An Automata Based Approach for Verifying Information Flow Properties [⋆]

Deepak D'Souza [a], Raghavendra K. R. [a] and Barbara Sprick [a]

[a]*Department of Computer Science and Automation*
Indian Institute of Science, Bangalore, India.

Abstract

We present an automated verification technique to verify trace based information flow properties for finite state systems. We show that the Basic Security Predicates (BSP's) defined and shown by Mantel [1] to be the building blocks of known trace based information flow properties, can be characterised in terms of regularity preserving language theoretic operations. This leads to a decision procedure for checking whether a finite state system satisfies a given BSP. Verification techniques in the literature (e.g. unwinding) are based on the structure of the transition system and are incomplete in some cases. In contrast, our technique is language based and complete for all information flow properties that can be expressed in terms of BSP's.

Key words: Verification, security, information flow

1 Introduction

Granting, restricting and controlling the flow of information is a core part of computing system security. In particular, confidential data needs to be protected from undesired accesses. Access control policies are defined to serve this task by specifying which accesses are allowed for which users. In general we distinguish between two types of policies: discretionary and mandatory access control policies. In discretionary access control – implemented for example in the UNIX operating system – every user can define which access is allowed

⋆ A preliminary version of this paper appeared as [2].
Email addresses: `deepakd@csa.iisc.ernet.in` (Deepak D'Souza), `raghavendrakr@csa.iisc.ernet.in` (Raghavendra K. R.), `sprick@csa.iisc.ernet.in` (Barbara Sprick).

for which users for his data. Mandatory access control models are rule based and control the global flow of information. Objects as well as subjects are hierarchically ordered and then access is granted in such a way that information can flow only from higher level subjects to lower level subjects. However, access control methods can only restrict *direct* information flow (over open channels). Information leakage over covert channels (e.g. Trojan Horses, observable behaviour and time or space availability, etc) is not controllable by access control methods.

In [3], Goguen and Meseguer first introduced the notion of *Non-Interference* as a means to control both direct as well as indirect information flow. Informally, Goguen and Meseguer distinguish between high level and low level users and describe non-interference as: *What one group of users does using a certain ability has no effect on what some other group of users does* [3]. More generally in this paradigm, a system is modelled in terms of high-level (or confidential) events and low-level (or "visible") events. A low-level user in the system has a good idea of the way the system works (in other words, he knows the model of the system). However, his view of events in the system is restricted to the visible events only. The question of how "secure" the system is in terms of information about confidential events being leaked, can be phrased as: given a sequence of visible events representing the "visible" view of a trace of the system, how much can a low-level user infer about the occurrence of confidential events in that trace?

Since Goguen and Meseguer's initial work, many such information flow properties have been proposed in the literature, which restrict in varying degrees the kind of information a low-level user may infer about confidential events in the system. These security properties include, among others, *non-inference* [4–6] (which requires that each system trace projected to low-level events is itself a possible trace of the system), *separability* [5] (which requires that every possible low-level subtrace interleaved with every possible high-level subtrace must be a possible behaviour of a system), *generalized non-interference* [7] (which requires that for every possible trace and every possible "perturbation" via deletion and insertions of confidential events, there is a "correction" via deletion and insertion of non-confidential events to the perturbation such that the resulting trace is again a possible trace of the system), and several others including *nondeducability* [8], *restrictiveness* [7], the *perfect security property* [6].

To give a simple example which illustrates the use of the non-inference property as a security property, consider the e-banking example below which is adapted from [9]. A bank user Alice is required to periodically change her PIN and register it with the bank. Alice can either generate a new PIN and send it to the bank, or simply send her old PIN to the bank. The bank replies with an acknowledgment ("accept") only in case she sends them a new PIN.

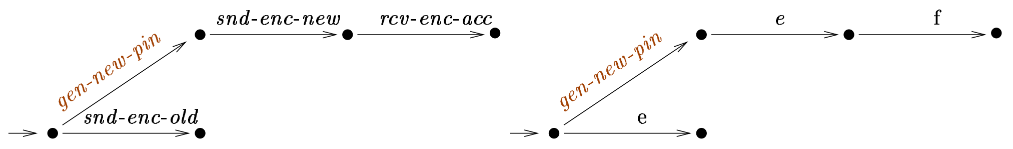

Fig. 1. System model T_0 and its abstraction T_0'

All communication is encrypted and sent over an open channel accessible to an adversary. The system is modelled as the transition system T_0 in Fig 1.

Both Alice and the bank are high-level users, while the adversary is a low-level user. The event *gen-new-pin* is the only confidential event in the system. However, though the events *snd-enc-new* (send encrypted PIN to the bank) and *rcv-enc-acc* (receive encrypted acknowledgment "accept" from the bank) are visible, the adversary can only make out the direction of the messages, and not their contents.

To model this, we further abstract the system as the transition system T_0' in Fig 1 where the visible event e represents the encrypted PIN sent by Alice to the bank, and the visible event f represents the acknowledgment sent by the bank to Alice.

It is evident that the system T_0' does not satisfy the non-inference property since the projection of the trace $gen\text{-}new\text{-}pin \cdot e \cdot f$ to visible events is $e \cdot f$ which is not a trace of the system. Thus, the adversary who sees the sequence of events $e \cdot f$ can infer that the event *gen-new-pin* must have occurred (and hence that Alice has changed her pin).

However, if the bank also replies with an acknowledgment ("reject") in the case when Alice sends her old PIN, then we get the system model T_1 which contains the *rcv-enc-rej* (receive encrypted acknowledgment "reject" from the bank) and its abstraction T_1' of Fig 2 which does indeed satisfy the non-inference property.

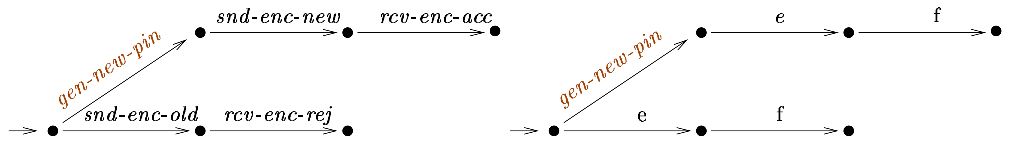

Fig. 2. System model T_1 and its abstraction T_1'

In [10,1] Mantel has presented an approach to uniformly formalize all known trace based information flow properties. Based on sets of traces as the system model, he defines a set of *basic security predicates (BSP's)*. Roughly, all BSP's require that for every trace and every "allowed" perturbation via deletion and insertion of confidential events, there is an "allowed" correction via deletion or insertion of non-confidential and non-visible events that results again in a trace of the system. The exact definition of "allowed" is given by each particular BSP. Mantel shows that all trace based security properties in the literature can be represented as conjunctions of these BSP's. For example generalized noninterference can be defined as the conjunction of the two BSP's *insertion (I)* and *deletion (D)*. A set of traces L satisfies the BSP I if for every perturbation of a trace that is obtained by inserting a confidential event after the last confidential event, there exists a correction of this perturbation such that the resulting trace is also in the language L. A set of traces L satisfies the BSP D if for every perturbation that is obtained by deleting the last confidential event, there exists a correction of this perturbation such that the resulting trace is also in the language L.

Our work is based on this modular framework of Mantel presented in [1]. We present an automated verification technique to check whether a finite state system satisfies a given basic security predicate. We define a set of language theoretic operations that represent the allowed perturbations and corrections for each BSP and show that the question of whether a set of traces L satisfies a BSP P boils down to checking whether a language L_1 is contained in a language L_2, where L_1 and L_2 are obtained from L by successive applications of the language-theoretic operations. Finally we show that the language-theoretic operations are regularity preserving. Thus if L is specified by a finite state transition system (and is hence regular), then L_1 and L_2 are also regular, and the question of whether $L_1 \subseteq L_2$ can be answered effectively.

As has been observed earlier in the literature, information flow properties are properties of *sets of traces* rather than properties of a *single* trace and hence cannot be handled by classical temporal logic based model checking approaches. Nevertheless our work shows it is possible to "model check" these properties by reducing them to the language inclusion problem for finite state systems.

Previous work dealing with the verification of such information flow properties (e.g. [11–14]) mainly employ "unwinding" theorems as the verification technique. These techniques are typically sufficient but not always necessary. We feel that this may be due to the fact that unwinding relations are based on the structure of the system rather than on the language of traces generated by the system. The only other work we are aware of which gives a decision procedure based on language inclusion is [15]. While they have addressed the properties of *non-deterministic noninterference* and *strong non-deterministic noninterference* (which is equivalent to the definition of noninference given in [5] and [4]), our approach gives a decision procedure for the whole class of information flow properties that can be expressed in terms of BSP's.

In the next section we introduce the language-theoretic operations we need to express the BSP's. In Section 3 we give equivalent definitions for the BSP's using these language-theoretic operations. Finally in Section 4 we show that the language-theoretic operations we have defined are regularity preserving.

2 Language-Theoretic Operations

By an alphabet we will mean a finite set of symbols representing *events* or *actions* of a system. For an alphabet Σ we use Σ^* to denote the set of finite strings over Σ. The null or empty string is represented by the symbol ϵ. For two strings α and β in Σ^* we write $\alpha\beta$ for the concatenation of α and β. A *language* L over Σ is just a subset of Σ^*.

A *marked* language M over an alphabet Σ is a language over the alphabet $\Sigma \cup \{\natural\}$, where '$\natural$' is a special "mark" symbol different from those in Σ, and each string in M contains exactly one occurrence of $\natural$.

For the rest of the paper we fix an alphabet of events Σ. We assume a partition of Σ into V, C, N, which in the framework of [1] correspond to events that are *visible*, *confidential*, and *neither* visible nor confidential, from a particular user's point of view.

Let L be a language over Σ and let M be a marked language over Σ. We define the following language-theoretic operations on L:

(1) Let $X \subseteq \Sigma$. The projection of a string $\tau \in \Sigma^*$ to X is written $\tau\restriction_X$ and is obtained from τ by deleting all events that are not elements of X. The projection of the language L to X, written $L\restriction_X$, is defined to be $\{\tau\restriction_X \mid \tau \in L\}$.

(2) *l-del*$(L) = \{\alpha\beta \mid c \in C,\ \beta\restriction_C = \epsilon, \text{ and } \alpha c\beta \in L\}$. Thus the operation *l-del* corresponds to the deletion of the last confidential event in a string. More precisely, *l-del*(L) contains all strings than can be obtained from a string in L by deleting the last C-event.

(3) *l-ins*$(L) = \{\alpha c\beta \mid c \in C,\ \beta\restriction_C = \epsilon, \text{ and } \alpha\beta \in L\}$. Thus *l-ins* corresponds to the insertion of a confidential event in strings of L, only at a position after which no confidential events occur.

(4) Let $X \subseteq \Sigma$. Then *l-ins-adm*$^X(L)$ is defined to be the set

$$\{\alpha c\beta \mid c \in C,\ \alpha\beta \in L,\ \beta\restriction_C = \epsilon, \text{ and } \exists\gamma c \in L : \gamma\restriction_X = \alpha\restriction_X\}.$$

Operation *l-ins-adm* (w.r.t. X) corresponds to insertion of "X-admissible" confidential events in strings of L. The insertion of a C-event is X-*admissible* after a prefix α in a string τ iff there exists another string $\gamma c \in L$ with γ projected to X being the same as α projected to X.

(5) *l-del-mark*$(L) = \{\alpha\natural\beta \mid c \in C,\ \alpha c\beta \in L, \text{ and } \beta\restriction_C = \epsilon\}$. The operation *l-del-mark* corresponds to the "marked" deletion of the last confidential event. More precisely, this operation replaces the last C-event in every string of L by the special mark symbol $\natural$.

(6) *l-ins-mark*$(L) = \{\alpha c\natural\beta \mid c \in C,\ \alpha\beta \in L,\ \beta\restriction_C = \epsilon\}$. The operation *l-ins-mark* corresponds to the "marked" insertion of a confidential event. It is similar to the operation *l-ins*, but additionally introduces a mark $\natural$ after the newly inserted C-event.

(7) Let $X \subseteq \Sigma$. Then we define *l-ins-adm-mark*$^X(L)$ to be the set

$$\{\alpha c\natural\beta \mid c \in C,\ \alpha\beta \in L,\ \beta\restriction_C = \epsilon, \text{ and } \exists\gamma c \in L : \gamma\restriction_X = \alpha\restriction_X\}.$$

Operation *l-ins-adm-mark* (w.r.t. X) corresponds to the marked insertion of X-admissible events. This operation is similar to *l-ins-adm*, but a mark $\natural$ is introduced after the newly inserted X-admissible event.

(8) *mark*$(L) = \{\alpha\natural\beta \mid \alpha\beta \in L\}$. Thus operation *mark* corresponds to the insertion of a mark at an arbitrary position in strings of L.

(9) Let $X \subseteq \Sigma$. Then the marked projection of the marked language M to X, written $M\restriction^m_X$, is defined as

$$M\restriction^m_X = \{\alpha\natural\beta' \mid \alpha\natural\beta \in M \text{ and } \beta' = \beta\restriction_X\}.$$

Thus marked projection operates on a marked language M and is similar to projection, but leaves every string intact upto the mark and projects to set X the suffix after the mark.

(10) Let $C' \subseteq C$ and $V' \subseteq V$. Then *l-del-con-mark*$_{C',V'}(L)$ is defined to be the set

$$\{\alpha v\natural\beta \mid c \in C',\ v \in V',\ \alpha cv\beta \in L \text{ and } \beta\restriction_C = \epsilon, \}.$$

Thus the operation *l-del-con-mark* (w.r.t. C' and V') corresponds to marked deletion in the "context" of an event in V'. This operation deletes the last confidential event c in a string and inserts a mark, provided c belongs to C' and and is immediately followed by a V'-event in the string. For convenience we place the mark *after* the V'-event.

(11) Let $C' \subseteq C$ and $V' \subseteq V$. Then *l-ins-con-mark*$_{C',V'}(L)$ is defined to be the set

$$\{\alpha cv\natural\beta \mid c \in C',\ v \in V',\ \alpha v\beta \in L, \text{ and } \beta\restriction_C = \epsilon\}.$$

Operation *l-ins-con-mark* (w.r.t. C' and V') thus corresponds to marked insertion of C'-events in the "context" of a V' event, in the sense above.

(12) Let $X \subseteq \Sigma$, $C' \subseteq C$ and $V' \subseteq V$. Then $\mathit{l\text{-}ins\text{-}adm\text{-}con\text{-}mark}^{X}_{C',V'}(L)$ is defined to be the set

$$\{\alpha c v \natural \beta \mid c \in C',\ v \in V',\ \alpha v \beta \in L,\ \beta \restriction_C = \epsilon, \text{ and } \exists \gamma c \in L,\ \gamma \restriction_X = \alpha \restriction_X\}.$$

The operation *l-ins-adm-con-mark* (w.r.t. X, C' and V') corresponds to the marked insertion of X-admissible C'-events in the context of a V' event. It is similar to *l-ins-con-mark* but allows only the insertion of X-admissible C'-events.

(13) Let $N' \subseteq N$ and $V' \subseteq V$. Then $\mathit{erase\text{-}con\text{-}mark}_{N',V'}(L)$ is defined to be the set

$$\{\alpha v \natural \beta \mid v \in V', \text{ and } \exists \delta \in (N')^* : \alpha \delta v \beta \in L\}.$$

The operation *erase-con-mark* (w.r.t. N' and V') corresponds to the marked erasure of N'-events in the context of V'-events. More precisely, $\mathit{erase\text{-}con\text{-}mark}_{N',V'}(L)$ contains all strings obtained from a string in L by the erasure of a consecutive sequence of N' events which end before a V' event v. The mark symbol is inserted *after* the event v in the string.

3 Expressing BSP's Language-Theoretically

We now express the basic security predicates (BSP's) of Mantel in terms of the language-theoretic operations defined in the previous section. We recall the definition of each predicate and show how it can be characterised using our language-theoretic operations.

For convenience we make use of the notation $\alpha =_Y \beta$ where $\alpha, \beta \in \Sigma^*$ and $Y \subseteq \Sigma$, to mean α and β are the same "modulo a correction on Y-events". More precisely, $\alpha =_Y \beta$ iff $\alpha \restriction_{\overline{Y}} = \beta \restriction_{\overline{Y}}$, where $\overline{Y}$ denotes $\Sigma - Y$. By extension, for languages L and M over Σ, we say $L \subseteq_Y M$ iff $L \restriction_{\overline{Y}} \subseteq M \restriction_{\overline{Y}}$.

In the definitions below, we assume L to be a language over Σ.

Definition 1 (R) *L satisfies the BSP R (Removal of events) iff for all $\tau \in L$ there exists $\tau' \in L$ such that $\tau' \restriction_C = \epsilon$ and $\tau' \restriction_V = \tau \restriction_V$.*

Lemma 2 *L satisfies R iff $L \restriction_V \subseteq_N L$.*

PROOF. ($\Rightarrow$:) Suppose L satisfies R. Let $\tau \in L \restriction_V$. Then there exists some τ' in L such that $\tau' \restriction_V = \tau$. Since L satisfies R and $\tau' \in L$, there exists τ'' in L

such that $\tau'' \lceil_C = \epsilon$ and $\tau'' \lceil_V = \tau' \lceil_V = \tau \lceil_V$. Since both τ'' and τ don't contain any C-events, they differ from each other only on N-events (i.e. $\tau =_N \tau''$). Thus τ belongs to L modulo a correction on N-events. Hence $L \lceil_V \subseteq_N L$.

($\Leftarrow$:) Suppose $L \lceil_V \subseteq_N L$. Let $\tau \in L$. Since $L \lceil_V \subseteq_N L$, there exists a string τ' in L, such that $\tau' =_N \tau \lceil_V$. Since $\tau \lceil_V$ has no C-events, $\tau' \lceil_C = \epsilon$ and $\tau' \lceil_V = (\tau \lceil_V) \lceil_V = \tau \lceil_V$. Hence R is satisfied. $\square$

Definition 3 (D) *L satisfies D (stepwise Deletion of events) iff for all $\alpha c \beta \in L$, such that $c \in C$ and $\beta \lceil_C = \epsilon$, we have $\alpha' \beta' \in L$ with $\alpha' =_N \alpha$ and $\beta' =_N \beta$.*

Lemma 4 *L satisfies D iff l-del$(L) \subseteq_N L$.*

PROOF. ($\Rightarrow$:) Assume L satisfies D and consider a string τ in *l-del*(L). Then τ must be of the form $\alpha\beta$ and there must exist $\alpha c \beta \in L$ such that $c \in C$, and $\beta \lceil_C = \epsilon$. Since L satisfies D, we must have $\alpha'\beta' \in L$ such that $\alpha' =_N \alpha$ and $\beta' =_N \beta$. But this just means that $\tau = \alpha\beta =_N \alpha'\beta'$. Hence *l-del*$(L) \subseteq_N L$.

($\Leftarrow$:) Suppose *l-del*$(L) \subseteq_N L$. Consider a string $\alpha c \beta$ in L with $c \in C$ and $\beta \lceil_C = \epsilon$. By the definition of *l-del*(L), we have $\alpha\beta \in$ *l-del*(L). Since *l-del*$(L) \subseteq_N L$, there exists $\tau \in L$ such that $\alpha\beta =_N \tau$. Thus τ' can be expressed as $\alpha'\beta'$ with $\alpha =_N \alpha'$ and $\beta =_N \beta'$. Hence D is satisfied. $\square$

Definition 5 (I) *L satisfies I (Insertion of events) iff for all $\alpha\beta \in L$ such that $\beta \lceil_C = \epsilon$, and for every $c \in C$, we have $\alpha' c \beta' \in L$, with $\beta' =_N \beta$ and $\alpha' =_N \alpha$.*

Lemma 6 *L satisfies I iff l-ins$(L) \subseteq_N L$.*

PROOF. ($\Rightarrow$:) Assume L satisfies I and consider a string τ in *l-ins*(L). Then τ will be of the form $\alpha c \beta$ for some $c \in C$ and there must exist $\alpha\beta \in L$ such that $\beta \lceil_C = \epsilon$. Since L satisfies I, there exists $\alpha' c \beta' \in L$ such that $\alpha' =_N \alpha$ and $\beta' =_N \beta$. But this just means that $\tau = \alpha c \beta =_N \alpha' c \beta'$. Hence *l-ins*$(L) \subseteq_N L$.

($\Leftarrow$:) Suppose *l-ins*$(L) \subseteq_N L$. Consider a string $\alpha\beta \in L$ with $\beta \lceil_C = \epsilon$. By the definition of *l-ins*(L), for any $c \in C$, we have $\alpha c \beta \in$ *l-ins*(L). Since *l-ins*$(L) \subseteq_N L$, there exists $\tau \in L$ such that $\alpha c \beta =_N \tau$. Thus τ can be expressed as $\alpha' c \beta'$ where $\alpha' =_N \alpha$ and $\beta =_N \beta$. Hence I is satisfied. $\square$

Definition 7 (IA) *Let $X \subseteq \Sigma$. Then L satisfies IA (Insertion of Admissible events) w.r.t X iff for all $\alpha\beta \in L$ such that $\beta \lceil_C = \epsilon$ and for some $c \in C$, there exists $\gamma c \in L$ with $\gamma \lceil_X = \alpha \lceil_X$, we have $\alpha' c \beta' \in L$ with $\beta' =_N \beta$ and $\alpha' =_N \alpha$.*

Lemma 8 *L satisfies IA iff l-ins-adm$^X(L) \subseteq_N L$.*

PROOF. ($\Rightarrow$:) Assume L satisfies *IA* and consider a string τ in $\mathit{l\text{-}ins\text{-}adm}^X(L)$. Then τ will be of the form $\alpha c\beta$ for some $c \in C$ and there must exist $\alpha\beta \in L$ such that $\beta \upharpoonright_C = \epsilon$ and also $\gamma c \in L$ with $\gamma \upharpoonright_X = \alpha \upharpoonright_X$. Since L satisfies *IA*, there exists $\alpha' c\beta' \in L$ with $\alpha' =_N \alpha$ and $\beta' =_N \beta$. But this just means that $\tau = \alpha c\beta =_N \alpha' c\beta'$. Hence $\mathit{l\text{-}ins\text{-}adm}^X(L) \subseteq_N L$.

($\Leftarrow$:) Suppose $\Leftrightarrow$ $\mathit{l\text{-}ins\text{-}adm}^X(L) \subseteq_N L$. Consider a string $\alpha\beta \in L$ with $\beta \upharpoonright_C = \epsilon$ and there exists $\gamma c \in L$ for some $c \in C$ with $\gamma \upharpoonright_X = \alpha \upharpoonright_X$. By the definition of $\mathit{l\text{-}ins\text{-}adm}^X(L)$, we have $\alpha c\beta \in \mathit{l\text{-}ins\text{-}adm}^X(L)$. Since $\mathit{l\text{-}ins\text{-}adm}^X(L) \subseteq_N L$, there exists $\tau \in L$ such that $\alpha c\beta =_N \tau$. Thus τ can be expressed as $\alpha' c\beta'$ such that $\alpha' =_N \alpha$ and $\beta' =_N \beta$. Hence *IA* is satisfied. □

Definition 9 (BSD) *L satisfies BSD (Backwards Strict Deletion) iff for all $\alpha c\beta \in L$ such that $c \in C$ and $\beta \upharpoonright_C = \epsilon$, we have $\alpha\beta' \in L$ with $\beta' =_N \beta$.*

Lemma 10 *L satisfies BSD iff $\mathit{l\text{-}del\text{-}mark}(L) \upharpoonright^m_N \subseteq \mathit{mark}(L) \upharpoonright^m_N$.*

PROOF. ($\Rightarrow$:) Assume L satisfies *BSD* and consider a string τ belonging to $\mathit{l\text{-}del\text{-}mark}(L) \upharpoonright^m_N$. Then τ must be of the form $\alpha\natural\beta'$, which comes from a string of the form $\alpha\natural\beta$ in $\mathit{l\text{-}del\text{-}mark}(L)$ with $\beta \upharpoonright_{\overline{N}} = \beta'$, which in turn must be such that $\alpha c\beta \in L$ for some $c \in C$ with $\beta \upharpoonright_C = \epsilon$. Since L satisfies *BSD*, we have $\alpha\beta'' \in L$ such that $\beta =_N \beta''$. By the definition of $\mathit{mark}(L)$, we have $\alpha\natural\beta'' \in \mathit{mark}(L)$. Deleting N-events from β'' results in β'. Thus $\alpha\natural\beta' = \tau$ must be in $\mathit{mark}(L) \upharpoonright^m_N$. Hence $\mathit{l\text{-}del\text{-}mark}(L) \upharpoonright^m_N$ is a subset of $\mathit{mark}(L) \upharpoonright^m_N$.

($\Leftarrow$:) Suppose $\mathit{l\text{-}del\text{-}mark}(L) \upharpoonright^m_N \subseteq \mathit{mark}(L) \upharpoonright^m_N$. Consider a string $\alpha c\beta \in L$, with $c \in C$ and $\beta \upharpoonright_C = \epsilon$. By the definition of $\mathit{l\text{-}del\text{-}mark}(L)$, we have $\alpha\natural\beta \in \mathit{l\text{-}del\text{-}mark}(L)$ and hence $\alpha\natural\beta' \in \mathit{l\text{-}del\text{-}mark}(L) \upharpoonright^m_N$ where β' is obtained from β by deleting N-events. Since $\mathit{l\text{-}del\text{-}mark}(L) \upharpoonright^m_N \subseteq \mathit{mark}(L) \upharpoonright^m_N$, we have $\alpha\natural\beta' \in \mathit{mark}(L) \upharpoonright^m_N$. Then there must exist $\alpha\natural\beta'' \in \mathit{mark}(L)$ where $\beta'' \upharpoonright_{\overline{N}} = \beta'$. Thus $\beta'' =_N \beta$. By the definition of $\mathit{mark}(L)$, we have $\alpha\beta'' \in L$. Hence *BSD* is satisfied. □

Definition 11 (BSI) *L satisfies BSI (Backwards Strict Insertion) iff for all $\alpha\beta \in L$ such that $\beta \upharpoonright_C = \epsilon$, and for every $c \in C$, we have $\alpha c\beta' \in L$, with $\beta' =_N \beta$.*

Lemma 12 *L satisfies BSI iff $\mathit{l\text{-}ins\text{-}mark}(L) \upharpoonright^m_N \subseteq \mathit{mark}(L) \upharpoonright^m_N$.*

PROOF. ($\Rightarrow$:) Assume L satisfies *BSI* and consider a string τ belonging to $\mathit{l\text{-}ins\text{-}mark}(L) \upharpoonright^m_N$. Then τ will be of the form $\alpha c\natural\beta'$ which comes from a string of the form $\alpha c\natural\beta$ in $\mathit{l\text{-}ins\text{-}mark}(L)$ which in turn must be such that $\beta \upharpoonright_{\overline{N}} = \beta'$, $c \in C$, $\alpha\beta \in L$ and $\beta \upharpoonright_C = \epsilon$. Since L satisfies *BSI*, we have $\alpha c\beta'' \in L$ such that $\beta =_N \beta''$. By the definition of $\mathit{mark}(L)$, we have $\alpha c\natural\beta'' \in \mathit{mark}(L)$. Deleting

N-events from β'' results in β'. Thus $\alpha c\natural\beta' = \tau$ must be in $mark(L)\upharpoonright^{m}_{\overline{N}}$. Hence *l-ins-mark*$(L)\upharpoonright^{m}_{\overline{N}}$ is a subset of $mark(L)\upharpoonright^{m}_{\overline{N}}$.

($\Leftarrow$:) Suppose *l-ins-mark*$(L)\upharpoonright^{m}_{\overline{N}}\subseteq mark(L)\upharpoonright^{m}_{\overline{N}}$. Consider a string $\alpha\beta \in L$ such that $\beta\upharpoonright_C = \epsilon$. By the definition of *l-ins-mark*(L), we have $\alpha c\natural\beta \in$ *l-ins-mark*(L) for any $c \in C$ and hence $\alpha c\natural\beta' \in$ *l-ins-mark*$(L)\upharpoonright^{m}_{\overline{N}}$ where β' is obtained from β by deleting N-events. Since *l-ins-mark*$(L)\upharpoonright^{m}_{\overline{N}} \subseteq mark(L)\upharpoonright^{m}_{\overline{N}}$, we have $\alpha c\natural\beta' \in mark(L)\upharpoonright^{m}_{\overline{N}}$. Then there must exist $\alpha c\natural\beta'' \in mark(L)$ where $\beta''\upharpoonright_{\overline{N}} = \beta'$. Thus $\beta'' =_N \beta$. By the definition of $mark(L)$, we have $\alpha c\beta'' \in L$. Hence *BSI* is satisfied. $\Box$

Definition 13 (BSIA) *Let $X \subseteq \Sigma$. Then L satisfies BSIA (Backwards Strict Insertion of Admissible events) w.r.t X iff for all $\alpha\beta \in L$ such that $\beta\upharpoonright_C = \epsilon$ and there exists $\gamma c \in L$ with $c \in C$ and $\gamma\upharpoonright_X = \alpha\upharpoonright_X$, we have $\alpha c\beta' \in L$ with $\beta' =_N \beta$.*

Lemma 14 *L satisfies BSIA iff l-ins-adm-mark$^X(L)\upharpoonright^{m}_{\overline{N}}\subseteq mark(L)\upharpoonright^{m}_{\overline{N}}$.*

PROOF. ($\Rightarrow$:) Assume L satisfies *BSIA* and consider a string τ belonging to *l-ins-adm-mark*$^X(L)\upharpoonright^{m}_{\overline{N}}$. Then τ will be of the form $\alpha c\natural\beta'$, which comes from a string of the form $\alpha c\natural\beta$ in *l-ins-adm-mark*$^X(L)$ with $\beta\upharpoonright_{\overline{N}} = \beta'$, which in turn must be such that $\beta\upharpoonright_C = \epsilon$, $c \in C$, $\alpha\beta \in L$, $\beta\upharpoonright_C = \epsilon$ and there exists $\gamma c \in L$ with $\gamma\upharpoonright_X = \alpha\upharpoonright_X$. Since L satisfies *BSIA*, we have $\alpha c\beta'' \in L$ where $\beta =_N \beta''$. By the definition of $mark(L)$, we have $\alpha c\natural\beta'' \in mark(L)$. Deleting N-events from β'' results in β'. Thus $\alpha c\natural\beta' = \tau$ must be in $mark(L)\upharpoonright^{m}_{\overline{N}}$. Hence *l-ins-adm-mark*$^X(L)\upharpoonright^{m}_{\overline{N}}$ is a subset of $mark(L)\upharpoonright^{m}_{\overline{N}}$.

($\Leftarrow$:) Suppose *l-ins-adm-mark*$^X(L)\upharpoonright^{m}_{\overline{N}}$ is a subset of $mark(L)\upharpoonright^{m}_{\overline{N}}$. Consider a string $\alpha\beta \in L$ such that $\beta\upharpoonright_C = \epsilon$ and there exists $\gamma c \in L$ with $c \in C$ and $\gamma\upharpoonright_X = \alpha\upharpoonright_X$. By the definition of *l-ins-adm-mark*$^X(L)$, we have $\alpha c\natural\beta \in$ *l-ins-adm-mark*$^X(L)$ and hence $\alpha c\natural\beta' \in$ *l-ins-adm-mark*$^X(L)\upharpoonright^{m}_{\overline{N}}$ where β' is obtained from β by deleting N-events. Since *l-ins-adm-mark*$^X(L)\upharpoonright^{m}_{\overline{N}}$ is a subset of $mark(L)\upharpoonright^{m}_{\overline{N}}$, we have $\alpha c\natural\beta' \in mark(L)\upharpoonright^{m}_{\overline{N}}$. Then there must exist $\alpha c\natural\beta'' \in mark(L)$ where $\beta''\upharpoonright_{\overline{N}} = \beta'$. Thus $\beta'' =_N \beta$. By the definition of $mark(L)$, there exists $\alpha c\beta'' \in L$. Hence *BSIA* is satisfied. $\Box$

Definition 15 (FCD) *Let $X \subseteq \Sigma$, $V' \subseteq V$, $C' \subseteq C$, and $N' \subseteq N$. Then L satisfies FCD (Forward Correctable Deletion) w.r.t V', C', N' iff for all $\alpha cv\beta \in L$ such that $c \in C'$, $v \in V'$ and $\beta\upharpoonright_C = \epsilon$, we have $\alpha\delta v\beta' \in L$ where $\delta \in (N')^*$ and $\beta' =_N \beta$.*

Lemma 16 *L satisfies FCD iff*

$$\textit{l-del-con-mark}_{C',V'}(L)\upharpoonright^{m}_{\overline{N}} \subseteq \textit{erase-con-mark}_{N',V'}(L)\upharpoonright^{m}_{\overline{N}}$$

PROOF. ($\Rightarrow$:) Assume L satisfies *FCD* and consider a string τ belonging to $\textit{l-del-con-mark}_{C',V'}(L) \restriction^m_N$. Then τ will be of the form $\alpha v \natural \beta'$ which comes from a string of the form $\alpha v \natural \beta$ in $\textit{l-del-con-mark}_{C',V'}(L)$ with $\beta \restriction_{\overline{N}} = \beta'$, which in turn must be such that $\beta \restriction_C = \epsilon$, $c \in C'$, $v \in V'$, and $\alpha c v \beta \in L$. Since L satisfies *FCD*, we have $\alpha \delta v \beta'' \in L$ where $\beta'' =_N \beta$. By the definition of $\textit{erase-con-mark}_{N',V'}(L)$, we have $\alpha v \natural \beta'' \in \textit{erase-con-mark}_{N',V'}(L)$. Deleting N-symbols from β'' results in β'. Thus $\alpha v \natural \beta'$ must be in $\textit{erase-con-mark}_{N',V'}(L) \restriction^m_N$. Hence $\textit{l-del-con-mark}_{C',V'}(L) \restriction^m_N$ is a subset of $\textit{erase-con-mark}_{N',V'}(L) \restriction^m_N$.

($\Leftarrow$:) Suppose $\textit{l-del-con-mark}_{C',V'}(L) \restriction^m_N \subseteq \textit{erase-con-mark}_{N',V'}(L) \restriction^m_N$. Consider a string $\alpha c v \beta \in L$, with $c \in C'$, $v \in V'$ and $\beta \restriction_C = \epsilon$. By the definition of $\textit{l-del-con-mark}_{C',N'}(L)$, we have $\alpha v \natural \beta \in \textit{l-del-con-mark}_{C',N'}(L)$ and hence $\alpha v \natural \beta' \in \textit{l-del-con-mark}_{C',N'}(L) \restriction^m_N$ where β' is obtained from β by deleting N-events. Since $\textit{l-del-con-mark}_{C',V'}(L) \restriction^m_N \subseteq \textit{erase-con-mark}_{N',V'}(L) \restriction^m_N$, we have $\alpha v \natural \beta' \in \textit{erase-con-mark}_{N',V'}(L) \restriction^m_N$. Then there must exist $\alpha v \natural \beta'' \in \textit{erase-con-mark}_{N',V'}(L)$ where $\beta'' \restriction_{\overline{N}} = \beta'$. Thus $\beta'' =_N \beta$. By the definition of $\textit{erase-con-mark}_{N',V'}(L)$, we have $\alpha \delta v \beta'' \in L$ for some δ such that $\delta \in (N')^*$. Hence *FCD* is satisfied. $\square$

Definition 17 (FCI) *Let $X \subseteq \Sigma$, $V' \subseteq V$, $C' \subseteq C$, and $N' \subseteq N$. Then L satisfies FCI (Forward Correctable Insertion) w.r.t C', V', N' iff for all $\alpha v \beta \in L$ such that $v \in V'$, $\beta \restriction_C = \epsilon$, and for every $c \in C'$ we have $\alpha c \delta v \beta' \in L$, with $\delta \in (N')^*$ and $\beta' =_N \beta$.*

Lemma 18 *L satisfies FCI iff*

$$\textit{l-ins-con-mark}_{C',V'}(L) \restriction^m_N \subseteq \textit{erase-con-mark}_{N',V'}(L) \restriction^m_N$$

PROOF. ($\Rightarrow$:) Assume L satisfies *FCI* and consider a string τ belonging to $\textit{l-ins-con-mark}_{C',V'}(L) \restriction^m_N$. Then τ will be of the form $\alpha c v \natural \beta'$ which comes from some string of the form $\alpha c v \natural \beta$ in $\textit{l-ins-con-mark}_{C',V'}(L)$ which in turn must be such that $c \in C'$, $v \in V'$, $\beta' = \beta \restriction_{\overline{N}}$, $\alpha v \beta \in L$ and $\beta \restriction_C = \epsilon$. Since L satisfies *FCI*, we have $\alpha c \delta v \beta'' \in L$ for some δ such that $\delta \in (N')^*$ and $\beta'' =_N \beta$. By the definition of $\textit{erase-con-mark}_{N',V'}(L)$, we have $\alpha c v \natural \beta'' \in \textit{erase-con-mark}_{N',V'}(L)$. Deleting N-symbols from β'' results in β'. Thus $\alpha c v \natural \beta'$ $= \tau$ must be in $\textit{erase-con-mark}_{N',V'}(L) \restriction^m_N$. Hence $\textit{l-ins-con-mark}_{C',V'}(L) \restriction^m_N$ is a subset of $\textit{erase-con-mark}_{N',V'}(L) \restriction^m_N$.

($\Leftarrow$:) Suppose $\textit{l-ins-con-mark}_{C',V'}(L) \restriction^m_N \subseteq \textit{erase-con-mark}_{N',V'}(L) \restriction^m_N$. Consider a string $\alpha v \beta \in L$ with $v \in V'$ and $\beta \restriction_C = \epsilon$. By the definition of $\textit{l-ins-con-mark}_{C',V'}(L)$, we have $\alpha c v \natural \beta \in \textit{l-ins-con-mark}_{C',V'}(L)$ for any $c \in C'$ and hence $\alpha c v \natural \beta' \in \textit{l-ins-con-mark}_{C',V'}(L) \restriction^m_N$ where β' is obtained from β by deleting N-events. Since $\textit{l-ins-con-mark}_{C',V'}(L) \restriction^m_N \subseteq \textit{erase-con-mark}_{N',V'}(L) \restriction^m_N$, we have $\alpha c v \natural \beta' \in \textit{erase-con-mark}_{N',V'}(L) \restriction^m_N$. Then there must exist $\alpha c v \natural \beta'' \in$

$\textit{erase-con-mark}_{N',V'}(L)$ where $\beta'' \upharpoonright_{\overline{N}} = \beta'$. Thus $\beta'' =_N \beta$. By the definition of $\textit{erase-con-mark}_{N',V'}(L)$, we have $\alpha c \delta v \beta'' \in L$. Hence *FCI* is satisfied. □

Definition 19 (FCIA) *Let $X \subseteq \Sigma$, $V' \subseteq V$, $C' \subseteq C$, and $N' \subseteq N$. Then L satisfies FCIA (Forward Correctable Insertion of admissible events) w.r.t. X, V', C', N' iff for all $\alpha v \beta \in L$ such that: $v \in V'$, $\beta \upharpoonright_C = \epsilon$, and there exists $\gamma c \in L$, with $c \in C'$ and $\gamma \upharpoonright_X = \alpha \upharpoonright_X$; we have $\alpha c \delta v \beta' \in L$ with $\delta \in (N')^*$ and $\beta' =_N \beta$.*

Lemma 20 *L satisfies FCIA (w.r.t. X, V', C', N') iff*

$$\textit{l-ins-adm-con-mark}^X_{C',V'}(L) \upharpoonright^m_{\overline{N}} \subseteq \textit{erase-con-mark}_{N',V'}(L) \upharpoonright^m_{\overline{N}} .$$

PROOF. (⇒:) Suppose L satisfies *FCIA* w.r.t. X, V', C', N'. Consider a string $\tau \in \textit{l-ins-adm-con-mark}^X_{C',V'}(L) \upharpoonright^m_{\overline{N}}$. τ must be of the form $\alpha c v \natural \beta'$, which comes from some string of the form $\alpha c v \natural \beta$ in $\textit{l-ins-adm-con-mark}^X_{C',V'}(L)$ which in turn must be such that $\beta' = \beta \upharpoonright_{\overline{N}}$, $c \in C'$, $v \in V'$, $\alpha v \beta \in L$, $\beta \upharpoonright_C = \epsilon$, and there exists $\gamma c \in L$ with $\gamma \upharpoonright_X = \alpha \upharpoonright_X$. Now since $\alpha v \beta \in L$ and L satisfies *FCIA*, there must exist $\alpha c \delta v \beta'' \in L$ with $\delta \in (N')^*$ and $\beta'' =_N \beta$. By the definition of $\textit{erase-con-mark}_{N',V'}(L)$, we have $\alpha c v \natural \beta'' \in \textit{erase-con-mark}_{N',V'}(L)$. Deleting N-events from β'' results in β'. Thus $\alpha c v \beta' = \tau$ must be in $\textit{erase-con-mark}_{N',V'}(L) \upharpoonright^m_{\overline{N}}$. Hence $\textit{l-ins-adm-con-mark}^X_{C',V'}(L) \upharpoonright^m_{\overline{N}}$ is a subset of $\textit{erase-con-mark}_{N',V'}(L) \upharpoonright^m_{\overline{N}}$.

(⇐:) Suppose $\textit{l-ins-adm-con-mark}^X_{C',V'}(L) \upharpoonright^m_{\overline{N}} \subseteq \textit{erase-con-mark}_{N',V'}(L) \upharpoonright^m_{\overline{N}}$. Consider a string of the form $\alpha v \beta \in L$, such that $v \in V'$, $\beta \upharpoonright_C = \epsilon$, and there exists $\gamma c \in L$ with $c \in C'$ and $\gamma \upharpoonright_X = \alpha \upharpoonright_X$. By the definition of $\textit{l-ins-adm-con-mark}^X_{C',V'}(L)$, $\alpha c v \natural \beta \in \textit{l-ins-adm-con-mark}^X_{C',V'}(L)$ and hence $\alpha c v \natural \beta' \in \textit{l-ins-adm-con-mark}^X_{C',V'}(L) \upharpoonright^m_{\overline{N}}$ where β' is obtained from β by deleting N-events. Since $\textit{l-ins-adm-con-mark}^X_{C',V'}(L) \upharpoonright^m_{\overline{N}} \subseteq \textit{erase-con-mark}_{N',V'}(L) \upharpoonright^m_{\overline{N}}$, we have $\alpha c v \natural \beta' \in \textit{erase-con-mark}_{N',V'}(L) \upharpoonright^m_{\overline{N}}$. Hence $\alpha c v \natural \beta''$ must belong to $\textit{erase-con-mark}_{N',V'}(L)$ for some β'' such that $\beta'' \upharpoonright_{\overline{N}} = \beta'$. Thus $\beta'' =_N \beta$. By the definition of $\textit{erase-con-mark}_{N',V'}(L)$, there exists $\delta \in (N')^*$ such that $\alpha c \delta v \beta'' \in L$. Hence L satisfies *FCIA*. □

Definition 21 (SR) *L satisfies SR (Strict Removal) iff for all $\tau \in L$ we have $\tau \upharpoonright_{\overline{C}} \in L$.*

Lemma 22 *L satisfies SR iff $L \upharpoonright_{\overline{C}} \subseteq L$.*

PROOF. (⇒:) Assume L satisfies *SR* and consider any string τ in $L \upharpoonright_{\overline{C}}$. There must exist some $\tau' \in L$ such that $\tau' \upharpoonright_{\overline{C}} = \tau$. Since L satisfies *SR*, we have $\tau = \tau' \upharpoonright_{\overline{C}} \in L$. Hence $L \upharpoonright_{\overline{C}} \subseteq L$.

($\Leftarrow$:) Suppose $L\restriction_{\overline{C}} \subseteq L$. Consider any string τ in L. Then $\tau\restriction_{\overline{C}} \in L\restriction_{\overline{C}}$. Since $L\restriction_{\overline{C}} \subseteq L$, we have $\tau\restriction_{\overline{C}} \in L$. Hence *SR* is satisfied. □

Definition 23 (SD) *L satisfies SD (Strict Deletion) iff for all $\alpha c\beta \in L$ such that $c \in C$ and $\beta\restriction_C = \epsilon$, we have $\alpha\beta \in L$.*

Lemma 24 *L satisfies SD iff l-del$(L) \subseteq L$.*

PROOF. ($\Rightarrow$:) Assume L satisfies *SD* and consider a string τ in *l-del*(L). Then τ will be of the form $\alpha\beta$ which comes from a string of the form $\alpha c\beta \in L$ such that $c \in C$ and $\beta\restriction_C = \epsilon$. Since L satisfies *SD*, we have $\alpha\beta \in L$. Hence *l-del*$(L) \subseteq L$.

($\Leftarrow$:) Suppose *l-del*$(L) \subseteq L$. Consider a string $\alpha c\beta \in L$ with $c \in C$ and $\beta\restriction_C = \epsilon$. By the definition of *l-del*(L), we have $\alpha\beta \in$ *l-del*(L). Since *l-del*$(L) \subseteq L$, we have $\alpha\beta \in L$. Hence *SD* is satisfied. □

Definition 25 (SI) *L satisfies SI (Strict Insertion) iff for all $\alpha\beta \in L$ such that $\beta\restriction_C = \epsilon$, and for every $c \in C$, we have $\alpha c\beta \in L$.*

Lemma 26 *L satisfies SI iff l-ins$(L) \subseteq L$.*

PROOF. ($\Rightarrow$:) Assume L satisfies *SI* and consider a string $\tau \in$ *l-ins*(L). Then τ will be of the form $\alpha c\beta$ which comes from a string of the form $\alpha\beta \in L$ such that $c \in C$ and $\beta\restriction_C = \epsilon$. Since L satisfies *SI*, we have $\alpha c\beta \in L$. Hence *l-ins*$(L) \subseteq L$.

($\Leftarrow$:) Suppose *l-ins*$(L) \subseteq L$. Consider a string $\alpha\beta \in L$ such that $\beta\restriction_C = \epsilon$. By the definition of *l-ins*(L), we have $\alpha c\beta \in$ *l-ins*(L) for any $c \in C$. Since *l-ins*$(L) \subseteq L$, we have $\alpha c\beta \in L$. Hence *SI* is satisfied. □

Definition 27 (SIA) *Let $X \subseteq \Sigma$. L satisfies SIA (Strict Insertion of Admissible events) w.r.t X iff for all $\alpha\beta \in L$ such that $\beta\restriction_C = \epsilon$ and there exists $\gamma c \in L$ with $c \in C$ and $\gamma\restriction_X = \alpha\restriction_X$, we have $\alpha c\beta \in L$.*

Lemma 28 *L satisfies SIA iff l-ins-adm$^X(L) \subseteq L$.*

PROOF. ($\Rightarrow$:) Assume L satisfies *SIA* and consider a string $\tau \in$ *l-ins-adm*$^X(L)$. Then τ will be of the form $\alpha c\beta$ which comes from the string of the form $\alpha\beta \in L$ such that $c \in C$, $\beta\restriction_C = \epsilon$ and there exists $\gamma c \in L$ with $c \in C$ and $\gamma\restriction_X = \alpha\restriction_X$. Since L satisfies *SIA*, we have $\tau = \alpha c\beta \in L$. Hence *l-ins-adm*$^X(L) \subseteq L$.

($\Leftarrow$:) Suppose $\textit{l-ins-adm}^X(L) \subseteq L$. Consider a string $\alpha\beta \in L$ such that $\beta \restriction_C = \epsilon$ and there exists $\gamma c \in L$ with $c \in C$ and $\gamma \restriction_X = \alpha \restriction_X$. By the definition of $\textit{l-ins-adm}^X(L)$, $\alpha c \beta \in \textit{l-ins-adm}^X(L)$. Since $\textit{l-ins-adm}^X(L) \subseteq L$, $\alpha c\beta \in L$. Hence *SIA* is satisfied. $\square$

4 Operations are Regularity Preserving

We now show how the language-theoretic characterisations of BSP's lead to a decision procedure for checking whether a finite-state system satisfies a given BSP. We first introduce the necessary terminology, beginning with the required notions in finite state automata.

A *(finite-state) transition system* over an alphabet Δ is a structure of the form $\mathcal{T} = (Q, s, \longrightarrow)$, where Q is a finite set of states, $s \in Q$ is the start state, and $\longrightarrow \subseteq Q \times \Delta \times Q$ is the transition relation. We write $p \xrightarrow{a} q$ to stand for $(p, a, q) \in \longrightarrow$, and use $p \xrightarrow{\alpha}{}^* q$ to denote the fact that we have a path labelled α from p to q in the underlying graph of the transition system $\mathcal{T}$. The language *accepted* (or *generated*) by the transition system $\mathcal{T}$ is defined to be $L(\mathcal{T}) = \{\alpha \in \Delta^* \mid s \xrightarrow{\alpha}{}^* q \text{ for some } q \in Q\}$.

A *(finite state) automaton* (FSA) over an alphabet Δ is of the form $\mathcal{A} = (Q, s, \longrightarrow, F)$ where $(Q, s, \longrightarrow)$ forms a transition system and $F \subseteq Q$ is a set of final states. The language accepted by $\mathcal{A}$ is defined to be $L(\mathcal{A}) = \{\alpha \in \Delta^* \mid s \xrightarrow{\alpha}{}^* q \text{ for some } q \in F\}$. A transition system can thus be thought of as an automaton in which all states are final.

It will be convenient to make use of automata with *ϵ-transitions.* Thus the automaton is also allowed transitions of the form $p \xrightarrow{\epsilon} q$. The language accepted by automata with ϵ-transitions is defined similarly, except that the ϵ labels don't contribute to the label of a path. As is well-known ϵ-transitions don't add to the expressive power of automata.

The class of languages accepted by FSA's is termed the class of *regular* languages. Regular languages are effectively closed under intersection and complementation. Moreover their language emptiness problem – i.e. given an FSA $\mathcal{A}$, is $L(\mathcal{A}) = \emptyset$? – is efficiently decidable (by simply checking if there is a final state reachable from the initial state). It follows that the language inclusion problem (whether $L(\mathcal{A}) \subseteq L(\mathcal{B})$?) is also decidable for automata, since we can check equivalently that $L(\mathcal{A}) \cap (\Delta^* - L(\mathcal{B})) = \emptyset$.

Returning now to our problem of verifying BSP's, we say that a system modelled as a finite-state transition system $\mathcal{T}$ satisfies a given BSP P iff $L(\mathcal{T})$ satisfies P. In the previous section we showed that the question of whether a language L satisfies P boils down to checking whether $L_1 \subseteq L_2$, where L_1 and L_2 are obtained from L by successive applications of some language-theoretic operations. If L is a regular language to begin with, and if each language-theoretic operation *op* of section 2 is *regularity preserving* (in the sense that if M is a regular language, then so is $op(M)$), then L_1 and L_2 are also regular languages and the question $L_1 \subseteq L_2$ can be effectively answered. To obtain a decision procedure for our BSP verification problem, it is thus sufficient to show that the language-theoretic operations are regularity preserving. In the rest of this section we concentrate on showing this.

The language operations of Section 2 are of the following kinds: they either take a language over Σ and return a language over Σ, or they take a language over Σ and return a marked language over Σ, or they take a marked language over Σ and return a marked language over Σ. In all cases we show that if they take a regular language, they return a regular language.

(1) *Projection*: Let L be a language over Σ accepted by an FSA $\mathcal{A}$, and let $X \subseteq \Sigma$. Then we can construct $\mathcal{A}'$ accepting $L\restriction_X$ by simply replacing transitions in $\mathcal{A}$ of the form $p \xrightarrow{a} q$, with $a \notin X$, by ϵ-transitions $p \xrightarrow{\epsilon} q$.

(2) *l-del*: Let L be a language over Σ, with $L = L(\mathcal{A})$. We construct $\mathcal{A}'$ for *l-del*(L) as follows. We create two copies of $\mathcal{A}$. The initial state of $\mathcal{A}'$ is the initial state of the first copy. In the second copy all transitions of the form $p \xrightarrow{c} q$ with $c \in C$ are deleted. In the first copy we add an ϵ-transition from a state p in the first copy to state q in the second copy if $p \xrightarrow{c} q$ in $\mathcal{A}$, with $c \in C$. The final states in the first copy are marked non-final and the final states in the second copy are retained as final.

This construction can be described formally as follows. Let $\mathcal{A} = (Q, s, \longrightarrow, F)$. Define $\mathcal{A}' = (Q', s', \longrightarrow', F')$ where $Q' = Q \times \{1,2\}$, $s' = (s,1)$, $\longrightarrow'$ is given by

$$
\begin{array}{llll}
(p,1) \xrightarrow{a}{}' (q,1) & \text{if} & p \xrightarrow{a} q \text{ in } \mathcal{A} \\
(p,1) \xrightarrow{\epsilon}{}' (q,2) & \text{if} & p \xrightarrow{c} q \text{ in } \mathcal{A} \text{ with } c \in C \\
(p,2) \xrightarrow{a}{}' (q,2) & \text{if} & p \xrightarrow{a} q \text{ and } a \notin C,
\end{array}
$$

and $F' = F \times \{2\}$.

The construction is depicted in Fig. 3.

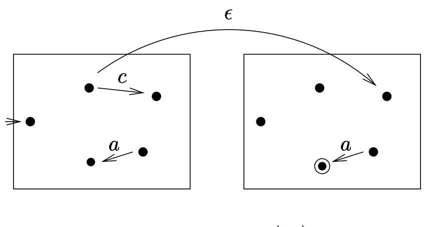

Fig. 3. *l-del*(L)

(3) *l-ins*: Let L be a language over Σ with $L = L(\mathcal{A})$. We construct $\mathcal{A}'$ for *l-ins*(L) as follows. We make two copies of $\mathcal{A}$. The start state of $\mathcal{A}'$ is the start state of the first copy, and the final states are the final states of the second copy. In the first copy for every transition $p \xrightarrow{a} q$ we add a c transition (for every $c \in C$) from p in the first copy to p in the second copy. The c-transitions for $c \in C$ are deleted from the second copy. The construction is depicted in Fig 4.

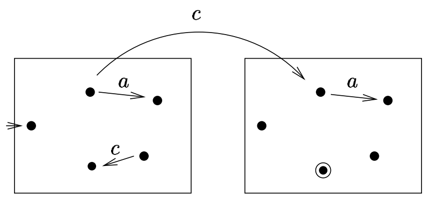

Fig. 4. *l-ins*(L)

(4) *l-ins-adm*: Let L be a language over Σ with $L = L(\mathcal{A})$, and let $X \subseteq \Sigma$. We construct $\mathcal{A}'$ for *l-ins-adm*$^X(L)$ as follows. We have two "copies" of $\mathcal{A}$. In the first copy, the states have two components: the first component keeps track of a state from $\mathcal{A}$, while the second keeps track of a *set* of states of $\mathcal{A}$ that are reachable by words that are X-equivalent to the current word being read. We have a transition labelled c, with $c \in C$, from a state (p, T) in the first copy to p in the second copy, provided T contains a state t from which it is possible to do a c and reach a final state. Once in the second copy, we allow only non-C transitions and retain the original final states.

More formally, we can define $\mathcal{A}'$ as follows. Let $\mathcal{A} = (Q, s, \longrightarrow, F)$ and let $\mathcal{B}$ be the automaton obtained from $\mathcal{A}$ by replacing transitions of the form $p \xrightarrow{a} q$ by $p \xrightarrow{\epsilon} q$ whenever $a \notin X$. Then $\mathcal{A}' = (Q', s', \longrightarrow', F')$ where $Q' = (Q \times 2^Q) \cup Q$; $s' = (s, S)$ where $S = \{q \in Q \mid s \xrightarrow{\epsilon}{}^* q \text{ in } \mathcal{B}\}$; $\longrightarrow'$ is given below:

$$\begin{array}{lll}
(p,T) \xrightarrow{a}{}' (q,T) & \text{if} & p \xrightarrow{a} q \text{ and } a \notin X \\
(p,T) \xrightarrow{a}{}' (q,U) & \text{if} & p \xrightarrow{a} q, a \in X, \text{ and} \\
 & & U = \{r \mid \exists t \in T, t \xrightarrow{a}{}^* r \text{ in } \mathcal{B}\} \\
(p,T) \xrightarrow{c}{}' p & \text{if} & \exists t \in T, q \in F : t \xrightarrow{c} q \text{ and } c \in C; \\
p \xrightarrow{a}{}' q & \text{if} & p \xrightarrow{a} q \text{ and } a \notin C.
\end{array}$$

and $F' = F$.

(5) *l-del-mark*: This construction is similar to *l-del* except that the label of the ϵ-transitions we add from the first copy to the second, is now $\natural$.

(6) *l-ins-mark*: The construction is similar to *l-ins*. Here instead of inserting a transition labelled c from the first copy to the second, we need to insert a transition labelled $c\natural$ from the first copy to the second. This can be

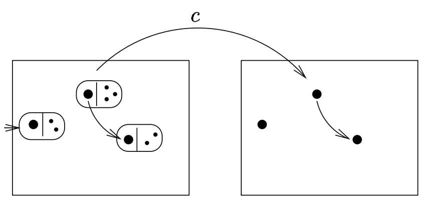

Fig. 5. *l-ins-adm*$^X(L)$

carried out by having a third copy of $\mathcal{A}$ placed between the first and second. The third copy has all its transitions deleted, and all its states are non-initial and non-final. A c transition from p in the first copy now goes to p in the third copy, and from p in the third copy we add a $\natural$ transition to p in the second copy.

(7) *l-ins-adm-mark*: The construction is similar to *l-ins-adm*. Instead of adding a c transition from the first copy to the second, we add one labelled $c\natural$ (once again this can be achieved using a third copy of $\mathcal{A}$).

(8) *mark*: Given $\mathcal{A}$ for $L \subseteq \Sigma^*$, we construct $\mathcal{A}'$ which accepts the marked language $mark(L)$. $\mathcal{A}$ is obtained from $\mathcal{A}$ as follows. We again use two copies of $\mathcal{A}$. The initial state of $\mathcal{A}'$ is the initial state of the first copy, and the final states are only those of the second copy. From every state in the first copy we add a transition labelled $\natural$ to its copy in the second.

(9) *Marked projection*: Given a marked language M, an FSA $\mathcal{A}$ accepting M, and $X \subseteq \Sigma$, we construct $\mathcal{A}'$ which accepts the marked language $M\restriction^m_X$. Once again we use two copies of $\mathcal{A}$. The initial state of the first copy is the initial state of $\mathcal{A}'$ and the final states of the second copy are the final states of $\mathcal{A}'$. From the first copy we delete transitions of the form $p \xrightarrow{\natural} q$ and add a transition labelled $\natural$ from p in the first copy to q in the second copy. In the second copy, we replace transition labels which are not in X by ϵ.

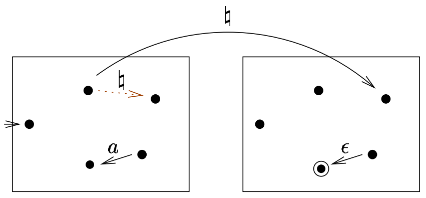

Fig. 6. $L\restriction^m_X$

(10) *l-del-con-mark*: Let L be a language over Σ and $\mathcal{A}$ be an FSA accepting L. Let $C' \subseteq C$ and $V' \subseteq V$. We construct $\mathcal{A}'$ accepting the marked language *l-del-con-mark*$_{C',V'}(L)$ as follows. We have four copies of $\mathcal{A}$. The second and third copies have all transitions deleted from them, and the fourth copy has all C transitions deleted from it. The initial state of the first copy is the initial state of $\mathcal{A}'$ and the final states of the fourth copy are the final states of $\mathcal{A}'$. For every transition $p \xrightarrow{c'} q$ with $c' \in C'$, we add an ϵ-transition from p in the first copy to q in the second copy. We add a v'-transition from a state r in the second copy to a state t in the third

copy iff $r \xrightarrow{v'} t$, with $v' \in V'$, is a transition in $\mathcal{A}$. Finally, we add a $\natural$-transition from each state u in the third copy to u in the fourth copy.

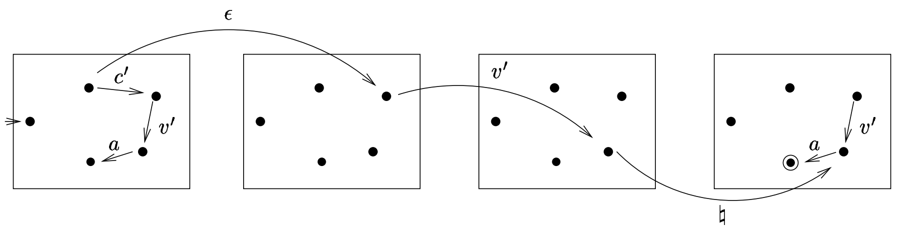

Fig. 7. *l-del-con-mark*$_{C',V'}(L)$

(11) *l-ins-con-mark*: Let L be a language over Σ and $\mathcal{A}$ be an FSA accepting L. Let $C' \subseteq C$ and $V' \subseteq V$. We construct $\mathcal{A}'$ accepting the marked language *l-ins-con-mark*$_{C',V'}(L)$ as follows. We have four copies of $\mathcal{A}$. The second and third copies have all transitions deleted from them, and the fourth copy has all C transitions deleted from it. The initial state of the first copy is the initial state of $\mathcal{A}'$ and the final states of the fourth copy are the final states of $\mathcal{A}'$. For every transition $p \xrightarrow{v'} q$ with $v' \in V'$, we add a c'-transition (for every $c' \in C'$) from p in the first copy to q in the second copy. We add a v'-transition from a state r in the second copy to a state t in the third copy iff $r \xrightarrow{v'} t$, with $v' \in V'$, is a transition in $\mathcal{A}$. Finally, we add a $\natural$-transition from each state u in the third copy to u in the fourth copy.

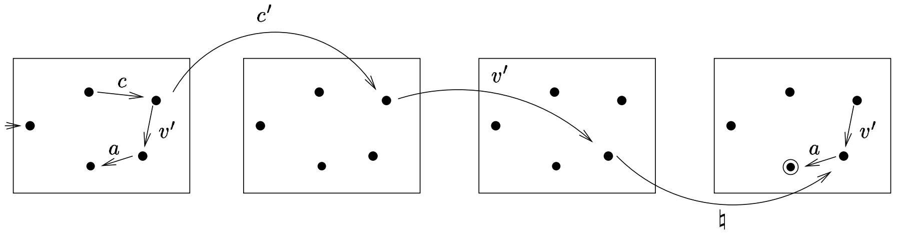

Fig. 8. *l-ins-con-mark*$_{C',V'}(L)$

(12) *l-ins-adm-con-mark*: Let L be a language over Σ with $L = L(\mathcal{A})$, and let $X \subseteq \Sigma$. Let $C' \subseteq C$ and $V' \subseteq V$. We construct $\mathcal{A}'$ for the language *l-ins-adm-con-mark*$^{X}_{C',V'}(L)$ as follows. We use four copies of $\mathcal{A}$. The first copy is exactly the same as in *l-ins-adm*$^{X}(L)$, where the states have two components, the first component keeping track of a state from $\mathcal{A}$, while the second keeps track of a *set* of states of $\mathcal{A}$ that are reachable by words that are X-equivalent to the current word being read. The second and third copies of $\mathcal{A}$ have all transitions deleted from them, and the fourth copy has all C transitions deleted from it. The initial state of the first copy is the initial state of $\mathcal{A}'$ and the final states of the fourth copy are the final states of $\mathcal{A}'$. We have a transition labelled c', with $c' \in C'$, from a state (p, T) in the first copy to p in the second copy, provided T contains a state t from which it is possible to do a c' and go to a final state. We

add a v'-transition from a state r in the second copy to a state u in the third copy iff $r \xrightarrow{v'} u$, with $v' \in V'$, is a transition in $\mathcal{A}$. Finally, we add a $\natural$-transition from each state w in the third copy to w in the fourth copy.

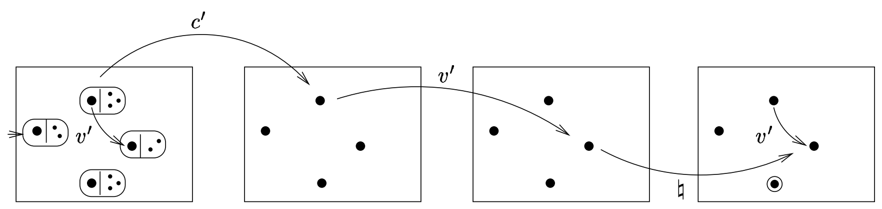

Fig. 9. *l-ins-adm-con-mark*$^{X}_{C',V'}(L)$

(13) *erase-con-mark*: Let $L \subseteq \Sigma^*$ and let $\mathcal{A}$ be an FSA with $L = L(\mathcal{A})$. Let $N' \subseteq N$ and $V' \subseteq V$. We construct $\mathcal{A}'$ accepting *erase-con-mark*$_{N',V'}(L)$ as follows. We have four copies of $\mathcal{A}$. The first and fourth copy have all their original transitions intact, the second has all transitions labeled with $a \notin N'$ deleted and transitions labelled n', with $n' \in N'$, replaced by ϵ-transitions; and the third has all its transitions deleted. We add an ϵ-transition from every state p in the first copy to p in the second copy; For every state p in the second copy such that $p \xrightarrow{v'} q$ in $\mathcal{A}$, we add a v'-transition from p in the second copy to q in the third copy. From every state r in the third copy we add a transition labelled $\natural$ to r in the fourth copy. The initial states of $\mathcal{A}'$ are the initial states of the first copy and the final states those of the fourth copy.

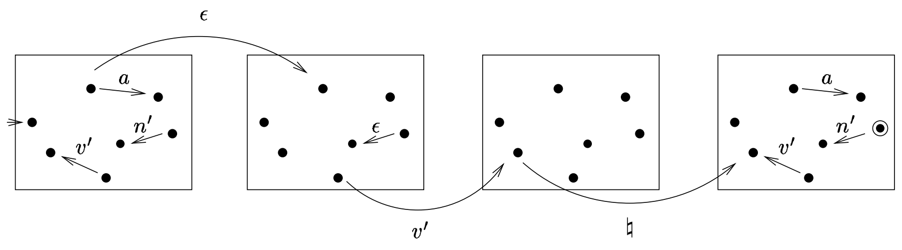

Fig. 10. *erase-con-mark*$_{N',V'}(L)$

5 Conclusion

We have demonstrated in this paper a way to automatically verify trace based information flow properties of finite state systems. We give characterisations of the properties in terms of language-theoretic operations on the set of traces of a system, rather than in terms of the structure of the system which is a stronger notion. This perhaps explains why we are able to obtain complete characterisations unlike the previous techniques in the literature.

The running time of our procedure can be seen to be exponential in the number of states of the given finite state transition system, in the worst case. This is because the automata constructions for the language-theoretic operations involve a blow-up in states of $O(n)$ in most cases, and $2^{O(n)}$ in the case of the BSP's based on the admissibility clause (here n is the number of states in the given transition system). Furthermore, no operation used on the right hand side of the containment (recall that our characterisations are typically of the form $op_1(L) \subseteq op_2(L)$) introduces an exponential blow-up. Thus in checking containment, we have to complement an automaton of size at most $O(n)$, and thus we have a bound of $2^{O(n)}$ in the worst case.

An interesting future direction to pursue is to see whether the technique of this paper can be extended to a more general logical language. The BSP's are all of the form "for every string in the language satisfying some conditions, there exists a string satisfying some other conditions" which are special cases of a first order logic interpreted over languages of strings. It would be interesting to investigate the decidability of such a logic when interpreted over regular languages, and to identify a natural decidable subclass of it which properly contains the BSP's of Mantel.

References

[1] H. Mantel, Possibilistic Definitions of Security – An Assembly Kit, in: Proceedings of the 13th IEEE Computer Security Foundations Workshop, IEEE Computer Society, Cambridge, UK, 2000, pp. 185–199.

[2] D. D'Souza, K. R. Raghavendra, B. Sprick, An automata based approach for verifying information flow properties, ENTCS 135 (1) (2005) 39–58.
URL `http://www.sciencedirect.com/science/journal/15710661`

[3] J. A. Goguen, J. Meseguer, Security policies and security models, in: Proc. IEEE Symp. on Security and Privacy, 1982, pp. 11–20.

[4] C. O'Halloran, A calculus of information flow, in: Proceedings of the European Symposium on Research in Computer Security, ESORICS 90, 1990.

[5] J. McLean, A general theory of composition for trace sets closed under selective interleaving functions, in: Proc. IEEE Symposium on Research in Security and Privacy, IEEE Computer Society Press, 1994, pp. 79 – 93.
URL `http://citeseer.nj.nec.com/mclean94general.html`

[6] A. Zakinthinos, E. S. Lee, A general theory of security properties, in: SP '97: Proceedings of the 1997 IEEE Symposium on Security and Privacy, IEEE Computer Society, Washington, DC, USA, 1997, p. 94.

[7] D. McCullough, Specifications for multilevel security and a hookup property, in: Proc. 1987 IEEE Symp. Security and Privacy, 1987.

[8] D. Sutherland, A model of information, in: Proceedings of the 9th National Computer Security Conference, 1986.

[9] D. Hutter, A. Schairer, Possibilistic information flow control in the presence of encrypted communication, in: 9th European Symposium on Research in Computer Security, LNCS, 2004.

[10] H. Mantel, A uniform framework for the formal specification and verification of information flow security, Ph.D. thesis, Universität des Saarlandes (2003).

[11] J. A. Goguen, J. Meseguer, Unwinding and inference control, in: Proc. IEEE Symp. on Security and Privacy, 1984, pp. 75–86.

[12] H. Mantel, Unwinding Possibilistic Security Properties, in: F. Cuppens, Y. Deswarte, D. Gollmann, M. Waidner (Eds.), European Symposium on Research in Computer Security (ESORICS), LNCS 1895, Springer, Toulouse, France, 2000, pp. 238–254.

[13] A. Bossi, R. Focardi, C. Piazza, S. Rossi, Bisimulation and unwinding for verifying possibilistic security properties, in: VMCAI 2003: Proceedings of the 4th International Conference on Verification, Model Checking, and Abstract Interpretation, Springer-Verlag, London, UK, 2003, pp. 223–237.

[14] R. Focardi, R. Gorrieri, The compositional security checker: A tool for the verification of information flow security properties, Software Engineering 23 (9) (1997) 550–571.
URL `citeseer.ist.psu.edu/article/focardi97compositional.htm%l`

[15] R. Focardi, R. Gorrieri, Automatic compositional verification of some security properties, in: Tools and Algorithms for Construction and Analysis of Systems, 1996, pp. 167–186.
URL `citeseer.ist.psu.edu/article/focardi96automatic.html`