Universidade Técnica de Lisboa
Instituto Superior Técnico

Electronic Money within My-Calculus

Pedro Miguel dos Santos Alves Madeira Adão

Applied Mathematics and Computation

Diploma Thesis

Supervisor:

Professor Paulo Mateus

July 2002

Acknowledgments

It would be difficult to mention everyone that somehow contributed and helped me throughout these months. To all of them I would like to thank for their contribution. Some of them deserve a special thanks, due to some small "special" things.

I would like to thank my supervisor, Professor Paulo Mateus, for his guidance, support and constant motivation throughout this year, and also to Professor Amílcar Sernadas, for having originally suggested this topic for my thesis and for the profitable discussions we had.

I would also thank to Professor Cristina Sernadas for her support and for the contribution of my development as a scientist. With her enthusiasm Theoretical Computer Science becomes fascinating.

I must thank to Professor António Ravara for the constant interest in my work, and for informal talks in crucial moments that were decisive in the last stage of my work. I wish to thank Professors Carlos Caleiro and Jaime Ramos for their fellowship and small discussions about almost everything.

I also thank to João Boavida and João Rasga for their help in the TEXnical aspects of this thesis.

I would like to thank all my friends and colleagues. A special acknowledgment goes to my Friends Ana, Carla, Cátia, Alexandre, João, Pedro, Ricardo and Vitor for their constant support and with whom I shared so many (very) good moments in this last five years.

Finally I would like to thank my family, without whom this was not possible, and to Ana for her presence, love and (infinite) support. Their love was essential for the completion of this (hard) task. All this work is dedicated to them.

The following entities have sponsored my work and my participation in Summer Schools:

- CLC (Center for Logic and Computation, Instituto Superior Técnico);
- FCT (Fundação para a Ciência e a Tecnologia) and EU FEDER through the FibLog project (POCTI/2001/MAT/37239) of CLC.

Contents

List of symbols

$\mathcal{N}$	Set of names
Var	Set of variables
$\mathcal{T}_{Spi}$	Set of Spi-Calculus terms
$\mathcal{P}_{Spi}$	Set of Spi-Calculus processes
$Proc_{Spi}$	Set of Spi-Calculus closed processes
$\rhd_{Spi}$	Reduction relation in Spi-Calculus
$\equiv_{Spi}$	Structural relation in Spi-Calculus
$\hookrightarrow_{Spi}$	Reaction relation in Spi-Calculus
τ	Silent Action
$\longrightarrow_{Spi}$	Commitment relation in Spi-Calculus
$\downarrow$	Exhibition of barb
$\Downarrow$	Convergence through a barb
$\simeq_{Spi}$	Testing equivalence
$\mathcal{A}$	Set of principals
$\mathcal{M} \backslash \mathcal{M}'$	Difference of finite multisets
$\mathcal{M} \uplus \mathcal{M}'$	Union of finite multisets
Σ	Set of states
$\mathcal{L}_{\mathcal{O}}$	Set of local operations
$\mathcal{T}$	Set of My-Calculus terms
$\mathcal{P}$	Set of My-Calculus processes
$Proc$	Set of My-Calculus closed processes
Ξ	Set of configurations
$\rhd$	Reduction relation
$\rhd^*$	Reflexive closure of the reduction relation
$\Rrightarrow$	Commitment relation over processes
$\longrightarrow$	Commitment relation over configurations
$\simeq_A$	A-strong equivalence
$\Longrightarrow$	Weak commitment relation over configurations
$\cong_A$	A-weak equivalence
$\oplus$	Union of states

Chapter 1

Introduction

The growth of the Internet and the increasing sophistication and availability of cryptographic tools have promised to bring commerce to new heights of efficiency. Efficiency suggests a number of things, including minimized human involvement, improved distribution of goods and a more rapid processing of transactions, namely due to the development of the payment methods, which need to be more secure and easy to use.

The idea of electronic money was introduced by Chaum, Fiat and Naor [CFN90]. Their system (later improved in [CdBvH$^+$90]) meets most of the ideal requisites listed below, but it is quite complex. Since then, new schemes have been designed in order to satisfy some desired "ideal" properties, namely off-line payments [Bra93, Bra95, Bra99, Fer93a, Fer94, Fer93b], tamper resistant mechanisms [CP93] and divisibility [OO92, Oka95].

The "ideal" requirements for an electronic cash system are:

- Security: Every party in an electronic cash system should be protected from a collusion between all the other parties.
- Off-line: There should be no need for communication with a central authority during payment.
- Fake Privacy: The Bank and all the shops should not be able to derive together any knowledge from their protocol transcripts about where a user spends his money.
- Privacy (untraceable): The bank should not be able to determine whether two payments were made by the same payer, even if all shops cooperate.
- Divisibility: A piece of money can be divided into smaller pieces each of which can be spent separately.

Some of these requirements are even undesirable for the regulator institutions, namely Treasury Department and Police. For instance the *privacy* property is not desirable for the National Authorities because they are concerned with problems such as money laundering, off-shore banking, etc. So, it will be necessary to trace the money for law enforcement purposes. The *off-line* requirement might also be discussed. Unless we have a physical device to avoid forgery [CP93], the off-line money will not prevent the double spending of money. This fraud cannot be detected at the time of spending, as payments are off-line. The solution that all electronic cash systems use is to detect the double-spending after the fact. At each payment the user is required to release information in response to a challenge from the shop. Such

release provides no clue to the identity of the user, but two releases are sufficient to identify the user uniquely. Unfortunately, sometimes this detection is too late!

Besides the discussion of what are the desired properties, another problem of the protocols in literature is that they are described informally and only by words. In these protocols we want to model interactive systems, so we should use a formalism that deals with interaction and communication. There are several formal methods to specify and analyze protocols. The most common is the process algebra approach [AG97a, AG98b, AG97b, BNP99, Hoa80, Low96, MMS02, MPW92, Mil99, MRTS01, Ros95, Sch98], but there are also logics for specification and verification of protocols, [AT91, BAN96, GM95, MCJ97, SC00, Syv91, SM96].

In this thesis we intend to design/specify a protocol to implement *e*-money. We intend to design an on-line protocol, where the money is represented by electronic files that are issued by a trusted authority. These tokens have an identification and a fixed value that are issued by a trusted authority, commonly the bank. This authority will monitor the transactions, so that multi-spending becomes impossible — at each time the bank will know exactly who is the owner of each token. We will not bother with traceability issues because we are more interested in avoiding multi-spending than in the privacy of the users. We hope that the extension for incorporating untraceability can be achieved using secure computation technics, [Can00, Can01]. In fact, the actual payment mechanisms are traceable so this "ideal" property is more a desired property than a mandatory one.

We will need some kind of memory to register the tokens that are issued and who is their owner at each time. Since we want to use a process algebra to specify our protocol, we can introduce the notion of state using parameterization. In the beginning, we tried to use Spi-Calculus, but since we assume that our communication channels are private and our notion of equivalence is different from the one in Spi-Calculus, we realized that we could restrict ourselves to CCS, or a small extension of it. So, we came up with **Mone**y**-Calculus** (My-Calculus), that is specially designed to deal with our problem. We do not use parameterization to incorporate memory but, instead, we associate with each process a multiset that has all the notes issued. This representation is more common in imperative programming than in functional programming, however, it is easier to work with. This process algebra is similar to the CCS, but our notion of equivalence is different.

In Chapter 2 we present the Spi-Calculus in order to illustrate how cryptographic problems can be treated and how authentication and secrecy properties can be expressed. With this we argue that it is possible to establish secure communication between two principals and so, we design our protocol assuming that all channels are private. We suppose that there exists a private network between all principals — a principal A will share two communication channels with any other principal X, namely an input channel c_{XA} and an output channel c_{AX}.

Further, in Chapter 3, we present our calculus and a suitable notion of equivalence. As in CCS and π-calculus, [Mil80, Mil89, MPW92], we start presenting a strong notion and then a weaker notion of equivalence. We prove that these relations are equivalence relations.

In Chapter 4 we specify our protocols, the withdrawal and the payment protocols. Finally, in Chapter 5, we show some properties of the protocol such as the impossibilities to: spend one token more than once, forge a token (create a token that does not exists) or use tokens from another principal.

We conclude this thesis in Chapter 6, with final remarks and setting a path to future work, namely we intend to extend our calculus, achieve compositionality results and treat the untraceability problem.

Chapter 2

Spi-Calculus

2.1 Introduction

Since the early 80's, interactive systems and concurrent communicating systems have been studied. In that time two models were independently conceived, CCS and CSP. The first one was developed by Milner, [Mil80, Mil89], while the second one was developed by Hoare, [Hoa80]. These models were presented as formalisms of *interactional* behaviour, and their main objective was to analyze properties of concurrent communicating processes. It was shown that these two models were able to represent not only interactive concurrent systems, such as communications protocols, but also much of what is common in traditional computation, e.g., data structures and storage regimes. In fact CCS was used to give a rigorous definition of a fairly powerful concurrent programming language.

Meanwhile an extension of the CCS was presented, the π-calculus, [MPW92, Mil99]. This extension was mainly motivated by the notion of *mobility*. This notion is related to the possibility of establishment of new communication channels between two devices and was impossible to treat in both CCS and CSP. Due to the development of protocols involving cryptographic primitives, the π-calculus was also extended and so, in 1997, Martín Abadi and Andrew Gordon presented the Spi-Calculus, [AG97a, AG97b, AG98b], designed for describing and analyzing cryptographic protocols.

The π-calculus primitives for channels are simple and yet powerful. Channels can be created and passed, for example from authentication servers to clients. The names of the channels that a process knows, determine the communication possibilities of that process. Channels may be *restricted*, so that only certain processes may communicate on them. The *scoping* rules of the π-calculus guarantee that the environment of a protocol (the attacker) cannot access a channel that was not, previously, given to him; scoping is thus the basis of communication security. Therefore, the π-calculus seems to be a convenient calculus of protocols for secure communication.

However, the π-calculus does not have a natural way to express cryptographic operations that are commonly used for implementing private channels in distributed systems. So, the main difference between the π-calculus and the Spi-Calculus, is that the latter includes cryptographic primitives such as *encryption* and *decryption*. It is possible to encode these primitives in the π-calculus, as processes, but the higher-level approach is more convenient because it is possible to axiomatize encryption and decryption directly in the operational semantics of the Spi-Calculus. In spite of using this higher-level approach, the fundamental ideas of the

π-calculus are maintained.

In the Spi-Calculus the protocols are stated as processes and the properties are proven using notions of protocols equivalence, [AG97b, AG98b, AG98a]. For instance, we can say that a protocol keeps a piece of data X secret by stating that the protocol with X is equivalent to the protocol with X', for every X'. Here, equivalence means equivalence in the eyes of the environment. The environment can interact with the protocol, attempting to create confusion between different messages or sessions. This definition of equivalence yields the desired properties for most of the security applications. Interestingly, the standard notion of bisimilarity cannot be taken as the relevant notion of equivalence.

Although the definition of equivalence makes reference to the environment, in the Spi-Calculus there is no need to give an explicit model of the environment. This is one of the main advantages of this approach. Writing such a model can be tedious and can lead to new arbitrariness and error. This specification can also lead us to underestimation of the environment's "power", which may cause flaws in the security. This conflict is resolved by letting the environment be an arbitrary Spi-Calculus process.

In this chapter we present the Spi-Calculus as in [AG98b]. We start by presenting the syntax of the Spi-Calculus. In Section 2.3 we present two notions of operational semantics and in the Section 2.4 we define the notion of equivalence in the Spi-Calculus. Finally we present some examples in the Section 2.5.

2.2 Syntax

For the definition of the Spi-Calculus syntax we assume defined the following sets:

- A countable set of *names*, $\mathcal{N}= \{a, b, c, \dots\}$;
- A countable set of *variables*, $Var= \{x, y, z, \dots\}$.

Notation 1. We reserve the names c and m to denote *channels* and k to denote *keys*.

Definition 2.1. The set of *Spi-Calculus terms*, $\mathcal{T}_{Spi}$, is defined inductively as follows:

- $0 \in \mathcal{T}_{Spi}$;
- $a \in \mathcal{T}_{Spi}$, provided that $a \in \mathcal{N}$;
- $x \in \mathcal{T}_{Spi}$, provided that $x \in Var$;
- $suc(t) \in \mathcal{T}_{Spi}$, provided that $t \in \mathcal{T}_{Spi}$;
- $(t_1, t_2) \in \mathcal{T}_{Spi}$, provided that $t_1, t_2 \in \mathcal{T}_{Spi}$;
- $\{t\}_k \in \mathcal{T}_{Spi}$, provided that $t \in \mathcal{T}_{Spi}$ and $k \in \mathcal{N}$.

Intuitively, terms have the following meaning:

- The *pairing* constructor, (t_1, t_2), means that any two terms can be combined in a pair.
- The *shared-key encryption* constructor, $\{t\}_k$, represents the ciphertext obtained by encrypting the term t under the key k using a shared-key cryptosystem.

In the standard π-calculus names are the only terms. In the Spi-Calculus the constructs 0, $suc(t)$ and (t_1, t_2) were added in order to simplify some proofs. The *shared-key encryption* was added in order to represent the encryption of terms.

Definition 2.2. The set of *Spi-Calculus processes*, $\mathcal{P}_{Spi}$, is defined inductively as follows:

- $\mathbf{0} \in \mathcal{P}_{Spi}$;
- $\overline{m}\langle t\rangle.P \in \mathcal{P}_{Spi}$, provided that $m \in \mathcal{N}$, $t \in \mathcal{T}_{Spi}$ and $P \in \mathcal{P}_{Spi}$;
- $m(x).P \in \mathcal{P}_{Spi}$, provided that $m \in \mathcal{N}$, $x \in Var$ and $P \in \mathcal{P}_{Spi}$;
- $P \mid Q \in \mathcal{P}_{Spi}$, provided that $P, Q \in \mathcal{P}_{Spi}$;
- $(\nu\, n)\, P \in \mathcal{P}_{Spi}$, provided that $n \in \mathcal{N}$ and $P \in \mathcal{P}_{Spi}$;
- $!P \in \mathcal{P}_{Spi}$, provided that $P \in \mathcal{P}_{Spi}$;
- $[t_1\ is\ t_2]\, .P \in \mathcal{P}_{Spi}$, provided that $t_1, t_2 \in \mathcal{T}_{Spi}$ and $P \in \mathcal{P}_{Spi}$;
- $let\ (x, y) = t\ in\ P \in \mathcal{P}_{Spi}$, provided that $t \in \mathcal{T}_{Spi}$, $x, y \in Var$ and $P \in \mathcal{P}_{Spi}$;
- $case\ t\ of\ 0\colon P\ suc(x)\colon Q \in \mathcal{P}_{Spi}$, provided that $t \in \mathcal{T}_{Spi}$, $x \in Var$ and $P, Q \in \mathcal{P}_{Spi}$;
- $case\ t\ of\ \{x\}_k\ in\ P \in \mathcal{P}_{Spi}$, provided that $t \in \mathcal{T}_{Spi}$, $x \in Var$, $k \in \mathcal{N}$ and $P \in \mathcal{P}_{Spi}$.

Notation 2. We write $P[t/x]$ to represent the substitution of each free occurrence of the variable x in process P by the term t.

Intuitively, processes have the following meanings:

- The *nil* process $\mathbf{0}$ does nothing.
- An *output process*, $\overline{m}\langle t\rangle.P$, is ready to output the term t on the channel m, and then to behave as P. The output only happens when there is a process ready to input from the channel m. An *input process*, $m(x).Q$, is ready to input from m and then to behave as $Q[t/x]$ where t is the message received. The variable x is bound in Q.
- A *composition*, $P \mid Q$, behaves as P and Q running in parallel.
- A *restriction*, $(\nu\, n)\, P$, is a process that makes a new, private name n, which may occur in P, and then behaves as P. The name n is bound in P.
- A *replication*, $!P$, behaves as infinite replicas of P running in parallel.
- The *match* process, $[t_1\ is\ t_2]\, .P$, behaves as P provided that t_1 and t_2 are the same term; otherwise it gets stuck, i.e., it does nothing.
- A *pair splitting* process, $let\ (x, y) = t\ in\ P$, behaves as $P[t_1/x][t_2/y]$ if t is the pair (t_1, t_2), and it gets stuck if t is not a pair. The variables x and y are bound in P.
- An *integer case* process, $case\ t\ of\ 0\colon P\ suc(x)\colon Q$, behaves as P if t is 0, as $Q[t_1/x]$ if t is $suc(t_1)$, for some term t_1, and otherwise it gets stuck. The variable x is bound in Q.

- A *shared-key decryption* process, *case* t *of* $\{x\}_k$ *in* P, attempts to decrypt t with the key k. If t has the form $\{t_1\}_k$, for some term t_1 and name k,then the process behaves as $P[t_1/x]$. Otherwise the process gets stuck. The variable x is bound in P.

Notation 3. We adopt some abbreviations where we omit the **0** process. As an example we use $m(x)$ as short for $m(x).\mathbf{0}$.

Notation 4. Given a family of processes $P_1, \ldots, P_k$, we let $\prod_{i\in\{1,\ldots,k\}} P_i$ be their k-way composition $P_1 \mid \cdots \mid P_k$.

The possibilities of communication of a process may change during computation. When a process sends a *restricted channel* as a message to a process outside the scope of the restriction, the scope is said to *extrude*, that is, it enlarges to embrace the process receiving the channel (we will see this in Example 2.2). So, the processes are *mobile* in the sense that their communication possibilities may change over time; they may learn the names of new channels via *scope extrusion*. Thus, a channel is a transferable capability for communication.

The (technical) idea is to use the *restriction* operator and *scope extrusion* from the π-calculus, as a formal model of possession and communication of secrets, such as cryptographic keys.

Example 2.1. The process $(\nu\, k)\, \overline{m}\langle\{0\}_k\rangle$ creates a name k and outputs the result of encrypting the numeral 0 with k on the channel m. Since m is not bound, anyone may receive $\{0\}_k$; however, since k is bound, this term cannot be successfully decrypted. In order to illustrate the use of decryption, we add a process that has the necessary key:

$$(\nu\, k)\, (\overline{m}\langle\{0\}_k\rangle \mid m(y).\mathit{case}\ y\ \mathit{of}\ \{x\}_k\ \mathit{in}\ \overline{m}\langle x\rangle)$$

The new process (on the right of $\mid$) tries to decrypt the message y that it receives through m using k, and sends the result x back through m.

We would also like to distinguish processes with free variables (or free names) from processes where all the variables (or names) are bounded. To achieve this end we introduce two definitions, *free variables* and *free names*.

Definition 2.3. We define the *free variables of a term*, $fv(t)$, inductively as follows:

- $fv(0) = \emptyset$;
- $fv(a) = \emptyset$, if $a \in \mathcal{N}$;
- $fv(x) = \{x\}$, if $x \in Var$;
- $fv(suc(t)) = fv(t)$;
- $fv((t_1, t_2)) = fv(t_1) \cup fv(t_2)$;
- $fv(\{t\}_k) = fv(t) \cup fv(k)$.

We define the *free variables of a process*, $fv(P)$, inductively as follows:

- $fv(\mathbf{0}) = \emptyset$;

- $fv(\overline{m}\langle t\rangle.P) = fv(t) \cup fv(P)$;
- $fv(m(x).P) = fv(P)\backslash\{x\}$;
- $fv(P \mid Q) = fv(P) \cup fv(Q)$;
- $fv((\nu\, n)\, P) = fv(P)$;
- $fv(!P) = fv(P)$;
- $fv([t_1\ is\ t_2]\, .P) = fv(t_1) \cup fv(t_2) \cup fv(P)$;
- $fv(let\ (x, y) = t\ in\ P) = fv(t) \cup fv(P)\backslash\{x, y\}$;
- $fv(case\ t\ of\ 0\colon P\ suc(x)\colon Q) = fv(t) \cup fv(P) \cup fv(Q)\backslash\{x\}$;
- $fv(case\ t\ of\ \{x\}_k\ in\ P) = fv(t) \cup fv(k) \cup fv(P)\backslash\{x\}$.

We say that a process (or a term) is *closed* if it has no free variables. We denote the set of all closed processes of Spi-Calculus by $Proc_{Spi}$.

Definition 2.4. We define the *free names of a term*, $fn(t)$, inductively as follows:

- $fn(0) = \emptyset$;
- $fn(a) = \{a\}$, if $a \in \mathcal{N}$;
- $fn(x) = \emptyset$, if $x \in Var$;
- $fn(suc(t)) = fn(t)$;
- $fn((t_1, t_2)) = fn(t_1) \cup fn(t_2)$;
- $fn(\{t\}_k) = fn(t) \cup fn(k)$.

We define the *free names of a process*, $fn(P)$, inductively as follows:

- $fn(\mathbf{0}) = \emptyset$;
- $fn(\overline{m}\langle t\rangle.P) = \{m\} \cup fn(t) \cup fn(P)$;
- $fn(m(x).P) = \{m\} \cup fn(P)$;
- $fn(P \mid Q) = fn(P) \cup fn(Q)$;
- $fn((\nu\, n)\, P) = fn(P)\backslash\{n\}$;
- $fn(!P) = fn(P)$;
- $fn([t_1\ is\ t_2]\, .P) = fn(t_1) \cup fn(t_2) \cup fn(P)$;
- $fn(let\ (x, y) = t\ in\ P) = fn(t) \cup fn(P)$;
- $fn(case\ t\ of\ 0\colon P\ suc(x)\colon Q) = fn(t) \cup fn(P) \cup fn(Q)$;
- $fn(case\ t\ of\ \{x\}_k\ in\ P) = fn(t) \cup fn(k) \cup fn(P)$.

Before starting the definition of the operational semantics we will make some standard but significant assumptions about cryptography:

- The only way to decrypt an encrypted packet is to know the corresponding key;
- An encrypted packet does not reveal the key that was used to encrypt it;
- There is enough redundancy in the messages to allow the decryption algorithm to detect whether a ciphertext has been encrypted with the expected key.

2.3 Operational Semantics

In [AG98b] there are two operational semantics presented for Spi-Calculus - the *reaction relation* and the *commitment relation*. The *reaction relation* is an adaptation of a similar idea introduced by Milner. The definition of reaction is rather elegant, but not convenient for proofs (because it relies on an auxiliary notion of structural equivalence). Therefore, an alternative characterization of reaction is provided defining the *commitment relation*, in style of [Mil99]. We will present both as in [AG98b].

2.3.1 The Reaction Relation

We define the *reaction relation* in three phases, according to [AG98b]. In the first one we have the definition of the *reduction relation*, $\rhd_{Spi}$. Then we define what is called *structural equivalence* of two processes, $\equiv_{Spi}$. Finally we present the definition of the *reaction relation*, $\hookrightarrow_{Spi}$.

Definition 2.5. The *reduction relation*, $\rhd_{Spi} \subseteq Proc \times Proc$, is defined as the least relation on closed processes defined by the following rules:

(RedRepl)	$!P$	$\rhd_{Spi}$	$P \mid !P$
(RedMatch)	$[t\ is\ t].P$	$\rhd_{Spi}$	P
(RedPair)	$let\ (x,y) = (t_1,t_2)\ in\ P$	$\rhd_{Spi}$	$P[t_1/x][t_2/y]$
(RedZero)	$case\ 0\ of\ 0\colon x\ suc(P)\colon Q$	$\rhd_{Spi}$	P
(RedSuc)	$case\ suc(n)\ of\ 0\colon x\ suc(P)\colon Q$	$\rhd_{Spi}$	$Q[n/x]$
(RedDecrypt)	$case\ \{t\}_k\ of\ \{x\}_k\ in\ P$	$\rhd_{Spi}$	$P[t/x]$

Informally we say that two processes are *structural equivalent* if one can be transformed into the other using the rules below.

Definition 2.6. The *structural equivalence*, $\equiv_{Spi} \subseteq Proc \times Proc$, is defined as the least relation on closed processes that satisfies the following equations and rules:

(StructNil)	$P \mid \mathbf{0}$	$\equiv_{Spi}$	P	
(StructComm)	$P \mid Q$	$\equiv_{Spi}$	$Q \mid P$	
(StructAssoc)	$P \mid (Q \mid R)$	$\equiv_{Spi}$	$(P \mid Q) \mid R$	
(StructSwitch)	$(\nu\, m)\,(\nu\, n)\,P$	$\equiv_{Spi}$	$(\nu\, n)\,(\nu\, m)\,P$	
(StructDrop)	$(\nu\, m)\,\mathbf{0}$	$\equiv_{Spi}$	$\mathbf{0}$	
(StructExtrusion)	$(\nu\, m)\,(P \mid Q)$	$\equiv_{Spi}$	$P \mid (\nu\, m)\,Q$	if $m \notin fn(P)$

$$\frac{P \rhd_{Spi} Q}{P \equiv_{Spi} Q} \text{ (StructRed)} \qquad \frac{}{P \equiv_{Spi} P} \text{ (StructRefl)}$$

$$\frac{P \equiv_{Spi} Q}{Q \equiv_{Spi} P} \text{ (StructSymm)} \qquad \frac{P \equiv_{Spi} Q \quad Q \equiv_{Spi} R}{P \equiv_{Spi} R} \text{ (StructTrans)}$$

$$\frac{P \equiv_{Spi} P'}{P \mid Q \equiv_{Spi} P' \mid Q} \text{ (StructPar)} \qquad \frac{P \equiv_{Spi} P'}{(\nu\, m)\, P \equiv_{Spi} (\nu\, m)\, P'} \text{ (StructRes)}$$

We are now ready to define the *reaction relation* of two closed processes. The previous relations were defined to allow the rearrangement of processes so that the reaction could be possible.

Definition 2.7. The *reaction relation* on closed processes, $\hookrightarrow_{Spi} \subseteq Proc \times Proc$, is defined as the least relation on closed processes that satisfies the following axiom,

$$\overline{c}\langle t\rangle.P \mid c(x).Q \hookrightarrow_{Spi} P \mid Q[t/x] \qquad \text{(ReactInter)}$$

and the following rules:

$$\frac{P \equiv_{Spi} P' \quad P' \hookrightarrow_{Spi} Q' \quad Q' \equiv_{Spi} Q}{P \hookrightarrow_{Spi} Q} \text{ (ReactStruct)}$$

$$\frac{P \hookrightarrow_{Spi} P'}{P \mid Q \hookrightarrow_{Spi} P' \mid Q} \text{ (ReactPar)} \qquad \frac{P \hookrightarrow_{Spi} P'}{(\nu\, n)\, P \hookrightarrow_{Spi} (\nu\, n)\, P'} \text{ (ReactRes)}$$

2.3.2 Commitment Relation in Spi-Calculus

In order to define the *commitment relation* we need two new syntactic forms - *abstractions* and *concretions*. An *abstraction* is an expression of the form $(x).P$ where x is a bound variable and P is a process. When F is the abstraction $(x).P$ and t is a term, we write $F(t)$ for $P[t/x]$. A *concretion* is an expression of the form $(\nu\, \vec{n})\, \langle t\rangle P$ where t is a term, P is a process and $\vec{n}$ are names that are bound in t and P. We will use C and D for concretions.

We define an *agent* as an abstraction, a concretion or a process. We will use the variables A and B when representing agents, and define $fv(A)$ and $fn(A)$ as the sets of free variables and free names of an agent A, respectively. The definitions of $fv(A)$ and $fn(A)$ are the expected extensions of the Definitions 2.4 and 2.3.

We now have to extend restriction and composition to arbitrary agents. We will do this using the following rules:

$$\begin{aligned}
(\nu\, m)\, (x).P &\triangleq (x).(\nu\, m)\, P; \\
R \mid (x).P &\triangleq (x).(R \mid P) \qquad \text{if } x \notin fv(R); \\
(\nu\, m)\, (\nu\, \vec{n})\, \langle t\rangle P &\triangleq \begin{cases} (\nu\, m, \vec{n})\, \langle t\rangle P & \text{if } m \in fn(t) \\ (\nu\, \vec{n})\, \langle t\rangle (\nu\, m)\, P & \text{otherwise} \end{cases}; \\
R \mid (\nu\, \vec{n})\, \langle t\rangle P &\triangleq (\nu\, \vec{n})\, \langle t\rangle (R \mid P) \qquad \text{if } \{\vec{n}\} \cap fn(R) = \emptyset.
\end{aligned}$$

In the first and third equation we also suppose that $m \notin \{\vec{n}\}$. We define the dual composition $A \mid R$ symmetrically.

The *interactions* of an abstraction $F = (x).P$ and a concretion $C = (\nu\,\vec{n})\,\langle t\rangle P$, $F@C$ and $C@F$, are defined as the processes:

$$F@C \triangleq (\nu\,\vec{n})\,(P[t/x] \mid Q);$$
$$C@F \triangleq (\nu\,\vec{n})\,(Q \mid P[t/x]).$$

Intuitively these processes represent the interaction of P and Q. It is the same as P and Q running in parallel and communicating using the same channel c.

We will now define how transitions work in Spi-Calculus.

Definition 2.8. We say that β is a *barb* if it is a name m (representing an input) or a co-name $\overline{m}$ (representing an output). We say that α is an *action* if it is a barb or the *silent action* τ.

Now we are able to introduce the *commitment relation* as in [AG98b].

Definition 2.9. The *commitment relation*, $\longrightarrow_{Spi}$, is written $P \xrightarrow{\alpha}_{Spi} A$, where P is a closed process, α is an action and A is a closed agent, and is defined inductively by the following rules:

$$\frac{}{m(x).P \xrightarrow{m}_{Spi} (x).P}\ (\text{CommIn}) \qquad \frac{}{\overline{m}\langle t\rangle.P \xrightarrow{\overline{m}}_{Spi} (\nu\,)\,\langle t\rangle P}\ (\text{CommOut})$$

$$\frac{P \xrightarrow{m}_{Spi} F \quad Q \xrightarrow{\overline{m}}_{Spi} C}{P \mid Q \xrightarrow{\tau}_{Spi} F@C}\ (\text{CommInter1}) \qquad \frac{P \xrightarrow{\overline{m}}_{Spi} C \quad Q \xrightarrow{m}_{Spi} F}{P \mid Q \xrightarrow{\tau}_{Spi} C@F}\ (\text{CommInter2})$$

$$\frac{P \xrightarrow{\alpha}_{Spi} A}{P \mid Q \xrightarrow{\alpha}_{Spi} A \mid Q}\ (\text{CommLPar}) \qquad \frac{Q \xrightarrow{\alpha}_{Spi} A}{P \mid Q \xrightarrow{\alpha}_{Spi} P \mid A}\ (\text{CommRPar})$$

$$\frac{P \xrightarrow{\alpha}_{Spi} A \quad \alpha \notin \{m, \overline{m}\}}{(\nu\,m)\,P \xrightarrow{\alpha}_{Spi} (\nu\,m)\,A}\ (\text{CommRes}) \qquad \frac{P \rhd_{Spi} Q \quad Q \xrightarrow{\alpha}_{Spi} A}{P \xrightarrow{\alpha}_{Spi} A}\ (\text{CommRed})$$

Whenever $P \xrightarrow{\alpha}_{Spi} A$, if the action α is a name, then A is an abstraction, if α is a co-name, A is a concretion, and if α is τ, A is a process. Therefore the commitment relation indexed by τ is a binary relation on $Proc$. We write $P \xrightarrow{\tau}_{Spi \equiv_{Spi}} Q$ when there exists a process R such that $P \xrightarrow{\tau}_{Spi} R$ and $R \equiv_{Spi} Q$.

Proposition 2.1. *$P \hookrightarrow_{Spi} Q$ if and only if $P \xrightarrow{\tau}_{Spi \equiv_{Spi}} Q$.*

Proof. See Appendix B of [AG98b]. □

2.4 Testing Equivalence

In Spi-Calculus, the notion of equivalence between two processes is called *testing equivalence*. In order to define it, we must define two auxiliary predicates, *exhibition of a barb* and *convergence through a barb*.

Definition 2.10. We say that a closed process P *exhibits barb* β, written $P{\downarrow}\beta$, if it satisfies the following rules:

$$\frac{}{m(x).P \downarrow m}\ \text{(BarbIn)} \qquad \frac{}{\overline{m}\langle t\rangle.P \downarrow \overline{m}}\ \text{(BarbOut)}$$

$$\frac{P \downarrow \beta}{P \mid Q \downarrow \beta}\ \text{(BarbPar)} \quad \frac{P \downarrow \beta \quad \beta \notin \{m, \overline{m}\}}{(\nu\, m)\, P \downarrow \beta}\ \text{(BarbRes)} \quad \frac{P \equiv_{Spi} Q \quad Q \downarrow \beta}{P \downarrow \beta}\ \text{(BarbStruct)}$$

Informally, a process P exhibiting a *barb* β, means that it can input or output immediately using the channel β. Another notion associated with this one is the notion of *convergence through a barb*.

Definition 2.11. We say that a process P *converges* through the barb β, written $P{\Downarrow}\beta$, if it exhibits barb β after a few reactions, this means,

$$\frac{P \downarrow \beta}{P \Downarrow \beta}\ \text{(ConvBarb)} \qquad \frac{P \hookrightarrow_{Spi} Q \quad Q \Downarrow \beta}{P \Downarrow \beta}\ \text{(ConvReact)}$$

We are now ready to define the notion of *testing equivalence*.

Definition 2.12.

(a) We define a *test* as a pair (R, β), where R is a closed process and β is a barb.

(b) We say that a process P *passes the test* (R, β) iff $(P \mid R) \Downarrow \beta$.

(c) We say that two processes P and Q are *testing equivalent*, written $P{\simeq_{Spi}}Q$, if

$$\forall \text{ test } (R, \beta), \qquad P \text{ passes the test iff } Q \text{ passes the test.}$$

Proposition 2.2.

(1) Structural equivalence implies testing equivalence, $\equiv_{Spi} \subseteq \simeq_{Spi}$.

(2) Testing equivalence is an equivalence relation.

(3) Testing equivalence is a congruence in Proc.

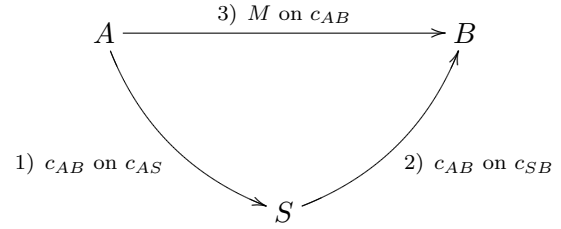
Proof. See Appendix D of [AG98b]. □

Proposition 2.3. *$P \downarrow \beta$ if and only if there exists an agent A such that $P \xrightarrow{\beta}_{Spi} A$.*

Proof. See Appendix B of [AG98b]. □

Proposition 2.4. *P passes the test (R, β) if an only if there exists an agent A and a process Q such that $P \mid R \xrightarrow{\tau}{}^{*}_{Spi} Q$ and $Q \xrightarrow{\beta}_{Spi} A$.*

Proof. See Appendix B of [AG98b]. □



Message 1 $A \rightarrow S : c_{AB}$ on c_{AS}
Message 2 $S \rightarrow B : c_{AB}$ on c_{SB}
Message 3 $A \rightarrow B : M$ on c_{AB}

Figure 2.1: Structure of the *Wide Mouthed Frog* protocol for establishment of a private channel between two principals.

2.5 Some Examples

As it was mentioned in the introduction, the π-calculus primitives for channels are simple and yet powerful. This power enables us to express some security protocols. It is common to find channels on which only a given set of principals is allowed to send data or listen. The set of principals may expand in the course of a protocol run, for example as the result of channel establishment. This property is easy to model in the π-calculus, via the restriction operation; the expansion of the set of principals that can access a channel corresponds to the *scope extrusion*.

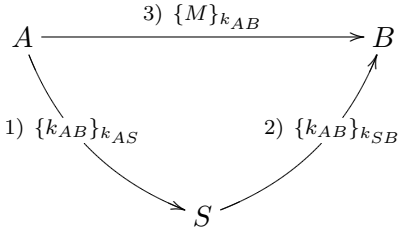
We present one small protocol, *Wide Mouth Protocol* [BAN96], that was designed for two principals, A and B, to exchange keys using an authentication server, S. In the first example we only use the π-calculus to explore the potential of the primitives for channels, using the protocol to send a message M, from A to B. We only create a private channel for communication between A and B. In the second example we will use the cryptographic primitives of the Spi-Calculus to design the WMF protocol. For both examples we state and prove what are called the *authenticity* and *secrecy* properties. This examples were also taken from [AG98b].

Example 2.2. Suppose that we have two principals, A and B, and a server S. Suppose also that each one of these principals shares a private channel with S, namely c_{AS} and c_{SB}. The principals A and B want to establish a new communication channel, c_{AB}, with the help of S. How can they achieve that? An informal description of the protocol is presented in Figure 2.1.

After passing the channel to B through S, A sends the message M on c_{AB}. In the π-calculus we formulate the protocol as follows:

$$\begin{aligned} A(M) &\triangleq (\nu\, c_{AB})\, \overline{c_{AS}}\langle c_{AB}\rangle.\overline{c_{AB}}\langle M\rangle; \\ S &\triangleq c_{AS}(x).\overline{c_{SB}}\langle x\rangle; \\ B &\triangleq c_{SB}(x).x(y).F(y); \\ Inst(M) &\triangleq (\nu\, c_{AS})\,(\nu\, c_{SB})\,(A(M) \mid S \mid B). \end{aligned}$$

We write $F(y)$ to represent what B does with the message that he receives. The restriction of the channels c_{AS} and c_{SB} represents the expected privacy guaranties for these channels. Another important fact to notice is the use of *scope extrusion*: A generates a new channel and sends it to the scope of B via S, that is, B is now able to communicate over a new channel, c_{AB}. For the discussion of authenticity the following specification is introduced:



Message 1 $A \rightarrow S : \{k_{AB}\}_{k_{AS}}$ on c_{AS}
Message 2 $S \rightarrow B : \{k_{AB}\}_{k_{SB}}$ on c_{SB}
Message 3 $A \rightarrow B : \{M\}_{k_{AB}}$ on c_{AB}

Figure 2.2: Structure of the *Wide Mouthed Frog* protocol for establishment of a private key between two principals.

$$
\begin{aligned}
A(M) &\triangleq (\nu\, c_{AB})\, \overline{c_{AS}}\langle c_{AB}\rangle.\overline{c_{AB}}\langle M\rangle;\\
S &\triangleq c_{AS}(x).\overline{c_{SB}}\langle x\rangle;\\
B_{Spec}(M) &\triangleq c_{SB}(x).x(y).F(M);\\
Inst_{Spec}(M) &\triangleq (\nu\, c_{AS})\,(\nu\, c_{SB})\,(A(M) \mid S \mid B_{Spec}(M)).
\end{aligned}
$$

If we look at this specification we observe that no matter what message B receives, he will always apply F to the message M. We obtain the following authenticity and secrecy properties:

Authenticity: $Inst(M) \simeq_{Spi} Inst_{Spec}(M)$ for all M;
Secrecy: $Inst(M) \simeq_{Spi} Inst(M')$ if $F(M) \simeq_{Spi} F(M')$ for all M and M'.

Proof. See Section 6 of [AG98b]. □

This example explored the π-calculus properties for channels. In the next example we present the WMF protocol as in the original version.

Example 2.3. Suppose that we have two principals, A and B, and a server S and that each of these shares a private key with S, namely k_{AS} and k_{SB}. The principals A and B want to establish a new communication key, k_{AB}, with the help of S. How can they achieve that? We illustrate this in the following example where we suppose that the channels c_{AS} and c_{SB} are not private.

An informal description of the protocol is given in Figure 2.2. After passing the key, k_{AB}, to B through S, A sends the message M on c_{AB} using the new key. Note that the key k_{AB} is always encrypted in the communications. In the Spi-Calculus we formulate the protocol as follows:

$$
\begin{aligned}
A(M) &\triangleq (\nu\, k_{AB})\, \overline{c_{AS}}\langle\{k_{AB}\}_{k_{AS}}\rangle.\overline{c_{AB}}\langle\{M\}_{k_{AB}}\rangle;\\
S &\triangleq c_{AS}(x).case\ x\ of\ \{y\}_{k_{AS}}\ in\ \overline{c_{SB}}\langle\{y\}_{k_{SB}}\rangle;\\
B &\triangleq c_{SB}(x).case\ x\ of\ \{y\}_{k_{SB}}\ in\ c_{AB}(z).case\ z\ of\ \{w\}_y\ in\ F(w);\\
Inst(M) &\triangleq (\nu\, k_{AS})\,(\nu\, k_{SB})\,(A(M) \mid S \mid B).
\end{aligned}
$$

We write $F(w)$ to represent what B does with the message that he receives. The restriction of the keys k_{AS} and k_{SB} represents the expected privacy guaranties for these keys.

For the discussion of authenticity we introduce the following specification:

$$\begin{aligned} A(M) &\triangleq (\nu\, k_{AB})\, \overline{c_{AS}}\langle\{k_{AB}\}_{k_{AS}}\rangle.\overline{c_{AB}}\langle\{M\}_{k_{AB}}\rangle; \\ S &\triangleq c_{AS}(x).case\ x\ of\ \{y\}_{k_{AS}}\ in\ \overline{c_{SB}}\langle\{y\}_{k_{SB}}\rangle; \\ B_{Spec}(M) &\triangleq c_{SB}(x).case\ x\ of\ \{y\}_{k_{SB}}\ in\ c_{AB}(z).case\ z\ of\ \{w\}_y\ in\ F(M); \\ Inst_{Spec}(M) &\triangleq (\nu\, k_{AS})\, (\nu\, k_{SB})\, (A(M) \mid S \mid B_{Spec}(M)). \end{aligned}$$

Again, if we look at this specification we observe that despite the message B receives, he will always apply F to the message M. As in the previous example, we obtain the following authenticity and secrecy properties:

Authenticity: $Inst(M) \simeq_{Spi} Inst_{Spec}(M)$ for all M;
Secrecy: $Inst(M) \simeq_{Spi} Inst(M')$ if $F(M) \simeq_{Spi} F(M')$ for all M and M'.

Proof. See Section 6 of [AG98b]. □

Chapter 3

My-Calculus

3.1 Introduction

In Chapter 2 we presented the Spi-Calculus, a calculus for analysis of communication protocols including cryptographic primitives. This calculus is extremely powerful for treating cryptographic problems, but as said in the introduction, we want to model an e-bank, and for that we need to join processes and some kind of memory (in order to register the coins available), therefore we will use configurations.

Usually, in CCS and in π-calculus we use parameterization in order to incorporate the notion of memory. A classical example is the example of the counter. The states of the counter are denoted $Count_n$, $n \geq 0$, with behaviour described by the following equations

$$Count_0 \triangleq inc.Count_1 + \overline{zero}.Count_0$$
$$Count_{n+1} \triangleq inc.Count_{n+2} + \overline{dec}.Count_n$$

and we set $Count \triangleq Count_0$. You can always increment by pressing the button inc; when it is $zero$ you can detect the fact by pressing the $\overline{zero}$ button; otherwise you can detect that it is nonzero and you can decrement the counter by pressing $\overline{dec}$.

In order to solve our problem we should have introduced this notion of memory in Spi-Calculus. However, if we analyze our problem, implementation of an e-bank, we realize that the main problems of the bank are not related to the privacy of communication, but with *forgery* and *stealing*. We do not want to worry about privacy problems, and so, we suppose that all communications are private. So, in spite of being possible to incorporate the notion of memory in the Spi-Calculus, we realize that testing equivalence is not appropriated to study our protocols.

Informally, the notion of equivalence presented in Section 3.5, states that two configurations are equivalent to a principal A if they cannot "cheat" A, that is, if for both configurations the state of A is equal after the execution of both protocols. For the applications in mind, it is enough to say that our protocols are equivalent to the bank, that is, no one can "cheat" the bank.

Since we do not need to worry about cryptographic problems, we will construct our calculus using the notions from the π-calculus leaving behind the constructors introduced by the Spi-Calculus.

In this chapter we start presenting the syntax of our calculus, Section 3.2. In the Section 3.3, we present the operational semantics of the calculus. Finally in Sections 3.4 and 3.5

we define the notion of equivalence for our calculus.

3.2 Syntax

This section will be similar to the Section 2.2. We assume defined the following sets:

- A countable set of *names*, $\mathcal{N} = \{a, b, c, \dots\}$;
- A countable set of *variables*, $Var = \{x, y, z, \dots\}$;
- A finite set of *principals*, $\mathcal{A}$= $\{A_1, A_2, \dots, A_n\}$.

The difference to the Chapter 2 is that in this one we also assume the set $\mathcal{A}$ to be defined, so that we can associate each state to a principal A. In order to define the notion of *state* we have to introduce the definition of finite multisets.

Definition 3.1. A *finite multiset* $\mathcal{M}$ over a set L is a map $\mathcal{M} : L \to \mathbb{N}$ such that $\mathcal{M}^{-1}(\mathbb{N}_1)$ is finite. We define the following operations on finite multisets:

(a) The *difference* of the multisets $\mathcal{M}$ and $\mathcal{M}'$ is the multiset $\mathcal{M}\backslash\mathcal{M}'$ where $(\mathcal{M}\backslash\mathcal{M}')(l) = max(0, \mathcal{M}(l) - \mathcal{M}'(l))$;

(b) The *union* of two multisets $\mathcal{M}$ and $\mathcal{M}'$ is the multiset $\mathcal{M} \uplus \mathcal{M}'$ where $(\mathcal{M} \uplus \mathcal{M}')(l) = \mathcal{M}(l) + \mathcal{M}'(l)$;

(c) We say that $l \in \mathcal{M}$ iff $\mathcal{M}(l) > 0$.

We define formally our notion of *state* as follows:

Definition 3.2. We define a *state*, $\sigma = \{\sigma_A\}_{A \in \mathcal{A}}$, to be a family of multisets indexed by the principals, where each σ_A represents the local state of the principal A. We denote the set of all states by Σ.

A first approach to the definition of state would be a family of sets. If we consider each σ_A as a set, we were restricting the possibility of a principal to have many copies of the same coin. We want to deal with the possibility of existing several copies of the coin and verify that our system has the desired properties. Clearly, the state of the bank will have special constraints, but this will be discussed in the Chapter 4. We also have to add some *operations* to deal with the state.

Observation 1. Each σ_A is a multiset over the set of terms, so it is a map $\sigma_A : \mathcal{T} \to \mathbb{N}$.

Definition 3.3. We define the set of *local operations*, $\mathcal{L}_{\mathcal{O}}$=$\{op, op_1, op_2, \dots, op_n\}$, to be the set of possible operations over states. Each of the operations is a family, $op = \{op_A\}_{A \in \mathcal{A}}$, where each op_A is an operation over the state of the principal A, σ_A.

We consider only three operations, *add*, *change* and *remove*. The semantic of these operations is described in the Section 3.3. Our set of local operations, $\mathcal{L}_{\mathcal{O}}$, is $\{add, change, remove\}$.

We continue with the definition of the *terms* and *processes* of My-Calculus. The definitions are similar to the ones of the previous chapter, but we do not introduce all the constructors. We only introduce those that will be useful in the design of our protocols.

Definition 3.4. The set of *My-Calculus terms*, $\mathcal{T}$, is defined inductively as follows:

- $a \in \mathcal{T}$, provided that $a \in \mathcal{N}$;
- $n \in \mathcal{T}$, provided that $n \in \mathbb{N}$;
- $x \in \mathcal{T}$, provided that $x \in Var$;
- $(t_1, t_2) \in \mathcal{T}$, provided that $t_1, t_2 \in \mathcal{T}$.

Notation 5. We use $(t_1, t_2, \ldots, t_{n-1}, t_n)$ as short for $(t_1, (t_2, (\ldots, (t_{n-1}, t_n))))$.

We may now define what is intended as a *well defined* operation.

Definition 3.5. We say that the operation *op* is *well defined* if:

- $op = add_A(t)$, where $A \in \mathcal{A}$ and $t \in \mathcal{T}$;
- $op = change_A(t_1, t_2)$, where $A \in \mathcal{A}$ and $t_1, t_2 \in \mathcal{T}$;
- $op = remove_A(t)$, where $A \in \mathcal{A}$ and $t \in \mathcal{T}$.

Definition 3.6. The set of *My-Calculus processes*, $\mathcal{P}$, is defined inductively as follows:

- $\mathbf{0} \in \mathcal{P}$;
- $op.P$, provided that *op* is a well defined operation and $P \in \mathcal{P}$;
- $\overline{m}\langle t \rangle.P \in \mathcal{P}$, provided that $m \in \mathcal{N}$, $t \in \mathcal{T}$ and $P \in \mathcal{P}$;
- $m(x).P \in \mathcal{P}$, provided that $m \in \mathcal{N}$, $x \in Var$ and $P \in \mathcal{P}$;
- $P \mid Q \in \mathcal{P}$, provided that $P, Q \in \mathcal{P}$;
- $(\nu\, n)\, P \in \mathcal{P}$, provided that $n \in \mathcal{N}$ and $P \in \mathcal{P}$;
- $!P \in \mathcal{P}$, provided that $P \in \mathcal{P}$;
- $[t_1\ is\ t_2]\, .P \in \mathcal{P}$, provided that $t_1, t_2 \in \mathcal{T}$ and $P \in \mathcal{P}$.

Intuitively, processes have the following meanings:

- The *nil* process $\mathbf{0}$ does nothing.
- The $op.P$ process means that the operation *op* is done and if it succeeds, the process behaves as P, otherwise it gets stuck, that is, it does nothing.
- An *output process* $\overline{m}\langle t \rangle.P$ is ready to output the term t on the channel m, and then to behave as P. The output only happens when there is a process ready to input from the channel m. An *input process* $m(x).Q$ is ready to input from m and then to behave as $Q[t/x]$ where t is the message received. x is bound in Q.
- The *composition* $P \mid Q$ behaves as P and Q running in parallel.

- The *restriction* $(\nu\, n)\, P$ is a process that makes a new, private name n, which may occur in P, and then behaves as P. The name n is bound in P.
- The *replication* $!P$ behaves as infinite replicas of P running in parallel.
- The *match* process $[t_1\ is\ t_2]\, .P$ behaves as P provided that t_1 and t_2 are the same term; otherwise it gets stuck.

The definitions of $fv(t)$, $fv(P)$, $fn(t)$ and $fn(P)$ are similar to the Definition 2.3 and Definition 2.4.

Definition 3.7. We define the *free variables of a term*, $fv(t)$, inductively as follows:

- $fv(a) = \emptyset$, if $a \in \mathcal{N}$;
- $fv(n) = \emptyset$, if $n \in \mathbb{N}$;
- $fv(x) = \{x\}$, if $x \in Var$;
- $fv((t_1, t_2)) = fv(t_1) \cup fv(t_2)$.

We define the *free variables of an operation*, $fv(op)$, as follows:

- $fv(add_A(t)) = fv(t)$;
- $fv(change_A(t_1, t_2)) = fv(t_1) \cup fv(t_2)$;
- $fv(remove_A(t)) = fv(t)$.

We define the *free variables of a process*, $fv(P)$, inductively as follows:

- $fv(\mathbf{0}) = \emptyset$;
- $fv(op.P) = fv(op) \cup fv(P)$;
- $fv(\overline{m}\langle t\rangle.P) = fv(t) \cup fv(P)$;
- $fv(m(x).P) = fv(P)\backslash\{x\}$;
- $fv(P \mid Q) = fv(P) \cup fv(Q)$;
- $fv((\nu\, n)\, P) = fv(P)$;
- $fv(!P) = fv(P)$;
- $fv([t_1\ is\ t_2]\, .P) = fv(t_1) \cup fv(t_2) \cup fv(P)$.

We say that a process (or a term) is *closed* if it has no free variables. We denote the set of all closed processes of My-Calculus by $Proc$.

Definition 3.8. We define the *free names of a term*, $fn(t)$, inductively as follows:

- $fn(a) = \{a\}$, if $a \in \mathcal{N}$;

- $fn(n) = \emptyset$, if $n \in \mathbb{N}$;
- $fn(x) = \emptyset$, if $x \in Var$;
- $fn((t_1, t_2)) = fn(t_1) \cup fn(t_2)$.

We define the *free names of an operation*, $fn(op)$, as follows:

- $fn(add_A(t)) = fn(t)$;
- $fn(change_A(t_1, t_2)) = fn(t_1) \cup fn(t_2)$;
- $fn(remove_A(t)) = fn(t)$.

We define the *free names of a process*, $fn(P)$, inductively as follows:

- $fn(\mathbf{0}) = \emptyset$;
- $fn(op.P) = fn(op) \cup fn(P)$;
- $fn(\overline{m}\langle t\rangle.P) = \{m\} \cup fn(t) \cup fn(P)$;
- $fn(m(x).P) = \{m\} \cup fn(P)$;
- $fn(P \mid Q) = fn(P) \cup fn(Q)$;
- $fn((\nu\, n)\, P) = fn(P) \backslash \{n\}$;
- $fn(!P) = fn(P)$;
- $fn([t_1\ is\ t_2]\, .P) = fn(t_1) \cup fn(t_2) \cup fn(P)$.

Definition 3.9. We define a *configuration* to be a pair (σ, P), where $\sigma \in \Sigma$ and $P \in \mathcal{P}$. We denote the set of all configurations by Ξ.

3.3 Semantics

As mentioned in the previous section we start by defining the semantics of the local operations. Informally, these operations have the following semantics:

- The operation $add_A(t)$ appends the term t to the state σ_A.
- The operation $change_A(t_1, t_2)$ replaces an occurrence of the term t_1 in σ_A with the term t_2. This operation succeeds if $t_1 \in \sigma_A$.
- The operation $remove_A(t)$ removes one occurrence of the term t from σ_A. This operation succeeds if $t \in \sigma_A$.

Formally, we define the semantics of these operations as a relation over configurations, the *reduction relation*. In this definition we use the notation of multisets introduced in the Definition 3.1. After defining the semantics of these operations, we define the operational semantics of our calculus via the *commitment relation*.

3.3.1 Reduction Relation

Definition 3.10. We define the *reduction relation* over configurations , $\rhd \subseteq \Xi \times \Xi$, as the least relation over configurations that satisfies the following equations and rules:

$$\text{(RedAdd)} \qquad (\sigma, add_X(t).P) \ \rhd \ (\sigma', P)$$

$$\text{where } \sigma' = \{\sigma'_A\}_{A\in\mathcal{A}} \text{ and } \sigma'_A = \begin{cases} \sigma_A \uplus \{t\} & \text{if } A = X \\ \sigma_A & \text{otherwise} \end{cases};$$

$$\text{(RedChange1)} \quad (\sigma, change_X(t_1, t_2).P) \ \rhd \ (\sigma', P) \quad \text{if} \quad t_1 \in \sigma_X$$

$$\text{where } \sigma' = \{\sigma'_A\}_{A\in\mathcal{A}} \text{ and } \sigma'_A = \begin{cases} \sigma_A \uplus \{t_2\}\backslash\{t_1\} & \text{if } A = X \\ \sigma_A & \text{otherwise} \end{cases};$$

$$\text{(RedChange2)} \quad (\sigma, change_X(t_1, t_2).P) \ \rhd \ (\sigma, \mathbf{0}) \quad \text{if} \quad t_1 \notin \sigma_X;$$

$$\text{(RedRemove1)} \qquad (\sigma, remove_X(t).P) \ \rhd \ (\sigma', P) \quad \text{if} \quad t \in \sigma_X$$

$$\text{where } \sigma' = \{\sigma'_A\}_{A\in\mathcal{A}} \text{ and } \sigma'_A = \begin{cases} \sigma_A\backslash\{t\} & \text{if } A = X \\ \sigma_A & \text{otherwise} \end{cases};$$

$$\text{(RedRemove2)} \qquad (\sigma, remove_X(t).P) \ \rhd \ (\sigma, \mathbf{0}) \quad \text{if} \quad t \notin \sigma_X;$$

$$\frac{(\sigma, P) \rhd (\sigma', Q) \quad (\sigma', Q) \rhd (\sigma'', R)}{(\sigma, P) \rhd (\sigma'', R)} \text{ (RedTrans)}$$

$$\frac{(\sigma, P) \rhd (\sigma', P')}{(\sigma, P \mid Q) \rhd (\sigma', P' \mid Q)} \text{ (RedLPar)} \qquad \frac{(\sigma, Q) \rhd (\sigma', Q')}{(\sigma, P \mid Q) \rhd (\sigma', P \mid Q')} \text{ (RedRPar)}$$

$$\frac{(\sigma, P) \rhd (\sigma', P')}{(\sigma, (\nu\, n)\, P) \rhd (\sigma', (\nu\, n)\, P')} \text{ (RedRest)} \qquad \frac{(\sigma, P) \rhd (\sigma', P')}{(\sigma, [x \ is \ x].P) \rhd (\sigma', P')} \text{ (RedMatch)}$$

Notation 6. We use $\rhd^*$ to denote the *reflexive closure* of the reduction relation.

We notice immediately that the rewriting system induced by this relation is not confluent.

Example 3.1. Assume that we have the following configuration, with only one principal A.

$$(\{\{\}\}, add_A(t) \mid change_A(t, t_1))$$

As we may see we can construct the following different trees:

$$\dfrac{\dfrac{\dfrac{}{(\{\{\}\}, add_A(t)) \rhd (\{\{t\}\}, \mathbf{0})}\text{(RedAdd)}}{(\{\{\}\}, add_A(t) \mid change_A(t, t_1))\rhd}\text{(RedLPar)} \quad \dfrac{\dfrac{}{(\{\{t\}\}, change_A(t, t_1))\rhd}\text{(RedChange1)}}{(\{\{t\}\}, \mathbf{0} \mid change_A(t, t_1))\rhd}\text{(RedRPar)}}{(\{\{\}\}, add_A(t) \mid change_A(t, t_1)) \rhd (\{\{t_1\}\}, \mathbf{0} \mid \mathbf{0})}\text{(RedTrans)}$$

and

$$\dfrac{\dfrac{\dfrac{}{(\{\{\}\}, change_A(t,t_1)) \rhd (\{\{\}\}, \mathbf{0})}\,(\text{RedChange2})}{(\{\{\}\}, add_A(t) \mid change_A(t,t_1)) \rhd (\{\{\}\}, add_A(t) \mid \mathbf{0})}\,(\text{RedRPar}) \quad \dfrac{\dfrac{}{(\{\{\}\}, add_A(t)) \rhd (\{\{t\}\}, \mathbf{0})}\,(\text{RedAdd})}{(\{\{\}\}, add_A(t) \mid \mathbf{0}) \rhd (\{\{t\}\}, \mathbf{0} \mid \mathbf{0})}\,(\text{RedLPar})}{(\{\{\}\}, add_A(t) \mid change_A(t,t_1)) \rhd (\{\{t\}\}, \mathbf{0} \mid \mathbf{0})}\,(\text{RedTrans})$$

3.3.2 Commitment Relation

The fact that names can be transmitted in interactions makes the π-calculus semantics naturally proliferate in two distinct families — *late* and *early* — depending on the operational intuition about input actions. The *late* paradigm interprets the derivative of the inputting process as a function of the name received, and then insists for an input move to be matched by a single input step. The more liberal, *early* semantics, allows an input to be matched by distinct moves, depending on the actual transmitted parameter. So, in the *early bisimulation* we only require that for each received name there is a matching transition. We define our *commitment relation* in the *early* style. We first define the *commitment relation* over processes and then extend it to commitment over configurations.

Definition 3.11. We define the *commitment relation over processes*, $\Rrightarrow \subseteq \mathcal{P} \times \mathcal{P}$, as the least relation over My-Calculus processes defined by the following rules:

$$\dfrac{}{\tau.P \overset{\tau}{\Rrightarrow} P}\,(\text{Tau}) \qquad \dfrac{}{x(y).P \overset{x(w)}{\Rrightarrow} P[w/y]}\,(\text{Input}) \qquad \dfrac{}{\overline{x}\langle y\rangle.P \overset{\overline{x}\langle y\rangle}{\Rrightarrow} P}\,(\text{Output})$$

$$\dfrac{P \overset{\overline{x}\langle y\rangle}{\Rrightarrow} P' \quad Q \overset{x(z)}{\Rrightarrow} Q'}{P \mid Q \overset{\tau}{\Rrightarrow} P' \mid Q'[y/z]}\,(\text{Com1}) \qquad \dfrac{P \overset{x(z)}{\Rrightarrow} P' \quad Q \overset{\overline{x}\langle y\rangle}{\Rrightarrow} Q'}{P \mid Q \overset{\tau}{\Rrightarrow} P'[y/z] \mid Q'}\,(\text{Com2})$$

$$\dfrac{P \overset{\alpha}{\Rrightarrow} P'}{P \mid Q \overset{\alpha}{\Rrightarrow} P' \mid Q}\,(\text{LPar}) \qquad \dfrac{Q \overset{\alpha}{\Rrightarrow} Q'}{P \mid Q \overset{\alpha}{\Rrightarrow} P \mid Q'}\,(\text{RPar})$$

$$\dfrac{P \overset{\alpha}{\Rrightarrow} P'}{[x\ is\ x].P \overset{\alpha}{\Rrightarrow} P'}\,(\text{Match}) \qquad \dfrac{P \overset{\alpha}{\Rrightarrow} P' \quad \alpha \notin \{m, \overline{m}\}}{(\nu\, m)\, P \overset{\alpha}{\Rrightarrow} (\nu\, m)\, P'}\,(\text{Res}) \qquad \dfrac{P \mid !P \overset{\alpha}{\Rrightarrow} P'}{!P \overset{\alpha}{\Rrightarrow} P'}\,(\text{Bang})$$

We are now ready to define the *commitment over configurations*.

Definition 3.12. We define the *commitment relation over configurations*, $\longrightarrow \subseteq \Xi \times \Xi$, as the least relation over configurations defined by the rule

$$\dfrac{(\sigma, P) \rhd^* (\sigma', P') \quad P' \overset{\alpha}{\Rrightarrow} P'' \quad (\sigma', P'') \rhd^* (\sigma'', P''')}{(\sigma, P) \overset{\alpha}{\longrightarrow} (\sigma'', P''')}\,(\text{Commit})$$

3.4 Strong Bisimulation

As said in the introduction we present in this section the first notion of equivalence, Definition 3.15. We do this in the standard way, that is, we define what is intended to be a *strong*

simulation of the configuration (σ_1, P) by the configuration (σ_2, Q), Definition 3.13. Then we say that two configurations are *strong bisimilar* if they simulate each other, Definition 3.14. Finally, we have that two configurations (σ_1, P) and (σ_2, Q) are strong equivalent if the pair $((\sigma_1, P), (\sigma_2, Q))$ is in some *strong bisimulation*. Intuitively this first notion of equivalence, says that the configuration (σ_2, Q) simulates (σ_1, P), for a principal A, if (σ_2, Q) can match all the transitions of (σ_1, P) and in the end both states are equal for the principal A.

Definition 3.13. A binary relation $\mathcal{S}$ over configurations, $\mathcal{S} \subseteq \Xi \times \Xi$, is an *A-strong simulation*, $A \in \mathcal{A}$, if, whenever $\big((\sigma_1, P), (\sigma_2, Q)\big) \in \mathcal{S}$,

(a) $(\sigma_1)_A = (\sigma_2)_A$;

(b) if $(\sigma_1, P) \rhd^* (\sigma_1', P')$ then there exists $(\sigma_2', Q') \in \Xi$ such that

$$(\sigma_2, Q) \rhd^* (\sigma_2', Q') \text{ and } ((\sigma_1', P'), (\sigma_2', Q')) \in \mathcal{S};$$

(c) if $(\sigma_1, P) \xrightarrow{\alpha} (\sigma_1', P')$ then there exists $(\sigma_2', Q') \in \Xi$ such that

$$(\sigma_2, Q) \xrightarrow{\alpha} (\sigma_2', Q') \text{ and } ((\sigma_1', P'), (\sigma_2', Q')) \in \mathcal{S}.$$

Notation 7. From now on we use $(\sigma_1, P)\mathcal{S}(\sigma_2, Q)$ as short for $\big((\sigma_1, P), (\sigma_2, Q)\big) \in \mathcal{S}$.

In general, we define the *converse* $\mathcal{S}^{-1}$ of a binary relation and the *composition* $\mathcal{S}_1\mathcal{S}_2$ of two binary relations as

$$\mathcal{S}^{-1} = \{(y, x) : x\mathcal{S}y\};$$
$$\mathcal{S}_1\mathcal{S}_2 = \{(x, z) : \text{exists } y, x\mathcal{S}_1 y \text{ and } y\mathcal{S}_2 z\}.$$

Definition 3.14. A binary relation $\mathcal{S}$ over configurations, $\mathcal{S} \subseteq \Xi \times \Xi$, is an *A-strong bisimulation*, $A \in \mathcal{A}$, if, both $\mathcal{S}$ and $\mathcal{S}^{-1}$ are A-strong simulations.

Now, we prove some properties of the A-strong bisimulations.

Proposition 3.1. *Let $\mathcal{S}_i$, $(i = 1, 2, \dots)$ be a family of A-strong bisimulations. Then the following relations are all A-strong bisimulations:*

(1) $\mathcal{I}d_\Xi$ (2) $\mathcal{S}_i^{-1}$
(3) $\mathcal{S}_i\mathcal{S}_j$ (4) $\mathcal{S} = \bigcup_{i \in I} \mathcal{S}_i$.

Proof. We only prove (3) and (4), because (1) is trivially verified and (2) is direct from the Definition 3.14. We start proving (3). Suppose that $(\sigma_1, P)\mathcal{S}_i\mathcal{S}_j(\sigma_3, R)$. This means that for some (σ_2, Q),

$$(\sigma_1, P)\mathcal{S}_i(\sigma_2, Q) \text{ and} \tag{3.1a}$$
$$(\sigma_2, Q)\mathcal{S}_j(\sigma_3, R). \tag{3.1b}$$

We start proving that,

$$(\sigma_1)_A = (\sigma_3)_A. \tag{$\star$}$$

Since both $\mathcal{S}_i$ and $\mathcal{S}_j$ are A-strong bisimulations, we have that

$$(\sigma_1)_A = (\sigma_2)_A \text{ and } (\sigma_2)_A = (\sigma_3)_A$$

This proves ($\star$). Now, we only prove the conditions (b) and (c) of Definition 3.13, definition of A-simulation, since the proof for $\mathcal{S}^{-1}$ is similar. We want to prove that

$$\begin{cases} \text{if } (\sigma_1, P) \rhd^* (\sigma_1', P') \text{ then there exists } (\sigma_3', R') \in \Xi \text{ such that} \\ \qquad\qquad (\sigma_3, R) \rhd^* (\sigma_3', R') \text{ and } (\sigma_1', P') \mathcal{S}_i \mathcal{S}_j (\sigma_3', R'); \\ \text{if } (\sigma_1, P) \xrightarrow{\alpha} (\sigma_1', P') \text{ then there exists } (\sigma_3', R') \in \Xi \text{ such that} \\ \qquad\qquad (\sigma_3, R) \xrightarrow{\alpha} (\sigma_3', R') \text{ and } (\sigma_1', P') \mathcal{S}_i \mathcal{S}_j (\sigma_3', R'). \end{cases} \tag{$*$}$$

Suppose that

$$(\sigma_1, P) \rhd^* (\sigma_1', P').$$

Since $\mathcal{S}_i$ is an A-strong bisimulation, $(\sigma_1, P) \mathcal{S}_i (\sigma_2, Q)$ and $(\sigma_1, P) \rhd^* (\sigma_1', P')$ we have that there exists $(\sigma_2', Q') \in \Xi$ such that

$$(\sigma_2, Q) \rhd^* (\sigma_2', Q') \text{ and } (\sigma_1', P') \mathcal{S}_i (\sigma_2', Q'). \tag{3.1c}$$

Since $\mathcal{S}_j$ is an A-strong bisimulation, $(\sigma_2, Q) \mathcal{S}_j (\sigma_3, R)$ and $(\sigma_2, Q) \rhd^* (\sigma_2', Q')$ we have that there exists $(\sigma_3', R') \in \Xi$ such that

$$(\sigma_3, R) \rhd^* (\sigma_3', R') \text{ and } (\sigma_2', Q') \mathcal{S}_j (\sigma_3', R'). \tag{3.1d}$$

So using (3.1c) and (3.1d) we have that there exists $(\sigma_3', R') \in \Xi$ such that

$$(\sigma_3, R) \rhd^* (\sigma_3', R') \text{ and } (\sigma_1', P') \mathcal{S}_i \mathcal{S}_j (\sigma_3', R').$$

This proves (b). To prove (c) we start supposing that

$$(\sigma_1, P) \xrightarrow{\alpha} (\sigma_1', P').$$

Since $\mathcal{S}_i$ is an A-strong bisimulation, $(\sigma_1, P) \mathcal{S}_i (\sigma_2, Q)$ and $(\sigma_1, P) \xrightarrow{\alpha} (\sigma_1', P')$ we have that there exists $(\sigma_2', Q') \in \Xi$ such that

$$(\sigma_2, Q) \xrightarrow{\alpha} (\sigma_2', Q') \text{ and } (\sigma_1', P') \mathcal{S}_i (\sigma_2', Q'). \tag{3.1e}$$

Since $\mathcal{S}_j$ is an A-strong bisimulation, $(\sigma_2, Q) \mathcal{S}_j (\sigma_3, R)$ and $(\sigma_2, Q) \xrightarrow{\alpha} (\sigma_2', Q')$ we have that there exists $(\sigma_3', R') \in \Xi$ such that

$$(\sigma_3, R) \xrightarrow{\alpha} (\sigma_3', R') \text{ and } (\sigma_2', Q') \mathcal{S}_j (\sigma_3', R'). \tag{3.1f}$$

So using (3.1e) and (3.1f) we have that there exists $(\sigma_3', R') \in \Xi$ such that

$$(\sigma_3, R) \xrightarrow{\alpha} (\sigma_3', R') \text{ and } (\sigma_1', P') \mathcal{S}_i \mathcal{S}_j (\sigma_3', R').$$

We have proved ($*$). This states that the composition of A-strong simulations is an A-strong simulation. Since $\mathcal{S}^{-1}$ is also an A-strong simulation, we have that the composition of A-strong bisimulations is an A-strong bisimulation.

To prove (4), all that we need is to observe that if $(\sigma_1, P) \mathcal{S} (\sigma_2, Q)$ then

$$(\sigma_1, P) \mathcal{S}_i (\sigma_2, Q) \text{ for some } i \in I.$$

From this, we notice immediately that $(\sigma_1)_A = (\sigma_2)_A$.
Similarly to the item (3) we only prove the result for A-simulations. The extension for A-bisimulations is obvious since both $\mathcal{S}$ and $\mathcal{S}^{-1}$ are A-simulations. Let us suppose that

$$(\sigma_1, P) \rhd^* (\sigma_1', P').$$

Since each $\mathcal{S}_i$ is an A-strong bisimulation, $(\sigma_1, P)\mathcal{S}_i(\sigma_2, Q)$ and $(\sigma_1, P) \rhd^* (\sigma_1', P')$ we have that there exists $(\sigma_2', Q') \in \Xi$ such that

$$(\sigma_2, Q) \rhd^* (\sigma_2', Q') \text{ and } (\sigma_1', P')\mathcal{S}_i(\sigma_2', Q').$$

Therefore, exists $(\sigma_2', Q') \in \Xi$ such that

$$(\sigma_2, Q) \rhd^* (\sigma_2', Q') \text{ and } (\sigma_1', P')\mathcal{S}(\sigma_2', Q').$$

This proves (b) of the definition of A-simulations. To prove (c) we start by supposing that

$$(\sigma_1, P) \xrightarrow{\alpha} (\sigma_1', P').$$

Since each $\mathcal{S}_i$ is an A-strong bisimulation, $(\sigma_1, P)\mathcal{S}_i(\sigma_2, Q)$ and $(\sigma_1, P) \xrightarrow{\alpha} (\sigma_1', P')$ we have that there exists $(\sigma_2', Q') \in \Xi$ such that

$$(\sigma_2, Q) \xrightarrow{\alpha} (\sigma_2', Q') \text{ and } (\sigma_1', P')\mathcal{S}_i(\sigma_2', Q').$$

Therefore, exists $(\sigma_2', Q') \in \Xi$ such that

$$(\sigma_2, Q) \xrightarrow{\alpha} (\sigma_2', Q') \text{ and } (\sigma_1', P')\mathcal{S}(\sigma_2', Q').$$

We have proved that the union of A-strong simulations is an A-strong simulation. The result is naturally extended to A-strong bisimulations, so $\mathcal{S} = \bigcup_{i\in I} \mathcal{S}_i$ is also an A-strong bisimulation. □

We are now ready to present our (first) notion of equivalence.

Definition 3.15. We say that two configurations (σ_1, P) and (σ_2, Q) are strong equivalent to a principal A or *A-strong equivalent*, and write $(\sigma_1, P) \simeq_A (\sigma_2, Q)$, if $(\sigma_1, P)\mathcal{S}(\sigma_2, Q)$ for some A-strong bisimulation $\mathcal{S}$. In other terms we define

$$\simeq_A = \bigcup \{\mathcal{S} : \mathcal{S} \text{ is an } A\text{-strong bisimulation}\}.$$

Proposition 3.2.

(1) $\simeq_A$ is the greatest A-strong bisimulation;

(2) $\simeq_A$ is an equivalence relation.

Proof. (1) is straightforward from Proposition 3.1. We just use the fact that the union of A-strong bisimulations is also an A-strong bisimulation to justify that $\simeq_A$ is an A-strong bisimulation. By the definition of $\simeq_A$ we have that it is the greatest A-strong bisimulation.

(2) is also a consequence of Proposition 3.1.

Reflexivity $(\sigma, P) \simeq_A (\sigma, P)$ because $\mathcal{I}d_\Xi$ is an A-strong bisimulation.

Symmetry If $(\sigma_1, P) \simeq_A (\sigma_2, Q)$ then $(\sigma_2, Q) \simeq_A (\sigma_1, P)$.

$$\begin{array}{lll} \text{If } (\sigma_1, P) \simeq_A (\sigma_2, Q) \text{ then } (\sigma_1, P)\mathcal{S}(\sigma_2, Q) & & \text{for some } A\text{-strong bisimulation } \mathcal{S} \\ \text{then } (\sigma_2, Q)\mathcal{S}^{-1}(\sigma_1, P) & & \\ \text{then } (\sigma_2, Q) \simeq_A (\sigma_1, P) & & \mathcal{S}^{-1} \text{ is an } A\text{-strong bisimulation.} \end{array}$$

Transitivity If $(\sigma_1, P) \simeq_A (\sigma_2, Q)$ and $(\sigma_2, Q) \simeq_A (\sigma_3, R)$ then $(\sigma_1, P) \simeq_A (\sigma_3, R)$.

$$\begin{array}{ll} \text{If } (\sigma_1, P) \simeq_A (\sigma_2, Q) \text{ then } (\sigma_1, P)\mathcal{S}_i(\sigma_2, Q) & \text{for some } A\text{-strong bisimulation } \mathcal{S}_i \\ \text{If } (\sigma_2, Q) \simeq_A (\sigma_3, R) \text{ then } (\sigma_2, Q)\mathcal{S}_j(\sigma_3, R) & \text{for some } A\text{-strong bisimulation } \mathcal{S}_j \end{array}$$

Since $\mathcal{S}_i\mathcal{S}_j$ is an A-strong bisimulation and $(\sigma_1, P)\mathcal{S}_i\mathcal{S}_j(\sigma_3, R)$, we have that

$$(\sigma_1, P) \simeq_A (\sigma_3, R). \qquad \square$$

Definition 3.16. A binary relation $\mathcal{S}$ over configurations, $\mathcal{S} \subseteq \Xi \times \Xi$, is an *$A$-strong bisimulation up to $\simeq_A$* if, whenever $(\sigma_1, P)\mathcal{S}(\sigma_2, Q)$,

(a) $(\sigma_1)_A = (\sigma_2)_A$;

(b) if $(\sigma_1, P) \rhd^* (\sigma_1', P')$ then there exists $(\sigma_2', Q') \in \Xi$ such that

$$(\sigma_2, Q) \rhd^* (\sigma_2', Q') \text{ and } (\sigma_1', P') \simeq_A \mathcal{S} \simeq_A (\sigma_2', Q');$$

(c) if $(\sigma_1, P) \xrightarrow{\alpha} (\sigma_1', P')$ then there exists $(\sigma_2', Q') \in \Xi$ such that

$$(\sigma_2, Q) \xrightarrow{\alpha} (\sigma_2', Q') \text{ and } (\sigma_1', P') \simeq_A \mathcal{S} \simeq_A (\sigma_2', Q');$$

(d) if $(\sigma_2, Q) \rhd^* (\sigma_2', Q')$ then there exists $(\sigma_1', P') \in \Xi$ such that

$$(\sigma_1, P) \rhd^* (\sigma_1', P') \text{ and } (\sigma_1', P') \simeq_A \mathcal{S} \simeq_A (\sigma_2', Q');$$

(e) if $(\sigma_2, Q) \xrightarrow{\alpha} (\sigma_2', Q')$ then there exists $(\sigma_1', P') \in \Xi$ such that

$$(\sigma_1, P) \xrightarrow{\alpha} (\sigma_1', P') \text{ and } (\sigma_1', P') \simeq_A \mathcal{S} \simeq_A (\sigma_2', Q').$$

Proposition 3.3. *If $\mathcal{S}$ is an A-strong bisimulation up to $\simeq_A$ and $(\sigma_1, P)\mathcal{S}(\sigma_2, Q)$ then $(\sigma_1, P) \simeq_A (\sigma_2, Q)$.*

Proof. We start by proving that $\simeq_A \mathcal{S} \simeq_A$ is an A-strong bisimulation, i.e., $\simeq_A \mathcal{S} \simeq_A \subseteq \simeq_A$.

Suppose that $(\sigma_1, P) \simeq_A \mathcal{S} \simeq_A (\sigma_2, Q)$. This means that there exists (σ_3, R) and (σ_4, T) such that

$$(\sigma_1, P) \simeq_A (\sigma_3, R)\mathcal{S}(\sigma_4, T) \simeq_A (\sigma_2, Q).$$

It is easy to prove that $(\sigma_1)_A = (\sigma_2)_A$. For that just notice that $(\sigma_1)_A = (\sigma_3)_A$ and $(\sigma_4)_A = (\sigma_2)_A$, because $\simeq_A$ is an A-strong bisimulation. Since $\mathcal{S}$ is an A-strong bisimulation up to $\simeq_A$, we have that $(\sigma_3)_A = (\sigma_4)_A$. So using transitivity we have $(\sigma_1)_A = (\sigma_2)_A$.

Now we have to prove (b), that is, if $(\sigma_1, P) \rhd^* (\sigma_1', P')$ then there exists $(\sigma_2', Q') \in \Xi$ such that

$$(\sigma_2, Q) \rhd^* (\sigma_2', Q') \text{ and } (\sigma_1', P') \mathcal{S} (\sigma_2', Q').$$

Suppose that $(\sigma_1, P) \rhd^* (\sigma_1', P')$. To illustrate the proof we sketch the following diagram:

$$\begin{array}{ccccc}
(\sigma_1, P) & \simeq_A & (\sigma_3, R) & \qquad (\sigma_3, R) \ \mathcal{S} \ (\sigma_4, T) \qquad & (\sigma_4, T) \simeq_A (\sigma_2, Q) \\
\triangledown_* & & \triangledown_* & \qquad *\triangle \qquad\qquad \triangle* \qquad & \triangledown_* \qquad \triangledown_* \\
(\sigma_1', P') & \simeq_A & (\sigma_3', R') & (\sigma_3', R') \simeq_A (\sigma_3'', R'') \ \mathcal{S} \ (\sigma_4'', T'') \simeq_A (\sigma_4', T') & (\sigma_4', T') \simeq_A (\sigma_2', Q')
\end{array}$$

Since $(\sigma_1, P) \simeq_A (\sigma_3, R)$ and $(\sigma_1, P) \rhd^* (\sigma_1', P')$, there exists (σ_3', R') such that

$$(\sigma_3, R) \rhd^* (\sigma_3', R') \text{ and } (\sigma_1', P') \simeq_A (\sigma_3', R'). \tag{3.2a}$$

Since $\mathcal{S}$ is an A-strong bisimulation up to $\simeq_A$, $(\sigma_3, R) \mathcal{S} (\sigma_4, T)$ and $(\sigma_3, R) \rhd^* (\sigma_3', R')$, there exists (σ_4', T') such that

$$(\sigma_4, T) \rhd^* (\sigma_4', T') \text{ and } (\sigma_3', R') \simeq_A (\sigma_3'', R'') \mathcal{S} (\sigma_4'', T'') \simeq_A (\sigma_4', T'). \tag{3.2b}$$

Finally, since $(\sigma_4, T) \simeq_A (\sigma_2, Q)$ and $(\sigma_4, T) \rhd^* (\sigma_4', T')$, we have that there exists (σ_2', Q') such that

$$(\sigma_2, Q) \rhd^* (\sigma_2', Q') \text{ and } (\sigma_4', T') \simeq_A (\sigma_2', Q'). \tag{3.2c}$$

By (3.2a), (3.2b) and (3.2c) we have that there exists (σ_2', Q') such that

$$(\sigma_2, Q) \rhd^* (\sigma_2', Q') \text{ and } (\sigma_1', P') \simeq_A (\sigma_3', R') \simeq_A (\sigma_3'', R'') \mathcal{S} (\sigma_4'', T'') \simeq_A (\sigma_4', T') \simeq_A (\sigma_2', Q').$$

Since $\simeq_A$ is an equivalence relation, in particular it is transitive, hence we obtain

$$(\sigma_2, Q) \rhd^* (\sigma_2', Q') \text{ and } (\sigma_1', P') \simeq_A (\sigma_3'', R'') \mathcal{S} (\sigma_4'', T'') \simeq_A (\sigma_2', Q'),$$

which means that there exists (σ_2', Q') such that

$$(\sigma_2, Q) \rhd^* (\sigma_2', Q') \text{ and } (\sigma_1', P') \simeq_A \mathcal{S} \simeq_A (\sigma_2', Q').$$

Now we prove (c), that is, if $(\sigma_1, P) \xrightarrow{\alpha} (\sigma_1', P')$ then there exists $(\sigma_2', Q') \in \Xi$ such that

$$(\sigma_2, Q) \xrightarrow{\alpha} (\sigma_2', Q') \text{ and } (\sigma_1', P') \mathcal{S} (\sigma_2', Q').$$

So, suppose that $(\sigma_1, P) \xrightarrow{\alpha} (\sigma_1', P')$. To illustrate the proof we sketch another diagram:

$$\begin{array}{ccccc}
(\sigma_1, P) & \simeq_A & (\sigma_3, R) & \qquad (\sigma_3, R) \ \mathcal{S} \ (\sigma_4, T) \qquad & (\sigma_4, T) \simeq_A (\sigma_2, Q) \\
\downarrow \alpha & & \downarrow \alpha & \qquad \swarrow \alpha \qquad\qquad \searrow \alpha \qquad & \downarrow \alpha \qquad \downarrow \alpha \\
(\sigma_1', P') & \simeq_A & (\sigma_3', R') & (\sigma_3', R') \simeq_A (\sigma_3'', R'') \ \mathcal{S} \ (\sigma_4'', T'') \simeq_A (\sigma_4', T') & (\sigma_4', T') \simeq_A (\sigma_2', Q')
\end{array}$$

Since $(\sigma_1, P) \simeq_A (\sigma_3, R)$ and $(\sigma_1, P) \xrightarrow{\alpha} (\sigma_1', P')$, there exists (σ_3', R') such that

$$(\sigma_3, R) \xrightarrow{\alpha} (\sigma_3', R') \text{ and } (\sigma_1', P') \simeq_A (\sigma_3', R'). \tag{3.2d}$$

Since $\mathcal{S}$ is an A-strong bisimulation up to $\simeq_A$, $(\sigma_3, R)\mathcal{S}(\sigma_4, T)$ and $(\sigma_3, R) \xrightarrow{\alpha} (\sigma_3', R')$, there exists (σ_4', T') such that

$$(\sigma_4, T) \xrightarrow{\alpha} (\sigma_4', T') \text{ and } (\sigma_3', R') \simeq_A (\sigma_3'', R'')\mathcal{S}(\sigma_4'', T'') \simeq_A (\sigma_4', T'). \tag{3.2e}$$

Finally, since $(\sigma_4, T) \simeq_A (\sigma_2, Q)$ and $(\sigma_4, T) \xrightarrow{\alpha} (\sigma_4', T')$, we have that there exists (σ_2', Q') such that

$$(\sigma_2, Q) \xrightarrow{\alpha} (\sigma_2', Q') \text{ and } (\sigma_4', T') \simeq_A (\sigma_2', Q'). \tag{3.2f}$$

By (3.2d), (3.2e) and (3.2f) we have that there exists (σ_2', Q') such that

$$(\sigma_2, Q) \xrightarrow{\alpha} (\sigma_2', Q') \text{ and } (\sigma_1', P') \simeq_A (\sigma_3', R') \simeq_A (\sigma_3'', R'')\mathcal{S}(\sigma_4'', T'') \simeq_A (\sigma_4', T') \simeq_A (\sigma_2', Q').$$

Since $\simeq_A$ is an equivalence relation, in particular it is transitive, hence we obtain

$$(\sigma_2, Q) \xrightarrow{\alpha} (\sigma_2', Q') \text{ and } (\sigma_1', P') \simeq_A (\sigma_3'', R'')\mathcal{S}(\sigma_4'', T'') \simeq_A (\sigma_2', Q'),$$

which means that there exists (σ_2', Q') such that

$$(\sigma_2, Q) \xrightarrow{\alpha} (\sigma_2', Q') \text{ and } (\sigma_1', P') \simeq_A \mathcal{S} \simeq_A (\sigma_2', Q').$$

The proof that $(\sigma_2, Q) \simeq_A \mathcal{S} \simeq_A (\sigma_1, P)$ implies $(\sigma_2, Q) \simeq_A (\sigma_1, P)$, conditions (d) and (e), is analogous so, we skip it. So far, we have proved that $\simeq_A \mathcal{S} \simeq_A$ is an A-strong bisimulation, so if $(\sigma_1, P) \simeq_A \mathcal{S} \simeq_A (\sigma_2, Q)$ then $(\sigma_1, P) \simeq_A (\sigma_2, Q)$.

It is trivial to prove that if $(\sigma_1, P)\mathcal{S}(\sigma_2, Q)$ then $(\sigma_1, P) \simeq_A \mathcal{S} \simeq_A (\sigma_2, Q)$, just notice that $\mathcal{I}d_\Xi$ is an A-strong bisimulation. With this final remark we prove the proposition. □

3.5 Weak Bisimulation

In the previous section we introduced a notion of equivalence that is sensible to all kinds of transitions. In this section we introduce a different notion that in not sensible to τ transitions. This approach turns possible the comparison of configurations that exhibit the same behaviour up to τ transitions. This is the usual notion of *weak bisimulation*. This will be our equivalence notion because we want to compare configurations that might have different τ transitions, but always exhibit the same state for the bank. Let us start with the definition of *weak transition* and then define, as in the previous section, *A-weak simulation*, *A-weak bisimulation* and *A-weak equivalence*.

Definition 3.17. We define the *weak commitment relation over configurations*, $\Longrightarrow \subseteq \Xi \times \Xi$, as

$$(\sigma, P) \Longrightarrow (\sigma', P') \text{ if } \exists_{n \geq 0} \quad (\sigma, P) \xrightarrow{\tau^n} (\sigma', P');$$
$$(\sigma, P) \overset{\alpha}{\Longrightarrow} (\sigma', P') \text{ if } \exists_{n_1 \geq 0, n_2 \geq 0} \quad (\sigma, P) \xrightarrow{\tau^{n_1}} \xrightarrow{\alpha} \xrightarrow{\tau^{n_2}} (\sigma', P').$$

Definition 3.18. A binary relation $\mathcal{S}$ over configurations, $\mathcal{S} \subseteq \Xi \times \Xi$, is an *A-weak simulation*, $A \in \mathcal{A}$, if, whenever $(\sigma_1, P)\mathcal{S}(\sigma_2, Q)$,

(a) $(\sigma_1)_A = (\sigma_2)_A$;

(b) if $(\sigma_1, P) \rhd^* (\sigma_1', P')$ then there exists $(\sigma_2', Q') \in \Xi$ such that

$$(\sigma_2, Q) \rhd^* (\sigma_2', Q') \text{ and } (\sigma_1', P') \mathcal{S} (\sigma_2', Q');$$

(c) if $(\sigma_1, P) \Longrightarrow (\sigma_1', P')$ then there exists $(\sigma_2', Q') \in \Xi$ such that

$$(\sigma_2, Q) \Longrightarrow (\sigma_2', Q') \text{ and } (\sigma_1', P') \mathcal{S} (\sigma_2', Q');$$

(d) if $(\sigma_1, P) \stackrel{\alpha}{\Longrightarrow} (\sigma_1', P')$ then there exists $(\sigma_2', Q') \in \Xi$ such that

$$(\sigma_2, Q) \stackrel{\alpha}{\Longrightarrow} (\sigma_2', Q') \text{ and } (\sigma_1', P') \mathcal{S} (\sigma_2', Q').$$

Definition 3.19. A binary relation $\mathcal{S}$ over configurations, $\mathcal{S} \subseteq \Xi \times \Xi$, is an *A-weak bisimulation* if both $\mathcal{S}$ and $\mathcal{S}^{-1}$ are A-weak simulations.

We may now compare A-strong and A-weak bisimulations. We state that A-strong bisimulation is strictly coarser than A-weak bisimulation, Proposition 3.6. To do this we prove the following lemma:

Lemma 3.4. *A binary relation $\mathcal{S}$ over configurations, $\mathcal{S} \subseteq \Xi \times \Xi$, is an A-weak simulation if and only if, whenever $(\sigma_1, P) \mathcal{S} (\sigma_2, Q)$,*

(1) $(\sigma_1)_A = (\sigma_2)_A$*;*

(2) if $(\sigma_1, P) \rhd^* (\sigma_1', P')$ *then there exists* $(\sigma_2', Q') \in \Xi$ *such that*

$$(\sigma_2, Q) \rhd^* (\sigma_2', Q') \text{ and } (\sigma_1', P') \mathcal{S} (\sigma_2', Q');$$

(3) if $(\sigma_1, P) \stackrel{\tau}{\longrightarrow} (\sigma_1', P')$ *then there exists* $(\sigma_2', Q') \in \Xi$ *such that*

$$(\sigma_2, Q) \Longrightarrow (\sigma_2', Q') \text{ and } (\sigma_1', P') \mathcal{S} (\sigma_2', Q');$$

(4) if $(\sigma_1, P) \stackrel{\alpha}{\longrightarrow} (\sigma_1', P')$ *then there exists* $(\sigma_2', Q') \in \Xi$ *such that*

$$(\sigma_2, Q) \stackrel{\alpha}{\Longrightarrow} (\sigma_2', Q') \text{ and } (\sigma_1', P') \mathcal{S} (\sigma_2', Q').$$

Proof. ($\Longrightarrow$) Suppose that $\mathcal{S}$ is an A-weak simulation and that $(\sigma_1, P) \mathcal{S} (\sigma_2, Q)$. It is obvious that $(\sigma_1)_A = (\sigma_2)_A$ because $\mathcal{S}$ is an A-weak simulation.

To prove (2) we start supposing that $(\sigma_1, P) \rhd^* (\sigma_1', P')$. Since $\mathcal{S}$ is an A-weak simulation and $(\sigma_1, P) \mathcal{S} (\sigma_2, Q)$, there exists $(\sigma_2', Q') \in \Xi$ such that

$$(\sigma_2, Q) \rhd^* (\sigma_2', Q') \text{ and } (\sigma_1', P') \mathcal{S} (\sigma_2', Q').$$

To prove (3) we start by assuming that $(\sigma_1, P) \stackrel{\tau}{\longrightarrow} (\sigma_1', P')$. By definition of weak-commitment we have that $(\sigma_1, P) \Longrightarrow (\sigma_1', P')$, $(n = 1)$. So, since $\mathcal{S}$ is an A-weak simulation, $(\sigma_1, P) \mathcal{S} (\sigma_2, Q)$ and $(\sigma_1, P) \Longrightarrow (\sigma_1', P')$, we have that there exists $(\sigma_2', Q') \in \Xi$ such that

$$(\sigma_2, Q) \Longrightarrow (\sigma_2', Q') \text{ and } (\sigma_1', P') \mathcal{S} (\sigma_2', Q').$$

To prove (4) we suppose that $(\sigma_1, P) \stackrel{\alpha}{\longrightarrow} (\sigma_1', P')$. By definition of weak-commitment we have that $(\sigma_1, P) \stackrel{\alpha}{\Longrightarrow} (\sigma_1', P')$, $(n_1 = 0, n_2 = 0)$. So, since $\mathcal{S}$ is an A-weak simulation,

$(\sigma_1, P)\mathcal{S}(\sigma_2, Q)$ and $(\sigma_1, P) \stackrel{\alpha}{\Longrightarrow} (\sigma_1', P')$, we have, by hypothesis, that there exists $(\sigma_2', Q') \in \Xi$ such that

$$(\sigma_2, Q) \stackrel{\alpha}{\Longrightarrow} (\sigma_2', Q') \text{ and } (\sigma_1', P')\mathcal{S}(\sigma_2', Q').$$

($\Longleftarrow$) Suppose now that $\mathcal{S}$ is a binary relation that satisfies the four conditions above and that $(\sigma_1, P)\mathcal{S}(\sigma_2, Q)$. The proofs of the conditions (a) and (b) of the A-weak simulation definition are obvious.

To prove the condition (c) of the A-weak simulation definition we start by assuming that $(\sigma_1, P) \Longrightarrow (\sigma_1', P')$. We have to prove that in this conditions, there exists $(\sigma_2', Q') \in \Xi$ such that

$$(\sigma_2, Q) \Longrightarrow (\sigma_2', Q') \text{ and } (\sigma_1', P')\mathcal{S}(\sigma_2', Q').$$

We will prove this by induction in the number of τ's.

- $n = 0$ — This is the case when $(\sigma_1', P') = (\sigma_1, P)$. It is obvious that $(\sigma_2', Q') \in \Xi$ exists, just consider that $(\sigma_2', Q') = (\sigma_2, Q)$ and $(\sigma_2, Q) \Longrightarrow (\sigma_2', Q')$ by 0 τ's;

- $n = 1$ — $(\sigma, P) \stackrel{\tau}{\longrightarrow} (\sigma', P')$. Since $(\sigma_1, P)\mathcal{S}(\sigma_2, Q)$, by hypothesis (3) there exists $(\sigma_2', Q') \in \Xi$ such that

  $$(\sigma_2, Q) \Longrightarrow (\sigma_2', Q') \text{ and } (\sigma_1', P')\mathcal{S}(\sigma_2', Q');$$

- $n+1$ — $(\sigma, P) \stackrel{\tau^n}{\longrightarrow} (\sigma'', P'') \stackrel{\tau}{\longrightarrow} (\sigma', P')$. Since $(\sigma_1, P)\mathcal{S}(\sigma_2, Q)$, by induction hypothesis, there exists $(\sigma_2'', Q'') \in \Xi$ such that

  $$(\sigma_2, Q) \Longrightarrow (\sigma_2'', Q'') \text{ and } (\sigma_1'', P'')\mathcal{S}(\sigma_2'', Q'').$$

  Using this last fact, $(\sigma_1'', P'')\mathcal{S}(\sigma_2'', Q'')$, and $(\sigma'', P'') \stackrel{\tau}{\longrightarrow} (\sigma', P')$, we have, by (3), that there exists $(\sigma_2', Q') \in \Xi$ such that

  $$(\sigma_2'', Q'') \Longrightarrow (\sigma_2', Q') \text{ and } (\sigma_1', P')\mathcal{S}(\sigma_2', Q').$$

  Since $(\sigma_2, Q) \Longrightarrow (\sigma_2'', Q'')$ and $(\sigma_2'', Q'') \Longrightarrow (\sigma_2', Q')$, we have that $(\sigma_2, Q) \Longrightarrow (\sigma_2', Q')$, and we conclude that there exists $(\sigma_2', Q') \in \Xi$, such that

  $$(\sigma_2, Q) \Longrightarrow (\sigma_2', Q') \text{ and } (\sigma_1', P')\mathcal{S}(\sigma_2', Q').$$

To prove the condition (d) of the A-weak simulation definition we start supposing that $(\sigma_1, P) \stackrel{\alpha}{\Longrightarrow} (\sigma_1', P')$. This states that

$$\exists_{n_1 \geq 0, n_2 \geq 0} \quad (\sigma_1, P) \stackrel{\tau^{n_1}}{\longrightarrow} (\sigma_1'', P'') \stackrel{\alpha}{\longrightarrow} (\sigma_1''', P''') \stackrel{\tau^{n_2}}{\longrightarrow} (\sigma_1', P'),$$

which is equivalent, by definition of weak commitment, to

$$(\sigma_1, P) \Longrightarrow (\sigma_1'', P'') \stackrel{\alpha}{\longrightarrow} (\sigma_1''', P''') \Longrightarrow (\sigma_1', P').$$

Since $(\sigma_1, P)\mathcal{S}(\sigma_2, Q)$ and $(\sigma_1, P) \Longrightarrow (\sigma_1'', P'')$, using (c), we have that there exists $(\sigma_2'', Q'') \in \Xi$ such that

$$(\sigma_2, Q) \Longrightarrow (\sigma_2'', Q'') \text{ and } (\sigma_1'', P'')\mathcal{S}(\sigma_2'', Q'').$$

Using the fact that $(\sigma_1'', P'')\mathcal{S}(\sigma_2'', Q'')$ and $(\sigma_1'', P'') \xrightarrow{\alpha} (\sigma_1''', P''')$, using hypothesis (4) we have that there exists $(\sigma_2''', Q''') \in \Xi$ such that

$$(\sigma_2'', Q'') \stackrel{\alpha}{\Longrightarrow} (\sigma_2''', Q''') \text{ and } (\sigma_1''', P''')\mathcal{S}(\sigma_2''', Q''').$$

Finally, using the fact that $(\sigma_1''', P''')\mathcal{S}(\sigma_2''', Q''')$ and $(\sigma_1''', P''') \Longrightarrow (\sigma_1', P')$, using again (c), we have that there exists $(\sigma_2', Q') \in \Xi$ such that

$$(\sigma_2''', Q''') \Longrightarrow (\sigma_2', Q') \text{ and } (\sigma_1', P')\mathcal{S}(\sigma_2', Q').$$

Now, we have that there exists $(\sigma_2', Q') \in \Xi$ such that

$$(\sigma_2, Q) \Longrightarrow (\sigma_2'', Q'') \stackrel{\alpha}{\Longrightarrow} (\sigma_2''', Q''') \Longrightarrow (\sigma_2', Q') \text{ and } (\sigma_1', P')\mathcal{S}(\sigma_2', Q').$$

So we have that there exists $(\sigma_2', Q') \in \Xi$ such that

$$(\sigma_2, Q) \stackrel{\alpha}{\Longrightarrow} (\sigma_2', Q') \text{ and } (\sigma_1', P')\mathcal{S}(\sigma_2', Q'). \qquad \square$$

With this lemma, to prove that (σ_1, P) is A-weak similar to (σ_2, Q), we only have to look at the strong transitions.

Proposition 3.5. *If $\mathcal{S}$ is an A-strong simulation, then $\mathcal{S}$ is an A-weak simulation.*

Proof. Let us suppose that $\mathcal{S}$ is an A-strong simulation and that $(\sigma_1, P)\mathcal{S}(\sigma_2, Q)$. We want to prove that $\mathcal{S}$ is an A-weak simulation. We will prove this using the characterization of A-weak simulations stated in Lemma 3.4. The conditions (1) and (2) of the lemma are straightforward because $\mathcal{S}$ is an A-strong simulation.

In order to prove condition (3), suppose that $(\sigma_1, P) \xrightarrow{\tau} (\sigma_1', P')$. Since $\mathcal{S}$ is an A-strong simulation and $(\sigma_1, P)\mathcal{S}(\sigma_2, Q)$, there exists $(\sigma_2', Q') \in \Xi$ such that

$$(\sigma_2, Q) \xrightarrow{\tau} (\sigma_2', Q') \text{ and } (\sigma_1', P')\mathcal{S}(\sigma_2', Q').$$

So using the definition of weak commitment, with $n = 1$, we have that there exists $(\sigma_2', Q') \in \Xi$ such that

$$(\sigma_2, Q) \Longrightarrow (\sigma_2', Q') \text{ and } (\sigma_1', P')\mathcal{S}(\sigma_2', Q').$$

The proof of (4) is similar. Let us suppose that $(\sigma_1, P) \xrightarrow{\alpha} (\sigma_1', P')$. Since $\mathcal{S}$ is an A-strong simulation and $(\sigma_1, P)\mathcal{S}(\sigma_2, Q)$, there exists $(\sigma_2', Q') \in \Xi$ such that

$$(\sigma_2, Q) \xrightarrow{\alpha} (\sigma_2', Q') \text{ and } (\sigma_1', P')\mathcal{S}(\sigma_2', Q').$$

So using the definition of weak commitment, with $n_1 = 0$ and $n_2 = 0$, we have that there exists $(\sigma_2', Q') \in \Xi$ such that

$$(\sigma_2, Q) \stackrel{\alpha}{\Longrightarrow} (\sigma_2', Q') \text{ and } (\sigma_1', P')\mathcal{S}(\sigma_2', Q'). \qquad \square$$

Corollary 3.6. *If $\mathcal{S}$ is an A-strong bisimulation, then $\mathcal{S}$ is an A-weak bisimulation.*

Proof. Straightforward from the Proposition 3.5, using the fact that both $\mathcal{S}$ and $\mathcal{S}^{-1}$ are A-strong simulations. $\square$

Proposition 3.7. *Let $\mathcal{S}_i$, $(i = 1, 2, \dots)$ be a family of A-weak bisimulations. Then the following relations are all A-weak bisimulations:*

$$(1)\ \mathcal{I}d_{\Xi} \qquad (2)\ \mathcal{S}_i^{-1}$$
$$(3)\ \mathcal{S}_i\mathcal{S}_j \qquad (4)\ \mathcal{S} = \bigcup_{i\in I} \mathcal{S}_i.$$

Proof. The proof is very similar to the proof of Proposition 3.1. We only have to substitute the strong transitions for weak transitions. □

Finally, we may define our notion of equivalence.

Definition 3.20. We say that two configurations (σ_1, P) and (σ_2, Q) are weak equivalent to a principal A or *A-weak equivalent*, and write $(\sigma_1, P)\cong_A(\sigma_2, Q)$, if $(\sigma_1, P)\mathcal{S}(\sigma_2, Q)$ for some A-weak bisimulation $\mathcal{S}$. In other terms we define

$$\cong_A = \bigcup\{\mathcal{S} : \mathcal{S} \text{ is an } A\text{-weak bisimulation}\}.$$

Corollary 3.8. $\simeq_A \subseteq \cong_A$.

Proof. Directly from Corollary 3.6. □

Proposition 3.9.

(1) $\cong_A$ is the greatest A-weak bisimulation;

(2) $\cong_A$ is an equivalence relation.

Proof. The proof is similar to the Proposition 3.2 using the result of Proposition 3.7. □

Now we are interested in having a result similar to the Proposition 3.3. For that we define what is an *A-weak bisimulation up to weak equivalence*.

Definition 3.21. A binary relation $\mathcal{S}$ over configurations, $\mathcal{S} \subseteq \Xi \times \Xi$, is an *A-weak bisimulation up to* $\cong_A$ if, whenever $(\sigma_1, P)\mathcal{S}(\sigma_2, Q)$,

(a) $(\sigma_1)_A = (\sigma_2)_A$;

(b) if $(\sigma_1, P) \rhd^* (\sigma_1', P')$ then there exists $(\sigma_2', Q') \in \Xi$ such that

$$(\sigma_2, Q) \rhd^* (\sigma_2', Q') \text{ and } (\sigma_1', P') \cong_A \mathcal{S} \cong_A (\sigma_2', Q');$$

(c) if $(\sigma_1, P) \Longrightarrow (\sigma_1', P')$ then there exists $(\sigma_2', Q') \in \Xi$ such that

$$(\sigma_2, Q) \Longrightarrow (\sigma_2', Q') \text{ and } (\sigma_1', P') \cong_A \mathcal{S} \cong_A (\sigma_2', Q');$$

(d) if $(\sigma_1, P) \stackrel{\alpha}{\Longrightarrow} (\sigma_1', P')$ then there exists $(\sigma_2', Q') \in \Xi$ such that

$$(\sigma_2, Q) \stackrel{\alpha}{\Longrightarrow} (\sigma_2', Q') \text{ and } (\sigma_1', P') \cong_A \mathcal{S} \cong_A (\sigma_2', Q');$$

(e) if $(\sigma_2, Q) \rhd^* (\sigma_2', Q')$ then there exists $(\sigma_1', P') \in \Xi$ such that

$$(\sigma_1, P) \rhd^* (\sigma_1', P') \text{ and } (\sigma_1', P') \simeq_A \mathcal{S} \simeq_A (\sigma_2', Q');$$

(f) if $(\sigma_2, Q) \Longrightarrow (\sigma_2', Q')$ then there exists $(\sigma_1', P') \in \Xi$ such that

$$(\sigma_1, P) \Longrightarrow (\sigma_1', P') \text{ and } (\sigma_1', P') \simeq_A \mathcal{S} \simeq_A (\sigma_2', Q');$$

(g) if $(\sigma_2, Q) \stackrel{\alpha}{\Longrightarrow} (\sigma_2', Q')$ then there exists $(\sigma_1', P') \in \Xi$ such that

$$(\sigma_1, P) \stackrel{\alpha}{\Longrightarrow} (\sigma_1', P') \text{ and } (\sigma_1', P') \simeq_A \mathcal{S} \simeq_A (\sigma_2', Q').$$

Proposition 3.10. *If $\mathcal{S}$ is an A-weak bisimulation up to $\cong_A$ and $(\sigma_1, P)\mathcal{S}(\sigma_2, Q)$ then $(\sigma_1, P) \cong_A (\sigma_2, Q)$.*

Proof. The proof is similar to the proof of Proposition 3.3. We only have to substitute the strong transitions for weak transitions, and strong equivalence for weak equivalence. □

Observation 2. Since any A-weak bisimulation up to $\cong_A$, $\mathcal{S}$, is an A-weak bisimulation, we can use the result of Lemma 3.4 when proving that $\mathcal{S}$ is an A-weak bisimulation.

Chapter 4

Electronic Money

4.1 Overview of the Problem

Nowadays, *e*-commerce is a strongly expanding universe. Due to efficiency and convenience, the number of customers is increasing exponentially. This expansion in number of customers has to be closely followed by a development of the payment methods, which need to be more secure and easier to use, and distribution techniques, which need to be faster and to have better delivery systems as well as good stock management. We will devote our attention to the payment methods' problem.

When we acquire any products through the Internet we usually have two methods of payment:

(a) Credit/debit cards;

(b) Cash on delivery (C.O.D).

While the former is more efficient and convenient, the latter is more secure (we only pay the parcel when it arrives and we do not send our credit card number over the Internet) — this leads us to a security/efficiency dilemma. At the present time *e*-commerce is just taking the first steps but, as we said before, the expected expansion of the market will, one day (sooner or later), cause serious problems if we do not solve this efficiency problem.

Besides the efficiency problem, the existing payment methods have some problems, for instance, when we buy something over the Internet using VISA, we have to pay a tax up to 2.5% for transaction fees. For the seller, the use of VISA is also an uncomfortable situation. Due to the possibility of fraud, a VISA transaction does not complete — meaning the seller does not get paid — for up to 90 days.

The ideal solution is to create a true digital payment method that provides lower transaction costs and immediate transaction processing.

To solve this problem several ideas have arise and among them the idea of *e*-coins [CFN90, CdBvH$^+$90]. Several papers have been published, namely [Bra93, Bra95, Bra99, Fer93a, Fer94, Fer93b, CP93, OO92, Oka95], but most of them incorporate the idea of using a tamper resistant smart card, or a similar hardware device, and the ones which do not use that device are not safe against double spending of coins. Our idea is, compromising some principles such as untraceability, to create *e*-money that can be used without any physical device. Without this physical device, the control has to be made by a regulator entity that has to be always

on-line. If, for some reason, the entity is off-line then, the service will be unavailable for that period.

We want each principal to have a virtual wallet with his *e*-coins. This coins should be represented by electronic files that can be *e*-mailed over the Internet. Similarly to the "physical" money, we must ensure that *e*-money fulfills some security properties, namely:

(a) Every note is issued by a bank, so *e*-money must also be issued by a trusted authority, such as an *e*-bank.

(b) Every note has a serial/identification number, so each *e*-coin must also have an identification that is unique among the other *e*-coins.

(c) Every note is, or is supposed to be, difficult to copy, but our *e*-coins will be represented by electronic files, which means they are easy to copy. So, besides the issuing process, our trusted authority must also be responsible for the regulation of the *e*-coins. Namely, the trusted authority must know, at any time, who is the owner of each coin.

(d) Every note cannot be spent more than once, so our *e*-coins must be protected against multi-spendable attacks, that is, a principal cannot copy an *e*-coin and then use both, the original and the copy.

One property that the usual money does not have, but we want to incorporate in our *e*-money, is the "anti-theft" property. For example if Oscar "steals" a coin from Bob, then Oscar will not be able to use it. To ensure this, the bank will know who is the owner of each coin.

Observation 3. We assume that the communication channels are private and authenticated, that is, when Bob receives a message on the channel c_{AB} he is sure that the message was sent by Alice.

Observation 4. We also have to assume some conditions about the bank:

(a) The Bank is always online — while the bank is off-line the service is unavailable;

(b) The Bank is honest and safe;

(c) The state of the Bank does not contain any repeated coins.

4.2 Withdrawal – A Simplification of the Problem

We present a simplification of the withdrawal protocol. We have a single principal that requests a single coin. We call this the *Simple Withdrawal Protocol (SWP)*. The protocol is illustrated in Figure 4.1.

We consider an auxiliary entity E for issuing processes. This way the Bank is only worried about payments, leaving the issuing process to E. Suppose that the wallet of A is empty, $\sigma_A = \emptyset$, and A wants to withdraw an *e*-coin with value v_1. We also suppose that the Bank Record and the Issuing Entity Record are empty, $\sigma_B = \emptyset$ and $\sigma_E = \emptyset$.

A-Wallet	**Issuing Entity**	**Bank Record**

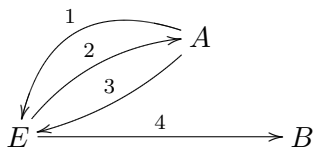


Figure 4.1: Withdrawal Protocol

To withdraw an e-coin with value v_1, all that A has to do is generate a time stamp ts_1 and send to E the information $(\boldsymbol{r}, id_A, ts_1, v_1)$, which means that he is *requesting* a coin, his identification is id_A, the time-stamp is ts_1 and the value he is asking for is v_1 (communication 1 of the Figure 4.1). The time stamp is only needed to distinguish different withdrawals that may be made at the same time. After that he adds to his wallet the information $(\boldsymbol{r}, ts_1, ?, v_1)$, that means he has *requested* a coin, with time-stamp ts_1 and value v_1.

$$A \xrightarrow{(\boldsymbol{r}, id_A, ts_1, v_1)} E \qquad B$$

A-Wallet	**Issuing Entity**	**Bank Record**
$(\boldsymbol{r}, ts_1, ?, v_1)$		

After receiving this information the Issuing Entity, E, generates an (unique) identification for the coin, id_{N_1}, sends to A the information $(\boldsymbol{i}, id_{N_1}, ts_1, v_1)$ (this means that the e-coin id_{N_1} is now *issued*, with the value v_1, in response to the communication with time-stamp ts_1) and adds $(\boldsymbol{r}, id_A, id_{N_1}, v_1)$ to the record — communication 2.

$$A \xleftarrow[(\boldsymbol{i}, id_{N_1}, ts_1, v_1)]{} E \qquad B$$

A-Wallet	**Issuing Entity**	**Bank Record**
$(\boldsymbol{r}, ts_1, ?, v_1)$	$(\boldsymbol{r}, id_A, id_{N_1}, v_1)$	

A replaces the temporary coin $(\boldsymbol{r}, ts_1, ?, v_1)$ by $(\boldsymbol{i}, ?, id_{N_1}, v_1)$ (the temporary coin is now *issued* with identification id_{N_1} and value v_1). Whenever A wishes to use this coin, the question mark will be substituted by the identification of the receiver. After changing his wallet, A communicates to E that he has already received the coin — sends the information $(\boldsymbol{i}, id_A, id_{N_1}, v_1)$, (communication 3).

A-Wallet	**Issuing Entity**	**Bank Record**
~~$(\boldsymbol{r}, ts_1, ?, v_1)$~~ $(\boldsymbol{i}, ?, id_{N_1}, v_1)$	$(\boldsymbol{r}, id_A, id_{N_1}, v_1)$	

$$A \xrightarrow{(\boldsymbol{i}, id_A, id_{N_1}, v_1)} E \qquad B$$

After receiving this information, E changes the record $(\boldsymbol{r}, id_A, id_{N_1}, v_1)$ to $(\boldsymbol{i}, id_A, id_{N_1}, v_1)$ which means that the coin is now really *issued*. Finally E communicates to the Bank that the coin is "good" for payment, sending the message $(\boldsymbol{i}, id_A, id_{N_1}, v_1)$ to the Bank (communication 4).

A-Wallet	**Issuing Entity**	**Bank Record**
$(\boldsymbol{i}, ?, id_{N_1}, v_1)$	~~$(\boldsymbol{r}, id_A, id_{N_1}, v_1)$~~ $(\boldsymbol{i}, id_A, id_{N_1}, v_1)$	

$$A \qquad\qquad E \xrightarrow{(\boldsymbol{i}, id_A, id_{N_1}, v_1)} B$$

Finally the Bank debits the account of A for v_1 and adds the e-coin to the list of coins that can be spent by A. The question mark will be replaced by the receiver when a transaction is made.

A-Wallet	**Issuing Entity**	**Bank Record**
$(\boldsymbol{i}, ?, id_{N_1}, v_1)$	$(\boldsymbol{i}, id_A, id_{N_1}, v_1)$	$(\boldsymbol{i}, id_A, id_{N_1}, v_1, ?)$

Formally the protocol is described as follows:

$$Request_{SWP}(v_1) \triangleq (\nu\, ts_1)\, \overline{c_{AE}}\langle(\boldsymbol{r}, id_A, ts_1, v_1)\rangle.add_A(\boldsymbol{r}, ts_1, ?, v_1);$$

$$\begin{aligned} E_{SWP} \triangleq\ & c_{AE}(x_1, x_2, x_3, x_4) \\ & [x_1\ is\ \boldsymbol{r}]\,.(\nu\, id_{N_1})\, \overline{c_{EA}}\langle(\boldsymbol{i}, id_{N_1}, x_3, x_4)\rangle.add_E(\boldsymbol{r}, x_2, id_{N_1}, x_4)\ \mid \\ & [x_1\ is\ \boldsymbol{i}]\,.change_E(\boldsymbol{r}, x_2, x_3, x_4, \boldsymbol{i}, x_2, x_3, x_4).\overline{c_{EB}}\langle(\boldsymbol{i}, x_2, x_3, x_4)\rangle; \end{aligned}$$

$$\begin{aligned} A_{SWP} \triangleq\ & c_{EA}(x_1, x_2, x_3, x_4). \\ & [x_1\ is\ \boldsymbol{i}]\,.change_A(\boldsymbol{r}, x_3, ?, x_4, \boldsymbol{i}, ?, x_2, x_4).\overline{c_{AE}}\langle(\boldsymbol{i}, id_A, x_2, x_4)\rangle; \end{aligned}$$

$$\begin{aligned} B_{SWP} \triangleq\ & c_{EB}(x_1, x_2, x_3, x_4). \\ & debt(x_2, x_4).add_B(x_1, x_2, x_3, x_4, ?); \end{aligned}$$

$$With_{SWP}(v_1) \triangleq (\nu\, c_{AE})\,(\nu\, c_{EA})\,(\nu\, c_{EB})\,(Request_{SWP}(v_1) \mid !A_{SWP} \mid !E_{SWP} \mid !B_{SWP}).$$

4.3 Payment – A Simplification of the Problem

As in the previous section we present a simplification of the protocol. Suppose that a principal A_{SPP}, wishes to purchase an item from V_{SPP} using a single coin, id_{N_1}, previously issued to him by the bank. We call this the *Simple Payment Protocol (SPP)*.

Suppose that the wallet of A has only the coin that he want to spend, $(\boldsymbol{i}, ?, id_{N_1}, v_1)$, $\sigma_A = \{(\boldsymbol{i}, ?, id_{N_1}, v_1)\}$, the record of B has only that same coin, $\sigma_B = \{(\boldsymbol{i}, id_A, id_{N_1}, v_1, ?)\}$, and the wallet of V is empty, $\sigma_V = \{\}$.

A-Wallet	**Bank Record**	**V-Wallet**
$(\boldsymbol{i}, ?, id_{N_1}, v_1)$	$(\boldsymbol{i}, id_A, id_{N_1}, v_1, ?)$	

The payment process is very simple. To start the payment, A changes the status of the coin, from *issued* to *transferred* and replaces the question mark for the identification of the seller. After that he communicates to V that he is *transferring* to him the coin id_{N_1}, with the value v_1, (communication 1).

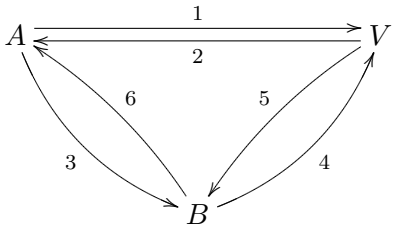
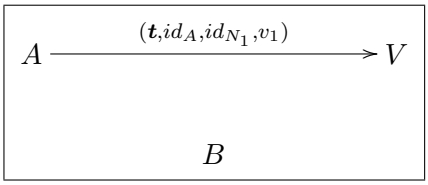


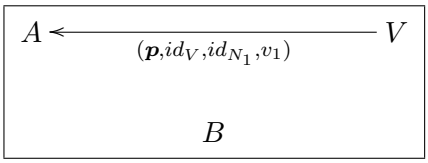
Figure 4.2: Payment Protocol

A-Wallet	**Bank Record**	**V-Wallet**
~~$(\boldsymbol{i}, ?, id_{N_1}, v_1)$~~ $(\boldsymbol{t}, id_V, id_{N_1}, v_1)$	$(\boldsymbol{i}, id_A, id_{N_1}, v_1, ?)$	



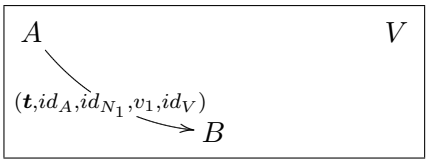
When V receives the message he had to his wallet the coin $(\boldsymbol{p}, id_A, id_{N_1}, v_1)$, which means that the state of the coin is *pending* and the actual owner is A. After that he acknowledges the reception of the message sending the message $(\boldsymbol{p}, id_V, id_{N_1}, v_1)$ to A, communication 2.

A-Wallet	**Bank Record**	**V-Wallet**
$(\boldsymbol{t}, id_V, id_{N_1}, v_1)$	$(\boldsymbol{i}, id_A, id_{N_1}, v_1, ?)$	$(\boldsymbol{p}, id_A, id_{N_1}, v_1)$



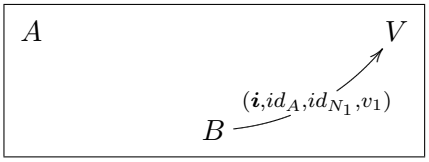
After the acknowledgment of V, A changes again the status of the coin from *transferred* to *spent* and communicates that fact to the Bank, ordering the transference of the coin to V, communication 3.

A-Wallet	**Bank Record**	**V-Wallet**
~~$(\boldsymbol{t}, id_V, id_{N_1}, v_1)$~~ $(\boldsymbol{s}, id_V, id_{N_1}, v_1)$	$(\boldsymbol{i}, id_A, id_{N_1}, v_1, ?)$	$(\boldsymbol{p}, id_A, id_{N_1}, v_1)$



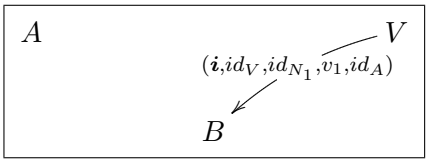
The Bank changes his record, changes the status of the coin from *issued* to *transferred* and asks V for the confirmation that he his selling something to A, communication 4.

A-Wallet	**Bank Record**	**V-Wallet**
$(\boldsymbol{s}, id_V, id_{N_1}, v_1)$	~~$(\boldsymbol{i}, id_A, id_{N_1}, v_1, ?)$~~ $(\boldsymbol{t}, id_A, id_{N_1}, v_1, id_V)$	$(\boldsymbol{p}, id_A, id_{N_1}, v_1)$



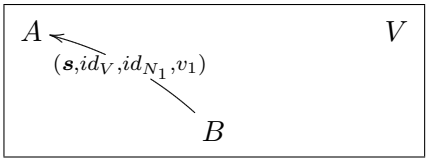
If V acknowledges the transaction, then exchanges in his wallet the status of the coin from *pending* to *issued*, and changes the "temporary" owner to ?, (field 2).

A-Wallet	**Bank Record**	**V-Wallet**
$(\boldsymbol{s}, id_V, id_{N_1}, v_1)$	$(\boldsymbol{t}, id_A, id_{N_1}, v_1, id_V)$	~~$(\boldsymbol{p}, id_A, id_{N_1}, v_1)$~~ $(\boldsymbol{i}, ?, id_{N_1}, v_1)$



After this communication, the Bank exchanges the owner of the coin and communicate to A that the transference is complete, communication 6.

A-Wallet	**Bank Record**	**V-Wallet**
$(\boldsymbol{s}, id_V, id_{N_1}, v_1)$	~~$(\boldsymbol{t}, id_A, id_{N_1}, v_1, id_V)$~~ $(\boldsymbol{i}, id_V, id_{N_1}, v_1, ?)$	$(\boldsymbol{i}, ?, id_{N_1}, v_1)$



So, all that A needs to do is to remove the coin from his wallet

A-Wallet	**Bank Record**	**V-Wallet**
~~$(\boldsymbol{s}, id_V, id_{N_1}, v_1)$~~	$(\boldsymbol{i}, id_V, id_{N_1}, v_1, ?)$	$(\boldsymbol{i}, ?, id_{N_1}, v_1)$

Formally, we describe the protocol as follows:

$$Buyer_{SPP}(id_{N_1}) \triangleq change_A(\boldsymbol{i}, ?, id_{N_1}, v_1, \boldsymbol{t}, id_V, id_{N_1}, v_1).\overline{c_{AV}}\langle(\boldsymbol{t}, id_A, id_{N_1}, v_1)\rangle;$$

$$A_{SPP} \triangleq c_{VA}(x_1, x_2, x_3, x_4).$$

$$[x_1 \ is \ \boldsymbol{p}] \, .change_A(\boldsymbol{t}, x_2, x_3, x_4, \boldsymbol{s}, x_2, x_3, x_4).\overline{c_{AB}}\langle(\boldsymbol{t}, id_A, x_3, x_4, x_2)\rangle \Big|$$

$$c_{BA}(x_1, x_2, x_3, x_4)$$

$$[x_1 \ is \ \boldsymbol{s}] \, .remove_A(\boldsymbol{s}, x_2, x_3, x_4);$$

$$V_{SPP} \triangleq c_{AV}(x_1, x_2, x_3, x_4)$$
$$[x_1 \ is \ \boldsymbol{t}] \, .add_V(\boldsymbol{p}, x_2, x_3, x_4).\overline{c_{VA}}\langle(\boldsymbol{p}, id_V, x_3, x_4)\rangle \Big|$$
$$c_{BV}(x_1, x_2, x_3, x_4)$$
$$[x_1 \ is \ \boldsymbol{i}] \, .change_V(\boldsymbol{p}, x_2, x_3, x_4, \boldsymbol{i}, ?, x_3, x_4).\overline{c_{VB}}\langle(\boldsymbol{i}, id_V, x_3, x_4, x_2)\rangle;$$

$$B_{SPP} \triangleq c_{AB}(x_1, x_2, x_3, x_4, x_5).$$
$$[x_1 \ is \ \boldsymbol{t}] \, .change_B(\boldsymbol{i}, x_2, x_3, x_4, ?, \boldsymbol{t}, x_2, x_3, x_4, x_5).\overline{c_{BV}}\langle(\boldsymbol{i}, x_2, x_3, x_4)\rangle \Big|$$
$$c_{VB}(x_1, x_2, x_3, x_4, x_5)$$
$$[x_1 \ is \ \boldsymbol{i}] \, .change_B(\boldsymbol{t}, x_5, x_3, x_4, x_2, \boldsymbol{i}, x_2, x_3, x_4, ?).\overline{c_{BA}}\langle(\boldsymbol{s}, x_2, x_3, x_4)\rangle;$$

$$Payment_{SPP}(id_{N_1}) \triangleq (\nu \, c_{AV}) \, (\nu \, c_{VA}) \, (\nu \, c_{AB}) \, (\nu \, c_{BA}) \, (\nu \, c_{BV}) \, (\nu \, c_{VB})$$
$$(Buyer_{SPP}(id_{N_1}) \mid !A_{SPP} \mid !V_{SPP} \mid !B_{SPP}).$$

This protocol has a small problem that cannot be specified in our calculus. For instance suppose that A wants to buy something from V, but he does not send the information to the bank. The transference of the coin will not terminate, so V will keep in his wallet a reference to the coin that A is trying to spend, but that coin will be forever in *pending* state. For this, it is needed a *garbage collector* system, in order to remove "lost" coins. This mechanism can be implemented locally, so this does not interfere with the communication process. Moreover, the implementation is very simple, just consider a timeout for each transaction.

Chapter 5

Properties of the Protocol

In Chapter 4, we presented a protocol for electronic payment. This protocol is only interesting if we prove some desired properties. One of the wished properties is that a principal cannot spend the same coin more than once. This property is not assured by most of the protocols in the literature, since most of them are off-line protocols, but our protocol satisfy this property, as stated in Theorem 5.2. We prove also that it is impossible to create new coins and spend them, Theorem 5.4, and that it is impossible to spend a coin that is issued to another principal, Theorem 5.6. Let us start with the multi-spending problem.

Lemma 5.1. *Suppose that the e-coin* $(\boldsymbol{i}, ?, id_{N_1}, v_1)$ *is issued to A, i.e.,* $(\boldsymbol{i}, id_A, id_{N_1}, v_1, ?) \in \sigma_B$ *and* $(\boldsymbol{i}, ?, id_{N_1}, v_1) \in \sigma_A$*. Then,*

$$\begin{aligned}&\big(\{\sigma_A, \sigma_B, \sigma_V\}, (\nu\, c_{AV})\,(\nu\, c_{VA})\,(\nu\, c_{AB})\,(\nu\, c_{BA})\,(\nu\, c_{BV})\,(\nu\, c_{VB})\\ &\qquad (Buyer_{SPP}(id_{N_1}) \mid !A_{SPP} \mid !V_{SPP} \mid !B_{SPP})\big) \cong_B\\ \big(\{\sigma_A \uplus \{(\boldsymbol{i}, ?, id_{N_1}, v_1)\}, \sigma_B, \sigma_V\}, &(\nu\, c_{AV})\,(\nu\, c_{VA})\,(\nu\, c_{AB})\,(\nu\, c_{BA})\,(\nu\, c_{BV})\,(\nu\, c_{VB})\\ &\qquad (Buyer_{SPP}(id_{N_1}) \mid\ Buyer_{SPP}(id_{N_1}) \mid !A_{SPP} \mid !V_{SPP} \mid !B_{SPP})\big).\end{aligned}$$

Proof. The proof is presented in the Appendix A. □

Theorem 5.2. *Our protocol is safe against (valid) e-coins multi-spending.*

Proof. The proof follows directly from Lemma 5.1. The Lemma states that a coin that is legally issued to A, $(\boldsymbol{i}, id_A, id_{N_1}, v_1, ?) \in \sigma_B$, can only be spent once by A , since the behaviour of the protocol that spends the coin twice is equal, for the bank, to the behaviour of the protocol that spends only once. □

Another important property to be satisfied is the *anti-forgery* property. We prove this in the next lemma.

Lemma 5.3. *Suppose that the e-coin* $(\boldsymbol{i}, ?, id_{N_1}, v_1)$ *is not issued to A, i.e.,* $(\boldsymbol{i}, id_A, id_{N_1}, v_1, ?) \notin \sigma_B$*. Then,*

$$\begin{aligned}&\big(\{\sigma_A, \sigma_B, \sigma_V\}, (\nu\, c_{AV})\,(\nu\, c_{VA})\,(\nu\, c_{AB})\,(\nu\, c_{BA})\,(\nu\, c_{BV})\,(\nu\, c_{VB})\\ &\qquad (!A_{SPP} \mid !V_{SPP} \mid !B_{SPP})\big) \cong_B\\ \big(\{\sigma_A \uplus \{(\boldsymbol{i}, ?, id_{N_1}, v_1)\}, \sigma_B, \sigma_V\}, &(\nu\, c_{AV})\,(\nu\, c_{VA})\,(\nu\, c_{AB})\,(\nu\, c_{BA})\,(\nu\, c_{BV})\,(\nu\, c_{VB})\\ &\qquad (Buyer_{SPP}(id_{N_1}) \mid !A_{SPP} \mid !V_{SPP} \mid !B_{SPP})\big).\end{aligned}$$

Proof. The proof is presented in the Appendix A. □

Theorem 5.4. *Our protocol is safe against e-coin forgery.*

Proof. Follows directly from the previous lemma. Notice that A is trying to spend a coin that is not issued to him, $(\boldsymbol{i}, id_A, id_{N_1}, v_1, ?) \notin \sigma_B$, and that the protocol where he spends it is A-weak equivalent to the protocol that does nothing. □

Finally we show that it is impossible to use a coin that is issued to other principal. This is similar to Lemma 5.3, but since we stated this property as a desired one, we prove this special case of the lemma above.

Lemma 5.5. *Suppose that the e-coin* $(\boldsymbol{i}, ?, id_{N_2}, v_2)$ *is issued to* V*, i.e.,* $(\boldsymbol{i}, id_V, id_{N_2}, v_2, ?) \in \sigma_B$ *and* $(\boldsymbol{i}, ?, id_{N_2}, v_2) \in \sigma_V$*. Then,*

$$\begin{aligned}&\big(\{\sigma_A, \sigma_B, \sigma_V\}, (\nu\, c_{AV})\,(\nu\, c_{VA})\,(\nu\, c_{AB})\,(\nu\, c_{BA})\,(\nu\, c_{BV})\,(\nu\, c_{VB})\\ &\qquad (!A_{SPP} \mid !V_{SPP} \mid !B_{SPP})\big) \cong_B\\ &\big(\{\sigma_A \uplus \{(\boldsymbol{i}, ?, id_{N_2}, v_2)\}, \sigma_B, \sigma_V\}, (\nu\, c_{AV})\,(\nu\, c_{VA})\,(\nu\, c_{AB})\,(\nu\, c_{BA})\,(\nu\, c_{BV})\,(\nu\, c_{VB})\\ &\qquad (Buyer_{SPP}(id_{N_2}) \mid !A_{SPP} \mid !V_{SPP} \mid !B_{SPP})\big).\end{aligned}$$

Proof. The proof is presented in the Appendix A. □

Theorem 5.6. *Our protocol is safe against e-coin stealing.*

Proof. Straightforward from the Lemma 5.5. □

Definition 5.1. We define the *union of states*, $\sigma' = \{\sigma'_A\}_{A \in \mathcal{A}}$ and $\sigma'' = \{\sigma''_A\}_{A \in \mathcal{A}}$ as the family $\sigma = \{\sigma'_A \uplus \sigma''_A\}_{A \in \mathcal{A}}$. We denote the union of states σ' and σ'' by $\sigma' \oplus \sigma''$.

Chapter 6

Final Remarks and Future Work

In Chapter 2, we presented the Spi-Calculus and sketched some examples, the proofs of the examples can be found in [AG98b].

Since the notion of equivalence of the Spi-Calculus was not appropriated we introduced a new calculus in Chapter 3 endowed with the desired equivalence notion. Our notion is a simple bisimulation that also looks at the state of the bank. Our notion of equivalence is not a congruence but this was not a problem for the proof of the properties of Chapter 5.

Our main contribution is the design of the protocols of Chapter 4. These protocols are simple protocols with only one buyer, one bank and one vendor, but we hope to get some compositionality results to extend it to a larger network. Mainly we want to discover what conditions the configuration (σ_2, O) has to satisfy so that if $(\sigma_1, P_1) \cong_B (\sigma_1', P_1')$ then $(\sigma_1 \oplus \sigma_2, P_1 \mid O) \cong_B (\sigma_1' \oplus \sigma_2, P_1' \mid O)$, i.e., we want to discover in which states and processes, the configurations (σ_1, P_1) and (σ_1', P_1') can be "immersed" in order to keep the same behaviour. For now, we conjecture that if the e-coins used by P_1 and P_1' are not issued in σ_2 then $(\sigma_1 \oplus \sigma_2, P_1 \mid O) \cong_B (\sigma_1' \oplus \sigma_2, P_1' \mid O)$.

In the future we intend to extend our protocol to any valuable such as shares, registered bonds, etc. We also like to incorporate the notion of privacy. We hope to achieve this using secure computation methods, [Can00, Can01], for instance the information of the coins is spread over a network of banks and each of them only has a small piece of information. It will be also interesting to incorporate the notion of *divisibility*, see Okamoto [OO92, Oka95]. With this notion we could in fact simulate transactions without requiring the exact amount of money, as we are able to give change.

We could also try to represent these protocols in an extended form of CCS, without the configurations. If we achieve this, some theoretical results from the CCS could be applied to the resolution of our problem.

We hope also to design a calculus where the theory for local operations is generic, and not considering only these three operations. It might be interesting to have the local operations also as observable actions. With this, our notion of equivalence will probably became a congruence.

Appendix A

Proofs of the Lemmas of Chapter 5

A.1 Proof of the Lemma 5.1

We want to prove that, if a principal tries to spend the same coin twice then the result, for the bank, is the same as just spending once. We prove this informally since the bisimulation that proves the result has 155 pairs. We want to prove that

$$\begin{aligned}&\big(\{\sigma_A,\sigma_B,\sigma_V\},(\nu\, c_{AV})\,(\nu\, c_{VA})\,(\nu\, c_{AB})\,(\nu\, c_{BA})\,(\nu\, c_{BV})\,(\nu\, c_{VB})\\&\qquad(Buyer_{SPP}(id_{N_1})\mid !A_{SPP}\mid !V_{SPP}\mid !B_{SPP})\big)\cong_B\\&\big(\{\sigma_A\uplus\{(\boldsymbol{i},?,id_{N_1},v_1)\},\sigma_B,\sigma_V\},(\nu\, c_{AV})\,(\nu\, c_{VA})\,(\nu\, c_{AB})\,(\nu\, c_{BA})\,(\nu\, c_{BV})\,(\nu\, c_{VB})\\&\qquad(Buyer_{SPP}(id_{N_1})\mid\ Buyer_{SPP}(id_{N_1})\mid !A_{SPP}\mid !V_{SPP}\mid !B_{SPP})\big).\end{aligned}$$

whenever $(\boldsymbol{i},?,id_{N_1},v_1)\in\sigma_A$ and $(\boldsymbol{i},id_A,id_{N_1},v_1,?)\in\sigma_B$.

We want to prove that if two payment protocols run in parallel using the same coin, id_{N_1}, only one will evolve into the final state; the other will get stuck when A communicates to the bank requesting the transference of the coin to V. If we look at a normal evolution of one protocol, we may consider the following configurations of the evolution:

$$\begin{aligned}E_1=\big(&\{\{(\boldsymbol{i},?,id_{N_1},v_1)\},\{(\boldsymbol{i},id_A,id_{N_1},v_1,?)\},\{\}\},(\nu\, c_{AV})\,(\nu\, c_{VA})\,(\nu\, c_{AB})\,(\nu\, c_{BA})\,(\nu\, c_{BV})\,(\nu\, c_{VB})\\&(Buyer_{SPP}(id_{N_1})\mid !A_{SPP}\mid !V_{SPP}\mid !B_{SPP})\big);\\E_2=\big(&\{\{(\boldsymbol{t},id_V,id_{N_1},v_1)\},\{(\boldsymbol{i},id_A,id_{N_1},v_1,?)\},\{\}\},(\nu\, c_{AV})\,(\nu\, c_{VA})\,(\nu\, c_{AB})\,(\nu\, c_{BA})\,(\nu\, c_{BV})\,(\nu\, c_{VB})\\&(\overline{c_{AV}}\langle(\boldsymbol{t},id_A,id_{N_1},v_1)\rangle\mid !A_{SPP}\mid !V_{SPP}\mid !B_{SPP})\big);\\E_3=\big(&\{\{(\boldsymbol{t},id_V,id_{N_1},v_1)\},\{(\boldsymbol{i},id_A,id_{N_1},v_1,?)\},\{\}\},(\nu\, c_{AV})\,(\nu\, c_{VA})\,(\nu\, c_{AB})\,(\nu\, c_{BA})\,(\nu\, c_{BV})\,(\nu\, c_{VB})\\&(!A_{SPP}\mid[\boldsymbol{t}\ is\ \boldsymbol{t}]\,.add_V(\boldsymbol{p},id_A,id_{N_1},v_1).\overline{c_{VA}}\langle(\boldsymbol{p},id_V,id_{N_1},v_1)\rangle\mid\\&c_{BV}(x_1,x_2,x_3,x_4).[x_1\ is\ \boldsymbol{i}]\,.change_V(\boldsymbol{p},x_2,x_3,x_4,\boldsymbol{i},?,x_3,x_4).\overline{c_{VB}}\langle(\boldsymbol{i},id_V,x_3,x_4,x_2)\rangle\mid\\&!V_{SPP}\mid!B_{SPP})\big);\\E_4=\big(&\{\{(\boldsymbol{t},id_V,id_{N_1},v_1)\},\{(\boldsymbol{i},id_A,id_{N_1},v_1,?)\},\{\}\},(\nu\, c_{AV})\,(\nu\, c_{VA})\,(\nu\, c_{AB})\,(\nu\, c_{BA})\,(\nu\, c_{BV})\,(\nu\, c_{VB})\\&(!A_{SPP}\mid add_V(\boldsymbol{p},id_A,id_{N_1},v_1).\overline{c_{VA}}\langle(\boldsymbol{p},id_V,id_{N_1},v_1)\rangle\mid\\&c_{BV}(x_1,x_2,x_3,x_4).[x_1\ is\ \boldsymbol{i}]\,.change_V(\boldsymbol{p},x_2,x_3,x_4,\boldsymbol{i},?,x_3,x_4).\overline{c_{VB}}\langle(\boldsymbol{i},id_V,x_3,x_4,x_2)\rangle\mid\\&!V_{SPP}\mid!B_{SPP})\big);\end{aligned}$$

$$
\begin{aligned}
E_5 = \big(&\{\{(\boldsymbol{t}, id_V, id_{N_1}, v_1)\}, \{(\boldsymbol{i}, id_A, id_{N_1}, v_1, ?)\}, \{(\boldsymbol{p}, id_A, id_{N_1}, v_1)\}\}, (\nu\, c_{AV})\,(\nu\, c_{VA})\,(\nu\, c_{AB})\,(\nu\, c_{BA})\\
&(\nu\, c_{BV})\,(\nu\, c_{VB})\,(!A_{SPP} \mid \overline{c_{VA}}\langle(\boldsymbol{p}, id_V, id_{N_1}, v_1)\rangle \mid\\
&c_{BV}(x_1, x_2, x_3, x_4).[x_1 \ is \ \boldsymbol{i}]\,.change_V(\boldsymbol{p}, x_2, x_3, x_4, \boldsymbol{i}, ?, x_3, x_4).\overline{c_{VB}}\langle(\boldsymbol{i}, id_V, x_3, x_4, x_2)\rangle \mid\\
&!V_{SPP} \mid !B_{SPP})\big);\\
E_6 = \big(&\{\{(\boldsymbol{t}, id_V, id_{N_1}, v_1)\}, \{(\boldsymbol{i}, id_A, id_{N_1}, v_1, ?)\}, \{(\boldsymbol{p}, id_A, id_{N_1}, v_1)\}\}, (\nu\, c_{AV})\,(\nu\, c_{VA})\,(\nu\, c_{AB})\,(\nu\, c_{BA})\\
&(\nu\, c_{BV})\,(\nu\, c_{VB})\\
&([\boldsymbol{p} \ is \ \boldsymbol{p}]\,.change_A(\boldsymbol{t}, id_V, id_{N_1}, v_1, \boldsymbol{s}, id_V, id_{N_1}, v_1).\overline{c_{AB}}\langle(\boldsymbol{t}, id_A, id_{N_1}, v_1, id_V)\rangle \mid\\
&c_{BA}(x_1, x_2, x_3, x_4).[x_1 \ is \ \boldsymbol{s}]\,.remove_A(\boldsymbol{s}, x_2, x_3, x_4) \mid !A_{SPP} \mid\\
&c_{BV}(x_1, x_2, x_3, x_4).[x_1 \ is \ \boldsymbol{i}]\,.change_V(\boldsymbol{p}, x_2, x_3, x_4, \boldsymbol{i}, ?, x_3, x_4).\overline{c_{VB}}\langle(\boldsymbol{i}, id_V, x_3, x_4, x_2)\rangle \mid\\
&!V_{SPP} \mid !B_{SPP})\big);\\
E_7 = \big(&\{\{(\boldsymbol{t}, id_V, id_{N_1}, v_1)\}, \{(\boldsymbol{i}, id_A, id_{N_1}, v_1, ?)\}, \{(\boldsymbol{p}, id_A, id_{N_1}, v_1)\}\}, (\nu\, c_{AV})\,(\nu\, c_{VA})\,(\nu\, c_{AB})\,(\nu\, c_{BA})\\
&(\nu\, c_{BV})\,(\nu\, c_{VB})\\
&(change_A(\boldsymbol{t}, id_V, id_{N_1}, v_1, \boldsymbol{s}, id_V, id_{N_1}, v_1).\overline{c_{AB}}\langle(\boldsymbol{t}, id_A, id_{N_1}, v_1, id_V)\rangle \mid\\
&c_{BA}(x_1, x_2, x_3, x_4).[x_1 \ is \ \boldsymbol{s}]\,.remove_A(\boldsymbol{s}, x_2, x_3, x_4) \mid !A_{SPP} \mid\\
&c_{BV}(x_1, x_2, x_3, x_4).[x_1 \ is \ \boldsymbol{i}]\,.change_V(\boldsymbol{p}, x_2, x_3, x_4, \boldsymbol{i}, ?, x_3, x_4).\overline{c_{VB}}\langle(\boldsymbol{i}, id_V, x_3, x_4, x_2)\rangle \mid\\
&!V_{SPP} \mid !B_{SPP})\big);\\
E_8 = \big(&\{\{(\boldsymbol{s}, id_V, id_{N_1}, v_1)\}, \{(\boldsymbol{i}, id_A, id_{N_1}, v_1, ?)\}, \{(\boldsymbol{p}, id_A, id_{N_1}, v_1)\}\}, (\nu\, c_{AV})\,(\nu\, c_{VA})\,(\nu\, c_{AB})\,(\nu\, c_{BA})\\
&(\nu\, c_{BV})\,(\nu\, c_{VB})\\
&(\overline{c_{AB}}\langle(\boldsymbol{t}, id_A, id_{N_1}, v_1, id_V)\rangle \mid\\
&c_{BA}(x_1, x_2, x_3, x_4).[x_1 \ is \ \boldsymbol{s}]\,.remove_A(\boldsymbol{s}, x_2, x_3, x_4) \mid !A_{SPP} \mid\\
&c_{BV}(x_1, x_2, x_3, x_4).[x_1 \ is \ \boldsymbol{i}]\,.change_V(\boldsymbol{p}, x_2, x_3, x_4, \boldsymbol{i}, ?, x_3, x_4).\overline{c_{VB}}\langle(\boldsymbol{i}, id_V, x_3, x_4, x_2)\rangle \mid\\
&!V_{SPP} \mid !B_{SPP})\big);\\
E_9 = \big(&\{\{(\boldsymbol{s}, id_V, id_{N_1}, v_1)\}, \{(\boldsymbol{i}, id_A, id_{N_1}, v_1, ?)\}, \{(\boldsymbol{p}, id_A, id_{N_1}, v_1)\}\}, (\nu\, c_{AV})\,(\nu\, c_{VA})\,(\nu\, c_{AB})\,(\nu\, c_{BA})\\
&(\nu\, c_{BV})\,(\nu\, c_{VB})\\
&(c_{BA}(x_1, x_2, x_3, x_4).[x_1 \ is \ \boldsymbol{s}]\,.remove_A(\boldsymbol{s}, x_2, x_3, x_4) \mid !A_{SPP} \mid\\
&c_{BV}(x_1, x_2, x_3, x_4).[x_1 \ is \ \boldsymbol{i}]\,.change_V(\boldsymbol{p}, x_2, x_3, x_4, \boldsymbol{i}, ?, x_3, x_4).\overline{c_{VB}}\langle(\boldsymbol{i}, id_V, x_3, x_4, x_2)\rangle \mid\\
&!V_{SPP} \mid [\boldsymbol{t} \ is \ \boldsymbol{t}]\,.change_B(\boldsymbol{i}, id_A, id_{N_1}, v_1, ?, \boldsymbol{t}, id_A, id_{N_1}, v_1, id_V).\overline{c_{BV}}\langle(\boldsymbol{i}, id_A, id_{N_1}, v_1)\rangle \mid\\
&c_{VB}(x_1, x_2, x_3, x_4, x_5).[x_1 \ is \ \boldsymbol{i}]\,.change_B(\boldsymbol{t}, x_5, x_3, x_4, x_2, \boldsymbol{i}, x_2, x_3, x_4, ?).\overline{c_{BA}}\langle(\boldsymbol{s}, x_2, x_3, x_4)\rangle \mid\\
&!B_{SPP})\big);\\
E_{10} = \big(&\{\{(\boldsymbol{s}, id_V, id_{N_1}, v_1)\}, \{(\boldsymbol{i}, id_A, id_{N_1}, v_1, ?)\}, \{(\boldsymbol{p}, id_A, id_{N_1}, v_1)\}\}, (\nu\, c_{AV})\,(\nu\, c_{VA})\,(\nu\, c_{AB})\,(\nu\, c_{BA})\\
&(\nu\, c_{BV})\,(\nu\, c_{VB})\\
&(c_{BA}(x_1, x_2, x_3, x_4).[x_1 \ is \ \boldsymbol{s}]\,.remove_A(\boldsymbol{s}, x_2, x_3, x_4) \mid !A_{SPP} \mid\\
&c_{BV}(x_1, x_2, x_3, x_4).[x_1 \ is \ \boldsymbol{i}]\,.change_V(\boldsymbol{p}, x_2, x_3, x_4, \boldsymbol{i}, ?, x_3, x_4).\overline{c_{VB}}\langle(\boldsymbol{i}, id_V, x_3, x_4, x_2)\rangle \mid\\
&!V_{SPP} \mid change_B(\boldsymbol{i}, id_A, id_{N_1}, v_1, ?, \boldsymbol{t}, id_A, id_{N_1}, v_1, id_V).\overline{c_{BV}}\langle(\boldsymbol{i}, id_A, id_{N_1}, v_1)\rangle \mid\\
&c_{VB}(x_1, x_2, x_3, x_4, x_5).[x_1 \ is \ \boldsymbol{i}]\,.change_B(\boldsymbol{t}, x_5, x_3, x_4, x_2, \boldsymbol{i}, x_2, x_3, x_4, ?).\overline{c_{BA}}\langle(\boldsymbol{s}, x_2, x_3, x_4)\rangle \mid\\
&!B_{SPP})\big);
\end{aligned}
$$

$E_{11} = \big(\{\{(\boldsymbol{s}, id_V, id_{N_1}, v_1)\}, \{(\boldsymbol{t}, id_A, id_{N_1}, v_1, id_V)\}, \{(\boldsymbol{p}, id_A, id_{N_1}, v_1)\}\}, (\nu\, c_{AV})\,(\nu\, c_{VA})\,(\nu\, c_{AB})\,(\nu\, c_{BA})$
$(\nu\, c_{BV})\,(\nu\, c_{VB})$
$(c_{BA}(x_1, x_2, x_3, x_4).[x_1\ is\ \boldsymbol{s}]\,.remove_A(\boldsymbol{s}, x_2, x_3, x_4) \mid !A_{SPP} \mid$
$c_{BV}(x_1, x_2, x_3, x_4).[x_1\ is\ \boldsymbol{i}]\,.change_V(\boldsymbol{p}, x_2, x_3, x_4, \boldsymbol{i}, ?, x_3, x_4).\overline{c_{VB}}\langle(\boldsymbol{i}, id_V, x_3, x_4, x_2)\rangle \mid$
$!V_{SPP} \mid \overline{c_{BV}}\langle(\boldsymbol{i}, id_A, id_{N_1}, v_1)\rangle \mid$
$c_{VB}(x_1, x_2, x_3, x_4, x_5).[x_1\ is\ \boldsymbol{i}]\,.change_B(\boldsymbol{t}, x_5, x_3, x_4, x_2, \boldsymbol{i}, x_2, x_3, x_4, ?).\overline{c_{BA}}\langle(\boldsymbol{s}, x_2, x_3, x_4)\rangle \mid$
$!B_{SPP}\big)$;

$E_{12} = \big(\{\{(\boldsymbol{s}, id_V, id_{N_1}, v_1)\}, \{(\boldsymbol{t}, id_A, id_{N_1}, v_1, id_V)\}, \{(\boldsymbol{p}, id_A, id_{N_1}, v_1)\}\}, (\nu\, c_{AV})\,(\nu\, c_{VA})\,(\nu\, c_{AB})\,(\nu\, c_{BA})$
$(\nu\, c_{BV})\,(\nu\, c_{VB})$
$(c_{BA}(x_1, x_2, x_3, x_4).[x_1\ is\ \boldsymbol{s}]\,.remove_A(\boldsymbol{s}, x_2, x_3, x_4) \mid !A_{SPP} \mid$
$[\boldsymbol{i}\ is\ \boldsymbol{i}]\,.change_V(\boldsymbol{p}, id_A, id_{N_1}, v_1, \boldsymbol{i}, ?, id_{N_1}, v_1).\overline{c_{VB}}\langle(\boldsymbol{i}, id_V, id_{N_1}, v_1, id_A)\rangle \mid !V_{SPP} \mid$
$c_{VB}(x_1, x_2, x_3, x_4, x_5).[x_1\ is\ \boldsymbol{i}]\,.change_B(\boldsymbol{t}, x_5, x_3, x_4, x_2, \boldsymbol{i}, x_2, x_3, x_4, ?).\overline{c_{BA}}\langle(\boldsymbol{s}, x_2, x_3, x_4)\rangle \mid$
$!B_{SPP}\big)$;

$E_{13} = \big(\{\{(\boldsymbol{s}, id_V, id_{N_1}, v_1)\}, \{(\boldsymbol{t}, id_A, id_{N_1}, v_1, id_V)\}, \{(\boldsymbol{p}, id_A, id_{N_1}, v_1)\}\}, (\nu\, c_{AV})\,(\nu\, c_{VA})\,(\nu\, c_{AB})\,(\nu\, c_{BA})$
$(\nu\, c_{BV})\,(\nu\, c_{VB})$
$(c_{BA}(x_1, x_2, x_3, x_4).[x_1\ is\ \boldsymbol{s}]\,.remove_A(\boldsymbol{s}, x_2, x_3, x_4) \mid !A_{SPP} \mid$
$change_V(\boldsymbol{p}, id_A, id_{N_1}, v_1, \boldsymbol{i}, ?, id_{N_1}, v_1).\overline{c_{VB}}\langle(\boldsymbol{i}, id_V, id_{N_1}, v_1, id_A)\rangle \mid !V_{SPP} \mid$
$c_{VB}(x_1, x_2, x_3, x_4, x_5).[x_1\ is\ \boldsymbol{i}]\,.change_B(\boldsymbol{t}, x_5, x_3, x_4, x_2, \boldsymbol{i}, x_2, x_3, x_4, ?).\overline{c_{BA}}\langle(\boldsymbol{s}, x_2, x_3, x_4)\rangle \mid$
$!B_{SPP}\big)$;

$E_{14} = \big(\{\{(\boldsymbol{s}, id_V, id_{N_1}, v_1)\}, \{(\boldsymbol{t}, id_A, id_{N_1}, v_1, id_V)\}, \{(\boldsymbol{i}, ?, id_{N_1}, v_1)\}\}, (\nu\, c_{AV})\,(\nu\, c_{VA})\,(\nu\, c_{AB})\,(\nu\, c_{BA})$
$(\nu\, c_{BV})\,(\nu\, c_{VB})$
$(c_{BA}(x_1, x_2, x_3, x_4).[x_1\ is\ \boldsymbol{s}]\,.remove_A(\boldsymbol{s}, x_2, x_3, x_4) \mid !A_{SPP} \mid$
$\overline{c_{VB}}\langle(\boldsymbol{i}, id_V, id_{N_1}, v_1, id_A)\rangle \mid !V_{SPP} \mid$
$c_{VB}(x_1, x_2, x_3, x_4, x_5).[x_1\ is\ \boldsymbol{i}]\,.change_B(\boldsymbol{t}, x_5, x_3, x_4, x_2, \boldsymbol{i}, x_2, x_3, x_4, ?).\overline{c_{BA}}\langle(\boldsymbol{s}, x_2, x_3, x_4)\rangle \mid$
$!B_{SPP}\big)$;

$E_{15} = \big(\{\{(\boldsymbol{s}, id_V, id_{N_1}, v_1)\}, \{(\boldsymbol{t}, id_A, id_{N_1}, v_1, id_V)\}, \{(\boldsymbol{i}, ?, id_{N_1}, v_1)\}\}, (\nu\, c_{AV})\,(\nu\, c_{VA})\,(\nu\, c_{AB})\,(\nu\, c_{BA})$
$(\nu\, c_{BV})\,(\nu\, c_{VB})$
$(c_{BA}(x_1, x_2, x_3, x_4).[x_1\ is\ \boldsymbol{s}]\,.remove_A(\boldsymbol{s}, x_2, x_3, x_4) \mid !A_{SPP} \mid !V_{SPP} \mid$
$[\boldsymbol{i}\ is\ \boldsymbol{i}]\,.change_B(\boldsymbol{t}, id_A, id_{N_1}, v_1, id_V, \boldsymbol{i}, id_V, id_{N_1}, v_1, ?).\overline{c_{BA}}\langle(\boldsymbol{s}, id_V, id_{N_1}, v_1)\rangle \mid !B_{SPP}\big)$;

$E_{16} = \big(\{\{(\boldsymbol{s}, id_V, id_{N_1}, v_1)\}, \{(\boldsymbol{t}, id_A, id_{N_1}, v_1, id_V)\}, \{(\boldsymbol{i}, ?, id_{N_1}, v_1)\}\}, (\nu\, c_{AV})\,(\nu\, c_{VA})\,(\nu\, c_{AB})\,(\nu\, c_{BA})$
$(\nu\, c_{BV})\,(\nu\, c_{VB})$
$(c_{BA}(x_1, x_2, x_3, x_4).[x_1\ is\ \boldsymbol{s}]\,.remove_A(\boldsymbol{s}, x_2, x_3, x_4) \mid !A_{SPP} \mid !V_{SPP} \mid$
$change_B(\boldsymbol{t}, id_A, id_{N_1}, v_1, id_V, \boldsymbol{i}, id_V, id_{N_1}, v_1, ?).\overline{c_{BA}}\langle(\boldsymbol{s}, id_V, id_{N_1}, v_1)\rangle \mid !B_{SPP}\big)$;

$E_{17} = \big(\{\{(\boldsymbol{s}, id_V, id_{N_1}, v_1)\}, \{(\boldsymbol{i}, id_V, id_{N_1}, v_1, ?)\}, \{(\boldsymbol{i}, ?, id_{N_1}, v_1)\}\}, (\nu\, c_{AV})\,(\nu\, c_{VA})\,(\nu\, c_{AB})\,(\nu\, c_{BA})$
$(\nu\, c_{BV})\,(\nu\, c_{VB})$
$(c_{BA}(x_1, x_2, x_3, x_4).[x_1\ is\ \boldsymbol{s}]\,.remove_A(\boldsymbol{s}, x_2, x_3, x_4) \mid !A_{SPP} \mid !V_{SPP} \mid$
$\overline{c_{BA}}\langle(\boldsymbol{s}, id_V, id_{N_1}, v_1)\rangle \mid !B_{SPP}\big)$;

$E_{18} = \big(\{\{(\boldsymbol{s}, id_V, id_{N_1}, v_1)\}, \{(\boldsymbol{i}, id_V, id_{N_1}, v_1, ?)\}, \{(\boldsymbol{i}, ?, id_{N_1}, v_1)\}\}, (\nu\, c_{AV})\,(\nu\, c_{VA})\,(\nu\, c_{AB})\,(\nu\, c_{BA})$
$(\nu\, c_{BV})\,(\nu\, c_{VB})$
$([\boldsymbol{s}\ is\ \boldsymbol{s}]\,.remove_A(\boldsymbol{s}, id_V, id_{N_1}, v_1) \mid !A_{SPP} \mid !V_{SPP} \mid !B_{SPP})\big)$;

$$E_{19} = \big(\{\{(\boldsymbol{s}, id_V, id_{N_1}, v_1)\}, \{(\boldsymbol{i}, id_V, id_{N_1}, v_1, ?)\}, \{(\boldsymbol{i}, ?, id_{N_1}, v_1)\}\}, (\nu\, c_{AV})\,(\nu\, c_{VA})\,(\nu\, c_{AB})\,(\nu\, c_{BA})\,(\nu\, c_{BV})\,(\nu\, c_{VB})\ (remove_A(\boldsymbol{s}, id_V, id_{N_1}, v_1) \mid !A_{SPP} \mid !V_{SPP} \mid !B_{SPP})\big);$$

$$E_{20} = \big(\{\{\}, \{(\boldsymbol{i}, id_V, id_{N_1}, v_1, ?)\}, \{(\boldsymbol{i}, ?, id_{N_1}, v_1)\}\}, (\nu\, c_{AV})\,(\nu\, c_{VA})\,(\nu\, c_{AB})\,(\nu\, c_{BA})\,(\nu\, c_{BV})\,(\nu\, c_{VB})\ (!A_{SPP} \mid !V_{SPP} \mid !B_{SPP})\big).$$

If we consider a wrong evolution of the protocol, when A is spending a coin that was not issued to him, we may consider the following evolution of configurations F, fake states.

$$F_1 = \big(\{\{(\boldsymbol{i}, ?, id_{N_1}, v_1)\}, \{\}, \{\}\}, (\nu\, c_{AV})\,(\nu\, c_{VA})\,(\nu\, c_{AB})\,(\nu\, c_{BA})\,(\nu\, c_{BV})\,(\nu\, c_{VB})\ (Buyer_{SPP}(id_{N_1}) \mid !A_{SPP} \mid !V_{SPP} \mid !B_{SPP})\big);$$

$$F_2 = \big(\{\{(\boldsymbol{t}, id_V, id_{N_1}, v_1)\}, \{\}, \{\}\}, (\nu\, c_{AV})\,(\nu\, c_{VA})\,(\nu\, c_{AB})\,(\nu\, c_{BA})\,(\nu\, c_{BV})\,(\nu\, c_{VB})\ (\overline{c_{AV}}\langle(\boldsymbol{t}, id_A, id_{N_1}, v_1)\rangle \mid !A_{SPP} \mid !V_{SPP} \mid !B_{SPP})\big);$$

$$F_3 = \big(\{\{(\boldsymbol{t}, id_V, id_{N_1}, v_1)\}, \{\}, \{\}\}, (\nu\, c_{AV})\,(\nu\, c_{VA})\,(\nu\, c_{AB})\,(\nu\, c_{BA})\,(\nu\, c_{BV})\,(\nu\, c_{VB})\ (!A_{SPP} \mid [\boldsymbol{t}\ is\ \boldsymbol{t}]\,.add_V(\boldsymbol{p}, id_A, id_{N_1}, v_1).\overline{c_{VA}}\langle(\boldsymbol{p}, id_V, id_{N_1}, v_1)\rangle \mid c_{BV}(x_1, x_2, x_3, x_4).[x_1\ is\ \boldsymbol{i}]\,.change_V(\boldsymbol{p}, x_2, x_3, x_4, \boldsymbol{i}, ?, x_3, x_4).\overline{c_{VB}}\langle(\boldsymbol{i}, id_V, x_3, x_4, x_2)\rangle \mid !V_{SPP} \mid !B_{SPP})\big);$$

$$F_4 = \big(\{\{(\boldsymbol{t}, id_V, id_{N_1}, v_1)\}, \{\}, \{\}\}, (\nu\, c_{AV})\,(\nu\, c_{VA})\,(\nu\, c_{AB})\,(\nu\, c_{BA})\,(\nu\, c_{BV})\,(\nu\, c_{VB})\ (!A_{SPP} \mid add_V(\boldsymbol{p}, id_A, id_{N_1}, v_1).\overline{c_{VA}}\langle(\boldsymbol{p}, id_V, id_{N_1}, v_1)\rangle \mid c_{BV}(x_1, x_2, x_3, x_4).[x_1\ is\ \boldsymbol{i}]\,.change_V(\boldsymbol{p}, x_2, x_3, x_4, \boldsymbol{i}, ?, x_3, x_4).\overline{c_{VB}}\langle(\boldsymbol{i}, id_V, x_3, x_4, x_2)\rangle \mid !V_{SPP} \mid !B_{SPP})\big);$$

$$F_5 = \big(\{\{(\boldsymbol{t}, id_V, id_{N_1}, v_1)\}, \{\}, \{(\boldsymbol{p}, id_A, id_{N_1}, v_1)\}\}, (\nu\, c_{AV})\,(\nu\, c_{VA})\,(\nu\, c_{AB})\,(\nu\, c_{BA})\,(\nu\, c_{BV})\,(\nu\, c_{VB})\ (!A_{SPP} \mid \overline{c_{VA}}\langle(\boldsymbol{p}, id_V, id_{N_1}, v_1)\rangle \mid c_{BV}(x_1, x_2, x_3, x_4).[x_1\ is\ \boldsymbol{i}]\,.change_V(\boldsymbol{p}, x_2, x_3, x_4, \boldsymbol{i}, ?, x_3, x_4).\overline{c_{VB}}\langle(\boldsymbol{i}, id_V, x_3, x_4, x_2)\rangle \mid !V_{SPP} \mid !B_{SPP})\big);$$

$$F_6 = \big(\{\{(\boldsymbol{t}, id_V, id_{N_1}, v_1)\}, \{\}, \{(\boldsymbol{p}, id_A, id_{N_1}, v_1)\}\}, (\nu\, c_{AV})\,(\nu\, c_{VA})\,(\nu\, c_{AB})\,(\nu\, c_{BA})\,(\nu\, c_{BV})\,(\nu\, c_{VB})\ ([\boldsymbol{p}\ is\ \boldsymbol{p}]\,.change_A(\boldsymbol{t}, id_V, id_{N_1}, v_1, \boldsymbol{s}, id_V, id_{N_1}, v_1).\overline{c_{AB}}\langle(\boldsymbol{t}, id_A, id_{N_1}, v_1, id_V)\rangle \mid c_{BA}(x_1, x_2, x_3, x_4).[x_1\ is\ \boldsymbol{s}]\,.remove_A(\boldsymbol{s}, x_2, x_3, x_4) \mid !A_{SPP} \mid c_{BV}(x_1, x_2, x_3, x_4).[x_1\ is\ \boldsymbol{i}]\,.change_V(\boldsymbol{p}, x_2, x_3, x_4, \boldsymbol{i}, ?, x_3, x_4).\overline{c_{VB}}\langle(\boldsymbol{i}, id_V, x_3, x_4, x_2)\rangle \mid !V_{SPP} \mid !B_{SPP})\big);$$

$$F_7 = \big(\{\{(\boldsymbol{t}, id_V, id_{N_1}, v_1)\}, \{\}, \{(\boldsymbol{p}, id_A, id_{N_1}, v_1)\}\}, (\nu\, c_{AV})\,(\nu\, c_{VA})\,(\nu\, c_{AB})\,(\nu\, c_{BA})\,(\nu\, c_{BV})\,(\nu\, c_{VB})\ (change_A(\boldsymbol{t}, id_V, id_{N_1}, v_1, \boldsymbol{s}, id_V, id_{N_1}, v_1).\overline{c_{AB}}\langle(\boldsymbol{t}, id_A, id_{N_1}, v_1, id_V)\rangle \mid c_{BA}(x_1, x_2, x_3, x_4).[x_1\ is\ \boldsymbol{s}]\,.remove_A(\boldsymbol{s}, x_2, x_3, x_4) \mid !A_{SPP} \mid c_{BV}(x_1, x_2, x_3, x_4).[x_1\ is\ \boldsymbol{i}]\,.change_V(\boldsymbol{p}, x_2, x_3, x_4, \boldsymbol{i}, ?, x_3, x_4).\overline{c_{VB}}\langle(\boldsymbol{i}, id_V, x_3, x_4, x_2)\rangle \mid !V_{SPP} \mid !B_{SPP})\big);$$

$$F_8 = \big(\{\{(\boldsymbol{s}, id_V, id_{N_1}, v_1)\}, \{\}, \{(\boldsymbol{p}, id_A, id_{N_1}, v_1)\}\}, (\nu\, c_{AV})\,(\nu\, c_{VA})\,(\nu\, c_{AB})\,(\nu\, c_{BA})\,(\nu\, c_{BV})\,(\nu\, c_{VB})\ (\overline{c_{AB}}\langle(\boldsymbol{t}, id_A, id_{N_1}, v_1, id_V)\rangle \mid c_{BA}(x_1, x_2, x_3, x_4).[x_1\ is\ \boldsymbol{s}]\,.remove_A(\boldsymbol{s}, x_2, x_3, x_4) \mid !A_{SPP} \mid c_{BV}(x_1, x_2, x_3, x_4).[x_1\ is\ \boldsymbol{i}]\,.change_V(\boldsymbol{p}, x_2, x_3, x_4, \boldsymbol{i}, ?, x_3, x_4).\overline{c_{VB}}\langle(\boldsymbol{i}, id_V, x_3, x_4, x_2)\rangle \mid !V_{SPP} \mid !B_{SPP})\big);$$

$F_9 = \big(\{\{(\boldsymbol{s}, id_V, id_{N_1}, v_1)\}, \{\}, \{(\boldsymbol{p}, id_A, id_{N_1}, v_1)\}\}, (\nu\, c_{AV})\,(\nu\, c_{VA})\,(\nu\, c_{AB})\,(\nu\, c_{BA})\,(\nu\, c_{BV})\,(\nu\, c_{VB})$
$(c_{BA}(x_1,x_2,x_3,x_4).[x_1\ is\ \boldsymbol{s}]\,.remove_A(\boldsymbol{s},x_2,x_3,x_4)\mid !A_{SPP}\mid$
$c_{BV}(x_1,x_2,x_3,x_4).[x_1\ is\ \boldsymbol{i}]\,.change_V(\boldsymbol{p},x_2,x_3,x_4,\boldsymbol{i},?,x_3,x_4).\overline{c_{VB}}\langle(\boldsymbol{i},id_V,x_3,x_4,x_2)\rangle\mid$
$!V_{SPP}\mid[\boldsymbol{t}\ is\ \boldsymbol{t}]\,.change_B(\boldsymbol{i},id_A,id_{N_1},v_1,?,\boldsymbol{t},id_A,id_{N_1},v_1,id_V).\overline{c_{BV}}\langle(\boldsymbol{i},id_A,id_{N_1},v_1)\rangle\mid$
$c_{VB}(x_1,x_2,x_3,x_4,x_5).[x_1\ is\ \boldsymbol{i}]\,.change_B(\boldsymbol{t},x_5,x_3,x_4,x_2,\boldsymbol{i},x_2,x_3,x_4,?).\overline{c_{BA}}\langle(\boldsymbol{s},x_2,x_3,x_4)\rangle\mid$
$!B_{SPP})\big);$

$F_{10} = \big(\{\{(\boldsymbol{s}, id_V, id_{N_1}, v_1)\}, \{\}, \{(\boldsymbol{p}, id_A, id_{N_1}, v_1)\}\}, (\nu\, c_{AV})\,(\nu\, c_{VA})\,(\nu\, c_{AB})\,(\nu\, c_{BA})\,(\nu\, c_{BV})\,(\nu\, c_{VB})$
$(c_{BA}(x_1,x_2,x_3,x_4).[x_1\ is\ \boldsymbol{s}]\,.remove_A(\boldsymbol{s},x_2,x_3,x_4)\mid !A_{SPP}\mid$
$c_{BV}(x_1,x_2,x_3,x_4).[x_1\ is\ \boldsymbol{i}]\,.change_V(\boldsymbol{p},x_2,x_3,x_4,\boldsymbol{i},?,x_3,x_4).\overline{c_{VB}}\langle(\boldsymbol{i},id_V,x_3,x_4,x_2)\rangle\mid$
$!V_{SPP}\mid change_B(\boldsymbol{i},id_A,id_{N_1},v_1,?,\boldsymbol{t},id_A,id_{N_1},v_1,id_V).\overline{c_{BV}}\langle(\boldsymbol{i},id_A,id_{N_1},v_1)\rangle\mid$
$c_{VB}(x_1,x_2,x_3,x_4,x_5).[x_1\ is\ \boldsymbol{i}]\,.change_B(\boldsymbol{t},x_5,x_3,x_4,x_2,\boldsymbol{i},x_2,x_3,x_4,?).\overline{c_{BA}}\langle(\boldsymbol{s},x_2,x_3,x_4)\rangle\mid$
$!B_{SPP})\big);$

$F_{11} = \big(\{\{(\boldsymbol{s}, id_V, id_{N_1}, v_1)\}, \{\}, \{(\boldsymbol{p}, id_A, id_{N_1}, v_1)\}\}, (\nu\, c_{AV})\,(\nu\, c_{VA})\,(\nu\, c_{AB})\,(\nu\, c_{BA})\,(\nu\, c_{BV})\,(\nu\, c_{VB})$
$(c_{BA}(x_1,x_2,x_3,x_4).[x_1\ is\ \boldsymbol{s}]\,.remove_A(\boldsymbol{s},x_2,x_3,x_4)\mid !A_{SPP}\mid$
$c_{BV}(x_1,x_2,x_3,x_4).[x_1\ is\ \boldsymbol{i}]\,.change_V(\boldsymbol{p},x_2,x_3,x_4,\boldsymbol{i},?,x_3,x_4).\overline{c_{VB}}\langle(\boldsymbol{i},id_V,x_3,x_4,x_2)\rangle\mid$
$!V_{SPP}\mid$
$c_{VB}(x_1,x_2,x_3,x_4,x_5).[x_1\ is\ \boldsymbol{i}]\,.change_B(\boldsymbol{t},x_5,x_3,x_4,x_2,\boldsymbol{i},x_2,x_3,x_4,?).\overline{c_{BA}}\langle(\boldsymbol{s},x_2,x_3,x_4)\rangle\mid$
$!B_{SPP})\big).$

We define $(\sigma_i')_X$ as the state of the principal X in the configuration E_i, $i \in \{1,\dots 20\}$. Similarly, we define $(\sigma_j'')_X$ as the state of the principal X in the configuration F_j, $j \in \{1,\dots 11\}$. Consider also the state σ as $\{\sigma_A \backslash \{(\boldsymbol{i},?,id_{N_1},v_1)\}, \sigma_B \backslash \{(\boldsymbol{i},id_A,id_{N_1},v_1,?)\}, \sigma_V\}$. Now, all that we have to notice is that the bisimulation that proves the equivalence is the bisimulation $\mathcal{S}_1$.

$$\mathcal{S}_1 = \bigcup_{i=1}^{20}\bigcup_{j=i}^{11}\Big((\sigma \oplus \sigma_i', E_i), (\sigma \oplus \sigma_i' \oplus \sigma_j'', E_i \mid F_j)\Big).$$

This is a B-weak bisimulation up to $\cong_B$, but using Proposition 3.10, we have a B-weak bisimulation.

A.2 Proof of the Lemma 5.3

In this section we prove that it will be not possible for A to "create" a coin and use it to pay to V. Since the second process gets stuck when A orders B to transfer the coin to V, the bisimulation that proves the result, $\mathcal{S}_2$, is very simple. We want to prove that

$$\big(\{\sigma_A, \sigma_B, \sigma_V\}, (\nu\, c_{AV})\,(\nu\, c_{VA})\,(\nu\, c_{AB})\,(\nu\, c_{BA})\,(\nu\, c_{BV})\,(\nu\, c_{VB})$$
$$(!A_{SPP}\mid !V_{SPP}\mid !B_{SPP})\big) \cong_B$$
$$\big(\{\sigma_A \uplus \{(\boldsymbol{i},?,id_{N_1},v_1)\}, \sigma_B, \sigma_V\}, (\nu\, c_{AV})\,(\nu\, c_{VA})\,(\nu\, c_{AB})\,(\nu\, c_{BA})\,(\nu\, c_{BV})\,(\nu\, c_{VB})$$
$$(Buyer_{SPP}(id_{N_1})\mid !A_{SPP}\mid !V_{SPP}\mid !B_{SPP})\big),$$

whenever $(\boldsymbol{i}, id_A, id_{N_1}, v_1, ?)\} \notin \sigma_B$.

$$
\begin{aligned}
\mathcal{S}_2 = \Big\{ & \Big(\big(\{\sigma_A, \sigma_B, \sigma_V\}, (\nu\, c_{AV})\,(\nu\, c_{VA})\,(\nu\, c_{AB})\,(\nu\, c_{BA})\,(\nu\, c_{BV})\,(\nu\, c_{VB})\,(!A_{SPP} \mid !V_{SPP} \mid !B_{SPP}) \big), \\
& \big(\{\sigma_A \uplus \{(\boldsymbol{i}, ?, id_{N_1}, v_1)\}, \sigma_B, \sigma_V\}, (\nu\, c_{AV})\,(\nu\, c_{VA})\,(\nu\, c_{AB})\,(\nu\, c_{BA})\,(\nu\, c_{BV})\,(\nu\, c_{VB}) \\
& \quad (Buyer_{SPP}(id_{N_1}) \mid !A_{SPP} \mid !V_{SPP} \mid !B_{SPP}) \big) \Big), \\
& \Big(\big(\{\sigma_A, \sigma_B, \sigma_V\}, (\nu\, c_{AV})\,(\nu\, c_{VA})\,(\nu\, c_{AB})\,(\nu\, c_{BA})\,(\nu\, c_{BV})\,(\nu\, c_{VB})\,(!A_{SPP} \mid !V_{SPP} \mid !B_{SPP}) \big), \\
& \big(\{\sigma_A \uplus \{(\boldsymbol{t}, id_V, id_{N_1}, v_1)\}, \sigma_B, \sigma_V\}, (\nu\, c_{AV})\,(\nu\, c_{VA})\,(\nu\, c_{AB})\,(\nu\, c_{BA})\,(\nu\, c_{BV})\,(\nu\, c_{VB}) \\
& \quad (\overline{c_{AV}}\langle(\boldsymbol{t}, id_A, id_{N_1}, v_1)\rangle \mid !A_{SPP} \mid !V_{SPP} \mid !B_{SPP}) \big) \Big), \\
& \Big(\big(\{\sigma_A, \sigma_B, \sigma_V\}, (\nu\, c_{AV})\,(\nu\, c_{VA})\,(\nu\, c_{AB})\,(\nu\, c_{BA})\,(\nu\, c_{BV})\,(\nu\, c_{VB})\,(!A_{SPP} \mid !V_{SPP} \mid !B_{SPP}) \big), \\
& \big(\{\sigma_A \uplus \{(\boldsymbol{t}, id_V, id_{N_1}, v_1)\}, \sigma_B, \sigma_V\}, (\nu\, c_{AV})\,(\nu\, c_{VA})\,(\nu\, c_{AB})\,(\nu\, c_{BA})\,(\nu\, c_{BV})\,(\nu\, c_{VB}) \\
& \quad (!A_{SPP} \mid [\boldsymbol{t}\ is\ \boldsymbol{t}]\,.add_V(\boldsymbol{p}, id_A, id_{N_1}, v_1).\overline{c_{VA}}\langle(\boldsymbol{p}, id_V, id_{N_1}, v_1)\rangle \mid \\
& \quad c_{BV}(x_1, x_2, x_3, x_4).[x_1\ is\ \boldsymbol{i}]\,.change_V(\boldsymbol{p}, x_2, x_3, x_4, \boldsymbol{i}, ?, x_3, x_4).\overline{c_{VB}}\langle(\boldsymbol{i}, id_V, x_3, x_4, x_2)\rangle \mid \\
& \quad !V_{SPP} \mid !B_{SPP}) \big) \Big), \\
& \Big(\big(\{\sigma_A, \sigma_B, \sigma_V\}, (\nu\, c_{AV})\,(\nu\, c_{VA})\,(\nu\, c_{AB})\,(\nu\, c_{BA})\,(\nu\, c_{BV})\,(\nu\, c_{VB})\,(!A_{SPP} \mid !V_{SPP} \mid !B_{SPP}) \big), \\
& \big(\{\sigma_A \uplus \{(\boldsymbol{t}, id_V, id_{N_1}, v_1)\}, \sigma_B, \sigma_V\}, (\nu\, c_{AV})\,(\nu\, c_{VA})\,(\nu\, c_{AB})\,(\nu\, c_{BA})\,(\nu\, c_{BV})\,(\nu\, c_{VB}) \\
& \quad (!A_{SPP} \mid add_V(\boldsymbol{p}, id_A, id_{N_1}, v_1).\overline{c_{VA}}\langle(\boldsymbol{p}, id_V, id_{N_1}, v_1)\rangle \mid \\
& \quad c_{BV}(x_1, x_2, x_3, x_4).[x_1\ is\ \boldsymbol{i}]\,.change_V(\boldsymbol{p}, x_2, x_3, x_4, \boldsymbol{i}, ?, x_3, x_4).\overline{c_{VB}}\langle(\boldsymbol{i}, id_V, x_3, x_4, x_2)\rangle \mid \\
& \quad !V_{SPP} \mid !B_{SPP}) \big) \Big), \\
& \Big(\big(\{\sigma_A, \sigma_B, \sigma_V\}, (\nu\, c_{AV})\,(\nu\, c_{VA})\,(\nu\, c_{AB})\,(\nu\, c_{BA})\,(\nu\, c_{BV})\,(\nu\, c_{VB})\,(!A_{SPP} \mid !V_{SPP} \mid !B_{SPP}) \big), \\
& \big(\{\sigma_A \uplus \{(\boldsymbol{t}, id_V, id_{N_1}, v_1)\}, \sigma_B, \sigma_V \uplus \{(\boldsymbol{p}, id_A, id_{N_1}, v_1)\}\}, (\nu\, c_{AV})\,(\nu\, c_{VA})\,(\nu\, c_{AB})\,(\nu\, c_{BA}) \\
& \quad (\nu\, c_{BV})\,(\nu\, c_{VB}) \\
& \quad (!A_{SPP} \mid \overline{c_{VA}}\langle(\boldsymbol{p}, id_V, id_{N_1}, v_1)\rangle \mid \\
& \quad c_{BV}(x_1, x_2, x_3, x_4).[x_1\ is\ \boldsymbol{i}]\,.change_V(\boldsymbol{p}, x_2, x_3, x_4, \boldsymbol{i}, ?, x_3, x_4).\overline{c_{VB}}\langle(\boldsymbol{i}, id_V, x_3, x_4, x_2)\rangle \mid \\
& \quad !V_{SPP} \mid !B_{SPP}) \big) \Big), \\
& \Big(\big(\{\sigma_A, \sigma_B, \sigma_V\}, (\nu\, c_{AV})\,(\nu\, c_{VA})\,(\nu\, c_{AB})\,(\nu\, c_{BA})\,(\nu\, c_{BV})\,(\nu\, c_{VB})\,(!A_{SPP} \mid !V_{SPP} \mid !B_{SPP}) \big), \\
& \big(\{\sigma_A \uplus \{(\boldsymbol{t}, id_V, id_{N_1}, v_1)\}, \sigma_B, \sigma_V \uplus \{(\boldsymbol{p}, id_A, id_{N_1}, v_1)\}\}, (\nu\, c_{AV})\,(\nu\, c_{VA})\,(\nu\, c_{AB})\,(\nu\, c_{BA}) \\
& \quad (\nu\, c_{BV})\,(\nu\, c_{VB}) \\
& \quad ([\boldsymbol{p}\ is\ \boldsymbol{p}]\,.change_A(\boldsymbol{t}, id_V, id_{N_1}, v_1, \boldsymbol{s}, id_V, id_{N_1}, v_1).\overline{c_{AB}}\langle(\boldsymbol{t}, id_A, id_{N_1}, v_1, id_V)\rangle \mid \\
& \quad c_{BA}(x_1, x_2, x_3, x_4).[x_1\ is\ \boldsymbol{s}]\,.remove_A(\boldsymbol{s}, x_2, x_3, x_4) \mid !A_{SPP} \mid \\
& \quad c_{BV}(x_1, x_2, x_3, x_4).[x_1\ is\ \boldsymbol{i}]\,.change_V(\boldsymbol{p}, x_2, x_3, x_4, \boldsymbol{i}, ?, x_3, x_4).\overline{c_{VB}}\langle(\boldsymbol{i}, id_V, x_3, x_4, x_2)\rangle \mid \\
& \quad !V_{SPP} \mid !B_{SPP}) \big) \Big),
\end{aligned}
$$

$$\Big(\big(\{\sigma_A,\sigma_B,\sigma_V\},(\nu\, c_{AV})\,(\nu\, c_{VA})\,(\nu\, c_{AB})\,(\nu\, c_{BA})\,(\nu\, c_{BV})\,(\nu\, c_{VB})\,(!A_{SPP}\mid !V_{SPP}\mid !B_{SPP})\big),$$

$$\big(\{\sigma_A \uplus \{(\boldsymbol{t},id_V,id_{N_1},v_1)\},\sigma_B,\sigma_V \uplus \{(\boldsymbol{p},id_A,id_{N_1},v_1)\}\},(\nu\, c_{AV})\,(\nu\, c_{VA})\,(\nu\, c_{AB})\,(\nu\, c_{BA})$$

$$(\nu\, c_{BV})\,(\nu\, c_{VB})$$

$$(change_A(\boldsymbol{t},id_V,id_{N_1},v_1,\boldsymbol{s},id_V,id_{N_1},v_1).\overline{c_{AB}}\langle(\boldsymbol{t},id_A,id_{N_1},v_1,id_V)\rangle \mid$$

$$c_{BA}(x_1,x_2,x_3,x_4).[x_1\ is\ \boldsymbol{s}]\,.remove_A(\boldsymbol{s},x_2,x_3,x_4) \mid !A_{SPP} \mid$$

$$c_{BV}(x_1,x_2,x_3,x_4).[x_1\ is\ \boldsymbol{i}]\,.change_V(\boldsymbol{p},x_2,x_3,x_4,\boldsymbol{i},?,x_3,x_4).\overline{c_{VB}}\langle(\boldsymbol{i},id_V,x_3,x_4,x_2)\rangle \mid$$

$$!V_{SPP}\mid !B_{SPP})\big)\Big),$$

$$\Big(\big(\{\sigma_A,\sigma_B,\sigma_V\},(\nu\, c_{AV})\,(\nu\, c_{VA})\,(\nu\, c_{AB})\,(\nu\, c_{BA})\,(\nu\, c_{BV})\,(\nu\, c_{VB})\,(!A_{SPP}\mid !V_{SPP}\mid !B_{SPP})\big),$$

$$\big(\{\sigma_A \uplus \{(\boldsymbol{s},id_V,id_{N_1},v_1)\},\sigma_B,\sigma_V \uplus \{(\boldsymbol{p},id_A,id_{N_1},v_1)\}\},(\nu\, c_{AV})\,(\nu\, c_{VA})\,(\nu\, c_{AB})\,(\nu\, c_{BA})$$

$$(\nu\, c_{BV})\,(\nu\, c_{VB})$$

$$(\overline{c_{AB}}\langle(\boldsymbol{t},id_A,id_{N_1},v_1,id_V)\rangle \mid$$

$$c_{BA}(x_1,x_2,x_3,x_4).[x_1\ is\ \boldsymbol{s}]\,.remove_A(\boldsymbol{s},x_2,x_3,x_4) \mid !A_{SPP} \mid$$

$$c_{BV}(x_1,x_2,x_3,x_4).[x_1\ is\ \boldsymbol{i}]\,.change_V(\boldsymbol{p},x_2,x_3,x_4,\boldsymbol{i},?,x_3,x_4).\overline{c_{VB}}\langle(\boldsymbol{i},id_V,x_3,x_4,x_2)\rangle \mid$$

$$!V_{SPP}\mid !B_{SPP})\big)\Big),$$

$$\Big(\big(\{\sigma_A,\sigma_B,\sigma_V\},(\nu\, c_{AV})\,(\nu\, c_{VA})\,(\nu\, c_{AB})\,(\nu\, c_{BA})\,(\nu\, c_{BV})\,(\nu\, c_{VB})\,(!A_{SPP}\mid !V_{SPP}\mid !B_{SPP})\big),$$

$$\big(\{\sigma_A \uplus \{(\boldsymbol{s},id_V,id_{N_1},v_1)\},\sigma_B,\sigma_V \uplus \{(\boldsymbol{p},id_A,id_{N_1},v_1)\}\},(\nu\, c_{AV})\,(\nu\, c_{VA})\,(\nu\, c_{AB})\,(\nu\, c_{BA})$$

$$(\nu\, c_{BV})\,(\nu\, c_{VB})$$

$$(c_{BA}(x_1,x_2,x_3,x_4).[x_1\ is\ \boldsymbol{s}]\,.remove_A(\boldsymbol{s},x_2,x_3,x_4) \mid !A_{SPP} \mid$$

$$c_{BV}(x_1,x_2,x_3,x_4).[x_1\ is\ \boldsymbol{i}]\,.change_V(\boldsymbol{p},x_2,x_3,x_4,\boldsymbol{i},?,x_3,x_4).\overline{c_{VB}}\langle(\boldsymbol{i},id_V,x_3,x_4,x_2)\rangle \mid$$

$$!V_{SPP} \mid [\boldsymbol{t}\ is\ \boldsymbol{t}]\,.change_B(\boldsymbol{i},id_A,id_{N_1},v_1,?,\boldsymbol{t},id_A,id_{N_1},v_1,id_V).\overline{c_{BV}}\langle(\boldsymbol{i},id_A,id_{N_1},v_1)\rangle \mid$$

$$c_{VB}(x_1,x_2,x_3,x_4,x_5).[x_1\ is\ \boldsymbol{i}]\,.change_B(\boldsymbol{t},x_5,x_3,x_4,x_2,\boldsymbol{i},x_2,x_3,x_4,?).\overline{c_{BA}}\langle(\boldsymbol{s},x_2,x_3,x_4)\rangle \mid$$

$$!B_{SPP})\big)\Big),$$

$$\Big(\big(\{\sigma_A,\sigma_B,\sigma_V\},(\nu\, c_{AV})\,(\nu\, c_{VA})\,(\nu\, c_{AB})\,(\nu\, c_{BA})\,(\nu\, c_{BV})\,(\nu\, c_{VB})\,(!A_{SPP}\mid !V_{SPP}\mid !B_{SPP})\big),$$

$$\big(\{\sigma_A \uplus \{(\boldsymbol{s},id_V,id_{N_1},v_1)\},\sigma_B,\sigma_V \uplus \{(\boldsymbol{p},id_A,id_{N_1},v_1)\}\},(\nu\, c_{AV})\,(\nu\, c_{VA})\,(\nu\, c_{AB})\,(\nu\, c_{BA})$$

$$(\nu\, c_{BV})\,(\nu\, c_{VB})$$

$$(c_{BA}(x_1,x_2,x_3,x_4).[x_1\ is\ \boldsymbol{s}]\,.remove_A(\boldsymbol{s},x_2,x_3,x_4) \mid !A_{SPP} \mid$$

$$c_{BV}(x_1,x_2,x_3,x_4).[x_1\ is\ \boldsymbol{i}]\,.change_V(\boldsymbol{p},x_2,x_3,x_4,\boldsymbol{i},?,x_3,x_4).\overline{c_{VB}}\langle(\boldsymbol{i},id_V,x_3,x_4,x_2)\rangle \mid$$

$$!V_{SPP} \mid change_B(\boldsymbol{i},id_A,id_{N_1},v_1,?,\boldsymbol{t},id_A,id_{N_1},v_1,id_V).\overline{c_{BV}}\langle(\boldsymbol{i},id_A,id_{N_1},v_1)\rangle \mid$$

$$c_{VB}(x_1,x_2,x_3,x_4,x_5).[x_1\ is\ \boldsymbol{i}]\,.change_B(\boldsymbol{t},x_5,x_3,x_4,x_2,\boldsymbol{i},x_2,x_3,x_4,?).\overline{c_{BA}}\langle(\boldsymbol{s},x_2,x_3,x_4)\rangle \mid$$

$$!B_{SPP})\big)\Big),$$

$$\Big(\big(\{\sigma_A, \sigma_B, \sigma_V\}, (\nu\, c_{AV})\,(\nu\, c_{VA})\,(\nu\, c_{AB})\,(\nu\, c_{BA})\,(\nu\, c_{BV})\,(\nu\, c_{VB})\,(!A_{SPP} \mid !V_{SPP} \mid !B_{SPP})\big),$$

$$\big(\{\sigma_A \uplus \{(\boldsymbol{s}, id_V, id_{N_1}, v_1)\}, \sigma_B, \sigma_V \uplus \{(\boldsymbol{p}, id_A, id_{N_1}, v_1)\}\}, (\nu\, c_{AV})\,(\nu\, c_{VA})\,(\nu\, c_{AB})\,(\nu\, c_{BA})$$

$$(\nu\, c_{BV})\,(\nu\, c_{VB})$$

$$(c_{BA}(x_1, x_2, x_3, x_4).[x_1\ is\ \boldsymbol{s}]\,.remove_A(\boldsymbol{s}, x_2, x_3, x_4) \mid !A_{SPP} \mid$$

$$c_{BV}(x_1, x_2, x_3, x_4).[x_1\ is\ \boldsymbol{i}]\,.change_V(\boldsymbol{p}, x_2, x_3, x_4, \boldsymbol{i}, ?, x_3, x_4).\overline{c_{VB}}\langle(\boldsymbol{i}, id_V, x_3, x_4, x_2)\rangle \mid !V_{SPP} \mid$$

$$c_{VB}(x_1, x_2, x_3, x_4, x_5).[x_1\ is\ \boldsymbol{i}]\,.change_B(\boldsymbol{t}, x_5, x_3, x_4, x_2, \boldsymbol{i}, x_2, x_3, x_4, ?).\overline{c_{BA}}\langle(\boldsymbol{s}, x_2, x_3, x_4)\rangle \mid$$

$$!B_{SPP})\big)\big)\Big\}.$$

The bisimulation $\mathcal{S}_2$ is just the normal evolution of the second protocol, always paired, with the initial state of the first protocol. It is easy to notice that the last pair of configurations have the same behaviour since the only differences are the input processes, but this behaviour can be imitated by the respective replications, and the states of A and V.

A.3 Proof of the Lemma 5.5

We describe briefly the proof of this lemma because it is very similar to the proof of Lemma 5.3. If we look at the proof of the Lemma 5.3, the only differences to this one are that the coin that is being used is issued, but to other principal, and that V will have the coin with number id_{N_2} twice in his wallet, one in the *issued* state, the valid one, and the other in the *pending* state, the forged one.

In the proof of the previous lemma, since the coin was not issued to A, $(\boldsymbol{i}, id_A, id_{N_1}, v_1, ?) \notin \sigma_B$, the process got stuck when A communicated the transference to the bank. This time, the process gets stuck in the exact same point because $(\boldsymbol{i}, id_A, id_{N_2}, v_1, ?) \notin \sigma_B$; the note is issued to V, $(\boldsymbol{i}, id_V, id_{N_2}, v_1, ?) \in \sigma_B$.

Index

Bibliography

[AG97a] M. Abadi and A. Gordon. A calculus for cryptographic protocols: The Spi Calculus. In Pfitzmann, Schunter, and Waidner, editors, *Proceedings of the* 4^{th} *ACM Conference on Computer and Communications Security*, pages 36–47, Zurich, Switzerland, April 1997. ACM Press. Full version available as Technical Report 414, Univ. Cambridge Computer Lab.

[AG97b] M. Abadi and A. Gordon. Reasoning about cryptographic protocols in the Spi Calculus. In A. Mazurkiewicz and J. Winkowski, editors, *Proceedings of the CONCUR'97,* 8^{th} *International Conference on Concurrency Theory*, volume 1243 of *Lecture Notes in Computer Science*, pages 59–73, Warsaw, Poland, July 1997. Springer-Verlag.

[AG98a] M. Abadi and A. Gordon. A bisimulation method for cryptographic protocols. *Nordic Journal of Computing*, 5(4):267–303, Winter 1998.

[AG98b] M. Abadi and A. Gordon. A calculus for cryptographic protocols: The Spi Calculus. Research Report 149, Digital Systems Research Center, Palo Alto, CA, USA, January 1998.

[AT91] M. Abadi and M. Tuttle. A semantics for a logic of authentication. In *Proceedings of the* 10^{th} *ACM Symposium on Principles of Distributed Computing*, pages 201–216. ACM Press, August 1991.

[BAN96] M. Burrows, M. Abadi, and R. Needham. A logic of authentication, from proceedings of the royal society, pp 233-271, volume 426, number 1871, 1989. In W. Stallings, editor, *Practical Cryptography for Data Internetworks*. IEEE Computer Society Press, 1996. A preliminary version appeared as Research Report 39, DEC Systems Research Center, Palo Alto, February 1989.

[BNP99] M. Boreale, R. De Nicola, and R. Pugliese. Proof techniques for cryptographic processes. In *Proceedings of the LICS'99,* 14^{th} *IEEE Symposium Logic in Computer Science*, pages 157–166. IEEE Computer Society Press, July 1999.

[Bra93] S. Brands. Untraceable off-line cash in wallets with observers (extended abstract). In D. Stinson, editor, *Proceedings of the CRYPTO '93 - Advances in Cryptology*, volume 773 of *Lecture Notes in Computer Science*, pages 302–318. Springer-Verlag, August 1993.

[Bra95] S. Brands. Off-line electronic cash based on secret-key certificates. In R. Baeza-Yates, E. Goles, and P. Goblete, editors, *Proceedings of the LATIN '95 - Second*

International Symposium of Latin American Theoretical Informatics, volume 911 of *Lecture Notes in Computer Science*, pages 131–166, Valparaiso, Chili, April 1995. Springer-Verlag. Also available as CWI technical report, CSR9506.

[Bra99] S. Brands. Electronic cash. In M. Atallah, editor, *Algorithms and Theory of Computation Handbook*, chapter 44. CRC Press, November 1999.

[Can00] R. Canetti. Security and composition of multi-party cryptographic protocols. *Journal of Cryptology*, 13(1):143–2002, 2000.

[Can01] R. Canetti. Universally composable security: a new paradigm for cryptographic protocols. Report, IBM T. J. Watson Research Center, October 2001.

[CdBvH+90] D. Chaum, B. den Boer, E. van Heyst, S. Mjølsnes, and A. Steenbeek. Effincient off-line electronic checks. In J. Quisquater and J. Vandewalle, editors, *Proccedings of the EUROCRYPT 89 - Advances in Cryptology*, Lecture Notes in Computer Science, pages 294–301. Springer-Verlag, 1990.

[CFN90] D. Chaum, A. Fiat, and M. Naor. Untraceable electronic cash. In S. Goldwasser, editor, *Proccedings of the CRYPTO '88 - Advances in Cryptology*, volume 403 of *Lecture Notes in Computer Science*, pages 319–327. Springer-Verlag, August 1990.

[CP93] D. Chaum and T. Pedersen. Wallet databases with observers. In E. Brickell, editor, *Proceedings of the CRYPTO '92 - Advances in Cryptology*, volume 740 of *Lecture Notes in Computer Science*, pages 89–105. Springer-Verlag, August 1993.

[Fer93a] N. Ferguson. Extensions of single-term coins. In D. Stinson, editor, *Proccedings of the CRYPTO '93 - Advances in Cryptology*, volume 773 of *Lecture Notes in Computer Science*, pages 292–301. Springer-Verlag, August 1993.

[Fer93b] N. Ferguson. Single term off-line coins. Technical Report CS-R9318, Center for Mathematics and Computer Science, Amsterdam, 1993.

[Fer94] N. Ferguson. Single term off-line coins. In *Proccedings of the EUROCRYPT '93 - Advances in Cryptology*, volume 765 of *Lecture Notes in Computer Science*, pages 318–328. Springer-Verlag, 1994.

[GM95] J. Gray and J. McLean. Using temporal logic to specify and verify cryptographic protocols (progress report). In *Proceedings of the 8^{th} IEEE Computer Security Foundations Workshop*, pages 108–116. IEEE Computer Society Press, 1995.

[Hoa80] C. A. R. Hoare. Communicating sequential processes. In R. McKeag and A. Macnaghten, editors, *On the construction of programs – an advanced course*, pages 229–254. Cambridge University Press, 1980.

[Low96] G. Lowe. Breaking and fixing the Needham-Shroeder public-key protocol using FDR. In T. Margaria and B. Steffen, editors, *Proccedings of the TACAS'96*, volume 1055 of *Lecture Notes in Computer Science*, pages 147–166. Springer-Verlag, 1996.

[MCJ97] W. Marrero, E. Clarke, and S. Jha. Model checking for security protocols. In *Proceedings of the DIMACS, Workshop on Design and Verication of Security Protocols*, 1997. A preliminary version appeared as Technical Report TR-CMU-CS-97-139, Carnegie Mellon University, May 1997.

[Mil80] R. Milner. *A Calculus of Communicating Systems*, volume 92 of *Lecture Notes in Computer Science*. Springer-Verlag, New York, NY, USA, 1980.

[Mil89] R. Milner. *Communication and Concurrency*. Prentice Hall, 1989.

[Mil99] R. Milner. *Communicating and Mobile Systems : The π-Calculus*. Cambridge University Press, June 1999.

[MMS02] P. Mateus, J. Mitchell, and A. Scedrov. Probabilistic polynomial calculus for secure computation analysis. On going paper, Section of Computer Science, Department of Mathematics, Instituto Superior Técnico, Lisboa, Portugal, 2002.

[MPW92] R. Milner, J. Parrow, and D. Walker. A calculus of mobile processes, parts I and II. *Information and Computation*, 100(1):1–40, 41–77, September 1992.

[MRTS01] J. Mitchell, A. Ramanathan, V. Teague, and A. Scedrov. A probabilistic polynomial-time calculus for analysis of cryptographic protocols. In S. Brookes and M. Mislove, editors, *Proceedings of the 17^{th} Annual Conference on the Mathematical Foundations of Programming Semantics*, volume 45 of *Electronic Notes in Theoretical Computer Science*, Arhus, Denmark, May 2001. Elsevier Science.

[Oka95] T. Okamoto. An efficient divisible electronic cash scheme. In D. Coppersmith, editor, *Proccedings of the CRYPTO '95 - Advances in Cryptology*, volume 963 of *Lecture Notes in Computer Science*, pages 438–451. Springer-Verlag, August 1995.

[OO92] T. Okamoto and K. Ohta. Universal electronic cash. In J. Feigenbaum, editor, *Proccedings of the CRYPTO '91 - Advances in Cryptology*, volume 576 of *Lecture Notes in Computer Science*, pages 324–337. Springer-Verlag, August 1992.

[Qua99] P. Quaglia. The π-calculus: Notes on labelled semantics. *Bulletin of the European Association for Theoretical Computer Science (BEATCS)*, 68:104–114, June 1999. Concurrency Column.

[Ros95] A. Roscoe. Modelling and verifying key-exchange using CSP and FDR. In *Proceedings of the 8^{th} IEEE Computer Security Foundations Workshop*. IEEE Computer Society Press, 1995.

[SC00] P. Syverson and I. Cervesato. The logic of authentication protocols. In R. Focardi and R. Gorrieri, editors, *Proceedings of the FOSAD'2000, International School on Foundations of Security Analysis and Design*, volume 2171 of *Lecture Notes in Computer Science*, pages 63–136, Bertinoro, Italy, September 2000. Springer.

[Sch98] S. Schneider. Verifying authentication protocols in CSP. In *Proceedings of the IEEE Transactions on Software Engineering*, volume 24, pages 741–758. IEEE Computer Society Press, 1998.

[SM96] P. Syverson and C. Meadows. A formal language for cryptographic protocol requirements. *Designs, Codes and Cryptography*, 7(1-2):27–59, 1996.

[Syv91] P. Syverson. The use of logic in the analysis of cryptographic protocols. In T. Lunt and J. McLean, editors, *Proceedings IEEE Symposium on Research in Security and Privacy*, pages 156–170. IEEE Computer Society, 1991.