Uppsala University

Non-Interference on Symbolic Transition System

Jeremy Dubreil

Superviser : Ivan Christoff

Master project hosted at INRIA Rennes in the team VerTeCs.

Contents

List of Figures

Introduction

The sake of security is now one of the most important challenges for the information technologies. The need for security has been increasing with the improvements of systems in sharing resources between users. For the conception of the first multi-users system, engineers had to find solutions to ensure confidentiality of data in a context several users are sharing memory. Nowadays, the situation is much more complex with the advent of the Internet newtwork and the explosion of the interaction possibilities. But, while the deployment of IPv6 is promising an unification of networks, and increasing number of communicating systems, the security remains experimental for most of the case. Indeed the degree of security for programs is often defined according to the number and the potential gravity of the existing attacks. Meanwhile, the use of formal methods to ensure functional properties, in the sense that the program really does what it is expected to do, is successfully increasing for embedded system or telecommunication protocols for example. But fruitful attempts for security properties are mostly confined to cryptographic protocols or very small system like credit cards. In the actual reseach, objectives are to define efficient methods to ensure security properties for distributed applications, running on different platform and communicating through the Internet.

For this project, we focus our studies on secrecy properties and especially for the case of information flow. In some case, it is possible for an attacker to infer some secret information about the system without any illegal actions. We say then that the secret is interfering with users actions. Indeed, for the cases we are going to study, we assume that the users have a complete knowledge of the system but in the meantime that some actions are unobservable. We say then that the attacker is inferring some secret information from one observation because, in the general case, he or she has to deduce it form the set of internal behaviours possibly leading to the same observation. In an

other domain, the diagnosis theory [1, 2, 3] gives some methods for monitoring the occurrences of particular events, faults for example, from a partial observation. As for secrecy, the monitor has to decide with certainty the occurrence of a fault from all the possibilities implying the same observation. We see that there are some similarities between the notions of diagnosis and the notion of interference and we try to give a definition of non-interference which allow us to extend the methods given by [3] for discrete event system diagnosis.

In opposition to what we have with cryptographic protocols for example, we wish to deal with security policies independent of the systems. For example the security policy may concern the occurrence of system calls invisible for users. Then we wish to have algorithm able to solve the interference problem for a class of security policies concerning these system calls and a set of systems using them. This idea will lead us to introduce the notion of environment which is basically a set of actions and a set of variable. Then only security policies involving the actions and the variables of the environment will be considered.

This report start with a short overview of security policies and the historic of non-interference notion in chapter 1. We will introduce in chapter 2 the modelling framework used for this internship in order to give a formal definitions of the notions of interference in chapter 3 and also, how the diagnosis problem can be understood as a special case of interference. Then we arrive in chapter 4 to main contribution of this report with the presentation of the algorithms used for non-interference checking or attacks generations. We will expose the case of property given by regular languages over the set of action for which the problem can be solved. For infinite system, the use of over-approximation of the set of behaviours will permit us to conclude with certainty for some case that a system is non-interfering but losing in the meantime certitude for systems our algorithm declares to be interfering. The conclusion of the document gives some limitations of our approach and some proposition for further works.

Chapter 1

Security Policies

For the sake of security, researchers generally divide the problems into three categories : **secrecy**, **integrity** and **availability**. The secrecy concerns the ability of an information system to allow or deny in the expected way the accessibility of some information. A good feature of multi-user operating systems can be, for example, the possibility to hide some files from the other users. In the same context most of the operating systems are designed to offer different possibilities of user right system management. In such a case, the integrity is satisfied if a user cannot illegally perform some critical settings. And, on the other hand, satisfying the availability means that every user can really do what the security policy allowed him, or her, to do. In general, these three notions are not independent from each other. Indeed a user bypassing the secrecy can access some crucial information and then corrupt the system. Moreover, damaging the system can change some of its functional properties and violate the availability. Therefore, a security policy can be described by a set of deontic logic rules but this policy has to be easy to understand and to deploy over an information system.

1.1 MLS, Multi Layer Secure models

Research about security of information system has started within the U.S. military domain after the second world war. The first attempts to formalise what is a security policy was a transposition of the pen-and-paper world to computer system. In this context, Lampson [4] introduce the Multi Layer Secure model based on the notion of subjects (active agents), objects (pas-

sive agents) and action the subject can do on objects (ex : read, write). Subjects and Objects are sorted following the idea of hierarchy in military organisation. For example :

Top Secret

—

Confidential

—

Public

The principle of MLS models is, for a given action, the rights of the subjects and the objects are set according to the security level they belong to. The rights can be described using an access control matrix where the columns concern the object's level and the line the subject's level :

	Top Secret	Confidential	Public
Top Secret	read, write	read	read
Confidential	write	read, write	read, write
Public			read

One of the main examples is the BLP model designed for the US Department of Defence. MLS can be described

1.1.1 The Bell and La Padula Model

In 1973, in a book known as the *Orange Book*, David D.E. Bell and L.J. La Padula[5] went ahead in the formalisation of MLS for access control. The BLP model aims to ensure secrecy or confidentiality of the objects and to avoid information flow from subjects with higher level to other subjects with lower level. For this we assume that read and write are the only two ways to access the object. The BLP model has been inspired of such a context and can be summarised by "no read up, no write down". Let S and O be the set of subjects and the set of objects. Let $(L, \leq)$ be a partially ordered set representing the confidentiality layers and a mapping $M : S \cup O \rightarrow L$

satisfying the property :

$$\forall s \in S, \forall o \in O,\ M(s) \leq M(o) \text{ or } M(o) \leq M(s)$$

The two following properties ensure that there cannot be any information flow downward w.r.t L :

The Simple Security Property :

$$access_{read}(s, o) \text{ if } M(o) \leq M(s)$$

The * Security Property :

$$access_{write}(s, o) \text{ if } M(s) \leq M(o)$$

Indeed, for two subjects s, s' such that $M(s) < M(s')$, s' cannot write in an object s can access. But we will see later that the BLP model is not sufficient to avoid all information flows. For example this concept is implemented in Unix systems to deal with password confidentiality. The file containing the passwords has a higher level than the users. A regular user can modify the parts corresponding to his or her password but cannot access the informations, even to his or her own password. But this model only deals with secrecy. A subject s with *confidential* level can write in some file with *top secret* level. Intentionally or not, s may then corrupt *top secret* informations.

1.1.2 The Biba integrity model

The Biba model is quite similar to BLP but The Simple Security Property and The * Security Property are inverted :

$$access_{read}(s, o) \text{ if } M(s) \leq M(o)$$

$$access_{write}(s, o) \text{ if } M(o) \leq M(s)$$

This ensures that regular users cannot modify some critical system files for example. To gain more flexibility, Biba and BLP can be enhanced with the concept of **Trusted Agents** allowed to bypass the * property to change the secrecy or the integrity requirement of some objects. Biba and BLP models are referred as a MAC model (Mandatory Access Control). MAC models don't allow the subject to transfer access grant to other subjects even if their confidentiality level are the same or lower.

1.1.3 The HRU model

In order to design a model more flexible than BLP or Biba, M. A. Harrison, W. L. Ruzzo and J. D. Ullman propose[6] that every object belongs to one subject which can grant or deny access of its objects to the other subjects.

This model becomes much more flexible but has to face undecidability problems which weren't in the previously described models. Indeed the authors show in the same article that it is in general impossible to decide if an access can be granted or not. In other words it is undecidable to know if the can become insecure only by applying the legal changes subjects are allowed to do on their objects.

1.2 Non-Interference model

The BLP model is designed to avoid confidential information to flow through the files from up to down. But this cannot avoid all the information channels. For example if a user with a low level try to create a file at a upper level. There is two possibilities : if a file with the same name already exist, there will be an error message, otherwise the user can create the file. This is enough to create a communication channel between high level users and low level one. Even is the bandwith might not be much, this violate the security policy. In 1982, Goguen and Mesenger first introduced in [7] the notion of non-interference to avoid cover channel problem. They gave the notion as follow :

Definition 1 (Goguen & Meseguer, 1982) *Given a system involving two different groups of user, what one group of users does has no effect on what other group of users can see.*

But this definition appears to be too restrictive to deal with a wider range of security concerns. For sake of generality, Focardi and Gorrieri gave an other definition for system described in the model algebra model [8]. They consider an partition of the actions into high level actions H and low level ones L and into inputs I and outputs O. .

Definition 2 (Focardi & Gorrieri, 1993) *A system M is non-interfering if the occurrences of high level actions H have no influence on the occurrences of the low level actions L.*

This notion of non-interference is more general than the one given by Goguen and Meseguer. Indeed, for a system M involving two groups of users, U_1 and U_2, we define the high level actions H to be the actions of U_1 and the low level actions L to be the actions of U_1. Then, what U_1 is doing correspond to the input actions of H and if M is non-interfering according to Focardi & Gorrieri, then U_1 actions have no influence on L, i.e. what U_2 is seeing, so M non-interfering according to Goguen & Meseguer. But we wish for this project to handle a more general case that the occurrences of high level input. For example we expect to be able to handle more dynamic properties like particular sequences of actions.

Definition 3 (Focardi & Gorrieri, 1993) *Let $M = (Q, \Lambda, Q_0, \rightarrow)$ be a LTS with $\Lambda = H \cup L$ and $\Lambda = I \cup O$. $\forall s \in \mathcal{L}(M)$ if $\pi_{I \cap H}(s) \neq \varepsilon$ then $\exists s' \in \mathcal{L}(M)$ such that $\pi_L(s') = \pi_L(s)$ and $\pi_{I \cap H}(s') = \varepsilon$.*

Chapter 2

Modeling framework

The system we are studying is this paper are described within the communicating automata theory. We will use this automaton and also automaton extended with typed variable to model both the systems and the security properties these systems are expected to satisfy. We mainly present in this chapter the notion of environment and the STS model. To study a case as general as possible, we do not expect automata to be finite. The set of states and the set of actions can have an arbitrary cardinal. We begin with presenting the Labelled Transition System (LTS) which is basically a possibly infinite automaton without accepting states.

2.1 The LTS model

Definition 4 *A LTS M is a tuple $(Q, Q_0, \Lambda, \rightarrow)$ with :*

- *Q is a set of states*
- *$Q_0 \subseteq Q$ is the subset of initial states*
- *Λ is a finite or infinite set of actions*
- *$\rightarrow \subseteq Q \times \Lambda \times Q$ is the transition relation*

Example of LTS. Let $M = \{q_0, q_1, q_2\}$, $Q_0 = \{q_0\}$, $\Lambda = \{a, b\}$:
Let $M = (Q, Q_0, \Lambda, \rightarrow)$ be an LTS, we use the following notations :

- We write $q \xrightarrow{\alpha} q'$ for $(q, \alpha, q') \in \rightarrow$ and $q \xrightarrow{\alpha}$ for $\exists q' : q \xrightarrow{\alpha} q'$.

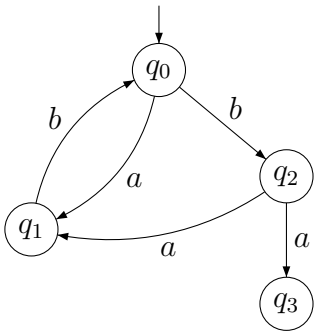


Figure 2.1: An example of LTS

- This notation is extended to Λ^* by $q \xrightarrow{s} q'$ when there is a sequence $\{q_0, q_1, ..., q_{n-1}\} \subseteq Q$ and $\{\alpha_0, \alpha_1, ..., \alpha_n\} \subseteq \Lambda$ such that $s = \alpha_0\alpha_1...\alpha_n$ and $\rho = q \xrightarrow{\alpha_0} q_0 \xrightarrow{\alpha_1} q_1... \xrightarrow{\alpha_{n-1}} q_{n-1} \xrightarrow{\alpha_n} q'$.

- $M \xrightarrow{s}$ means that $\exists q_0 \in Q_0$ such that $q_0 \xrightarrow{s}$. We define the set of sequences of M by $\mathcal{L}(M) = \{s \in \Lambda^*,\ M \xrightarrow{s}\}$. $\rho = q_0 \xrightarrow{s} q$ is called a run in M, $\mathcal{R}(M)$ is the set of all runs. $\mathcal{R}(M) \subset Q_0 \cdot (\Lambda \cdot Q)^*$.

- When there is a set of final states $Q_m \subset Q$ then $\mathcal{L}(M, Q_m) = \{s \in \Lambda^* : \exists q_m \in Q_m,\ M \xrightarrow{s} q_m\}$ is the language recognised by M. If Q is finite and Λ is a finite set of actions then $(Q, \Lambda, Q_0, Q_m, \rightarrow)$ is a finite automaton.

- M is deterministic if there is only one initial state, $|Q_0| = 1$, and $\forall (q, q', q'') \in Q^3, \forall a \in \Lambda,\ q \xrightarrow{a} q' \wedge q \xrightarrow{a} q'' \Rightarrow q' = q''$

For a set X with the concatenation operator $\cdot$ and a subset $A \subset X$ we define the projection :

$$\begin{aligned} \pi_A : X^* &\rightarrow A^* \\ \epsilon &\mapsto \epsilon \\ x \cdot \omega &\mapsto x \cdot \pi_A(\omega) \text{ if } x \in A \\ x \cdot \omega &\mapsto \pi_A(\omega) \text{ otherwise} \end{aligned}$$

For a set of state Q and a set of action Λ, we extend π to $Q \cdot (\Lambda \cdot Q)^*$. If there is a set of observable actions $\Lambda_o \subset \Lambda$ then we write $\mathcal{T}(M) = \pi_{\Lambda_o}(\mathcal{R}(M))$ the set of traces of M. In order to introduce some of the notations used further,

let V be a set of typed variables. For $v \in V$, $Dom(v)$ is the domain (i.e. type) of v. We note with $Dom(V) = \times_{v \in V} Dom(v)$ the set of vectors of variables' valuations. A vector of valuation is denoted by $\vec{q} \in Dom(V)$. For two set of variables V and V' such that $V \cap V' = \emptyset$ we write $\langle \vec{q}, \vec{q'} \rangle \in Dom(V \cup V')$ the valuation vector in $V \cup V'$ created with $\vec{q} \in Dom(V)$ and $\vec{q'} \in Dom(V')$. In the other direction, for $\vec{q} \in Dom(V \cup V')$ we note by $\vec{q_{|_V}} \in Dom(V)$ and $\vec{q_{|_{V'}}} \in Dom(V')$ the two vectors such that $\vec{q} = \langle \vec{q_{|_V}}, \vec{q_{|_{V'}}} \rangle$.

Definition 5 *An environment $\mathcal{E} = (\mathcal{V}, \Sigma)$ is a set of typed variable variables $\mathcal{V}$ and a finite set of actions Σ. We have a partition $\Sigma = \Sigma_o \cup \Sigma_{uo}$ into the observable[1] and the internal actions. An observable action $a \in \Sigma_o$ can take a communication parameter which type P_a only depends of a. The internal actions take the empty tuple as parameter. Finally, we define $\Lambda = \Lambda_o \cup \Sigma_{uo}$ the set of all possible events taking into account the parameters of observable actions. We note $R_{\mathcal{E}} = Dom(\mathcal{V}) \cdot (\Lambda \cdot Dom(\mathcal{V}))^*$.*

2.2 The STS model

The Symbolic Transition System model is a way to describe finite automata extended with variables. The result is a system (STS) with a possibly infinite set of states. The main advantage of the STS model is that it can be used to describe a wide range of communicating systems thanks to the use of typed variables and the possibility for actions to take communication parameters. The STS model is suitable to model processes described in an imperative manner using a finite set of primitives. We only consider here systems built with actions form the environment $\mathcal{E} = (\mathcal{V}, \Sigma)$.

Definition 6 *A Symbolic Transition System $\mathcal{M}$ is a tuple (V, Θ, Σ, T) where:*

- *V is a finite set of typed variables. $L \subset V$ is a set of locality labels[2] such that $Dom(L)$ is finite*
- *Θ is the initial condition, i.e. a predicate on the variables*

[1] Usually, we consider a partition of the observable actions into inputs and outputs : $\Sigma_o = \Sigma_? \cup \Sigma_!$. But this distinction won't be useful for our concern.

[2] $l \in L$ is a label giving the name of the control points (i.e branching) during the execution of the program. $Dom(L)$ would be the set of state of an automaton without variables

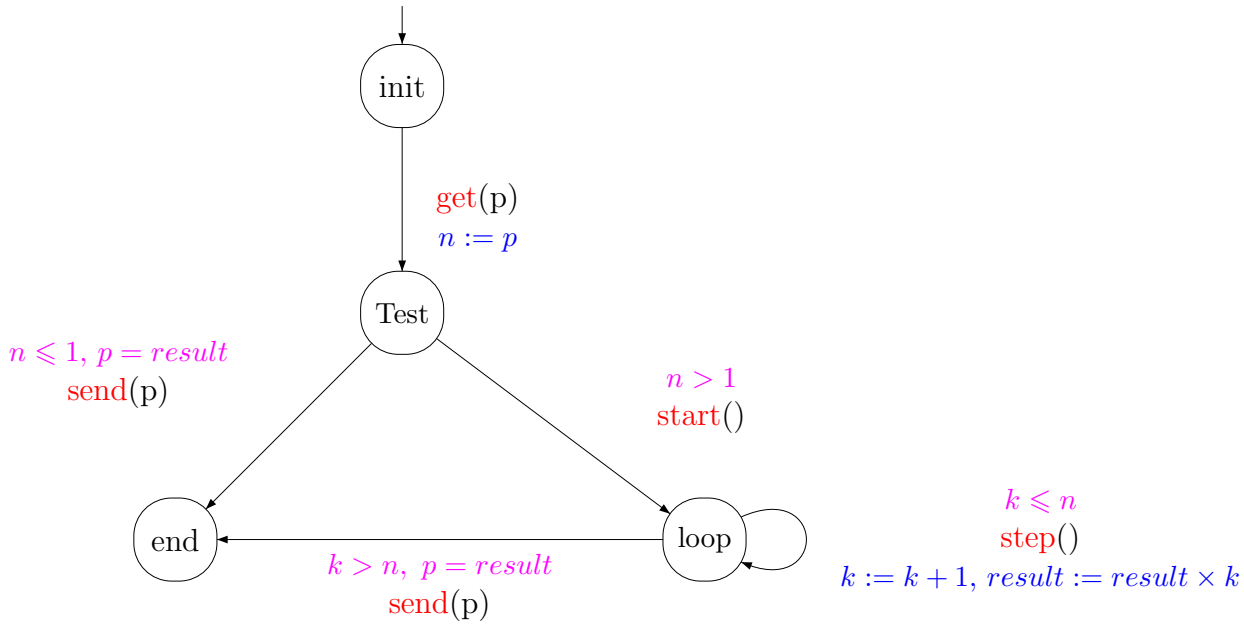
- *T is a finite set of transition. We note $[a, p, G, A] \in T$ written also $[a(p),\ G(\vec{q}, p),\ \vec{q} := A(\vec{q}, p)] \in T$, means that $a \in \Sigma$, $p \in P_a$, G is a guard over the current valuation $\vec{q}$ and the parameter p and A is an assignment of the variables also depending on $\vec{q}$ and p. We may write $[\alpha,\ G(\vec{q}),\ \vec{q} := A(\vec{q})] \in T$ for $\alpha \in \Sigma_{uo}$*

We assume that the guards are written in a theory where satisfiability is decidable.

Example :

This simple example shows how to formalise the function $n \mapsto n!$ with the STS model

$Fact = (V, \Theta, \Sigma, T)$, $L = \{l\}$, $Dom(l) = \{\text{init, test, loop, end }\}$,$V_p = \{n, tmp, result\}$, $Dom(n) = Dom(tmp) = Dom(result) = \mathbb{N}$, $\Theta : (l, n, tmp, result) = (\text{init}, 1, 1, 1)$, $\Sigma_o = \{$ get, send $\}$, $\Sigma_{uo} = \{$ step, start $\}$



2.3 Semantics of an STS

Definition 7 *The semantics of an STS $\mathcal{M} = (V, \Theta, \Sigma, T)$ is an LTS $[\mathcal{M}] = (Q, Q_0, \Lambda, \rightarrow)$ where :*

- $Q = Dom(V) = \times_{v \in V} Dom(v)$
- Q_0 *is the set of vectors of* Q *satisfying the initial condition* Θ. $Q_0 = \{\vec{q} \in Q,\ \vec{q} \models \Theta\}$
- $\rightarrow$ *is defined by the inference rule :*

$$\frac{\vec{q} \in Q, [a, p, G, A] \in T, G(\vec{q}, p), \vec{q'} = A(\vec{q}, p)}{(\vec{q}, (a, p), \vec{q'}) \in \rightarrow}$$

We note in the same way the corresponding notions of $\mathcal{M}$ and $[\mathcal{M}]$. So we have $\mathcal{R}(\mathcal{M}) \triangleq \mathcal{R}([\mathcal{M}])$, $\mathcal{L}(\mathcal{M}) \triangleq \mathcal{L}([\mathcal{M}])$, $\mathcal{T}(\mathcal{M}) \triangleq \mathcal{T}([\mathcal{M}])$. For $V' \subset V$, we extend the notation $\vec{q}_{|_{V'}}$ to the set of runs : for $\rho \in \mathcal{R}(\mathcal{M})$, $\rho = \vec{q_0} \stackrel{\alpha_1}{\rightarrow} \vec{q_1} \stackrel{\alpha_2}{\rightarrow} \cdots \stackrel{\alpha_n}{\rightarrow} \vec{q_n} \in \mathcal{R}(\mathcal{M})$, $\vec{q_0}, \vec{q_1}, \cdots \vec{q_n} \in Dom(V)$, $\alpha_0, \alpha_1, \cdots \alpha_n \in \Lambda$ then $\rho_{|_{V'}} = \vec{q_0}_{|_{V'}} \stackrel{\alpha_1}{\rightarrow} \vec{q_1}_{|_{V'}} \stackrel{\alpha_2}{\rightarrow} \cdots \stackrel{\alpha_n}{\rightarrow} \vec{q_n}_{|_{V'}}$

Chapter 3

Notion of Non-Interference

Here comes the main contribution of the project. Provided a secret given by the security policy, the purpose is to provide some methods to prove that there cannot be any flow of secret information. Using different model, automata, LTS and STS mostly, used to describe both the systems and eventually secrecy properties, we will see how this problem can be understood as a reachability problem. For this, we focus our work to the case the secrecy properties defined within an environment $\mathcal{E} = (\mathcal{V}, \Sigma)$ like properties over sets of runs. This framework allows us to cope with a large number of secrecy problems.

3.1 Some introductory examples

We give here some basic examples to explain the outlines of non-interference.

First example :

let M be a LTS with $\Lambda = \{h, p, l_1, l_2\}$, $\Lambda_o = \{l_1, l_2\}$. The security property claims that the event p has to be hidden to users. p is not an observable event. But with the knowledge of the system behaviour (i.e the complete graph of possibilities), users can infer that p has occurred by observing the event l_2. Such a system is then not secure because the occurence of p is interfering with what the users can see.

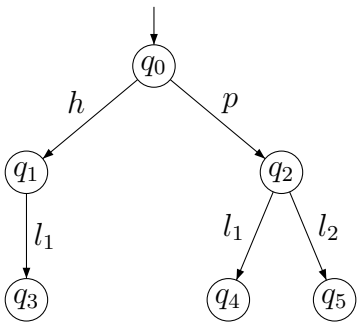


Figure 3.1: An example of interference

Second example :

The next example details a case of information flow in a coffee machine. We consider some robbers interested in gathering some money. For this goal they use to break the coffee machines but they only do it when they are sure whether it is full of cash. Technically the machines cannot accept money when it is already full and therefore give the money back. The Security Policy claims that users cannot access any information about the cash stack but a coffee machine following this policy can still be unsafe.

We define a **User** interacting with the actions {coin, confirm, cancel, coffee}. The **Coffee Machine** uses the alphabet :

- Λ_o = {coinIn, coinOut, confirm, cancel, coffeeOut}
- $\Lambda \backslash \Lambda_o$ = {isCashFull, cashFull, cashNotFull, isCoffeeEmpty, coffeeEmpty, coffeeNotEmpty}

In the example of figure 3.2, we see that the observation of **coinIn** $\cdot$ **coinOut** is characteristic of the occurrence of $full$. Indeed there is no other sequences without any occurrence of $full$ such that the projection on the observable actions gives **coinIn** $\cdot$ **coinOut**. Using this fact, robbers can infer that the machine is full of cash and this machine fail then to satisfy this secrecy policy because users can indirectly get information about the state of the cash stack.

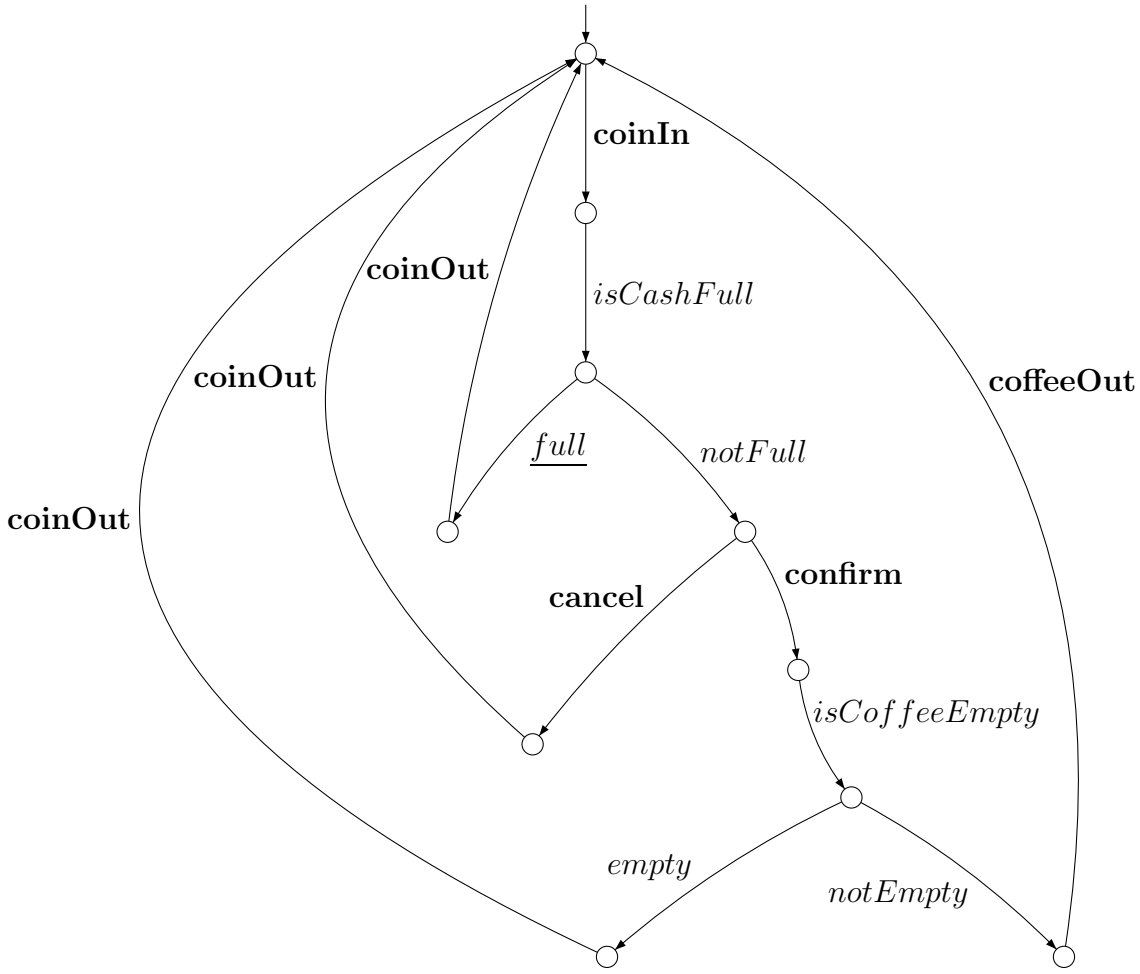


Figure 3.2: Unsafe coffee machine

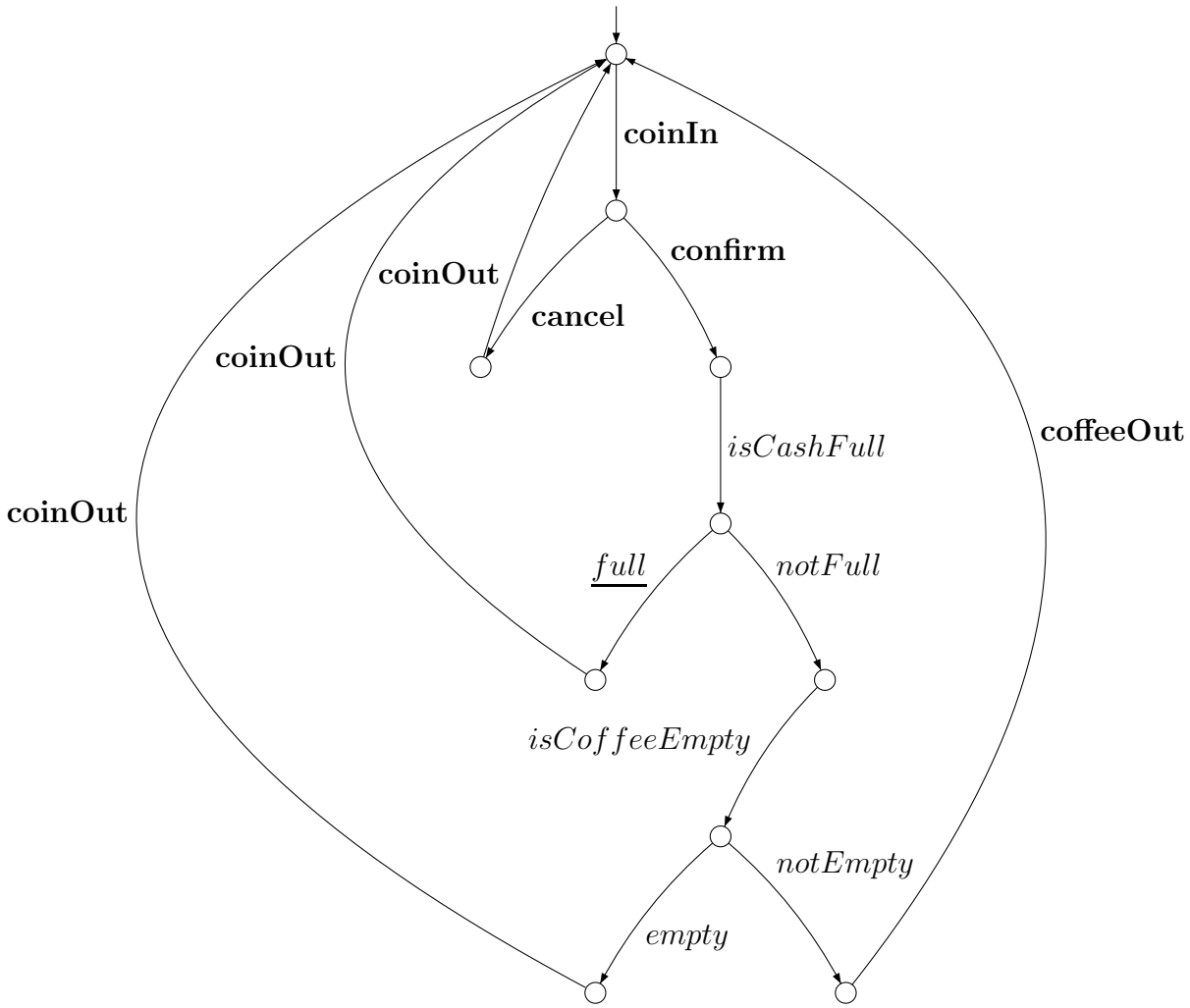


Figure 3.3: Non-interfering coffee machine

This model satisfy the secrecy policy because users cannot infer the occurrence of $full$ from their observations . Indeed, for every sequence where $full$ occurs, there is an other sequence without any occurrence of $full$ leading to the same observation. Users cannot infer with certainty that $full$ has occured. This points out the main idea for ensuring non-interference in a system. To hide the occurrence of a private action to the users, for each sequence containing secret actions, the system needs to provide an other possible sequence with the same trace but without any occurrence of secret

actions. In other words the system is secure if users cannot make any diagnosis of secret information. We will formalise this now.

3.2 Definition of Non-Interference

3.2.1 general definition

We consider in this section an environment $\mathcal{E} = (\mathcal{V}, \Sigma)$, $\Lambda = \Sigma_{uo} \cup \Lambda_o$ with $\Lambda_o = \cup\{a(p), a \in \Sigma_o, p \in P_a\}$, $R_\mathcal{E} = Dom(\mathcal{V}) \cdot (\Lambda \cdot Dom(\mathcal{V}))^*$ and a security property ψ defined over $\mathcal{P}(R_\mathcal{E})$. Let $\mathcal{M} = (V, \Sigma, \Theta, T)$ be a STS such that $\mathcal{V} \subset V$. We say that a run $\rho \in \mathcal{R}(\mathcal{M})$ is compatible with a trace $t \in \mathcal{T}(\mathcal{M})$ if $\pi_{\Lambda_o}(\rho) = t$ (i.e. $\rho \in \pi^{-1}_{\Lambda_o}(t) \cap \mathcal{R}(\mathcal{M})$)[1].

Definition 8 *A property ψ is interfering with a trace $t \in \mathcal{T}(\mathcal{M})$ if the set of all the runs is $\mathcal{M}$ compatibles with t is satisfying ψ. formally, we write :*

$$I_\psi(\mathcal{M}, t) : (\pi^{-1}_{\Lambda_o}(t) \cap \mathcal{R}(\mathcal{M}))_{|\mathcal{V}} \models \psi$$

Definition 9 *A system $\mathcal{M}$ is interfering for the property ψ if we have $\exists t \in \mathcal{T}(\mathcal{M})$, $I_\psi(\mathcal{M}, t)$. We write then $I_\psi(\mathcal{M})$ and $NI_\psi(\mathcal{M})$ if $\mathcal{M}$ is non-interfering.*

$$I_\psi(\mathcal{M}) : \exists t \in \mathcal{T}(\mathcal{M}),\ (\pi^{-1}_{\Lambda_o}(t) \cap \mathcal{R}(\mathcal{M}))_{|\mathcal{V}} \models \psi$$
$$NI_\psi(\mathcal{M}) : \forall t \in \mathcal{T}(\mathcal{M}),\ (\pi^{-1}_{\Lambda_o}(t) \cap \mathcal{R}(\mathcal{M}))_{|\mathcal{V}} \not\models \psi$$

3.2.2 Interference and diagnosis

This definition allow us to make a link with the definition of diagnosis given in [3] for failure diagnosis. The idea of diagnosis is to be able to infer with certainty the occurrence of an non observable behaviours. For example the occurrence of failure is typically not directly observable but we wish to be able to infer that a failure has happend. In [3], it is required that only a bounded number of observations is needed to be sure of the occurrence of failures. Given a property Ω over $R_\mathcal{E}$ we define ψ_Ω over $\mathcal{P}(R_\mathcal{E})$ by $\forall \tilde{\rho} \in \mathcal{P}(R_\mathcal{E}),\ \tilde{\rho} \models \psi_\Omega \triangleq \forall \rho \in \tilde{\rho},\ \rho \models \Omega$.

[1] This imply that there is no internal loop in $\mathcal{M}$ otherwise, $\pi^{-1}_{\Lambda_o}(t) \cap \mathcal{R}(\mathcal{M})$ may be infinite. The definition of internal loop is given in chapter 4

Definition 10 *A STS $\mathcal{M}$ is Ω-diagnosable, and we write $D_\Omega(\mathcal{M})$ if $\exists n \in \mathbb{N}$ such that :*

$$\forall \rho \in \mathcal{R}(\mathcal{M}),\ \rho \models \Omega \ \Rightarrow\ \forall t \in \Lambda_o^*, |t| \geqslant n, \pi_{\Lambda_o}(\rho)\cdot t \in \mathcal{T}(\mathcal{M}),\ I_\psi(\mathcal{M}, \pi_{\Lambda_o}(\rho)\cdot t)$$

$|t|$ denotes here the Length of the trace t. According to this definition, a system is Ω-diagnosticable if the fact that an run is satisfying Ω is interfering with the observation.

Chapter 4

Model transformations

Definition 11 *A state $\vec{q} \in Dom(V)$ is reachable if there is a sequence $s \in \Lambda^*$ such that $q_0 \xrightarrow{s} q$. We write $Reach(\mathcal{M})$ the set of reachable states.*

Property 1 *It is not decidable in general to know if a given state $\vec{q}$ is reachable in a STS.*

We will find more precisions of this problem on this article [9].

4.1 Composition of two STS

The synchronous parallel composition of two STS is a way to model processes synchronisation on common actions. We assume the processes do not share memory (variables) for composition.

Definition 12 *The composition of two STS $\mathcal{M}_1 = (V_1, \Theta_1, \Sigma, T_1)$ and $\mathcal{M}_2 = (V_2, \Theta_2, \Sigma, T_2)$, is the STS $\mathcal{M}_1 \parallel \mathcal{M}_2 = (V, \Theta, \Sigma, T)$ where :*

- *$V = V_1 \biguplus V_2$ is the disjoint union[1] of V_1 and V_2*
- *Θ is defined by :*

$$\vec{q} = \langle \vec{q_1}, \vec{q_2} \rangle \in Dom(V),\ \frac{\vec{q_1} \models \Theta_1,\ \vec{q_2} \models \Theta_2}{\vec{q} \models \Theta}$$

[1] meaning that $V = V_1 \cup V_2$ and $V_1 \cap V_2 = \emptyset$

- *T is defined by the inference rules :*

  $\vec{q_1} \in Dom(V_1)$, $\vec{q_2} \in Dom(V_2)$, $\vec{q} = \langle \vec{q_1}, \vec{q_2} \rangle \in Dom(V)$,

$$\frac{a \in \Sigma_o,\ [a(p),\ G_1(\vec{q_1}, p),\ \vec{q_1'} := A_1(\vec{q_1}, p)] \in T_1,\ [a(p),\ G_2(\vec{q_2}, p),\ \vec{q_2'} := A_2(\vec{q_2}, p)] \in T_2}{[a(p),\ G_1(\vec{q_1}, p) \wedge G_2(\vec{q_2}, p),\ \langle \vec{q_1'}, \vec{q_2'} \rangle := \langle A_1(\vec{q_1}, p),\ A_2(\vec{q_2}, p) \rangle] \in T}$$

$$\alpha \in \Sigma_{uo},\ i, j \in \{1, 2\}, i \neq j,\ \frac{[\alpha,\ G_i(\vec{q_i}),\ \vec{q_i'} := A_i(\vec{q_i})] \in T_i}{[\alpha,\ G_i(\vec{q_i}),\ \langle \vec{q_i'}, \vec{q_j'} \rangle := \langle A_i(\vec{q_i}), \vec{q_j} \rangle] \in T}$$

Property 2 $\mathcal{T}(\mathcal{M}_1 \parallel \mathcal{M}_2) = \mathcal{T}(\mathcal{M}_1) \cap \mathcal{T}(\mathcal{M}_2)$

4.2 Product of two STS

The synchronous product is an other way of creating a system based on the interaction of two others. The definition of product is based on the possibility to share the variables as well as all the actions, observable and internal.

Definition 13 *An STS $\mathcal{M} = (V_\mathcal{M}, \Theta_\mathcal{M}, \Sigma, T_\mathcal{M})$ is compatible for product with STS $\mathcal{N} = (V_\mathcal{N}, \Theta_\mathcal{N}, \Sigma, T_\mathcal{N})$ if :*

- $V_\mathcal{M} \subset V_\mathcal{N}$
- *The initial condition $\Theta_\mathcal{N}$ doesn't depend of the valuation of the variables $V_\mathcal{M}$*
- *The assignments of $T_\mathcal{N}$ only modify the valuations of the values $V_\mathcal{N} \backslash V_\mathcal{M}$*

Definition 14 *The synchronous product of two STS $\mathcal{M} = (V_\mathcal{M}, \Theta_\mathcal{M}, \Sigma, T_\mathcal{M})$ and $\mathcal{N} = (V_\mathcal{N}, \Theta_\mathcal{N}, \Sigma, T_\mathcal{N})$ is a STS $\mathcal{M} \rtimes \mathcal{N} = (V, \Theta, \Sigma, T)$ such that :*

- $V = V_\mathcal{N}$
- $\forall \vec{q} \in Dom(N)$, $\vec{q} \models \Theta$ *if* $\vec{q} \models \Theta_\mathcal{N}$ *and* $\vec{q_{|V_\mathcal{M}}} \models \Theta_\mathcal{M}$
- *if* $[a, p, G_\mathcal{M}, A_\mathcal{M}] \in T_\mathcal{M}$ *and* $[a, p, G_\mathcal{N}, A_\mathcal{N}] \in T_\mathcal{N}$ *then* $[a, p, G, A] \in T$ *with G and A defined by :* $\forall \vec{q} \in Dom(V),\ \forall p \in P_a$

– $G(\vec{q},p) \triangleq G_{\mathcal{N}}(\vec{q},p) \wedge G_{\mathcal{M}}(q_{|\vec{V_{\mathcal{M}}}},p)$

– $A(\vec{q},p) \triangleq \langle A_{\mathcal{M}}(q_{|\vec{V_{\mathcal{M}}}},p), (A_{\mathcal{N}}(\vec{q},p))_{|V_{\mathcal{N}}\setminus V_{\mathcal{M}}}\rangle$

Definition 15 *We say that $\mathcal{N}$ is nonintrusive for product if for all STS $\mathcal{M}$ compatible for product with $\mathcal{N}$, the behaviours of $\mathcal{M}$ are preserved after product. In other words : $\forall \rho = \vec{q} \xrightarrow{s} \vec{q'} \in \mathcal{R}(\mathcal{M}),\ \exists \vec{q_{\mathcal{N}}}, \vec{q'_{\mathcal{N}}} \in Dom(V_{\mathcal{N}} \setminus V_{\mathcal{M}}),\ \langle q, q_{\mathcal{N}}\rangle \xrightarrow{s} \langle q', q'_{\mathcal{N}}\rangle \in \mathcal{R}(\mathcal{M})$.*

4.3 ε-closure of an STS

With the idea that calculating the ε-closure of an automaton, we give here one method to get rid of the internal actions in an STS. Starting from a system $\mathcal{M}$, the problem is to compute a new system $\varepsilon(\mathcal{M})$ such that $\mathcal{L}(\varepsilon(\mathcal{M})) = \mathcal{T}(\varepsilon(\mathcal{M})) = \mathcal{T}(\mathcal{M})$ and $\varepsilon(\mathcal{M})$ doesn't have any internal transition.

A loop in $\mathcal{M}$ is a run $\rho = \vec{q} \xrightarrow{s} \vec{q'},\ s \in \Lambda^*$ such that $\vec{q_{|_L}}$ and $\vec{q'_{|_L}}$. An internal loop is a loop ρ such that $\pi(\rho) = \varepsilon$. We assume in the following that all the systems we consider do not have internal loop.

Definition 16 *For an STS $\mathcal{M}$ without internal loop, the symbolic ε-closure of $M = (V, \Theta, \Sigma, T)$ is an STS noted $\varepsilon(\mathcal{M}) = (V, \Theta, \Sigma_o, T_\varepsilon)$ with T defined by :*

$$\vec{q_0}, \vec{q_1}, \cdots \vec{q_n} \in Dom(V),\ \alpha_1\alpha_2\cdots\alpha_{n-1} \in \Sigma_{uo}^*,\ a \in \Sigma_o,$$

$$\frac{(\forall i \in [\![1, n-1]\!],\ [\alpha_i,\ G_{\alpha_i}(\vec{q_{i-1}}),\ \vec{q_i} := A_{\alpha_i}(\vec{q_{i-1}})] \in T),\ [a(p),\ G(\vec{q_{n-1}}, p),\ \vec{q_n} := A(\vec{q_{n-1}}, p)] \in T}{[a(p),\ G_\varepsilon(\vec{q_0}, p),\ \vec{q_n} := A_\varepsilon(\vec{q_0}, p)] \in T_\varepsilon}$$

where $G_\varepsilon(\vec{q_0}, p) = G_{\alpha_0}(\vec{q_0}) \wedge G_{\alpha_1} \circ A_{\alpha_0}(\vec{q_0}) \wedge \cdots \wedge G_{\alpha_{n-1}} \circ A_{\alpha_{n-2}} \circ \cdots \circ A_{\alpha_0}(\vec{q_0}) \wedge G(A_{\alpha_{n-1}} \circ \cdots \circ A_{\alpha_0}(\vec{q_0}), p)$ and $A_\varepsilon(\vec{q_0}, p) = A(A_{\alpha_{n-1}} \circ \cdots \circ A_{\alpha_0}(\vec{q_0}), p)$.

This means that $\vec{q_0} \xrightarrow{\alpha} \vec{q_1} \xrightarrow{a(p)} \vec{q_2}$ in $\mathcal{M}$ becomes $\vec{q_0} \xrightarrow{a(p)} \vec{q_2}$ in $\varepsilon(\mathcal{M})$.

Example :

The procedure for calculating the ε-closure does terminate since the transition set is finite according to the definition of STS.

For $\vec{q} \in Dom(V), t \in \mathcal{T}(\mathcal{M})$, we note with $\Delta_{\mathcal{M}}(\vec{q}, t)$ the set of valuations reachable with the observation t passing from the state $\vec{q}$. $\Delta_{\mathcal{M}}(\vec{q}, t) = \{\vec{q'} \in Dom(V),\ \exists s \in \mathcal{L}(\mathcal{M}),\ s = s_1 s_2,\ \pi_{\Lambda_o}(s) = t\ \wedge\ \vec{q_0} \xrightarrow{s_1} \vec{q} \xrightarrow{s_2} \vec{q'}\}$.

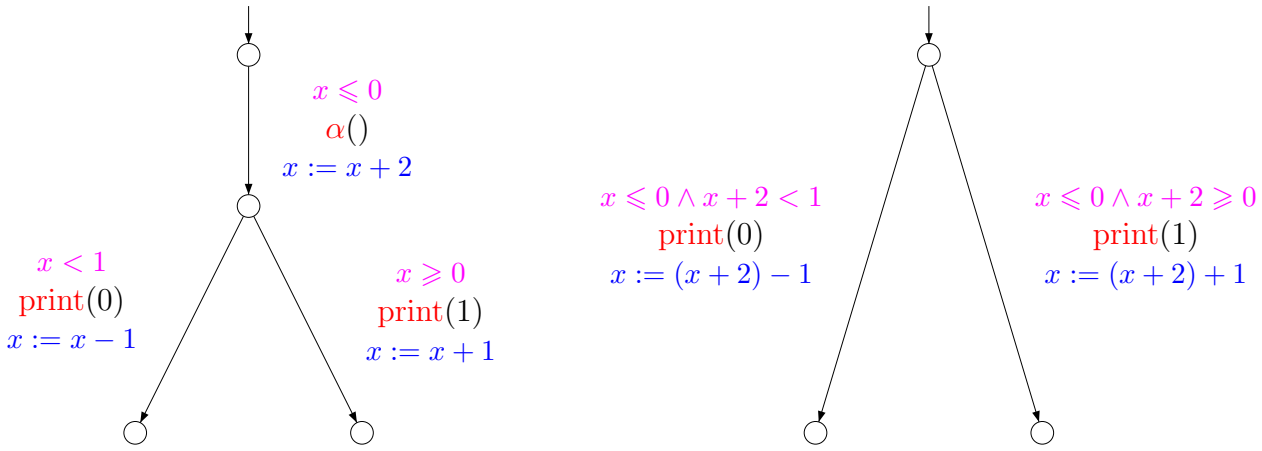


Figure 4.1: Calculating the ε-closure of a STS

4.4 Determination of an STS

Definition 17 *An STS $\mathcal{M}$ is said to be deterministic if the LTS $[\mathcal{M}]$ is deterministic.*

This is the natural definition expressing the idea that there is no ambiguity while running the STS. But checking if $[\mathcal{M}]$ is deterministic is undecidable in general because the reachability of one particular set is undecidable and it is then impossible to check if $[\mathcal{M}]$ is deterministic for every state $\vec{q}$. To avoid this we give a less restrictive notion called syntactical determinism.

For $l \in Dom(L)$ we define $Next(l, a) = \{[a, p, G, A] \in T,\ \exists(\vec{q}, p) \in Dom(V) \times P_a,\ \vec{q_{|_L}} = l \wedge G(\vec{q}, p)$ holds$\}$

Definition 18 *An STS $\mathcal{M}$ is syntactically deterministic for $l \in Dom(L)$ and $a \in \Sigma$ if for each pair of transitions $t_1, t_2 \in Next(l, a)$ with guards G_1 and G_2, we have $t_1 = t_2$ or the predicate $G_1 \wedge G_2$ is unsatisfiable.*

Property 3 *If a STS $\mathcal{M}$ is syntactically deterministic, then $\mathcal{M}$ is deterministic*

proof : We assume that the STS $\mathcal{M}$ is syntactically deterministic. let $\vec{q}, \vec{q_1}, \vec{q_2} \in Dom(V)$ such that $\vec{q} \overset{a(p)}{\rightarrow} \vec{q_1}$ and $\vec{q} \overset{a(p)}{\rightarrow} \vec{q_2}$ in $[\mathcal{M}]$. Let $t_1, t_2 \in$

$Next(\vec{q_{|_L}})$, $t_1 = [a, p, G_1, A_1]$, $t_2 = [a, p, G_2, A_2]$. $\vec{q_1} := A_1(\vec{q}, p) \neq \vec{q_2} := A_2(\vec{q}, p)$ implies that $A_1 \neq A_2$ and then that $t_1 \neq t_2$. According to the definition of a syntactically deterministic STS we also have in that case that $G_1 \wedge G_2$ cannot be satisfied and this face an impossibility. So we have $\vec{q_1} = \vec{q_2}$. We conclude that $[\mathcal{M}]$ is a deterministic LTS and hence that $\mathcal{M}$ is deterministic.

There is a terminating procedure to check the syntactical determinism of a STS because $Dom(L)$ and T are finite sets. In the following, we will only deal with syntactical determinism calling it determinism for simplification.

Definition 19 *An LTS $M = (Q, Q_0, \Lambda, \rightarrow)$ is deterministic in $q \in Q$ with lookahead k if $\forall a \in \Sigma$, $\forall s \in \Lambda^k$, we have $q \xrightarrow{a} q_1 \xrightarrow{s} \wedge\ q \xrightarrow{a} q_2 \xrightarrow{s} \ \Rightarrow q_1 = q_2$.*

An LTS is deterministic with lookahead k when for a local non-determinim $q \xrightarrow{a} q_1 \wedge q \xrightarrow{a} q_2$ we can decide which state q_1 or q_2 the system is after the observation of a by observing a sequence of length at most k after a.

Definition 20 *An STS $\mathcal{M}$ has lookahead k in a location $l \in Dom(L)$ if $\forall \vec{q} \in Dom(V)$, $\vec{q_{|_L}} = l$, $[\mathcal{M}]$ is deterministic with lookahead k in $\vec{q}$. When it exists, $look(l, \mathcal{M})$ denotes the smallest lookahead in l and we extend with $look(l, \mathcal{M}) := \infty$ when such a lookahead doesn't exist. We also define $look(\mathcal{M}) := max\{look(l, \mathcal{M}),\ l \in L\}$. $look(\mathcal{M}) \in \mathbb{N} \cup \{\infty\}$*

Theorem 1 *An STS is deterministic in l iff $look(l, \mathcal{M}) = 0$ and $\mathcal{M}$ is deterministic iff $look(\mathcal{M}) = 0$*

Now we will study the class of STS for which we can compute a STS $det(\mathcal{M})$ of $\mathcal{M}$ such that $\mathcal{L}(det(\mathcal{M})) = \mathcal{L}(\mathcal{M})$ and $det(\mathcal{M})$ is deterministic. More complete explanations of this can be found in [10]. This calculation will be usefull for deciding if it is possible to gather some secret information about a system through the observable actions. The main idea of calculating $det(\mathcal{M})$ is to postpone the guards and the assignments until we know the branches the runs has been. This example describes the procedure :

The sign $\langle l_0, \langle l_1, l_2 \rangle \rangle$ means that we keep the information that after the occurrence of a the run can have been in location l_1 or l_2. But the occurrence of b or c allow us to distinguish between the two possibilities and then run the appropriate guards and assignments. Also, to keep the informations contained in the parameters, we add some variables to $det(\mathcal{M})$ used to store this parameters in order to modify correctly the transition coming after.

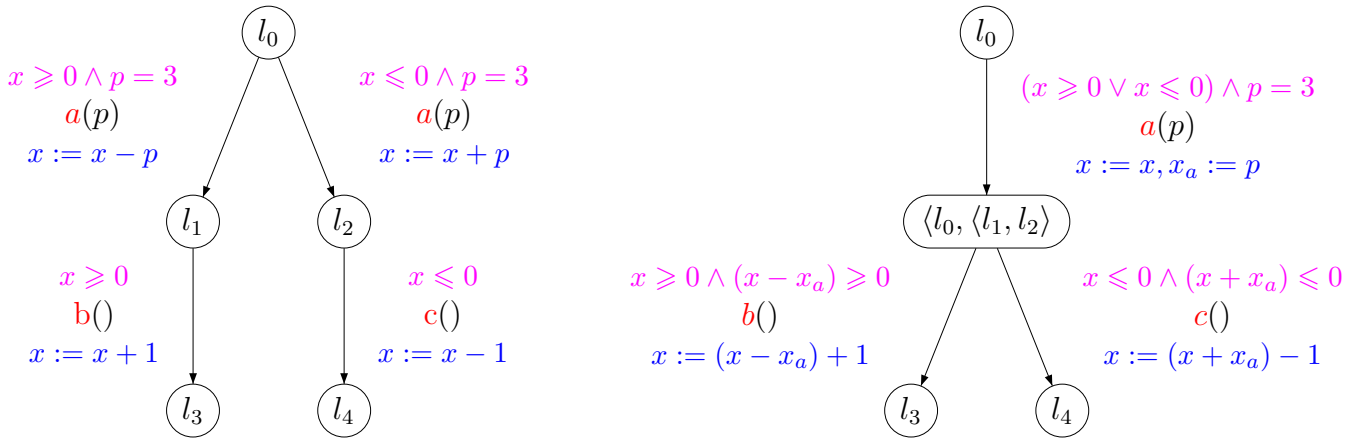


Figure 4.2: Basic step for calculating $det(\mathcal{M})$

Theorem 2 *For every STS $\mathcal{M}$, there is a procedure to calculate $det(\mathcal{M})$ if $look(\mathcal{M}) \neq \infty$.*

A complete poof of this theorem using the ideas described above can be found in [10].

Chapter 5

Checking Non-Interference

5.1 Case of finite automata

We will study now the case of finite systems and properties dealing with the occurrences of actions. In this context the environment is $\mathcal{E} = (\emptyset, \Lambda)$ with $\Lambda = \Lambda_o \cup \Lambda_{uo}$ a finite set of actions and the projection $\pi_{\Lambda_o} : \Lambda^* \to \Lambda_o^*$. Let $M = (Q, \Lambda, Q_0, \to)$ be a finite LTS. Let ψ be a property over $\mathcal{P}(\Lambda^*)$. We wish to create a function giving if we have $I_\psi(M, t)$ or $\neg I_\psi(M, t)$ for $t \in \mathcal{T}(M)$. For this we consider the LTS $\chi_\psi(M) = (\mathcal{Q}, \Lambda_o, \mathcal{Q}_0, \to_\chi)$ with $Q_\chi \subset \mathcal{P}(Q) \times \Lambda^*$, $\mathcal{Q}_0 = (Q_0, \emptyset)$ and $\to_\chi$ defined by :

$$\frac{(\tilde{q}, \tilde{s}) \in \mathcal{Q},\ t \in \mathcal{T}(M)}{(\tilde{q}, \tilde{s}) \to_\chi (\Delta_M(\tilde{q}, t), \pi_{\Lambda_o}^{-1}(t) \cap \mathcal{L}(M))}$$

We define in $\chi_\psi(M)$ a set of final state $\mathcal{Q}_F = \{(\tilde{q}, \tilde{s}) \in \mathcal{Q},\ \tilde{s} \models \psi\}$. $\chi_\psi(M)$ is a deterministic LTS such that $\mathcal{L}(\chi_\psi(M), \mathcal{Q}_F)$ is the set of traces $t \in \mathcal{T}(M),\ I_\psi(M, t)$. In the general case $\chi_\psi(M)$ is not finite and it is then not decidable to know if $\mathcal{Q}_F$ is reachable or not. Hence, it is undecidable to know if $I\psi(M)$ or $NI\psi(M)$. We will study now some cases where we can give a construction such than $\chi_\psi(M)$ such that the question $\mathcal{L}(\chi_\psi(M), \mathcal{Q}_F) \stackrel{?}{=} \emptyset$ is decidable.

Case of a property given by a regular language

We consider In this section the case where the secrecy can be described like a property[1] φ over the set of sequences Λ^*. This property φ is satisfied when the secrecy is violated. To fit with the general definition, we define $\psi : \forall \tilde{s} \in \mathcal{P}(\Lambda^*),\ \tilde{s} \models \psi \stackrel{\Delta}{=} \forall s \in \tilde{s},\ s \models \varphi$. We will later examine the case of secrecy properties given by regular languages. An observer can deduce the secret throw its observation $t \in \mathcal{T}(M)$ when all the sequences in $\mathcal{L}(M)$ are violating the secret.

Proposition 1 *For a finite LTS $M = (Q, \Lambda, Q_0 \rightarrow)$ with $\Lambda_0 \subset \Lambda$ the observable actions and φ a secrecy property over Λ^* we have :*

$$I_\psi(M) \Leftrightarrow \exists t \in \mathcal{T}(M),\ \forall s \in \pi^{-1}(t) \cap \mathcal{L}(M),\ s \models \varphi$$

If the property φ is given by a regular language L_φ then we can derive the non-interference problems to some other more easily computable problems within the automata theory. Then, we will see that I_ψ can be converted to a reachability problem.

Proposition 2 *let φ be a property given by a regular language L_φ, we can then express the notion of interference by :*

$$I_\psi(M) \Leftrightarrow \exists t \in \mathcal{T}(M),\ \pi^{-1}(t) \cap \mathcal{L}(M) \subseteq L_\varphi$$

Now, let $A_\varphi = (Q^\varphi, \Lambda, Q_0^\varphi, Q_f^\varphi, \rightarrow)$ be a deterministic complete automaton such that $\mathcal{L}(A_\varphi, Q_f^\varphi) = L_\varphi$ and $\mathcal{L}(A_\varphi, Q^\varphi \setminus Q_f^\varphi) = L_{\bar{\varphi}}$. $M \times A_\varphi = (Q \times Q^\varphi, \Lambda, Q_0 \times Q_0^\varphi, Q \times Q_f^\varphi, \rightarrow)$ given with the classical product of two automata.

Proposition 3 *With the previous definition of M and A_φ we have :*

$$NI_\psi(M) \Leftrightarrow \pi_{\Lambda_o}(\mathcal{L}(M \times A_\varphi, (Q \times Q_f^\varphi) \setminus (Q \times Q_f^\varphi))) = \mathcal{T}(M)$$

proof : Let $K = \mathcal{L}(M \times A_\varphi, (Q \times Q_f^\varphi) \setminus (Q \times Q_f^\varphi))$. K is the set of sequences of M not violating the secrecy policy. Provided $t \in \mathcal{T}(M)$, we assume that we have $\pi_{\Lambda_o}(K) = \mathcal{T}(M)$. In that case $t \in \pi_{\Lambda_o}(K)$ and there is $s \in \mathcal{L}(M \times A_\varphi),\ \pi(s) = t$ such that $s \in L_{\bar{\varphi}}$. Hence we have $s \in \pi^{-1}(t) \cap \mathcal{L}(M),\ s \not\models \varphi$. This prove $NI_\psi(M)$. In the other sense,

[1] We assume this property to be decidable.

we assume $NI_\psi(M)$ and let $t \in \mathcal{T}(M)$. According to the definition of NI and proposition 2 , we have $\exists s \in \pi^{-1}(t) \cap \mathcal{L}(M) \cap \bar{L_\varphi}$. But we have $K = \mathcal{L}(M) \cap \mathcal{L}(A_\varphi, Q^\varphi \setminus Q_f^\varphi) = \mathcal{L}(M) \cap \bar{L_\varphi}$ and then $\pi(s) = t \in \pi_{\Lambda_o}(K)$. Since $\pi_{\Lambda_o}(K) \subseteq \pi_{\Lambda_o}(\mathcal{L}(M)) = \mathcal{T}(M)$, we have the equivalence.

For example, for the coffee machine given by the figure 3.2, the secrecy property is given by $s \models \varphi : s \in (\Lambda^* \; full \; \Lambda^*)$. We have[2] $\pi_{\Lambda_o}(K) = (prefix(\{coinIn \cdot (confirm \cdot coffeeOut \; + \; cancel \cdot coinOut))\}))^*$ but with $t = coinIn \cdot coinOut$, $t \in \mathcal{T}(M)$ and $t \notin \pi_{\Lambda_o}(K)$, hence, according to the proposition 3, this model is interfering. Using the same argument, one can prove that the model given in figure 3.3 is non-interfering.

This proposition gives a way to prove the non-interference by checking a languages inclusion. Generally, checking a language inclusion leads to a state reachability problem. We will see how the non-interference problem can be solved by checking reachability in a finite automata. Let $\chi_M \stackrel{\Delta}{=} det(\varepsilon(M \times A_\varphi))$ with the set of final states $\mathcal{Q}_F \stackrel{\Delta}{=} \{\tilde{q} \in \mathcal{P}(Q \times Q^\varphi, \forall q = (q_M, q_{A_\varphi}) \in Q \times Q^\varphi, \; q_{A_\varphi} \in Q_f^\varphi\}$. The operators det and ε are the classical determinisation and ε-closure operator for finite automata.

Theorem 3 *Using the previous notation, we have :*

$$NI_\psi(M) \Leftrightarrow \mathcal{L}(\chi_\psi(M), \mathcal{Q}_F) = \emptyset$$

Proof : The proof is directly comming form the definition of $\chi_\psi(M)$: a trace t of M leading to $\mathcal{Q}_F$) in $\chi_\psi(M)$ is a trace such that all the sequences in M compatibles with t are satisfying the property φ.

Example of calculation of $\chi_\psi(M)$

For a property P over a set X, we identify the property and subset of X where the property is satisfied. We write then P for $\{x \in X \mid x \models P\}$ and $\bar{P} = X \setminus P$.

[2]The operator $prefix \; : \Lambda^* \to \mathcal{P}(\Lambda^*)$ gives the set of prefixes of a word $\omega \in \Lambda^*$.

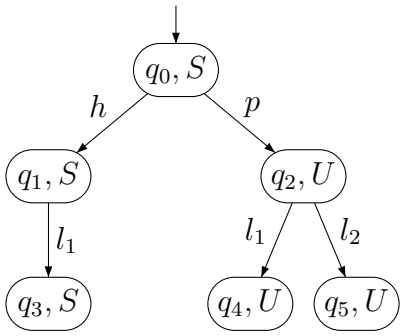


Figure 5.1: Example of calculation of $\chi_\psi(M)$

5.2 Extention of non-interference to the STS model

We consider now the case of a security properties described with STS. Obviously, this approach cannot deal with everything we can expect. For example, it is difficult with this approach to deal with security properties like "Users cannot infer any information about the variable X". But on the other hand the STS model will be suitable a to deal with properties consering particular system behaviours. For Instance "The server protocol must disconnect a client sending a sequence of action characteristic of an attack". Let $\mathcal{E} = (\mathcal{V}, \Sigma)$ be an environment. The secrecy property φ is given by the nonintrusive STS $P_\varphi = (V_\varphi, \Theta_\varphi, \Sigma, T_\varphi)$ with $\mathcal{V} \subset V_\varphi$ and such that the set of locations $Dom(l_\varphi) = S \cup U,\ S \cap U = \emptyset$. S, for "safe", is the set of locations where the secrecy is not violated, i.e. φ is not satisfied. And U, for "unsafe" , is the set of locations where the secrecy is violated. We write here $\mathcal{R}(P_\varphi, U) = \{\rho = \vec{q_0} \xrightarrow{s} \vec{q} \in \mathcal{R}(P_\varphi),\ \vec{q_0} \models \Theta_\varphi,\ \vec{q_{|_{l_\varphi}}} \in U\}$ the set of runs violating the secrecy. $\mathcal{R}(P_\varphi, S)$ is defined in the same way and we have $\mathcal{R}(P_\varphi) = \mathcal{R}(P_\varphi, U) \cup \mathcal{R}(P_\varphi, S)$. Also, the property φ gives the property $\psi : \tilde{\rho} \in R_\mathcal{E} \cdot (\Sigma \cdot Dom(V_\varphi))^*, \tilde{\rho} \models \psi$ iff $\forall \rho \in \tilde{\rho},\ \rho \models \varphi$. In other words, using the STS P_φ, $\tilde{\rho} \models \psi$ if and only if all the runs $\rho \in \tilde{\rho}$ are ending in a location labeled as unsafe, i.e. $\tilde{\rho} \models \psi \Leftrightarrow \forall \rho = \vec{q_0} \xrightarrow{s} \vec{q} \in \tilde{\rho},\ \vec{q}_{|_{l_\varphi}} \in U$. Let now $\mathcal{M} = (V, \Sigma, \Theta, T)$ be a STS compatible for product with P_φ and such that $\mathcal{V} \subset V \subset V_\varphi$. We assume that there is no internal loop in the product $\mathcal{M} \rtimes P_\varphi$ and that $look(\mathcal{M} \rtimes P_\varphi) \neq \infty$. With this assuptions, we can calculate $\chi_\psi(\mathcal{M}) = det(\varepsilon(\mathcal{M} \rtimes P_\varphi))$. Let $\Theta_F^\varphi \triangleq \{\tilde{q} \in \mathcal{P}(Dom(V_\varphi)),\ \forall \vec{q} \in \tilde{\rho}, \vec{q}_{|_{l_\varphi}} \in U\}$.

Θ_F^φ is the the set of macro-states constituted of states reached by violating the secrecy property. This construction of $\chi_\psi(\mathcal{M})$ gives the important result:

Theorem 4 *With the hypothesis given above, for every system $\mathcal{M}$ compatible for product with P_φ such that we can compute $\chi_\psi(\mathcal{M}) = det(\varepsilon(\mathcal{M} \rtimes P_\varphi))$, we have :*

$$NI_\psi(\mathcal{M}) \Leftrightarrow \mathcal{L}(\chi_\varphi(\mathcal{M}), \Theta_F^\varphi) = \emptyset$$

and, generally, $\mathcal{L}(\chi_\varphi(\mathcal{M}), \Theta_F^\varphi)$ is the set of traces an attacker can use to infer secret information.

As we have seen in the first chapter, the Symbolic Transition System has to face with different problems since reachability may be undecidable. Thanks to the definition of syntactical determinism, we have a strict subclass of STS for which determinism is decidable. However, it is still undecidable to know if $\mathcal{L}(\chi_\varphi(\mathcal{M}), \Theta_F^\varphi)$ is empty or not, because it also deals with checking the reachability of the macro-state Θ_F^φ in $\chi_\varphi(\mathcal{M})$. Using the same idea that presented in [9], we will solve this problem by approximate analysis using an overapproximation $\chi_\varphi(\mathcal{M})^\alpha$ where the reachability of Θ_F^φ is decidable. Because of the overapproximation, we loose the completness of the algorithms but, for some cases, we can still give an answer with certainty to the NI problem:

Theorem 5 *Using the same notation than the theorem 4, we have :*

$$\mathcal{L}(\chi_\varphi(\mathcal{M})^\alpha, \Theta_F^\varphi) = \emptyset \;\Rightarrow\; NI(\mathcal{M}, \varphi)$$

and if $t \in \mathcal{T}(\mathcal{M})$ is a real attack in $\mathcal{M}$, then $t \in \mathcal{L}(\chi_\varphi(\mathcal{M})^\alpha, \Theta_F^\varphi)$

Proof : By definition of overapproximation, we have $\mathcal{R}(\chi_\varphi(\mathcal{M})) \subseteq \mathcal{R}(\chi_\varphi(\mathcal{M})^\alpha)$ and then $\mathcal{L}(\chi_\varphi(\mathcal{M}), \Theta_F^\varphi) \subseteq \mathcal{L}(\chi_\varphi(\mathcal{M})^\alpha, \Theta_F^\varphi)$. So, according to the theorem 4, $\mathcal{L}(\chi_\varphi(\mathcal{M})^\alpha, \Theta_F^\varphi) = \emptyset \Rightarrow \mathcal{L}(\chi_\varphi(\mathcal{M}), \Theta_F^\varphi) \Rightarrow NI_\psi(\mathcal{M})$. Also, if there is a attack $t \in \mathcal{T}(\mathcal{M})$, $I_\psi(\mathcal{M}, t)$ then $t \in \mathcal{L}(\chi_\varphi(\mathcal{M}), \Theta_F^\varphi) \subset \mathcal{L}(\chi_\varphi(\mathcal{M})^\alpha, \Theta_F^\varphi)$. Finally, $\mathcal{L}(\chi_\varphi(\mathcal{M})^\alpha, \Theta_F^\varphi)$ gives a set of potential attacks but we have to check if they correspond to real real traces of $\mathcal{M}$.

Conclusion

This project has been the occasion to propose a definition of interference general enough to model most of the security properties we can find in the literature. Nevertheless, with this definition, the interference problem is often becoming intractable because of the explosion of possibilities and, in this report, we had to make more restrictive assumption about the security properties derive an easier problem. Indeed, we gave computable solutions for properties given by automata or extended automata but, even if there are interesting security policies which can be modelled in that way, we do not know if this approach is relevant for practical case. Furthermore, it can be interesting to extend the class of properties ψ over the set of runs for which the calculation of χ_ψ terminates. Further work can be done in the way to make an abstraction a the set of runs $\tilde{\rho}$ keeping only the relevant informations and merging similar states. This is basically what is done when the property is given by an automata because the important informations about the past or the future of one sequence are given by the automata. To follow this idea, it can be attractive to study security properties given by Horn logic for example. Also, an other possible future work can be to see how this notion of non-interference and the related algorithms can be implemented in the case of real communicating process framework like CORBA for example.

Bibliography

[1] S. Lafortune K. Sinaamohideen M. Sampath, R. Sengupta and D. Teneketzis. Diagnosability of discrete event systems. *IEEE Transactions on Automatic Control*, 1995.

[2] S. Lafortune K. Sinaamohideen M. Sampath, R. Sengupta and D. Teneketzis. Failure diagnosis using discrete event models. *IEEE Transactions on Control System Technology*, 1996.

[3] T. Jéron, H. Marchand, S. Pinchinat, and M-O. Cordier. Supervision patterns in discrete event systems diagnosis. Technical Report 1784, IRISA, February 2006.

[4] Butler W. Lampson. A user machine in a time-sharing system. *Proc. IEEE 54*, 1966.

[5] David D.E. Bell and L.J. La Padula. Secure computer system: Unified exposition and multics interpretation.

[6] JD Ullman MA Harrison, WL Ruzzo. Protection in operating systems. *Communications of the ACM*, 1976.

[7] J. A. Goguen and J. Meseguer. Security policies and security models. pages 11–20, April 1982.

[8] Riccardo Focardi and Roberto Gorrieri. An information flow security property for ccs. In *Proceedings of Second North American Process Algebra Workshop*, 1993.

[9] B. Jeannet, T. Jéron, V. Rusu, and E. Zinovieva. Symbolic test selection based on approximate analysis. In *11th Int. Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS'05)Volume 3440 of LNCS*, Edinburgh (Scottland), April 2005.

[10] T. Jéron, H. Marchand, and V. Rusu. Symbolic determinisation of extended automata. Technical Report 1176, IRISA, February 2006.