An Automata Based Approach for Verifying Information Flow Properties

Deepak D'Souza   Raghavendra K. R.   Barbara Sprick

IISc-CSA-TR-2005-4
http://archive.csa.iisc.ernet.in/TR/2005/4/

Computer Science and Automation
Indian Institute of Science, India

April 2005

An Automata Based Approach for Verifying Information Flow Properties

Deepak D'Souza Raghavendra K. R.
and Barbara Sprick
Department of Computer Science and Automation
Indian Institute of Science, Bangalore, India.
{deepakd,raghavendrakr,sprick}@csa.iisc.ernet.in

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 (BSPs) defined by Mantel in [5], which are shown 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 BSPs.

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, as e.g. implemented in the UNIX operating system, every user can define for his data, which access is allowed for which users. 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 [4], 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-intereference as *What one group of users does using a certain ability has no effect on what some other group of users does* [4]. More precisely, their original notion of non-interference says that two systems (or users) S_1 and S_2 are non-interfering if the output of S_2 does not depend on the input of S_1.

Since Goguen and Meseguer's initial work, many more definitions about non-interference have been proposed in the literature. They all follow the same principle of a low level entity not being able to infer too much information about a high level user or high level activity in general. These security properties include, among others, *non-inference* [10, 9, 12] (which requires that each system behavior projected to low level behavior is itself a possible behavior), *separability* [9] (which requires that every possible low level behavior interleaved with every possible high level behaviour must be a possible behaviour of a system), *generalized non-interference* [8] (which requires that for every possible trace and every possible perturbation there is a correction to the perturbation such that the resulting trace is again a possible trace of the system), *nondeducability* [11], *restrictiveness* [8], the *perfect security property* [12], and many more.

Though all these properties follow the main idea of ensuring, that information is not leaked from high level users to low level users, they differ in their strictness as well as in the type of system they are defined for.

In [7, 5] Mantel has presented an approach to uniformly formalize all known trace based information flow properties. Based on sets of traces as the system model, Mantel has defined a set of *basic security predicates (BSPs).* He shows that all known trace based security properties can be represented as conjunctions of these BSPs. For example generalized noninterference can be defined as the conjunction of the two BSPs *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 obtained by inserting or deleting certain non-confidential events, 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 that is obtained by inserting or deleting certain non-confidential events, such that the resulting trace is also in the language L.

Our work is based on the modular framework presented in [5]. We present an automated verification technique to check whether a finite state system satisfies a given basic security predicate. Our approach is language based rather than structure based. We define a set of language theoretic operations 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 defined 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, the BSPs are properties of sets of traces rather than properties of traces and hence cannot be handled by classical model checking approaches. Nonetheless, our work gives a method to "model check" these properties by reducing them to the language inclusion problem for finite state systems.

Previous work dealing with the verification of trace based security properties (e.g.[6, 1, 3]) mainly employ unwinding theorems as verification technique for information flow properties. These techniques are typically sufficient though not necessary in all cases. 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 [2]. 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 [9] and [10]), our approach gives a decision procedure for the whole class of information flow properties that can be expressed in terms of BSPs.

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 occurence 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 [5] correspond to events that are *visible*, *confidential*, and *neither* visible nor confidential, from a particular user's point of view.

Definition 2.1 (Language-theoretic operations) *Let L be a language over Σ modelling sets of possible traces of a system and let M be a marked language over Σ. Let X be a subset of Σ.*

We define the following language-theoretic operations on L:

1. $L\restriction_X := \{\tau\restriction_X \mid \tau \in L\}$*, where $\tau\restriction_X$ is obtained from τ by deleting all events from τ that are not elements of X.*
2. *l-del*$(L) := \{\alpha\beta \mid \alpha c\beta \in L,\ \beta\restriction_C = \epsilon\}$.
   Operation l-del corresponds to the deletion of the last confidential event in a string. More precisely, this operation deletes the last occuring C-event from every string in L.
3. *l-ins*$(L) := \{\alpha c\beta \mid \alpha\beta \in L,\ \beta\restriction_C = \epsilon,\ c \in C\}$.
   Operation l-ins corresponds to the insertion of confidential events in strings of L. More precisely, l-ins contains all strings $\gamma \in \Sigma^*$ obtained by inserting a C-event in a string $\tau \in L$, in a position after which no C-events occur.
4. *l-ins-adm*$^X(L) :=$
   $\{\alpha c\beta \mid \alpha\beta \in L,\ \beta\restriction_C = \epsilon,$ *there exists* $\gamma c \in L,\ \gamma\restriction_X = \alpha\restriction_X,\ c \in C\}$.
   Operation l-ins-adm^X corresponds to admissible insertion of confidential events in strings of L. More precisely, this operation is similar to l-ins but allows only the insertion of admissible *C-events. The insertion of an event $c \in C$ is* admissible *after a prefix α in a string τ iff there exists another string $\gamma c \in L$ with γ projected to the set X being equal to α projected to X.*
5. *l-del-mark*$(L) := \{\alpha\natural\beta \mid \alpha c\beta \in L,\ \beta\restriction_C = \epsilon\}$.
   Operation l-del-mark corresponds to marked deletion of the last confidential event. More precisely, this operation replaces the last event $c \in C$ in every string of L by the special mark symbol $\natural$.
6. *l-ins-mark*$(L) := \{\alpha c\natural\beta \mid \alpha\beta \in L,\ \beta\restriction_C = \epsilon,\ c \in C\}$.
   Operation l-ins-mark corresponds to marked insertion of a confidential

event. This operation is similar to l-ins, but additionally introduces a mark $\natural$ after the newly inserted symbol.

7. l-ins-adm-mark$^X(L) :=$
$\{\alpha c\natural\beta \mid \alpha\beta \in L,\ \beta\upharpoonright_C = \epsilon,$ there exists $\gamma c \in L,\ \gamma\upharpoonright_X = \alpha\upharpoonright_X,\ c \in C\}$.
Operation l-ins-adm-mark^X corresponds to marked insertion of admissible events. More precisely, this operation is similar to l-ins-adm^X, but a mark $\natural$ is introduced after the newly inserted (admissible) symbol c in the string.

8. mark$(L) := \{\alpha\natural\beta \mid \alpha\beta \in L\}$.
Operation mark corresponds to the insertion of a mark at an arbitrary position. More precisely mark contains all strings which can be obtained by the insertion of the mark symbol in an arbitrary position of a string in L.

9. $M\upharpoonright^m_X := \{\alpha\natural\beta' \mid \alpha\natural\beta \in L,\ \beta' = \beta\upharpoonright_X\}$.
This operation corresponds to a marked projection. More precisely, this 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 $V' \subseteq V$.
l-del-con-mark$_{C',V'}(L) := \{\alpha v\natural\beta \mid \alpha c v\beta \in L,\ \beta\upharpoonright_C = \epsilon,\ c \in C',\ v \in V'\}$.
Operation l-del-con-mark corresponds to marked deletion in the "context" of an event in V'. More precisely, this operation replaces the last confidential event c in a string by the mark symbol, provided c belongs to C' and and is immediately followed by a V' event in the string.

11. Let $C' \subseteq C$ and $V' \subseteq V$.
l-ins-con-mark$_{C',V'}(L) := \{\alpha c v\natural\beta \mid \alpha v\beta \in L,\ \beta\upharpoonright_C = \epsilon,\ c \in C',\ v \in V'\}$.
Operation l-ins-con-mark corresponds to marked insertion in the context of a V' event. More precisely, l-ins-con-mark contains all strings obtained by inserting a C' event at a point in a string after which no confidential events occur and which is immediately followed by a V' event v; the mark symbol is also inserted after the event v.

12. Let $C' \subseteq C$ and $V' \subseteq V$.
l-ins-adm-con-mark$^X_{C',V'}(L) :=$
$\{\alpha c v\natural\beta \mid \alpha v\beta \in L,\ \beta\upharpoonright_C = \epsilon,$ there exists $\gamma c \in L,\ \gamma\upharpoonright_X = \alpha\upharpoonright_X,\ c \in C',\ v \in V'\}$.
Operation l-ins-adm-con-mark^X corresponds to the marked insertion of admissible events in the context of a V' event. This operation is similar

to l-ins-con-mark but allows only the insertion of admissible *C-symbols, where admissibility is defined as for operation l-ins-adm^X.*

13. *Let $N' \subseteq N$ and $V' \subseteq V$.*
erase-con-mark$_{N',V'}(L) := \{\alpha v \natural \beta \mid \alpha\delta v\beta \in L,\ \delta \in (N')^*,\ v \in V'\}$. Operation erase-con-mark corresponds to the marked erasure of N'-events. More precisely, erase-con-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 also inserted after the event v in the string.

3 Expressing BSP's Language-Theoretically

We now express the *basic security predicates* (BSPs) of Mantel in terms of the language-theoretic operations just defined, and the usual subset relation. For convenience we make use of the notation $L \subseteq_Y M$, where L and M are languages over Σ and $Y \subseteq \Sigma$, to mean L is a subset of M "modulo a correction on Y events". More formally, $L \subseteq_Y M$ iff $L\upharpoonright_{\overline{Y}} \subseteq M \upharpoonright_{\overline{Y}}$, where $\overline{Y}$ denotes $\Sigma - Y$.

Definition 3.1 (R) *L satisfies R (Removal of Events) iff for all $\tau \in L$ there exists $\tau' \in L$ such that $\tau' \upharpoonright_C = \epsilon$ and $\tau' \upharpoonright_V = \tau \upharpoonright_V$).*

Lemma 3.1 *R is satisfied by $L \Leftrightarrow L \upharpoonright_V \subseteq_N L$.*

Proof $\Rightarrow$: Consider any string τ in $L \upharpoonright_V$. There exists some τ' in L such that $\tau' \upharpoonright_V = \tau$. All the symbols in τ belong to V. Since R is satisfied by L, there exists τ'' in L such that $\tau'' \upharpoonright_C = \epsilon$ and $\tau'' \upharpoonright_V = \tau' \upharpoonright_V$. τ'' differs from τ with only N symbols. τ belongs to L modulo corrections of N. Hence $L \upharpoonright_V \subseteq_N L$.

$\Leftarrow$: Consider any string τ in L. Since $L \upharpoonright_V \subseteq_N L$, there exists a string τ' in L, such that it is equivalent to $\tau \upharpoonright_V$ modulo corrections of N-symbols. $\tau' \upharpoonright_C = \epsilon$ since $\tau \upharpoonright_V$ has no C-symbols. τ' is equivalent to $\tau \upharpoonright_V$ upto corrections with respect to N-symbols. Hence R is satisfied. □

Definition 3.2 (D) *L satisfies D (Stepwise Deletion of events) iff for all $\alpha c \beta \in L$, $c \in C$ such that $\beta \upharpoonright_C = \epsilon$, we have $\alpha'\beta' \in L$ with $\alpha' \upharpoonright_{V \cup C} = \alpha \upharpoonright_{V \cup C}$ and $\beta' \upharpoonright_{V \cup C} = \beta \upharpoonright_{V \cup C}$.*

Lemma 3.2 *D is satisfied by $L \Leftrightarrow$ l-del$(L) \subseteq_N L$.*

Proof $\Rightarrow$: Consider a string τ in *l-del*(L). τ will be of the form $\alpha\beta$ with $\beta\restriction_C = \epsilon$ where By the definition of *l-del*(L), there exists $\alpha c\beta \in L$ for some $c \in C$. $\alpha'\beta' \in L$ such that $\alpha' \restriction_{V\cup C} = \alpha \restriction_{V\cup C}$ and $\beta' \restriction_{V\cup C} = \beta \restriction_{V\cup C}$ (D is satisfied). $\tau = \alpha\beta$ belongs to L upto corrections of N-symbols. Hence *l-del*$(L) \subseteq_N L$.

$\Leftarrow$: Consider a string τ of the form $\alpha c\beta$ in L with $\beta \restriction_C = \epsilon$. By the definition of *l-del*(L), there exists $\alpha\beta \in$ *l-del*(L). Since *l-del*$(L) \subseteq_N L$, there exists $\tau' \in L$ such that τ and τ' are equivalent upto corrections of N-symbols. τ' can be expressed as $\alpha'\beta'$ such that $\alpha' \restriction_{V\cup C} = \alpha \restriction_{V\cup C}$ and $\beta' \restriction_{V\cup C} = \beta \restriction_{V\cup C}$ Hence D is satisfied. □

Definition 3.3 (I) *L satisfies I (Insertion of events) iff for all $\alpha\beta \in L$ such that $\beta \restriction_C = \epsilon$, we have $\alpha' c\beta' \in L$, for every $c \in C$ with $\beta' \restriction_{V\cup C} = \beta \restriction_{V\cup C}$, $\alpha' \restriction_{V\cup C} = \alpha \restriction_{V\cup C}$.*

Lemma 3.3 *I is satisfied by $L \Leftrightarrow$ l-ins$(L) \subseteq_N L$.*

Proof $\Rightarrow$: Consider a string τ in *l-ins*(L). τ will be of the form $\alpha c\beta$ with $\alpha\beta \in L, \beta \restriction_C = \epsilon$ and $c \in C$. Since I is satisfied by L, there exists $\alpha' c\beta' \in L$ such that $\alpha' \restriction_{V\cup C} = \alpha \restriction_{V\cup C}$ and $\beta' \restriction_{V\cup C} = \beta \restriction_{V\cup C}$. $\tau = \alpha c\beta$ belongs to L upto corrections of N-symbols. Hence *l-ins*$(L) \subseteq_N L$.

$\Leftarrow$: Consider a string $\tau \in L$ of the form $\alpha\beta$, where $\beta \restriction_C = \epsilon$. By the definition of *l-ins*(L), there exists $\alpha c\beta \in$ *l-ins*(L) for any $c \in C$. Since *l-ins*$(L) \subseteq_N L$, there exists τ' in L such that τ and τ' are equivalent upto corrections of N-symbols. τ' can be expressed as $\alpha' c\beta'$ where $\alpha' \restriction_{V\cup C} = \alpha \restriction_{V\cup C}$ and $\beta' \restriction_{V\cup C} = \beta \restriction_{V\cup C}$. Hence I is satisfied. □

Definition 3.4 (IA^X) *L satisfies IA^X (Insertion of X-admissible events) iff for all $\alpha\beta \in L$ with $\beta \restriction_C = \epsilon$ and there exists $\gamma c \in L$, $c \in C$ with $\gamma \restriction_X = \alpha \restriction_X$, we have $\alpha' c\beta' \in L$ with $\beta' \restriction_{V\cup C} = \beta \restriction_{V\cup C}$, $\alpha' \restriction_{V\cup C} = \alpha \restriction_{V\cup C}$).*

Lemma 3.4 *IA^X is satisfied by $L \Leftrightarrow$ l-ins-adm$^X(L) \subseteq_N L$.*

Proof $\Rightarrow$: Consider a string τ in *l-ins-adm*$^X(L)$. τ will be of the form $\alpha c\beta$ for some $c \in C$ such that there exists $\gamma c \in L$ with $\gamma \restriction_X = \alpha \restriction_X$ and $\alpha\beta \in L$. Since IA^X is satisfied, there exists $\alpha' c\beta' \in L$ with $\alpha' \restriction_{V\cup C} = \alpha \restriction_{V\cup C}$ and $\beta' \restriction_{V\cup C} = \beta \restriction_{V\cup C}$. $\tau = \alpha c\beta$ belongs to L upto corrections of N-symbols. Hence *l-ins-adm*$^X(L) \subseteq_N L$.

$\Leftarrow$: Consider a string $\tau \in L$ of the form $\alpha\beta$ with $\beta \restriction_C = \epsilon$ and there exists $\gamma c \in L$ with $\gamma \restriction_X = \alpha \restriction_X$ for some $c \in C$. By the definition of *l-ins-adm*$^X(L)$, there exists $\alpha c\beta \in$ *l-ins-adm*$^X(L)$. Since *l-ins-adm*$^X(L) \subseteq_N L$, there exists $\tau' \in L$ such that τ' and τ are equivalent upto corrections of N-symbols. τ' can be expressed as $\alpha' c\beta'$ such that $\alpha' \restriction_{V\cup C} = \alpha \restriction_{V\cup C}$ and $\beta' \restriction_{V\cup C} = \beta \restriction_{V\cup C}$. Hence IA^X is satisfied. □

Definition 3.5 (*BSD*) *L satisfies BSD (Backwards Strict Deletion) iff for all $\alpha c\beta \in L$, $c \in C$ such that $\beta \upharpoonright_C = \epsilon$, we have $\alpha\beta' \in L$ with $\beta' \upharpoonright_{V\cup C} = \beta \upharpoonright_{V\cup C}$.*

Lemma 3.5 *BSD is satisfied by $L \Leftrightarrow$ l-del-mark$(L) \upharpoonright_{\frac{m}{N}} \subseteq mark(L) \upharpoonright_{\frac{m}{N}}$.*

Proof $\Rightarrow$: Consider a string $\tau \in$ *l-del-mark*(L). τ can be expressed as $\alpha\natural\beta$ with $\beta \upharpoonright_C = \epsilon$ and $\alpha c\beta \in L$ for some $c \in C$. There exists $\alpha\natural\beta' \in$ *l-del-mark*$(L) \upharpoonright_{\frac{m}{N}}$ where β' is β with N-symbols deleted. Since *BSD* is satisfied by L, there exists $\alpha\beta'' \in L$ where $\beta \upharpoonright_{V\cup C} = \beta'' \upharpoonright_{V\cup C}$. By the definition of $mark(L)$, there exists $\alpha\natural\beta'' \in mark(L)$. Deleting N-symbols from β'' results in β'. So, $\alpha\natural\beta' \in mark(L) \upharpoonright_{\frac{m}{N}}$. Hence *l-del-mark*$(L) \upharpoonright_{\frac{m}{N}} \subseteq mark(L) \upharpoonright_{\frac{m}{N}}$.

$\Leftarrow$: Consider a string $\tau \in L$ of the form $\alpha c\beta$, $c \in C$ with $\beta \upharpoonright_C = \epsilon$. By the definition of *l-del-mark*(L), there exists $\alpha\natural\beta \in$ *l-del-mark*(L). By the definition of *marked projection*, there exists $\alpha\natural\beta' \in$ *l-del-mark*$(L) \upharpoonright_{\frac{m}{N}}$ where β' is β with N-symbols deleted. Since *l-del-mark*$(L) \upharpoonright_{\frac{m}{N}} \subseteq mark(L) \upharpoonright_{\frac{m}{N}}$, $\alpha\natural\beta' \in mark(L) \upharpoonright_{\frac{m}{N}}$. There exists $\alpha\natural\beta'' \in mark(L)$ for some β'' such that $\beta'' \upharpoonright_{\overline{N}} = \beta'$. β'' is equivalent to β upto corrections of N-symbols. By the definition of $mark(L)$, there exists $\alpha\beta'' \in L$. Hence *BSD* is satisfied. □

Definition 3.6 (*BSI*) *L satisfies BSI (Backwards Strict Insertion) iff for all $\alpha\beta \in L$ such that $\beta \upharpoonright_C = \epsilon$, we have $\alpha c\beta' \in L$, for every $c \in C$ with $\beta' \upharpoonright_{V\cup C} = \beta \upharpoonright_{V\cup C}$.*

Lemma 3.6 *BSI is satisfied by $L \Leftrightarrow$ l-ins-mark$(L) \upharpoonright_{\frac{m}{N}} \subseteq mark(L) \upharpoonright_{\frac{m}{N}}$.*

Proof $\Rightarrow$: Consider a string $\tau \in$ *l-ins-mark*(L). τ can be expressed as $\alpha c\natural\beta$, $c \in C$ with $\beta \upharpoonright_C = \epsilon$ and $\alpha\beta \in L$. There exists $\alpha c\natural\beta' \in$ *l-ins-mark*$(L) \upharpoonright_{\frac{m}{N}}$ where β' is β with N-symbols deleted. Since *BSI* is satisfied by L, there exists $\alpha c\beta'' \in L$ where $\beta \upharpoonright_{V\cup C} = \beta'' \upharpoonright_{V\cup C}$. By the definition of $mark(L)$, there exists $\alpha c\natural\beta'' \in mark(L)$. Deleting N-symbols from β'' results in β'. So, $\alpha c\natural\beta' \in mark(L) \upharpoonright_{\frac{m}{N}}$. Hence *l-ins-mark*$(L) \upharpoonright_{\frac{m}{N}} \subseteq mark(L) \upharpoonright_{\frac{m}{N}}$.

$\Leftarrow$: Consider a string $\tau \in L$ of the form $\alpha\beta$ with $\beta \upharpoonright_C = \epsilon$. By the definition of *l-ins-mark*(L), there exists $\alpha c\natural\beta \in$ *l-ins-mark*(L) for any $c \in C$. By the definition of $L \upharpoonright_{\frac{m}{N}}$, there exists $\alpha c\natural\beta' \in$ *l-ins-mark*$(L) \upharpoonright_{\frac{m}{N}}$ where β' is β with N-symbols deleted. Since *l-ins-mark*$(L) \upharpoonright_{\frac{m}{N}} \subseteq mark(L) \upharpoonright_{\frac{m}{N}}$, $\alpha c\natural\beta' \in mark(L) \upharpoonright_{\frac{m}{N}}$. There exists $\alpha c\natural\beta'' \in mark(L)$ for some β'' such that $\beta'' \upharpoonright_{\overline{N}} = \beta'$. β'' is equivalent to β upto corrections of N-symbols. By the definition of $mark(L)$, there exists $\alpha c\beta'' \in L$. Hence *BSI* is satisfied. □

Definition 3.7 (*BSIA^X*) *L satisfies BSIA^X (Backwards Strict Insertion of X-admissible events) iff for all $\alpha\beta \in L$ with $\beta \upharpoonright_C = \epsilon$ and there exists $\gamma c \in L$, $c \in C$ with $\gamma \upharpoonright_X = \alpha \upharpoonright_X$, we have $\alpha c\beta' \in L$ with $\beta' \upharpoonright_{V\cup C} = \beta \upharpoonright_{V\cup C}$.*

Lemma 3.7 *$BSIA^X$ is satisfied by L $\Leftrightarrow$ l-ins-adm-mark$^X(L) \upharpoonright^{m}_{\overline{N}} \subseteq$ mark$(L) \upharpoonright^{m}_{\overline{N}}$.*

Proof $\Rightarrow$: Consider a string $\tau \in$ *l-ins-adm-mark*$^X(L)$. τ can be expressed as $\alpha c \natural \beta$, $c \in C$ with $\beta \upharpoonright_C = \epsilon$ and $\alpha\beta \in L$ and there exists $\gamma c \in L$ with $\gamma \upharpoonright_X = \alpha \upharpoonright_X$. There exists $\alpha c \natural \beta' \in$ *l-ins-adm-mark*$^X(L) \upharpoonright^{m}_{\overline{N}}$ where β' is β with N-symbols deleted. Since $BSIA^X$ is satisfied, there exists $\alpha c \beta'' \in L$ where β and β'' are equivalent upto corrections of N-symbols. By the definition of *mark*(L), there exists $\alpha c \natural \beta'' \in$ *mark*(L). Deleting N-symbols from β'' results in β'. So, $\alpha c \natural \beta' \in$ *mark*$(L) \upharpoonright^{m}_{\overline{N}}$. Hence *l-ins-adm-mark*$^X(L) \upharpoonright^{m}_{\overline{N}}$ is a subset of *mark*$(L) \upharpoonright^{m}_{\overline{N}}$.

$\Leftarrow$: Consider a string $\tau \in L$ of the form $\alpha\beta$ with $\beta \upharpoonright_C = \epsilon$ and there exists $\gamma c \in L$, $c \in C$ with $\gamma \upharpoonright_X = \alpha \upharpoonright_X$. By the definition of *l-ins-adm-mark*$^X(L)$, there exists $\alpha c \natural \beta \in$ *l-ins-adm-mark*$^X(L)$. There exists $\alpha c \natural \beta' \in$ *l-ins-adm-mark*$^X(L) \upharpoonright^{m}_{\overline{N}}$ where β' is β with N-symbols deleted. Since *l-ins-adm-mark*$^X(L) \upharpoonright^{m}_{\overline{N}} \subseteq$ *mark*$(L) \upharpoonright^{m}_{\overline{N}}$, $\alpha c \natural \beta' \in$ *mark*$(L) \upharpoonright^{m}_{\overline{N}}$. There exists $\alpha c \natural \beta'' \in$ *mark*(L) for some β'' such that $\beta'' \upharpoonright_{\overline{N}} = \beta'$. β'' is equivalent to β upto corrections of N-symbols. By the definition of *mark*(L), there exists $\alpha c \beta'' \in L$. Hence $BSIA^X$ is satisfied. □

Definition 3.8 (*FCD*) *L satisfies FCD (Forward Correctable Deletion) iff for all $\alpha c v \beta \in L$, $c \in C'$, $v \in V'$ with $\beta \upharpoonright_C = \epsilon$ we have $\alpha \delta v \beta' \in L$ where $\delta \in (N')^*$ and $\beta' \upharpoonright_{V \cup C} = \beta \upharpoonright_{V \cup C}$.*

Lemma 3.8 *FCD is satisfied by L $\Leftrightarrow$*
l-del-con-mark$_{C',V'}(L) \upharpoonright^{m}_{\overline{N}} \subseteq$ erase-con-mark$_{N',V'}(L) \upharpoonright^{m}_{\overline{N}}$.

Proof $\Rightarrow$: Consider a string τ in *l-del-con-mark*$_{C',V'}(L)$. τ can be expressed as $\alpha v \natural \beta$ where $\alpha c v \beta \in L$, $c \in C'$, $v \in V'$ with $\beta \upharpoonright_C = \epsilon$. There exists $\alpha v \natural \beta' \in$ *l-del-con-mark*$_{C',V'}(L) \upharpoonright^{m}_{\overline{N}}$, where β' is β with N-symbols deleted. Since *FCD* is satisfied by L, there exists $\alpha \delta v \beta'' \in L$ where β'' and β are equivalent upto corrections of N-symbols, with $\delta \in (N')^*$ By the definition of *erase-con-mark*$_{N',V'}(L)$, there exists $\alpha v \natural \beta'' \in$ *erase-con-mark*$_{N',V'}(L)$. Deleting N-symbols from β'' results in β'. So, $\alpha v \natural \beta' \in$ *erase-con-mark*$_{N',V'}(L) \upharpoonright^{m}_{\overline{N}}$. Hence *l-del-con-mark*$_{C',V'}(L) \upharpoonright^{m}_{\overline{N}} \subseteq$ *erase-con-mark*$_{N',V'}(L) \upharpoonright^{m}_{\overline{N}}$.

$\Leftarrow$: Consider a string $\tau \in L$ of the form $\alpha c v \beta$, $c \in C'$, $v \in V'$ with $\beta \upharpoonright_C = \epsilon$. By the definition of *l-del-con-mark*$_{C',N'}(L)$, there exists $\alpha v \natural \beta \in$ *l-del-con-mark*$_{C',N'}(L)$. By the definition of *Marked Projection*, there exists $\alpha v \natural \beta' \in$ *l-del-con-mark*$_{C',N'}(L) \upharpoonright^{m}_{\overline{N}}$ with $\beta' = \beta \upharpoonright_{\overline{N}}$. As *l-del-con-mark*$_{C',V'}(L) \upharpoonright^{m}_{\overline{N}} \subseteq$ *erase-con-mark*$_{N',V'}(L) \upharpoonright^{m}_{\overline{N}}$ we have $\alpha v \natural \beta' \in$ *erase-con-mark*$_{N',V'}(L) \upharpoonright^{m}_{\overline{N}}$. There exists $\alpha v \natural \beta'' \in$ *erase-con-mark*$_{N',V'}(L)$ where $\beta'' \upharpoonright_{\overline{N}} = \beta'$. By the definition of *erase-con-mark*$_{N',V'}(L)$, there exists $\alpha \delta v \beta'' \in L$ with $\delta \in (N')^*$. β and β'' are

equivalent upto correction of N-symbols. This proves that *FCD* is satisfied.
□

Definition 3.9 (*FCI*) *L satisfies FCI (Forward Correctable Insertion) iff for all* $\alpha v\beta \in L$, $v \in V'$ *such that* $\beta \upharpoonright_C = \epsilon$, *we have* $\alpha c\delta v\beta' \in L$, *for every* $c \in C'$ *with* $\delta \in (N')^*$ *and* $\beta' \upharpoonright_{V\cup C} = \beta \upharpoonright_{V\cup C}$.

Lemma 3.9 *FCI is satisfied by L* $\Leftrightarrow$
l-ins-con-mark$_{C',V'}(L) \upharpoonright_{\frac{m}{N}} \subseteq$ *erase-con-mark*$_{N',V'}(L) \upharpoonright_{\frac{m}{N}}$.

Proof $\Rightarrow$: Consider a string $\tau \in$ *l-ins-con-mark*$_{C',V'}(L)$. τ can be expressed as $\alpha cv\natural\beta$, $c \in C'$, $v \in V'$ with $\alpha v\beta \in L$ and $\beta \upharpoonright_C = \epsilon$. There exists $\alpha cv\natural\beta' \in$ *l-ins-con-mark*$_{C',V'}(L) \upharpoonright_{\frac{m}{N}}$ where β' is β with N-symbols deleted. Since *FCI* is satisfied by L, there exists $\alpha c\delta v\beta'' \in L$ with $\delta \in (N')^*$ and $\beta'' \upharpoonright_{V\cup C} = \beta \upharpoonright_{V\cup C}$. There exists $\alpha cv\natural\beta'' \in$ *erase-con-mark*$_{N',V'}(L)$. Deleting N-symbols from β'' results in β'. So, $\alpha cv\natural\beta' \in$ *erase-con-mark*$_{N',V'}(L) \upharpoonright_{\frac{m}{N}}$. Hence *l-ins-con-mark*$_{C',V'}(L) \upharpoonright_{\frac{m}{N}}$ is a subset of *erase-con-mark*$_{N',V'}(L) \upharpoonright_{\frac{m}{N}}$.

$\Leftarrow$: Consider a string $\tau \in L$ of the form $\alpha v\beta$, $vinV'$ with $\beta \upharpoonright_C = \epsilon$. By the definition of *l-ins-con-mark*$_{C',V'}(L)$, there exists $\alpha cv\natural\beta \in$ *l-ins-con-mark*$_{C',V'}(L)$, $c \in C'$. There exists $\alpha cv\natural\beta' \in$ *l-ins-con-mark*$_{C',V'}(L) \upharpoonright_{\frac{m}{N}}$ with β' is β with N-symbols deleted. Since *l-ins-con-mark*$_{C',V'}(L) \upharpoonright_{\frac{m}{N}} \subseteq$ *erase-con-mark*$_{N',V'}(L) \upharpoonright_{\frac{m}{N}}$, $\alpha cv\natural\beta' \in$ *erase-con-mark*$_{N',V'}(L) \upharpoonright_{\frac{m}{N}}$. There exists $\alpha cv\natural\beta'' \in$ *erase-con-mark*$_{N',V'}(L)$ where $\beta'' \upharpoonright_{\overline{N}} = \beta'$. β'' and β are equivalent upto corrections of N-symbols. By the definition of *erase-con-mark*$_{N',V'}(L)$, there exists $\alpha c\delta v\beta'' \in L$. Hence *FCI* is satisfied. □

Definition 3.10 (*FCIA*^{X}) *L satisfies FCIA*^{X} *(Forward Correctable Insertion of X-admissible events) iff for all* $\alpha v\beta \in L$, $v \in V'$ *with* $\beta \upharpoonright_C = \epsilon$ *and there exists* $\gamma c \in L$, $c \in C'$ *with* $\gamma \upharpoonright_X = \alpha \upharpoonright_X$, *we have* $\alpha c\delta v\beta' \in L$ *with* $\delta \in (N')^*$ *and* $\beta' \upharpoonright_{V\cup C} = \beta \upharpoonright_{V\cup C}$.

Lemma 3.10 *FCIA*^{X} *is satisfied by L* $\Leftrightarrow$ *l-ins-adm-con-mark*$^X_{C',V'}(L) \upharpoonright_{\frac{m}{N}} \subseteq$ *erase-con-mark*$_{N',V'}(L) \upharpoonright_{\frac{m}{N}}$.

Proof $\Rightarrow$: Consider a string $\tau \in$ *l-ins-adm-con-mark*$^X_{C',V'}(L)$. τ can be expressed as $\alpha cv\natural\beta$, $c \in C'$, $v \in V'$ with $\alpha v\beta \in L$ and $\beta \upharpoonright_C = \epsilon$ and there exists $\gamma c \in L$ with $\gamma \upharpoonright_X = \alpha \upharpoonright_X$. There exists $\alpha cv\natural\beta' \in$ *l-ins-adm-con-mark*$^X_{C',V'}(L) \upharpoonright_{\frac{m}{N}}$ where β' is β with N-symbols deleted. Since *FCIA*^{X} is satisfied by L, there exists $\alpha c\delta v\beta'' \in L$ with $\delta \in (N')^*$ and $\beta'' \upharpoonright_{V\cup C} = \beta \upharpoonright_{V\cup C}$. By the definition of *erase-con-mark*$_{N',V'}(L)$, $\alpha cv\natural\beta'' \in$ *erase-con-mark*$_{N',V'}(L)$. Deleting N-symbols from β'' results in β'. So, $\alpha cv\natural\beta' \in$ *erase-con-mark*$_{N',V'}(L) \upharpoonright_{\frac{m}{N}}$. Hence *l-ins-adm-con-mark*$^X_{C',V'}(L) \upharpoonright_{\frac{m}{N}}$ is a subset of *erase-con-mark*$_{N',V'}(L) \upharpoonright_{\frac{m}{N}}$.

$\Leftarrow$: Consider a string $\tau \in L$ of the form $\alpha v\beta$, $vinV'$ with $\beta\restriction_C = \epsilon$ and there exists $\gamma c \in L$ with $\gamma\restriction_X = \alpha\restriction_X$. By the definition of $\textit{l-ins-adm-con-mark}^X_{C',V'}(L)$, there exists $\alpha cv\natural\beta \in \textit{l-ins-adm-con-mark}^X_{C',V'}(L)$, $c \in C'$. There exists $\alpha cv\natural\beta' \in \textit{l-ins-adm-con-mark}^X_{C',V'}(L)\restriction^m_{\overline{N}}$ with β' is β with N-symbols deleted. From $\textit{l-ins-adm-con-mark}^X_{C',V'}(L)\restriction^m_{\overline{N}} \subseteq \textit{erase-con-mark}_{N',V'}(L)\restriction^m_{\overline{N}}$ it follows that $\alpha cv\natural\beta' \in \textit{erase-con-mark}_{N',V'}(L)\restriction^m_{\overline{N}}$. There exists $\alpha cv\natural\beta'' \in \textit{erase-con-mark}_{N',V'}(L)$ where $\beta''\restriction_{\overline{N}} = \beta'$. β'' and β are equivalent upto corrections of N-symbols. By the definition of $\textit{erase-con-mark}_{N',V'}(L)$, there exists $\alpha c\delta v\beta'' \in L$. Hence $\textit{FCIA}^X$ is satisfied. □

Definition 3.11 (*SR*) *L satisfies SR (Strict Removal) iff for all $\tau \in L$ we have $\tau\restriction_{\overline{C}} \in L$.*

Lemma 3.11 *SR is satisfied by $L \Leftrightarrow L\restriction_{\overline{C}} \subseteq L$.*

Proof $\Rightarrow$: Consider any string τ in $L\restriction_{\overline{C}}$. By the definition of projection, there exists τ' in L such that $\tau'\restriction_{\overline{C}} = \tau$. Since *SR* is satisfied by L, $\tau = \tau'\restriction_{\overline{C}} \in L$. Hence $L\restriction_{\overline{C}} \subseteq L$.

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

Definition 3.12 (*SD*) *L satisfies SD (Strict Deletion) iff for all $\alpha c\beta \in L$, $c \in C$ such that $\beta\restriction_C = \epsilon$, we have $\alpha\beta \in L$.*

Lemma 3.12 *SD is satisfied by $L \Leftrightarrow$ l-del$(L) \subseteq L$.*

Proof $\Rightarrow$: Consider a string τ in *l-del*(L). τ can be expressed as $\alpha\beta$ with $\beta\restriction_C = \epsilon$ and $\alpha c\beta \in L$ for some $c \in C$. Since *SD* is satisfied by L, there exists $\alpha\beta \in L$. Hence *l-del*$(L) \subseteq L$.

$\Leftarrow$: Consider a string τ of the form $\alpha c\beta \in L$, $c \in C$. By the definition of *l-del*(L), there exists $\alpha\beta \in$ *l-del*(L). Since *l-del*$(L) \subseteq L$, $\alpha\beta \in L$. Hence SD is satisfied. □

Definition 3.13 (*SI*) *L satisfies SI (Strict Insertion) iff for all $\alpha\beta \in L$ such that $\beta\restriction_C = \epsilon$, we have $\alpha c\beta \in L$, for every $c \in C$.*

Lemma 3.13 *SI is satisfied by $L \Leftrightarrow$ l-ins$(L) \subseteq L$.*

Proof $\Rightarrow$: Consider a string $\tau \in L$. τ can be expressed as $\alpha c\beta$, $c \in C$ such that $\beta\restriction_C = \epsilon$ with $\alpha\beta \in L$. Since *SI* is satisfied by L, there exists $\alpha c\beta \in L$. Hence *l-ins*$(L) \subseteq L$.

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

Definition 3.14 (SIA^X) *L satisfies SIA^X (Strict Insertion of X-admissible events) iff for all $\alpha\beta \in L$ such that $\beta\restriction_C = \epsilon$ and there exists $\gamma c \in L$, $c \in C$ with $\gamma\restriction_X = \alpha\restriction_X$, we have $\alpha c\beta \in L$.*

Lemma 3.14 *SIA^X is satisfied by $L \Leftrightarrow$ l-ins-adm$^X(L) \subseteq L$.*

Proof $\Rightarrow$: Consider a string $\tau \in$ *l-ins-adm*$^X(L)$. τ can be expressed as $\alpha c\beta$ with $\beta\restriction_C = \epsilon$ and there exists $\gamma c \in L$, $c \in C$ with $\gamma\restriction_X = \alpha\restriction_X$. Since *SIA*^{X} is satisfied by L, there exists $\tau = \alpha c\beta \in L$. Hence *l-ins-adm*$^X(L) \subseteq L$.

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

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 $\mathcal{T}$. More precisely we define $\xrightarrow{\alpha}{}^*$ inductively by saying $p \xrightarrow{\epsilon}{}^* p$ for all $p \in Q$, and $p \xrightarrow{\alpha a}{}^* q$ whenever there exists $r \in Q$ such that $p \xrightarrow{\alpha}{}^* r$ and $r \xrightarrow{a} q$. The language *accepted* (or *generated*) by the transition system $\mathcal{T}$ is defined to be $L(\mathcal{T}) = \{\alpha \in \Delta^* \mid p \xrightarrow{\alpha}{}^* q \text{ for some } q \in Q\}$.

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.* Here 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. ϵ-transitions don't add the to the expressive power of automata, as one can give a language equivalent automaton $\mathcal{B}$ for a given automaton with ϵ-transitions $\mathcal{A}$ by adding transitions of the form $p \xrightarrow{a} q$ whenever $p \xrightarrow{a}{}^{*} q$ in $\mathcal{A}$, and then deleting the ϵ-transitions.

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 thus 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 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 give 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 wrt X.* 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 of the form $p \xrightarrow{a} q$, with $a \notin X$, in $\mathcal{A}$, by an ϵ-transition $p \xrightarrow{\epsilon} q$.

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 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 the final states in the second copy are retained.

   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

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

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

The construction is depicted in Fig. 1.

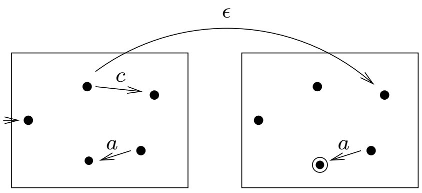

Figure 1: *l-del*(L)

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 2.

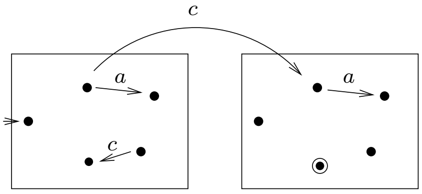

Figure 2: *l-ins*(L)

4. *l-ins-adm*^{X}. 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:

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

and $F' = F$.

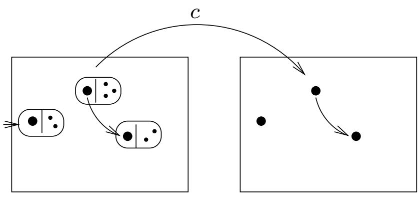

Figure 3: *l-ins-adm*$^X(L)$

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 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 neither initial nor 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*^{X}. The construction is similar to *l-ins-adm*^{X}. 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 the same state in the second copy.

9. *Marked projection* $L\upharpoonright^m_X$. Given a marked language M, an FSA $\mathcal{A}$ accepting M, and $X \subseteq \Sigma$, we construct $\mathcal{A}'$ which accepts the marked language $M\upharpoonright^m_X$. 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 transition 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.

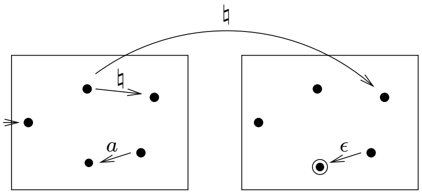

Figure 4: $L\upharpoonright^m_X$

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.

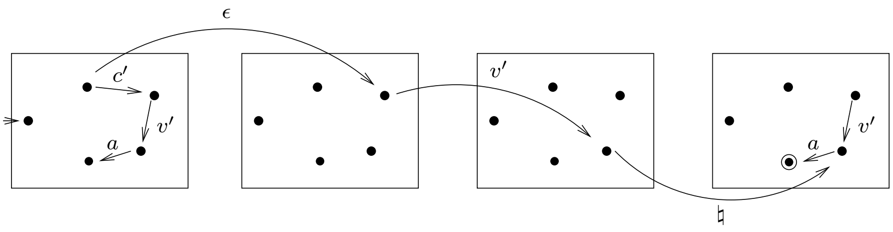

Figure 5: $\textit{l-del-con-mark}_{C',V'}(L)$

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 $\textit{l-ins-con-mark}_{C',V'}(L)$ 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.

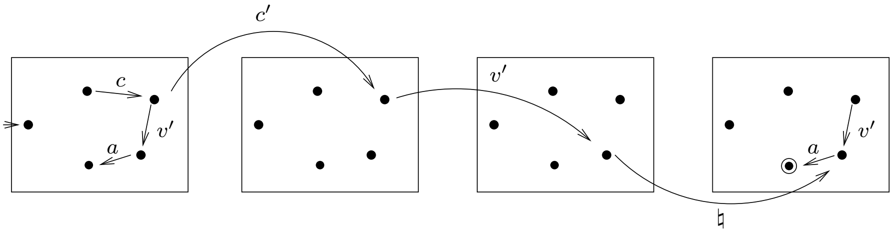

Figure 6: $\textit{l-ins-con-mark}_{C',V'}(L)$

12. $\textit{l-ins-adm-con-mark}^X$. 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 $\textit{l-ins-adm-mark}^X(C')V'L$ as follows. We use four "copies" of $\mathcal{A}$. The first copy is exactly the same as in $\textit{l-ins-adm}^X(L)$, 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'. 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.

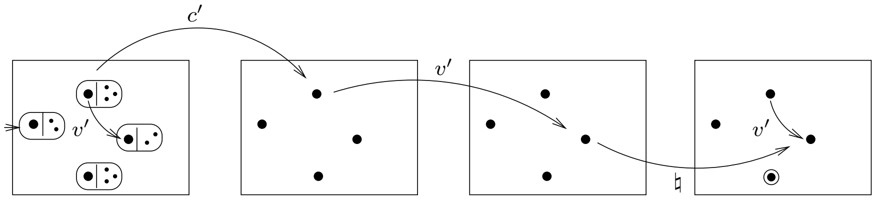

Figure 7: *l-ins-adm-con-mark*$^X_{C',V'}(L)$

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 p in the third copy we add a transition labelled $\natural$ to p 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.

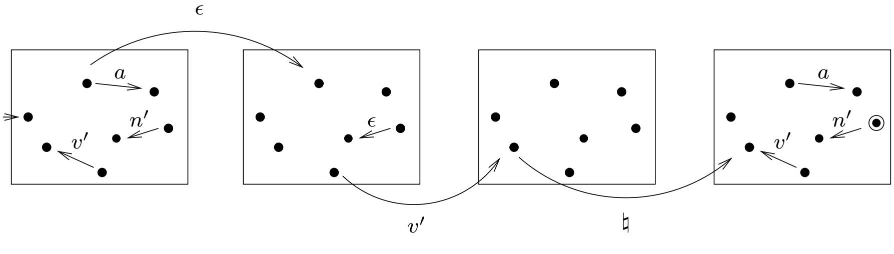

Figure 8: *erase-con-mark*$_{N',V'}(L)$

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 wost 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.

References

[1] Annalisa Bossi, Riccardo Focardi, Carla Piazza, and Sabina 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*, pages 223–237, London, UK, 2003. Springer-Verlag.

[2] Riccardo Focardi and Roberto Gorrieri. Automatic compositional verification of some security properties. In *Tools and Algorithms for Construction and Analysis of Systems*, pages 167–186, 1996.

[3] Riccardo Focardi and Roberto Gorrieri. The compositional security checker: A tool for the verification of information flow security properties. *Software Engineering*, 23(9):550–571, 1997.

[4] J. A. Goguen and J. Meseguer. Security policies and security models. In *Proc. IEEE Symp. on Security and Privacy*, pages 11–20, April 1982.

[5] Heiko Mantel. Possibilistic Definitions of Security – An Assembly Kit. In *Proceedings of the 13th IEEE Computer Security Foundations Workshop*, pages 185–199, Cambridge, UK, July 3–5 2000. IEEE Computer Society.

[6] Heiko Mantel. Unwinding Possibilistic Security Properties. In F. Cuppens, Y. Deswarte, D. Gollmann, and M. Waidner, editors, *European Symposium*

on Research in Computer Security (ESORICS), LNCS 1895, pages 238–254, Toulouse, France, October 4-6 2000. Springer.

[7] Heiko Mantel. *A Uniform Framework for the Formal Specification and Verification of Information Flow Security.* PhD thesis, Universität des Saarlandes, 2003.

[8] Daryl McCullough. Specifications for multilevel security and a hookup property. In *Proc. 1987 IEEE Symp. Security and Privacy*, 1987.

[9] John McLean. A general theory of composition for trace sets closed under selective interleaving functions. In *Proc. IEEE Symposium on Research in Security and Privacy*, pages 79 – 93. IEEE Computer Society Press, 1994.

[10] Colin O'Halloran. A calculus of information flow. In *Proceedings of the European Symposium on Research in Computer Security, ESORICS 90*, 1990.

[11] David Sutherland. A model of information. In *Proceedings of the 9th National Computer Security Conference*, 1986.

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