in *non-deterministic* *systems* with intransitive policies created a major gap which has prevented the migration of research results on information flow control into practice. In this article, we have constructed a bridge over this gap by proposing an approach to information flow control which is compatible with intransitive flow and which can be applied to non-deterministic systems. We have argued that our approach only accepts systems as secure if they are intuitively *secure wrt. a given flow policy* (cf. Section 3.5). Thus, the same kind of insecurities are detected as in Rushby's approach for *deterministic systems*. Consequently, our approach also suffers from the limitations identified in [RG99]. However, these are limitations of flow policies in general (cf. Section 5). Although the properties of our solution are similar to the ones of Rushby's solution, our formalization differs considerably. This is a necessary difference because our work is based on a different system model, i.e. event systems, which is compatible with non-determinism while Rushby's state machines are deterministic.

We have integrated our approach to information flow control for intransitive flow policies into our previously proposed assembly kit [Man00a]. To us it was very appealing that this did not require major changes to the assembly kit but only the definition of novel BSPs. The unwinding conditions we have presented are also similar to the ones for the transitive case [Man00b]. We are confident that the presented approach provides a suitable basis for applying information flow control to distributed systems. Our approach is the first proposal which can be used for such systems in the context of intransitive information flow. However, we neither claim that this is the only solution nor that it is the optimal one. In order to improve this solution further research will be useful which, in our opinion, should be driven by experiences from case studies. We plan to experiment with our approach in case studies in future work.

**Acknowledgments.** This work benefited from discussions with Dieter Hutter and Axel Schairer. The author would like to thank Serge Autexier and Alexandra Heidger for many valuable comments on the presentation.

## References

[FM99] Riccardo Focardi and Fabio Martinelli. A Uniform Approach to the Definition of Security Properties. In *FM'99 – Formal Methods (vol. 1)*, LNCS 1708, pages 794–813. Springer, 1999.

[GM82] J. A. Goguen and J. Meseguer. Security Policies and Security Models. In *Proceedings of the IEEE Symposium on Security and Privacy*, pages 11–20, Oakland, CA, April 26–28 1982.

[GM84] J. A. Goguen and J. Meseguer. Inference Control and Unwinding. In *Proceedings of the IEEE Symposium on Security and Privacy*, pages 75–86, Oakland, CA, April 29–May 2 1984.

[JT88] Dale M. Johnson and F. Javier Thayer. Security and the Composition of Machines. In *Proceedings of the Computer Security Foundations Workshop*, pages 72–89, Franconia, NH, June 1988.

[Man00a] Heiko Mantel. Possibilistic Definitions of Security – An Assembly Kit –. In *Proceedings of the IEEE Computer Security Foundations Workshop*, pages 185–199, Cambridge, UK, July 3–5 2000. IEEE Computer Society.

[Man00b] Heiko Mantel. Unwinding Possibilistic Security Properties. In *European Symposium on Research in Computer Security (ESORICS)*, pages 238–254, LNCS 1895, Toulouse, France, October 4-6 2000. Springer.

[McC87] Daryl McCullough. Specifications for Multi-Level Security and a Hook-Up Property. In *Proceedings of the IEEE Symposium on Security and Privacy*, pages 161–166, Oakland, CA, April 27–29 1987.

[McL96] John McLean. A General Theory of Composition for a Class of "Possibilistic" Security Properties. *IEEE Transaction on Software Engineering*, 22(1):53–67, January 1996.

[O'H90] Colin O'Halloran. A Calculus of Information Flow. In *Proceedings of the European Symposium on Research in Computer Security (ESORICS)*, pages 147–159, Toulouse, France, October 24–26 1990.

[Pin95] Sylvan Pinsky. Absorbing Covers and Intransitive Non-Interference. In *Proceedings of the IEEE Symposium on Security and Privacy*, pages 102–113, Oakland, CA, May 8–10 1995.

[RG99] A.W. Roscoe and M.H. Goldsmith. What is intransitive noninterference? In *Proceedings of the 12th IEEE Computer Security Foundations Workshop*, pages 228–238, Mordano, Italy, June 28–30 1999.

[RS99] P.Y.A. Ryan and S.A. Schneider. Process Algebra and Non-interference. In *Proceedings of the 12th IEEE Computer Security Foundations Workshop*, pages 214–227, Mordano, Italy, June 28–30 1999.

[Rus92] John Rushby. Noninterference, Transitivity, and Channel-Control Security Policies. Technical Report CSL-92-02, SRI International, 1992.

[Sch00] Steve Schneider. *Concurrent and real-time systems : the CSP approach*. John Wiley, Chichester, England ; New York, 2000.

[SRS+00] G. Schellhorn, W. Reif, A. Schairer, P. Karger, V. Austel, and D. Toll. Verification of a Formal Security Model for Multiapplicative Smart Cards. In *European Symposium on Research in Computer Security (ESORICS)*, pages 17–36, LNCS 1895, Toulouse, France, October 4-6 2000. Springer.

[Sut86] D. Sutherland. A Model of Information. In *9th National Computer Security Conference*, September 1986.

[vG90] R.J. van Glabbeek. The Linear Time – Branching Time Spectrum. In *Proceedings of CONCUR'90, Theories of Concurrency: Unification and Extensions*, pages 278–297, LNCS 458. Springer, 1990.

[WJ90] J. Todd Wittbold and Dale M. Johnson. Information Flow in Nondeterministic Systems. In *Proceedings of the IEEE Symposium on Research in Security and Privacy*, pages 144–161, Oakland, CA, May 1990.

[ZL97] Aris Zakinthinos and E.S. Lee. A General Theory of Security Properties. In *Proceedings of the IEEE Symposium on Security and Privacy*, pages 94–102, Oakland, CA, May 4–7 1997.