COPPE UFRJ
Instituto Alberto Luiz Coimbra de Pós-Graduação e Pesquisa de Engenharia

# MODEL CHECKING DOLEV-YAO MULTI-AGENT EPISTEMIC LOGIC

Anna Carolina Carvalho Moreira de Oliveira

Dissertação de Mestrado apresentada ao Programa de Pós-graduação em Engenharia de Sistemas e Computação, COPPE, da Universidade Federal do Rio de Janeiro, como parte dos requisitos necessários à obtenção do título de Mestre em Engenharia de Sistemas e Computação.

Orientador: Mario Roberto Folhadela Benevides

Rio de Janeiro
Março de 2018

MODEL CHECKING DOLEV-YAO MULTI-AGENT EPISTEMIC LOGIC

Anna Carolina Carvalho Moreira de Oliveira

DISSERTAÇÃO SUBMETIDA AO CORPO DOCENTE DO INSTITUTO ALBERTO LUIZ COIMBRA DE PÓS-GRADUAÇÃO E PESQUISA DE ENGENHARIA (COPPE) DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE MESTRE EM CIÊNCIAS EM ENGENHARIA DE SISTEMAS E COMPUTAÇÃO.

Examinada por:

RIO DE JANEIRO, RJ – BRASIL
MARÇO DE 2018

Oliveira, Anna Carolina Carvalho Moreira de

Model Checking Dolev-Yao Multi-Agent Epistemic Logic/Anna Carolina Carvalho Moreira de Oliveira. – Rio de Janeiro: UFRJ/COPPE, 2018.

X, 55 p.: il.; $29,7$cm.

Orientador: Mario Roberto Folhadela Benevides

Dissertação (mestrado) – UFRJ/COPPE/Programa de Engenharia de Sistemas e Computação, 2018.

Referências Bibliográficas: p. 45 – 46.

1. knowledge. 2. security. 3. logic. 4. epistemic. 5. strips. 6. planning. 7. checker. I. Benevides, Mario Roberto Folhadela. II. Universidade Federal do Rio de Janeiro, COPPE, Programa de Engenharia de Sistemas e Computação. III. Título.

*To my mother and my father for*
*the support. And my boyfriend for*
*the partnership in academic life.*

# Acknowledgments

I would like to thank my advisor Mario Benevides for the opportunity and guidance since I was an undergraduate.

I must express my gratitude to my boyfriend Luiz Cláudio F. Fernandez for stay always with me, for our moments since graduation and for all the work developed together since undergraduate research.

I would also like to acknowledge Ivan Varzinczak for working with us in future possibilities for our project. And Simon Kramer for the talks about our work.

I also wish to thank Vitor Machado for his assistance.

And I would like to extend my gratitude to my family and my boyfriend's family for their support.

Resumo da Dissertação apresentada à COPPE/UFRJ como parte dos requisitos necessários para a obtenção do grau de Mestre em Ciências (M.Sc.)

VERIFICADOR DE MODELOS PARA LÓGICA EPISTÊMICA DOLEV-YAO

Anna Carolina Carvalho Moreira de Oliveira

Março/2018

Orientador: Mario Roberto Folhadela Benevides

Programa: Engenharia de Sistemas e Computação

Utilizamos a internet para quase tudo, inclusive para coisas que exigem um alto nível de sigilo, como por exemplo transações bancárias. Isso nos faz pensar na segurança necessária para manter seguro esse ambiente, que constantemente sofre ataques. Por esse motivo o estudo da criptografia está em constante avanço. Mas além disso é importante também que os protocolos de segurança não contenha vulnerabilidade. Dolev e Yao em 1983 conseguiram perceber que no protocolo de chave pública, que é amplamente utilizado até hoje, é possível um usuário malicioso descobrir o conteúdo de uma comunicação entre outros usuário da rede sem quebra da chave criptográfica, somente através de trocas de mensagens. Baseado no modelo apresentado nesse artigo, nós desenvolvemos uma extensão da lógica epistêmica, que expressa conhecimento e crença, para avaliar se os protocolos são seguros. E em cima desse nosso novo sistema, nós o traduzimos para a linguagem de strips, onde podemos automaticamente testar todas as possíveis ações de um intruso para descobrir se será possível ele conseguir o conteúdo da mensagem. E com esse novo formato do nosso sistema, nós desenvolvemos um verificador de modelo.

Abstract of Dissertation presented to COPPE/UFRJ as a partial fulfillment of the requirements for the degree of Master of Science (M.Sc.)

MODEL CHECKING DOLEV-YAO MULTI-AGENT EPISTEMIC LOGIC

Anna Carolina Carvalho Moreira de Oliveira

March/2018

Advisor: Mario Roberto Folhadela Benevides

Department: Systems Engineering and Computer Science

We use the web for almost everything, including for actions that require high level of secrecy, for example, banking transactions. It make us think about the necessary security to keep this environment safe, that constantly suffers cyber attacks. For this reason, the study of cryptography is always in advance. Besides, It is important do not have vulnerability in security protocol too. In 1983, Dolev and Yao realize that in public key protocol, which is widely used even nowadays, a malicious user in a network can discover the contents of communications between other users simply by eavesdropping on the exchange of messages. Based on the model introduced in the Dolev and Yao's article, we propose an epistemic logic extension to evaluate if a security protocol is safe. We transcripted this extension to strips language, where we can automatically test every possible intruder actions to discover contents of messages. And with this new notation we developed a model checker.

# Contents

# List of Figures

# Chapter 1

# Introduction

The internet is widely used by everyone, yet we are always susceptible to risks. In network traffic there are great quantities of sensitive information, therefore it's important that we use mechanisms to protect them. For this purpose, we need good cryptography algorithms and secure protocols too.

When dealing with communications safety, we are mostly concerned with the encryption used on the network. Encryption is important for keeping secrecy, but so is the logic of the protocols themselves.

Dolev and Yao proved in 1983 [1] that a malicious user in a network can discover the contents of communications between other users simply by eavesdropping on the exchange of messages. If the protocol itself is not safe, the dependence on solid encryption algorithms is increased even further.

We used this paper by Dolev and Yao as the basis for the logic presented here, due to its relevance in the field. We will present this model in Section 2.2.

In our research, we encountered many different approaches for the analysis of security protocols. Another expressive work is "A Logic of Authentication" by Burrows, Abadi and Needham [2]. In this article they proposed the BAN Logic, which however could hardly be considered an usual logic, due to the lack of certain classical operators on its syntax. However it's interesting to study their methods, and we present this in more detail in Section 2.3.

Abadi has other relevant works in the field of communications security. Abadi and Rogaway consider a distinction between a formal logic approach, and an algebraic one (computational model). However they also proposed a reconciliation of the two aspects in [3]. We do not discuss this paper in more detail in this work, since our proposal consists of a formal model similar to the articles previously mentioned. An example of a computational model that we analyzed for this work is Spy-Calculus by Abadi and Gordon [4].

The general idea presented in the work of Dolev and Yao is to reason about what an intruder can extract from private communications. Considering this, we decided to expand multi-agent epistemic logic, which are logics of knowledge and beliefs, to reason about security protocols. We further discuss epistemic logics in Section 2.1, and our own proposed system in Chapter 3.

In 1951, von Wright introduced the notion of work with modal logic to reasoning about knowledge [5]. Hintikkas's work defined a semantic to the notion of knowledge and belief [6]. With this, the area expanded in philosophy. Afterwards, fields such as computer science, artificial intelligence, linguistic realized the real potential for develop researches on epistemic logic.

Another important notion used in this project is the planning problem. It came from artificial intelligence progress, by necessity to program actions in robots and functions to evaluate every possibilities given a set of actions. Strips was a pioneer planner and was developed for the "Shakey the robot" project, designed at SRI International. Shakey was the first mobile robot controlled by artificial intelligence.

Finally, in this work we also introduce a model checker for our logic in Chapter 5. As we intend to provide this automated approach for the analysis of our logic, we considered the current state of the art in epistemic logic model checking, and also the existing implementations of authentication protocols. We decided to transcript our language into strips (Chapter 4), with which planning problems can be represented, and then we proceeded to develop our model checker. In the strips model, we are able to search across all possibilities automatically, something we further explain with the aid of examples in Section 2.4.

# Chapter 2

# A background of logic

In this chapter we discuss concepts of major importance, that serve as the base of our work. We present some definitions of multi-agent epistemic logic. Next, we describe the Dolev-Yao model, and introduce examples extracted from their paper which we later transcript to, and analyze in our own language. After that, we provide notions of the BAN Logic, and then introduce the operations used in planning problems.

## 2.1 Multi-Agents Epistemic Logic

We consider the multi-agent epistemic logic of [7–10], which is commonly named $S_5$ logic. In this logic it is possible to represent knowledge ($K$) and beliefs ($B$) of agents, and this formalization can be used for several purposes such as security protocol analysis, machine learning and game theory.

In this type of logic there exists some possible worlds where information cannot be known with certainty by some of its agents. For instance, consider a card game where the knowledge of agents is limited, and they need to work with possibilities to advance the game. Epistemic logic can formally represent all possible states and agents' doubts and certainties.

### 2.1.1 Language and semantics

**Definition 1** *The language of multi-agent epistemic logic consists of a set $\Phi$ of countably many propositional symbols, a finite set of agents $\mathcal{A}$, boolean connectives $\neg$, $\wedge$ and $\rightarrow$, and a modality $K_a$ for each agent $a$. The formulas are defined as follows:*

$$\varphi ::= p \mid \top \mid \neg\varphi \mid \varphi_1 \wedge \varphi_2 \mid \varphi_1 \rightarrow \varphi_2 \mid K_a\varphi$$

*where $a \in \mathcal{A}$ and $p \in \Phi$.*

We use a standard abbreviations:

- $\varphi_1 \vee \varphi_2 \equiv \neg(\neg\varphi_1 \wedge \neg\varphi_2)$;
- $\top \equiv \neg\bot$;
- $\varphi_1 \leftrightarrow \varphi_2 \equiv (\varphi_1 \rightarrow \varphi_2) \wedge (\varphi_2 \rightarrow \varphi_1)$;
- $B_a\varphi \equiv \neg K_a \neg\varphi$.

**Definition 2** *A multi-agent epistemic logic frame is a tuple $\mathcal{F} = (W, \sim_a)$. where $W$ is a non-empty set of states and $\sim_a$ is a reflexive, transitive and symmetric binary relation over $W$, for each $a \in \mathcal{A}$*

We say that model is reflexive when the states have a relation with itself (Figure 2.1(a)). Defined as:

$$\forall w \in W \rightarrow wRw$$

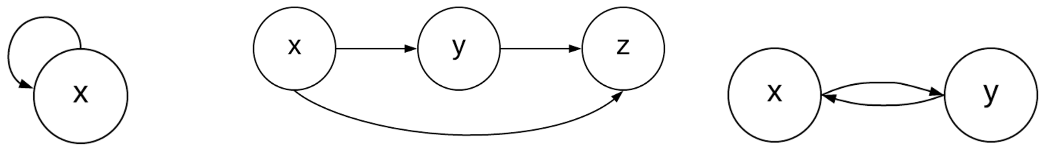
In transitive relation, if a state $x$ has a relation with state $y$ and $y$ has a relation with state $z$ then $x$ must have a relation with $z$ (Figure 2.1(b)). Defined as:

$$\forall w,\ w',\ w'' \in W,\ wRw' \wedge w'Rw'' \rightarrow wRw''$$

And in a model with symmetric binary we need to have a relation $(y, x)$ when we have the relation $(x, y)$ (Figure 2.1(c)). Defined as:

$$\forall w,\ w',\ wRw' \rightarrow w'Rw$$

Where $R$ is a simple relation. In Kripke model we represent the reflexive, transitive and symmetric binary relation with a simple edge (Figure 2.2).



(a) It is a reflexive frame (b) It is a transitive frame (c) It is a symmetric frame

Figure 2.1: In this figure we have a reflexive model, a transitive model and symmetric model

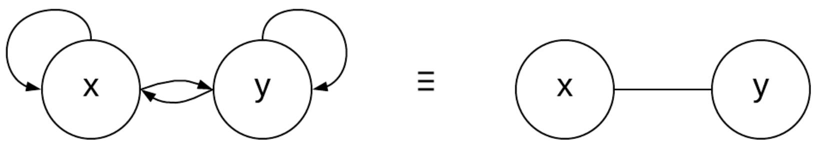


Figure 2.2: A Model with reflexive, transitive and symmetric binary relations and how we represent in Kripke model.

**Definition 3** *A multi-agent epistemic logic model (Kripke model) is $\mathcal{M} = (\mathcal{F}, V)$. Where $\mathcal{F}$ is an epistemic frame and $V$ is a valuation function $V : \Phi \to 2^W$.*

**Definition 4** *Given a multi-agent epistemic logic model $\mathcal{M} = \langle S, \sim_a, V\rangle$ and $s \in S$, the notion of satisfaction $\mathcal{M}, s \vDash \varphi$ is defined as follows:*

- $\mathcal{M}, s \vDash p$ *if and only if* $s \in V(p)$
- $\mathcal{M}, s \vDash \neg\varphi$ *if and only if* $\mathcal{M}, s \nvDash \varphi$
- $\mathcal{M}, s \vDash \varphi \wedge \psi$ *if and only if* $\mathcal{M}, s \vDash \varphi$ *e* $\mathcal{M}, s \vDash \psi$
- $\mathcal{M}, s \vDash \varphi \to \psi$ *if and only if* $\mathcal{M}, s \vDash \varphi$ *then* $\mathcal{M}, s \vDash \psi$
- $\mathcal{M}, s \vDash K_a\varphi$ *if and only if for all* $s' \in W$, *if* $s \sim_a s'$ *then* $\mathcal{M}, s' \vDash \varphi$

### 2.1.2 Axiomatization

All instantiations of propositional tautologies are valid axioms. In the multi-agent epistemic logic also we have the following axioms that are valid in reflexive, transitive and symmetric frames. And the inference rules are Modus Ponens (M.P.), Universal Generalization (U.G.) and Substitution (U.B.)

1. $K_a(\varphi \to \psi) \to (K_a\varphi \to K_a\psi)$
2. $K_a\varphi \to \varphi$
3. $K_a\varphi \to K_aK_a\varphi$
4. $\neg K_a\varphi \to K_a\neg K_a\varphi$

**Inference Rules**

M.P. $\varphi, \varphi \to \psi/\psi$ U.G. $\varphi/K_a\varphi$ U.B. $\varphi/\sigma\varphi$
where $\sigma$ is a map uniformly substituting formulas for propositional variables.

### 2.1.3 Example

**Example 1** *Consider three agents, Amy (A), Ben (B) and Carl (C), playing a card game. In this game there are three cards* 0, 1 *and* 2. *Ben gets a card with number* 1, *and only he sees its value. After that, Carl gets card* 2 *and he sees card* 0 *on the table. Only Carl can see the values of these two cards. Amy does not get a card, and she does not see any cards either. Next, we show the epistemic model for this situation.*

*The vertices represent the possible states, and edges denote the states that an agent can't distinguish.*

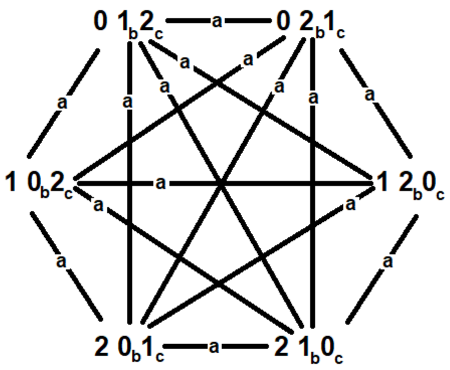


Figure 2.3: A Kripke model only with edges $a$ of example 1

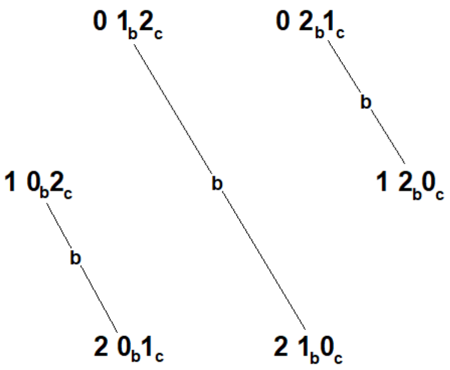


Figure 2.4: A Kripke model only with edges $b$ of example 1

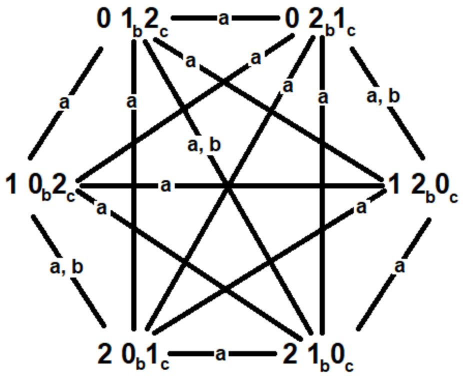


Figure 2.5: A Kripke model of example 1

*Here, we can infer some formulas, for example:*

- $01_b2_c \not\models K_a1_b$, *which means that in state* $01_b2_c$, *it is not true that Amy knows Ben's card is* 1*;*
- $02_b1_c \models (\neg K_b1_c) \wedge (K_b2_b)$, *here we say that in state* $02_b1_c$ *Ben does not know that Carl has card* 1*, and Ben knows card* 2 *is his card;*
- $20_b1_c \models K_c1_c \wedge K_c0_b$, *means that in the state* $20_b1_c$*, Carl knows that he has the card* 1*, and knows that Ben has card* 0*;*
- $12_b0_c \models K_b\neg(\neg 0_c \wedge \neg 1_c) \equiv 12_b0_c \models K_b(0_c \vee 1_c)$, *this formula represents the fact that Ben knows that Carl has either card* 0 *or card* 1*.*

*We can see in this model that to Amy any state is possible, since she has no information to exclude any of the possibilities. In contrast, Carl has no doubts, since he has sufficient information to distinguish between all possible states. Ben, on the other hand, has some information but not enough to know exactly who has every card.*

## 2.2 Dolev-Yao Model

Dolev and Yao's paper [1] is very important to the research of formal verification methods for security protocols. They present arguments showing that secure encryption protocols do not ensure privacy. If the logic of the communication protocol itself is not safe, a malicious user can discover contents of other users communications simply by looking at the exchange of messages.

### 2.2.1 Public Key Protocol

The Dolev-Yao model uses a public key protocol [11, 12], where each user has two functions: $E_X$ is a public encode function and every agent in the network knows it. And $D_X$ is a decode function that only user $X$ has knowledge about. It is required that $E_X D_X = D_X E_X$, and if any user $Y$ receives $E_X(M)$, nothing about message $M$ can be extracted.

### 2.2.2 Examples

**Example 2** *In this example agent $A$ send message $M$ to agent $B$. However a malicious user $Z$ intercepts the encoded message, and forwards it to $B$ as if $Z$ were the original source.*

$$\begin{tikzcd} A \arrow[r] & (A, E_B(M), B) \arrow[r] & B \end{tikzcd}$$

Figure 2.6: Example 2 - $A$ sends a message to $B$

$$\begin{tikzcd} A \arrow[r] \arrow[dr, "{(A,E_B(M),B)}"'] & {|} \arrow[r] & B \\ & Z \arrow[ur, "{(Z,E_B(M),B)}"'] & \end{tikzcd}$$

(a) $Z$ intercepts the message

$$\begin{tikzcd} A \arrow[r] \arrow[dr, "{(A,E_B(M),B)}"'] & {|} \arrow[r] & B \arrow[dl, "{(B,E_Z(M),Z)}"] \\ & Z & \end{tikzcd}$$

(b) $B$ replies to $Z$

Figure 2.7: Example 2 - $Z$ intercepts the message and $B$ replies to $Z$

1. *User $A$ sends a message to user $B$. [Figure 2.6]*

*2. Intruder $Z$ intercepts a message sent from $A$ to $B$. [Figure 2.7(a)]*

*3. Intruder $Z$ sends message $(Z, E_B(M), B)$ to $B$. [Figure 2.7(a)]*

*4. $B$ sends message $(B, E_Z(M), Z)$ to $Z$. [Figure 2.7(b)]*

*5. Intruder $Z$ decodes $E_Z(M)$ and obtains $M$.*

**Example 3** *Agent $A$ sends a message $MA$ to $B$, and $B$ replies with message $M$ encoded with the key of the user provided in the encrypted message itself, which results in the message being unreadable by an intruder $Z$.*

$$\begin{tikzcd} A \arrow[r] & (A, E_B(MA), B) \arrow[r] & B \end{tikzcd}$$

Figure 2.8: Example 3 - $A$ sends a message to $B$

$$\begin{tikzcd} A \arrow[r] \arrow[dr, "{(A,E_B(MA),B)}"'] & \mid \arrow[r] & B \\ & Z \arrow[ur, "{(Z,E_B(MA),B)}"'] & \end{tikzcd}$$

(a) $Z$ intercepts the message

$$\begin{tikzcd} A \arrow[r] \arrow[dr, "{(A,E_B(MA),B)}"'] & \mid \arrow[r] & B \arrow[dl, "{(B,E_A(MB),Z)}"] \\ & Z & \end{tikzcd}$$

(b) $B$ replies to $Z$

Figure 2.9: Example 3 - $Z$ intercepts the message and $B$ replies to $Z$

*1. $A$ sends message $MA$ to $B$. [Figure 2.8]*

*2. Intruder $Z$ intercepts the message sent from $A$ to $B$. [Figure 2.9(a)]*

*3. Intruder $Z$ sends message $(Z, E_B(MA), B)$ to $B$. [Figure 2.9(a)]*

*4. $B$ sends the message $(B, E_A(MB), Z)$ to $Z$. [Figure 2.9(b)]*

*5. Intruder $Z$* **cannot** *decode $E_A(MB)$ to obtain $M$.*

*It can be proved that this protocol is secure against any arbitrary behaviour of the intruder.*

**Example 4** *Now, agent $A$ sends message $E_B(E_B(M)A)$ to $B$. And when $B$ replies, an intruder $Z$ intercepts it. $Z$ uses $E_B(M)A$ as $M'$, and sends $E_B(E_B(M')Z)$ to $A$ which leads to $Z$ eventually being able to read $M$.*

$$\begin{tikzcd} A \arrow[r] & (A, E_B(E_B(M)A), B) \arrow[r] & B \end{tikzcd}$$

(a) $A$ sends a message to $B$

$$\begin{tikzcd} B \arrow[r] & (B, E_A(E_A(M)B), A) \arrow[r] & A \end{tikzcd}$$

(b) $B$ replies to $A$

Figure 2.10: Example 4 - $A$ sends a message to $B$ and $B$ replies to $A$

$$\begin{tikzcd} A & \mid \arrow[l] & B \arrow[l] \arrow[dl, "(B{,}E_A(E_A(M)B){,}A)"] \\ & Z \arrow[ul, "(Z{,}E_A(E_A(M')Z){,}A)"] & \end{tikzcd}$$

(a) $B$ replies to $A$ and $Z$ intercepts

$$\begin{tikzcd} A \arrow[dr, "(A{,}E_Z(E_Z(M')A){,}Z)"'] & \mid \arrow[l] & B \arrow[l] \arrow[dl, "(B{,}E_A(E_A(M)B){,}A)"] \\ & Z & \end{tikzcd}$$

(b) $A$ replies to $Z$

$$\begin{tikzcd} A & \mid \arrow[l] & B \arrow[l] \arrow[dl, "(B{,}E_A(E_A(M)B){,}A)"] \\ & Z \arrow[ul, "(Z{,}E_A(E_A(M)Z){,}A)"] & \end{tikzcd}$$

(c) $Z$ replies to $A$

$$\begin{tikzcd} A \arrow[dr, "(A{,}E_Z(E_Z(M)A){,}Z)"'] & \mid \arrow[l] & B \arrow[l] \arrow[dl, "(B{,}E_A(E_A(M)B){,}A)"] \\ & Z & \end{tikzcd}$$

(d) $A$ replies to $Z$ again

Figure 2.11: Example 4 - $B$ replies to $A$ and $Z$ intercepts. $A$ replies to $Z$. Also $Z$ replies to $A$ and $A$ replies to $Z$ again.

1. *$A$ sends message $E_B(E_B(M)A)$ to $B$. [Figure 2.10(a)]*

2. *$B$ replies message $E_A(E_A(M)B)$ to $A$. [Figure 2.10]*

3. *The intruder $Z$ intercepts the message sent from $B$ to $A$. [Figure 2.11(a)]*
4. *Intruder $Z$ denote $E_A(M)B$ by $M'$.*
5. *Intruder $Z$ sends message $(Z, E_A(E_A(M')Z), A)$ to $A$. [Figure 2.11(a)]*
6. *$A$ sends message $(A, E_Z(E_Z(M')A), Z)$ to $Z$. [Figure 2.11(b)]*
7. *Intruder $Z$ decodes $E_Z(M')$ and obtains $E_A(M)$.*
8. *Intruder $Z$ sends message $(Z, E_A(E_A(M)Z), A)$ to $A$. [Figure 2.11(c)]*
9. *$A$ sends message $(A, E_Z(E_Z(M)A), Z)$ to $Z$. [Figure 2.11(d)]*
10. *Intruder $Z$ decodes $E_Z(M)$ and obtains $M$.*

### 2.2.3 Rules

The rules defined here are not presented on the same terms as the original paper, however they can be easily derived as well with the theory presented. Consider a set of keys $\mathcal{K} = \{K_1, ...\}$, and $\{M\}_K$ represents a message $M$ encrypted with key $K$.

*Reflexivity*

$$\frac{M \in T}{T \vdash M}$$

*Encryption*

$$\frac{T \vdash K \quad T \vdash M}{T \vdash \{M\}_K}$$

*Decryption*

$$\frac{T \vdash \{M\}_K \quad T \vdash K}{T \vdash M}$$

Pair-Composition

$$\frac{T \vdash M \quad T \vdash N}{T \vdash (M, N)}$$

Pair-Decomposition

$$\frac{T \vdash (M, N)}{T \vdash M} \qquad \frac{T \vdash (M, N)}{T \vdash N}$$

## 2.3 BAN Logic

In another well-known article about authentication in security protocols by M. Burrows, M. Abadi e R. Needham published in 1990 [2], they proposed the BAN logic, which name derives from the authors' initials. This is an important work in the field of authentication. It's similar to Hoare logic and somewhat distant from usual propositional or classical logics. In the paper they analyzed two famous protocols, Kerberos and Andrew Secure RPC Handshake. We used our proposed system to analyze these as well, and the results are presented in Appendix A.

### 2.3.1 Symbols

The symbols of the logic corresponds to the users of the network, the keys used in the protocols, and statements.

By convention, to denote different kinds of keys we use $K_{ab}$, $K_{as}$ and $K_{bs}$ for shared ones, $K_a$, $K_b$ and $K_s$ for public ones, and $K_a^{-1}$, $K_b^{-1}$ and $K_s^{-1}$ for secret ones. To represent specific statements we use $N_a$, $N_b$ and $N_s$. Also we typically use $A$, $B$ for specific agents and $S$ for server agent.

To range over the symbols, we frequently use $P$, $Q$ for agents and $R$ , $X$, $Y$ for statements and $K$ for encryption keys.

### 2.3.2 Syntax

This syntax is quite unlike that of classical logic. For instance, the only propositional connective is the conjunction, which is denoted by a comma (,). These conjunctions respects properties such as associativity and commutativity.

To compose the syntax the system has $P$ **believes** $X$, $P$ **sees** $X$, $P$ **said** $X$, $P$ **controls** $X$ (meaning that $P$ has jurisdiction over $X$) and **fresh**($X$) (it is important in BAN Logic to know if a formula $X$ is fresh). We also have $P \overset{K}{\leftrightarrow} Q$, which represents that $P$ and $Q$ may use the shared key $K$ to communicate; $\overset{K}{\mapsto} P$ meaning that $P$ has $K$ as a public key; $P \overset{X}{\rightleftharpoons} Q$ which means that a formula $X$ is a secret known only to $P$ and $Q$; $\{X\}_K$ representing formula $X$ encrypted with the key $K$; and $\langle X\rangle_Y$ which represents the formula $X$ combined with the formula $Y$. In actual implementations, $\langle X\rangle_Y$ means $X$ is simply concatenated with the formula $Y$.

Therefore, we have the following syntax:

$$\varphi ::= P \textbf{ believes } X \mid P \textbf{ sees } X \mid P \textbf{ said } X \mid P \textbf{ controls } X \mid \textbf{fresh } X$$

$$\psi ::= P \overset{K}{\leftrightarrow} Q \mid \overset{K}{\mapsto} P \mid P \overset{X}{\rightleftharpoons} Q \mid \langle X\rangle_Y$$

Where $\varphi$ works with information and $\psi$ works with keys and passwords.

### 2.3.3 Logical Postulates

In their paper, Burrows, Abadi and Needham showed us the necessity to distinguish between what happened in past, to what's occurring in the present. In their work, they consider that the present starts at the beginning of the protocol execution. It's also important to keep in mind that the notion of belief is very important in BAN Logic, as it is essential to define if an user can trust an incoming message. With that in mind, we proceed to the rules introduced by them.

The three following rules are concerned with the meaning of messages. This rules represent that if $P$ believes in key or secret authenticity and sees a information encoded with this key or secret, then $P$ believes that $Q$ said the information.

$For\, shared\, keys$

$$\frac{P\,\textbf{believes}\,Q\overset{K}{\leftrightarrow}P, \quad P\,\textbf{sees}\,\{X\}_K}{P\,\textbf{believes}\,Q\,\textbf{said}\,X}$$

$For\, public\, keys$

$$\frac{P\,\textbf{believes}\,\overset{K}{\mapsto}Q, \quad P\,\textbf{sees}\,\{X\}_{K^{-1}}}{P\,\textbf{believes}\,Q\,\textbf{said}\,X}$$

$For\, shared\, secrets$

$$\frac{P\,\textbf{believes}\,Q\overset{Y}{\rightleftharpoons}P, \quad P\,\textbf{sees}\,\langle X\rangle_Y}{P\,\textbf{believes}\,Q\,\textbf{said}\,X}$$

The following rule is concerned with whether an user has jurisdiction over some information. It expresses that if $P$ believes that $Q$ has jurisdiction over a information and believes in it, then $P$ believes in this information.

$$\frac{P\,\textbf{believes}\,Q\,\textbf{controls}\,X, \quad P\,\textbf{believes}\,Q\,\textbf{believes}\,X}{P\,\textbf{believes}\,X}$$

These rules are related to what an agent sees. This rules are equivalent pair decomposition and decoded rules.

$$\frac{P\,\textbf{sees}\,(X,Y)}{P\,\textbf{sees}\,X} \qquad\qquad \frac{P\,\textbf{sees}\,\langle X\rangle_Y}{P\,\textbf{sees}\,X}$$

$$\frac{P\,\textbf{believes}\,Q \overset{K}{\leftrightarrow} P, \quad P\,\textbf{sees}\,\{X\}_K}{P\,\textbf{sees}\,X}$$

$$\frac{P\,\textbf{believes}\,\overset{K}{\mapsto} P, \quad P\,\textbf{sees}\,\{X\}_K}{P\,\textbf{sees}\,X}$$

$$\frac{P\,\textbf{believes}\,\overset{K}{\mapsto} Q, \quad P\,\textbf{sees}\,\{X\}_{K^{-1}}}{P\,\textbf{sees}\,X}$$

This rule determines if an information is recent. In this postulate we have that If a piece of information in a pair composition is fresh, then $P$ believes that all pair composition is recent.

$$\frac{P\,\textbf{believes}\quad \textbf{fresh}(X)}{P\,\textbf{believes}\quad \textbf{fresh}\,(X,Y)}$$

### 2.3.4 On Quantifiers in Delegations

The following captures how jurisdiction delegation works:

$$A\ \textbf{believes}\ S\ \textbf{controls}\ A \overset{K}{\leftrightarrow} B$$

$$A\,\textbf{believes}\,\forall K.(S\,\textbf{controls}\,A \overset{K}{\leftrightarrow} B)$$

$$A\,\textbf{believes}\,\forall K.(S\,\textbf{controls}\,B\,\textbf{controls}\,A \overset{K}{\leftrightarrow} B)$$

$$A\,\textbf{believes}\,S\,\textbf{controls}\,\forall K.(B\,\textbf{controls}\,A \overset{K}{\leftrightarrow} B)$$

$$\frac{P\,\textbf{believes}\,\forall V_1\ \ldots\ V_n.(Q\,\textbf{controls}\,X)}{P\,\textbf{believes}\,Q'\,\textbf{controls}\,X'}$$

### 2.3.5 Example

**Example 5** *This is the Example 1 of Dolev and Yao's paper written in BAN Logic notation. The example that A send a message M to B, but a intruder Z intercepts the encoded message and forwards it to B.*

1. $m_1 : A \longrightarrow B : \{m\}_{K_B}$
2. $m_2 : Z \longrightarrow B : \{m\}_{K_B}$
3. $m_3 : B \longrightarrow Z : \{m\}_{K_Z}$
4. $B$ *believes* $A \overset{K_B}{\leftrightarrow} B$
5. $Z$ *believes* $B \overset{K_Z}{\leftrightarrow} Z$
6. $m_1 : Z$ *sees* $\{m\}_{K_B}$
7. $m_2 : B$ *sees* $\{m\}_{K_B}$
8. $B$ *sees* $m$ *(agent sees rule)*
9. $m_3 : Z$ *sees* $\{m\}_{K_Z}$
10. $Z$ *sees* $m$ *(agent sees rule)*

*In this example, the items beginning with $m_x$ represent the protocol itself, that is, the sequence of messages which were exchanged. The others are inferences made with the BAN Logic syntax and its derived rules.*

## 2.4 Planning Problem

The planning problem consists of an automated process to check if a goal is achievable, given the starting state and the defined actions. With the information acquired in the process, it's possible to construct the possibility graph [13, 14].

This is interesting for the field of artificial intelligence because it results in a tree of possibilities given a set of actions and an initial state, with which one can find a path to reach the intended state. This can be applied in robot and games programming, for performance analysis, and in some decision making processes.

**Definition 5** *The classic planning problem is a tuple $\langle S, Ac, s_0, S_G\rangle$, where $S$ is the set of all possible states, $Ac$ a set of actions, $s_0 \in S$ is the initial state, and $S_G \subseteq S$ is a set containing the possible goals.*

### 2.4.1 Example

One of the widely known examples in planning problem is the "blocks world". We have three blocks $A$, $B$ and $C$ sitting on a table and we want stack them in a way that the $C$ stay in bottom and $A$ in top. An action can only be applied to one block at a time, and an upper block is required to move, so you can move the bottom of the stack. And only one block can stay directly on top of another.

Here we have propositions to represent the table, the blocks (using a variable to distinguish each one), when a block is on another block or the table ($On$) and if the block has nothing on top of it ($Clear$). Also, the actions are move, when a block sitting on a table or another block and we move to the top of different block, and move to table.

- $Table$;
- $Block(x)$;
- $On(x, y)$;
- $Clear(x)$;
- Action( $Move(z, x, y)$,
  - Precondition: $On(z, x) \wedge Clear(z) \wedge Clear(y) \wedge Block(z) \wedge Block(y) \wedge (z \neq x) \wedge (z \neq y) \wedge (x \neq y)$
  - Effect: $On(z, y) \wedge Clear(x) \wedge \neg On(z, x) \wedge \neg Clear(y)$ )
- Action( $MoveToTable(z, x)$,
  - Precondition: $On(z, x) \wedge Clear(z) \wedge Clear(Table) \wedge Block(z) \wedge Table \wedge (z \neq x)$
  - Effect: $On(z, Table) \wedge Clear(x) \wedge \neg On(z, x) \wedge \neg Clear(Table)$ )

It is necessary to specify an initial state and a goal, as follows:

- Initial( $Block(A) \wedge Block(B) \wedge Block(C) \wedge On(A, Table) \wedge On(B, Table) \wedge On(C, Table) \wedge Clear(A) \wedge Clear(B) \wedge Clear(C)$ )
- Goal( $On(A, B) \wedge On(B, C) \wedge Clear(Table) \wedge Clear(Table)$ )

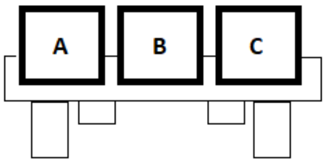


Figure 2.12: Initial state

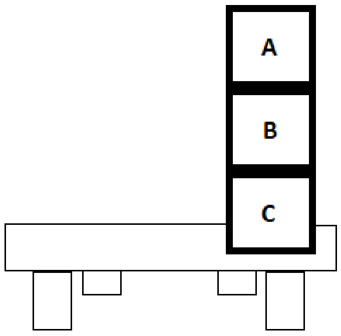


Figure 2.13: Goal

### 2.4.2 STRIPS (Stanford Research Institute Problem Solver)

The Stanford Research Institute Problem Solver is a problem solver developed by Fikes and Nilsson in 1971 [15, 16]. The language was created with the goal of implementing efficient operators. It was developed to be a planner for the first mobile robot controlled by artificial inteligence, "Shakey the robot".

In Strips, the problem space is formed by an initial state, a set of operator with their effects, and goal conditions. The search space is a set of all possible worlds that are transversed to locate a goal. It applies the operators which change the current state, until it reaches the goal conditions. An operator consists of a set of preconditions and effects, which can be in a delete list, or an add list.

They introduced the search strategy as a search tree formed by goal, subgoals and models generated in process. In this representation, each node has $(\langle world\ model\rangle, \langle goal\ list\rangle)$.

#### Example

Now we introduce a simplification of example presented by them.

In this example, we can realize the potential of this system. When the robot receives the task, he use the set of actions given to map all possibilities and analyze what the sequence of actions must be executed to achieve the goal.

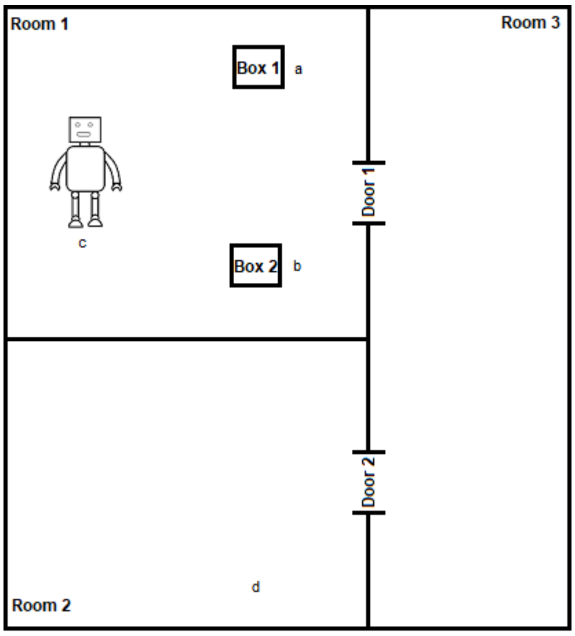


Figure 2.14: Example of strips - Robot

### Initial World

- $(\forall x \forall y \forall z)[Connects(x, y, z) \Rightarrow Connects(x, z, y)]$
- $Connects(Door\ 1, Room\ 1, Room\ 3)$
- $Connects(Door\ 2, Room\ 2, Room\ 3)$
- $LocInRoom(d, Room\ 2)$
- $At(Box\ 1, a)$
- $At(Box\ 2, b)$
- $AtRobot(c)$
- $Type(Box\ 1, Box)$
- $Type(Box\ 2, Box)$
- $Type(Door\ 1, Door)$
- $Type(Door\ 2, Door)$
- $InRoom(Box\ 1, Room\ 1)$
- $InRoom(Box\ 2, Room\ 1)$
- $InRoom(Robot, Room\ 1)$

- $Pushable(Box\ 1)$
- $Pushable(Box\ 2)$
- $OnFloor$

**Operators**

- *goto1(m)*: Robot goes to m.
  - Preconditions: $OnFloor \wedge \exists x[InRoom(Robot, x) \wedge LocInRoom(m, x)]$
  - Delete list: $AtRobot(S), NextTo(Robot, S)$
  - Add list: $AtRobot(m)$
- *goto2(m)*: Robot goes next to object m.
  - Preconditions: $OnFloor \wedge \exists x[InRoom(Robot, x) \wedge LocInRoom(m, x)] \vee \exists x \exists y[InRoom(Robot, x) \wedge Connects(m, x, y)]$
  - Delete list: $AtRobot(S)$, $NextTo(Robot, S)$
  - Add list: $NextTo(Robot, m)$
- *pushto(m, n)*: Robot pushes object m next to object n.
  - Preconditions: $Pushable(m) \wedge OnFloor \wedge NextTo(Robot, m) \wedge \{\exists x[InRoom(m, x) \wedge InRoom(n, x)] \vee \exists x \exists y[InRoom(m, x) \wedge Connects(n, x, y)]\}$
  - Delete list: $AtRobot(S)$, $NextTo(Robot, S)$, $NextTo(S, m)$, $At(m, S)$, $NextTo(m, S)$
  - Add list: $NextTo(Robot, m)$, $NextTo(m, n)$, $NextTo(n, m)$
- *gothrudoor(k, l, m)*: Robot goes through door k from room l into room m.
  - Preconditions: $NextTo(Robot, k) \wedge Connects(k, l, m) \wedge InRoom(Robot, l) \wedge OnFloor$
  - Delete list: $AtRobot(S)$, $NextTo(Robot, S)$, $InRoom(Robot, S)$
  - Add list: $InRoom(Robot, m)$

**Tasks**

1. Push two boxes together

- Goal: $NextTo(Box\ 1, Box\ 2)$
- Strips solution: $\{goto2(Box\ 2)$, $pushto(Box\ 2, Box\ 1)\}$

2. Go to a location in another room

   - Goal: $AtRobot(d)$
   - Strips solution: $\{goto2(Door\ 1)$, $gothrudoor(Door\ 1, Room\ 1, Room\ 3)$, $goto2(Door\ 2)$, $gothrudoor(Door\ 2, Room\ 3, Room\ 2)$, $goto1(d)\}$

# Chapter 3

# Dolev-Yao Multi-Agent Epistemic Logic

In Dolev and Yao's model (seen in section 2.2), the focus is on reasoning about an intruder. The idea is mapping possible actions and knowledge acquired by the agents. On the other hand, epistemic logic (seen in section 2.1) can formally express what the agents know about the world. So we attempted to introduce substantial elements to better represent the information learned by an outsider during communications.

This system is an extension to the epistemic logic of knowledge and beliefs, to reason about security protocols [17]. We can use the expressive power of $S_5$ to think about users' possibilities, and what an user can discover during communications. More importantly, what information an intruder can extract from these communications.

## 3.1 Language

The language is a multi-agent epistemic logic extension. Now, we have expressions beyond the propositions in the formulas. With this, we can denote keys, pair composition of information and information encoded under specific key. Also, we have the same standard abbreviations to express the $\vee$, $\bot$ and $\leftrightarrow$.

**Definition 6** *The Dolev-Yao multi-agent epistemic logic language consists of an enumerable set $\Phi$ of propositional symbols, a finite set of agents $\mathcal{A}$, an enumerable set of keys $\mathcal{K} = \{k_1, \cdots\}$, boolean connectives $\neg$, $\wedge$ and $\rightarrow$, and a modality $K_a$ for each agent $a$. The expressions and formulas are defined as follows:*

$$E ::= p \mid k \mid (E_1, E_2) \mid \{E\}_k$$

*where $k \in \mathcal{K}$.*

$$\varphi ::= e \mid \top \mid \neg\varphi \mid \varphi_1 \wedge \varphi_2 \mid \varphi_1 \rightarrow \varphi_2 \mid K_a\varphi$$

*where $e \in E$ e $a \in \mathcal{A}$.*

## 3.2 Semantics

Regarding the semantics, the definitions of frame and satisfaction are the same as in standard multi-agent epistemic logics. However, three restrictions were added to the valuation function, which we found necessary for the soundness proof developed later on.

**Definition 7** *A frame of Dolev-Yao multi-agent epistemic logic is a tuple $\mathcal{F} = (W, \sim_a)$. Where $W$ is a non-empty set of states and $\sim_a$ is a reflexive, transitive and symmetric binary relation over $W$, for each $a \in \mathcal{A}$.*

**Definition 8** *A Dolev-Yao multi-agent epistemic logic model is $\mathcal{M} = (\mathcal{F}, V)$. Where $\mathcal{F}$ is a Dolev-Yao multi-agent epistemic frame and $V$ is a valuation function $V : E \to 2^W$ satisfying the following conditions:*

1. *$V(m) \cap V(k) \subseteq V(\{m\}_k)$;*
2. *$V(\{m\}_k) \cap V(k) \subseteq V(m)$;*
3. *$V(m) \cap V(n) = V((m, n))$.*

**Definition 9** *Given a Dolev-Yao multi-agent epistemic logic model $\mathcal{M} = \langle S, \sim_a, V \rangle$ and $s \in S$. The notion of satisfaction $\mathcal{M}, s \models \varphi$ is defined by:*

- *$\mathcal{M}, s \models e$ if and only if $s \in V(e)$;*
- *$\mathcal{M}, s \models \neg\varphi$ if and only if $\mathcal{M}, s \not\models \varphi$;*
- *$\mathcal{M}, s \models \varphi \wedge \psi$ if and only if $\mathcal{M}, s \models \varphi$ and $\mathcal{M}, s \models \psi$;*
- *$\mathcal{M}, s \models \varphi \to \psi$ if and only if $\mathcal{M}, s \models \varphi$ then $\mathcal{M}, s \models \psi$;*
- *$\mathcal{M}, s \models K_a\varphi$ if and only if for all $s' \in S$, if $s \sim_a s'$ then $\mathcal{M}, s' \models \varphi$.*

## 3.3 Axiomatization

In the axiomatization, we have the same axioms and inference rules as those from multi-agent epistemic logic. We have also added the three last axioms of the list below:

1. All instantiations of propositional tautologies

2. $K_a(\varphi \to \psi) \to (K_a\varphi \to K_a\psi)$,

3. $K_a\varphi \to \varphi$,

4. $K_a\varphi \to K_aK_a\varphi \quad (+$ *introspection*$)$,

5. $\neg K_a\varphi \to K_a\neg K_a\varphi \quad (-$ *introspection*$)$,

6. $m \wedge k \to \{m\}_k \quad ($*encryption*$)$,

7. $\{m\}_k \wedge k \to m \quad ($*decryption*$)$,

8. $m \wedge n \leftrightarrow (m, n) \quad ($*pair composition & decomposition*$)$.

**Inference Rules**

M.P. $\varphi, \varphi \to \psi/\psi$ U.G. $\varphi/K_a\varphi$ U.B. $\varphi/\sigma\varphi$
where $\sigma$ is a map uniformly substituting formulas for propositional variables.

**Lemma 1** *The following formulas are theorems of* $\mathbf{S5_{DY}}$*:*

*1.* $K_am \wedge K_ak \to K_a\{m\}_k$,

*2.* $K_a\{m\}_k \wedge K_ak \to K_am$,

*3.* $K_am \wedge K_an \leftrightarrow K_a(m, n)$.

**Proof:** This proof is straightforward from axioms 6, 7, 8, axiom 2 and inference rule U.G. and the the fact that the $K_a$ distribuit over the conjunction ($\vdash K_a(\varphi \wedge \psi) \leftrightarrow (K_a\varphi \wedge K_a\psi)$).

$\triangle$

**Theorem 1** $\mathbf{S5_{DY}}$ *is sound and complete w.r.t. the class of* $\mathbf{S5_{DY}}$ *models.*

**Proof:** *The soundness and completeness proof are in section 3.4 and section 3.5 respectively.*

$\triangle$

## 3.4 Soundness

Axioms 1, 2, 3, 4, 5 and inference rules are standard in multi-agent epistemic logic, so we only prove the soundness of axioms that we added in system, 6, 7, 8.

**Lemma 2** *The following axioms are sound.*

1. $\Vdash m \wedge k \to \{m\}_k$ *(encryption)*

2. $\Vdash \{m\}_k \wedge k \to m$ *(decryption)*

3. $\Vdash m \wedge n \leftrightarrow (m, n)$ *(pair composition & decomposition)*

**Proof:**

1. *Suppose we have a Dolev-Yao multi-agent epistemic model $M$ and a state $w$ s.t. $M, w \Vdash m \wedge k$. Then we have that $M, w \Vdash m$ and $M, w \Vdash k$. But this is if and only if $w \in V(m)$ and $w \in V(k)$. By condition 1 of definition 8 we have that $w \in V(\{m\}_k)$ and thus $M, w \Vdash \{m\}_k$ and $M, w \Vdash m \wedge k \to \{m\}_k$.*
2. *(ii) and (iii) are analogous to (i), but we use condition 2 and 3 of definition 8 respectively.*

$\triangle$

## 3.5 Completeness

Here, we have the proof of completeness of our system.

**Definition 10** *Given a set of formulas $\Gamma$, we say:*

1. *$\Gamma$ is* **inconsistent** *if there exists a subset $\{\alpha_1, \ldots, \alpha_n\} \subseteq \Gamma$, such that $\vdash \neg(\alpha_1 \wedge \ldots \wedge \alpha_n)$. $\Gamma$ is* **consistent** *if it is not inconsistent;*

2. *$\Gamma$ is* **maximal** *if for any formula $\alpha$, either $\alpha \in \Gamma$, or $\neg\alpha \in \Gamma$;*

3. *$\Gamma$ is* **maximal consistent** *if it is both maximal and consistent. In this case, we say that $\Gamma$ is a* **MCS** *(Maximal Consistent Set).*

**Proposition 1 (MCS Properties)** *Let $\Gamma$ be a MCS. So:*

1. *for all $\phi$, either $\phi \in \Gamma$, or $\neg\phi \in \Gamma$, but not both;*

2. *$\Gamma$ is closed under Modus Ponens: if $\phi \in \Gamma$ and $\phi \to \psi$, then $\psi \in \Gamma$;*

3. *$\forall \phi, \psi, \phi \vee \psi \in \Gamma$ iff either $\phi \in \Gamma$, or $\psi \in \Gamma$;*

4. *$\forall \phi, \psi, \phi \wedge \psi \in \Gamma$ iff both $\phi \in \Gamma$, and $\psi \in \Gamma$;*

5. *all theorems of $S5_{DY} \subseteq \Gamma$;*

6. *if $m \in \Gamma$ and $k \in \Gamma$, then $\{m\}_k \in \Gamma$;*

7. *if* $\{m\}_k \in \Gamma$ *and* $k \in \Gamma$*, then* $m \in \Gamma$*;*

8. $(m, n) \in \Gamma$ *iff both* $m \in \Gamma$ *and* $n \in \Gamma$*.*

**Proof:**

- *1 by maximality, one of them must be in* $\Gamma$*;*
- *2 suppose* $\psi \notin \Gamma$*, so* $\{\phi, \phi \to \psi, \neg\psi\} \subseteq \Gamma$*, which is an absurd because* $\{\phi, \phi \to \psi, \neg\psi\}$ *is inconsistent. Therefore* $\psi \in \Gamma$*;*
- *3 and 4 analogous to 2;*
- *5 for all theorems* $\phi \in S5_{DY}$, $\vdash \phi$*. Suppose* $\neg\phi \in \Gamma$*, as* $\Gamma$ *is consistent,* $\Gamma \vdash \neg\phi$*, which is a contradiction. So* $\neg\phi \notin \Gamma$*. By maximality,* $\phi \in \Gamma$*. Therefore, all axioms of* $S5_{DY} \subseteq \Gamma$*.*
- *6, 7 and 8 follow straightforward from 2 and 5.*

$\triangle$

**Lemma 3 (Lindenbaum's Lemma)** *For any consistent set* $\Sigma$*, there is a set* $\Sigma^+$ *such that* $\Sigma \subseteq \Sigma^+$ *and* $\Sigma^+$ *is a MCS.*

**Proof:** *Let* $\phi_0, \phi_1, \phi_2, \ldots$ *be an enumeration of the formulas of our language. We define the set* $\Sigma^+$ *as the union of a chain of consistent sets as follows:*

- $\Sigma_0 = \Sigma$*;* $\Sigma_{i+1} = \begin{cases} \Sigma_i \cup \{\phi_{i+1}\}, & \textit{if it is consistent} \\ \Sigma_i \cup \{\neg\phi_{i+1}\}, & \textit{otherwise} \end{cases}$

  **Claim***:* $\Sigma_k$ *is consistent for all* $k$*. We prove that by induction on* $k$*.*

  **Base Case:** $\Sigma_0 = \Sigma$ *is consistent by hypothesis.*

  **Induction Hypothesis:** *suppose that* $\Sigma_k$ *is consistent.*

  *Now, we want to show that* $\Sigma_{k+1}$ *is also consistent. By construction, we have:*

$$\Sigma_{k+1} = \begin{cases} \Sigma_k \cup \{\phi_{k+1}\}, & \textit{if it is consistent} \\ \Sigma_k \cup \{\neg\phi_{k+1}\}, & \textit{otherwise} \end{cases}$$

  *By this construction, we have directly that* $\Sigma_{k+1}$ *is consistent too.*

  *Therefore,* $\Sigma_n$ *is consistent for all* $n$*.*

- $\Sigma^+ \cup_{i \geq 0} \Sigma_i$*. Now we have to prove that* $\Sigma^+$ *is a MCS.*

  $\Sigma^+$ *is consistent. Because otherwise some finite subset of the set* $\Sigma_i \subseteq \Sigma^+$ *would be inconsistent, but we just proved that all* $\Sigma_i$ *are consistent. So,* $\Sigma^+$ *is consistent. (by item 1 of definition 10)*

*$\Sigma^+$ is maximal. Because given any formula $\phi$, either $\phi \in \Sigma_k$, or $\neg\phi \in \Sigma_k$, for some $k$. Then $\Sigma_k \subseteq \Sigma^+$. So, $\Sigma^+$ is maximal. Therefore $\Sigma^+$ is a MCS.*

$\triangle$

**Definition 11 (Canonical Model)** *Let $\Lambda$ be a set of formulas. The canonical model $\mathfrak{M}$ over $\Lambda$ is the triple $(W^\Lambda, \sim_a^\Lambda, V^\Lambda)$, where:*

1. *$W^\Lambda$ is the set of all $\Lambda$-MCS;*
2. *$\sim_a^\Lambda$, the canonical relation, is the binary relation on $W^\Lambda$, for each agent $a \in \mathcal{A}$, defined by $w \sim_a^\Lambda v$ if for all formula $\psi$: $K_a\psi \in w \Rightarrow \psi \in v$.*
3. *$V^\Lambda$, the canonical valuation, defined as $V^\Lambda(m) = \{w \in W^\Lambda | m \in w\}$, where $m \in E$.*

*$\mathfrak{F} = (W^\Lambda, \sim_a^\Lambda)$ is called the canonical frame.*

**Lemma 4 (Existence Lemma)** *Let $\Gamma \in W^\Lambda$ be MCS such that $B_a\phi \in \Gamma$. Then exists a MCS $\Sigma$ such that $\{\varphi | K_a\varphi \in \Gamma\} \cup \{\phi\} \subseteq \Sigma$.*

**Proof:** *We first prove that $\Lambda^- = \{\varphi | K_a\varphi \in \Gamma\} \cup \{\phi\}$ is consistent. Suppose that $\Lambda^-$ is inconsistent. So, for the finite subset $\varphi_1, \cdots, \varphi_n$ we have that $\neg(\varphi_1 \wedge \cdots \wedge \varphi_n \wedge \phi)$ is a theorem.*
$\vdash_\Lambda \neg(\varphi_1 \wedge \cdots \wedge \varphi_n \wedge \phi)$
$\vdash_\Lambda \varphi_1 \wedge \cdots \wedge \varphi_n \to \neg\phi$, *propositional tautology*
$\vdash_\Lambda K_a(\varphi_1 \wedge \cdots \wedge \varphi_n \to \neg\phi)$, *Universal Generalization inference rule*
$\vdash_\Lambda K_a\varphi_1 \wedge \cdots \wedge K_a\varphi_n \to K_a\neg\phi$, *axiom ii*

*By hypothesis, $K_a\varphi_1 \in \Gamma, \ldots, K_a\varphi_n \in \Gamma$, so, by property 2 of proposition 1, $K_a\neg\phi \in \Gamma$, and also, $\neg B_a\phi \in \Gamma$, which is a contradiction. Thus, $\Lambda^-$ is consistent and by lemma 3, there exists a MCS extension $\Sigma$ that extends $\Lambda^-$.*

$\triangle$

**Lemma 5 (Truth Lemma)** *For any formula $\phi$, $\mathfrak{M}^\Lambda, w \Vdash \phi \Leftrightarrow \phi \in w$.*

**Proof:** *By induction on the length of $\phi$.*

**Base Case:** $\mathfrak{M}^\Lambda, w \Vdash e$, *iff* $w \in V^\Lambda(e)$ *iff* $e \in w$.

**Induction Hypothesis:** *It holds for* $|\phi| < n$: $\mathfrak{M}^\Lambda, w \Vdash \phi \Leftrightarrow \phi \in w$.

**Booleans:** *follows from the property 1 of Proposition 1.*

**Knowledge Operator:**

$\Rightarrow$ *Suppose* $\mathfrak{M}^\Lambda, w \Vdash K_a\phi$ *(1) and* $K_a\phi \notin w$*. So, by maximality, we have that* $B_a\neg\phi \in w$*. So, by Lemma 4 there exists a* $v$ *such that* $\{\varphi | K_a\varphi \in w\} \cup \neg\phi \subseteq v$ *(2); by definition 11* $w \sim_a^\Lambda v$*. From (1), for all* $w'$*, if* $w \sim_a^\Lambda w'$ *then* $\mathfrak{M}^\Lambda, w' \Vdash \phi$*, By the induction hypothesis,* $\phi \in w'$ *for all* $w'$ *and in particular* $\phi \in v$*, which is a contradiction with (2). Thus,* $K_a\phi \in w$

$\Leftarrow$ *Suppose* $K_a\phi \in w$ *and* $\mathfrak{M}^\Lambda, w \nVdash K_a\phi$*, then there exists* $v$ *such that* $w \sim_a^\Lambda v$ *and* $\mathfrak{M}^\Lambda, v \Vdash \neg\phi$*. But, by induction hypothesis* $\neg\phi \in v$*. By definition 11 if* $w \sim_a^\Lambda v$*, for all formula* $\psi$*:* $K_a\psi \in w \Rightarrow \psi \in v$*. So,* $\phi \in v$*, which is a contradiction. Thus,* $\mathfrak{M}^\Lambda, w \Vdash K_a\phi$

$\triangle$

**Lemma 6** *The canonical model relations* $\sim_a^\Lambda$ *are reflexive, transitive and symmetric.*

**Proof:** *This follows from the definition of* $\sim_a^\Lambda$ *and this proof can be found in epistemic and modal logic literature [7–9].*

$\triangle$

**Theorem 2** *The canonical model* $\mathfrak{M}^{S5_{DY}}$ *is a* $\mathbf{S5_{DY}}$ *models.*

**Proof:** *First we prove that* $\mathfrak{M}^{S5_{DY}}$ *satisfies the conditions 1, 2, and 3 of definition 8:*

- *Suppose we have* $w \in V(m) \cap V(k)$ *for a generic state* $w \in W^{S5_{DY}}$*. So, we have that* $w \in V(m)$ *and* $w \in V(k)$*. Also,* $\mathfrak{M}^{S5_{DY}}, w \Vdash m$ *and* $\mathfrak{M}^{S5_{DY}}, w \Vdash k$*, which entails* $\mathfrak{M}^{S5_{DY}}, w \Vdash m \wedge k$*. Using axiom 6,* $\mathfrak{M}^{S5_{DY}}, w \Vdash \{m\}_k$*. By the Truth Lemma (Lemma 5), we have that* $\{m\}_k \in w$*, that is,* $w \in V(\{m\}_k)$*. Thus* $V(m) \cap V(k) \subseteq V(\{m\}_k)$ *(condition 1 of definition 8).*
- *The proofs of conditions 2 and 3 of definition 8 are analogous to the above proof, but we use axioms 7 and 8, instead of b respectively.*

  *Together with Lemma , we are done.*

$\triangle$

**Theorem 3 - Completeness** *Let* $\Sigma$ *be a consistent set of formulas. Then,* $\Sigma$ *is satisfiable .*

**Proof:** *By the Existence Lemma (Lemma 4), there exists a MCS* $\Sigma^+$ *, such that* $\Sigma \subseteq \Sigma^+$ *and by the Truth Lemma (Lemma 5),* $\mathfrak{M}, \Sigma^+ \models \Sigma$*.*

$\triangle$

## 3.6 Examples

**Example 6** *We recover the first example (example 2) from the original paper by Dolev and Yao. The example that the agent $A$ send a message $M$ to agent $B$, but a malicious user $Z$ intercepts the encoded message and forwards it to $B$. We have three agents $A, B$ and $Z$. We assume that $K_{XY} = K_{YX}$ for all agents $X$ and $Y$.*

0. $KB_0 = \{K_A k_{AB},\ K_B k_{AB},\ K_B k_{BZ},\ K_Z k_{BZ},\ K_A m\}$ *initial knowledge*

$$\begin{tikzcd}
\text{1.} & K_a\{AB\} \arrow[d, "send_{AB}(\{m\}_{k_{AB}})"'] & L1.1 \\
& - - - \arrow[d, "Z\ intercepts"'] & \\
\text{1.} & KB_1 := KB_0 \cup K_Z\{m\}_{k_{AB}} \arrow[d, "send_{ZB}(\{m\}_{k_{AB}})"'] & \\
\text{2.} & KB_2 := KB_1 \cup K_B\{m\}_{k_{AB}} &
\end{tikzcd}$$

$$K_B m \qquad L1.2$$

$$\begin{tikzcd}
& K_B\{m\}_{k_{ZB}} \arrow[d, "send_{BZ}(\{m\}_{k_{BZ}})"'] & L1.1 \\
\text{3.} & KB_3 := KB_2 \cup K_Z\{m\}_{k_{BZ}} &
\end{tikzcd}$$

$$K_Z m \qquad L1.2$$

*The intruder $Z$ knows $M$*

**Example 7** *We recover the second example (example 3) from the original paper by Dolev and Yao. $A$ sends a message $Mk_{AB}$ to $B$, and $B$ replies to the intruder with the same message encrypted with the key that was encoded with the message. Again, we assume that $K_{XY} = K_{YX}$ for all agents $X$ and $Y$.*

0. $KB_0 = \{K_A k_{AB},\ K_B k_{AB},\ K_B k_{BZ},\ K_Z k_{BZ},\ K_A m\}$ *initial knowledge*

$$KB_0 \vdash K_A(k_{AB}, m) \qquad L1.3$$

$$\begin{tikzcd}
 & KB_0 \vdash K_A\{(k_{AB}, m)\}_{k_{AB}} \arrow[d, "send_{AB}(\{(k_{AB},m)\}_{k_{AB}})"'] & L1.1 \\
 & - - - \arrow[d, "Z\ intercepts"'] & \\
1. & KB_1 := KB_0 \cup K_Z\{(k_{AB}, m)\}_{k_{AB}} \arrow[d, "send_{ZB}(\{(k_{AB},m)\}_{k_{AB}})"'] & \\
2. & KB_2 := KB_1 \cup K_B\{(k_{AB}, m)\}_{k_{AB}} &
\end{tikzcd}$$

$$K_B(k_{AB}, m) \qquad L1.2$$

$$K_B m \qquad L1.3$$

$$\begin{tikzcd}
 & K_B\{(k_{AB}, m)\}_{k_{AB}} \arrow[d, "send_{BZ}(\{(k_{AB},m)\}_{k_{AB}})"'] & L1.1 \\
3. & KB_3 := KB_2 \cup K_Z\{(k_{AB}, m)\}_{k_{AB}} &
\end{tikzcd}$$

$$KB_3 \nvdash K_Z m$$

*Intruder $Z$ does not know the message $M$*

**Example 8** *We recover the third example (example 4) from the original paper by Dolev and Yao. $A$ sends a message $(\{M\}_{k_{AB}}, k_{AB})$ to $B$ and $B$ replies, but the intruder intercepts the message and starts communicating directly with $A$. Again, we assume that $K_{XY} = K_{YX}$ for all agents $X$ and $Y$.*

0. $KB_0 = \{K_A k_{AB},\ K_A k_{AZ},\ K_B k_{AB},\ K_Z k_{AZ},\ K_A m\}$ *initial knowledge*

$$KB_0 \vdash K_A\{m\}_{k_{AB}} \qquad L1.1$$

$$KB_0 \vdash K_A(k_{AB}, \{m\}_{k_{AB}}) \qquad L1.3$$

$$\begin{tikzcd}
 & KB_0 \vdash K_A\{(k_{AB}, \{m\}_{k_{AB}})\}_{k_{AB}} \arrow[d, "send_{AB}(\{(k_{AB},\{m\}_{k_{AB}})\}_{k_{AB}})"'] & L1.1 \\
1. & KB_1 := KB_0 \cup K_B\{(k_{AB}, \{m\}_{k_{AB}})\}_{k_{AB}} &
\end{tikzcd}$$

$$K_B(k_{AB}, \{m\}_{k_{AB}}) \qquad L1.2$$

$$K_B\{m\}_{k_{AB}} \qquad L1.3$$

$$K_Bm \qquad L1.2$$

$$\begin{tikzcd} K_B\{(k_{AB},\{m\}_{k_{AB}})\}_{k_{AB}} \arrow[d, "send_{BA}(\{(k_{AB},\{m\}_{k_{AB}})\}_{k_{AB}})"'] \\ - - - \arrow[d, "Z\ intecepts"'] \\ KB_2 := KB_1 \cup K_Z\{(k_{AB},\{m\}_{k_{AB}})\}_{k_{AB}} \end{tikzcd} \qquad L1.1$$

2.

$$K_Z\{m'\}_{k_{AB}} = \{(k_{AB},\{m\}_{k_{AB}})\}_{k_{AB}}$$

$$\begin{tikzcd} K_Z\{(k_{AZ},\{m'\}_{k_{AB}})\}_{k_{AZ}} \arrow[d, "send_{ZA}(\{(k_{AZ},\{m'\}_{k_{AB}})\}_{k_{AZ}})"'] \\ KB_3 := KB_2 \cup K_A\{(k_{AZ},\{m'\}_{k_{AB}})\}_{k_{AZ}} \end{tikzcd} \qquad L1.1$$

3.

$$K_A(k_{AZ},\{m'\}_{k_{AB}}) \qquad L1.2$$

$$K_A\{m'\}_{k_{AB}} \qquad L1.3$$

$$K_Am' \qquad L1.2$$

$$\begin{tikzcd} K_A\{(k_{AZ},\{m'\}_{k_{AZ}})\}_{k_{AZ}} \arrow[d, "send_{AZ}(\{(k_{AZ},\{m'\}_{k_{AZ}})\}_{k_{AZ}})"'] \\ KB_4 := KB_3 \cup K_Z\{(k_{AZ},\{m'\}_{k_{AZ}})\}_{k_{AZ}} \end{tikzcd} \qquad L1.1$$

4.

$$K_Z(k_{AZ},\{m'\}_{k_{AZ}}) \qquad L1.2$$

$$K_Z\{m'\}_{k_{AZ}} \qquad L1.3$$

$$K_Zm' \qquad L1.2$$

$$K_Z(k_{AB},\{m\}_{k_{AB}})$$

$$K_Z\{m\}_{k_{AB}} \qquad L1.3$$

$$\begin{tikzcd} K_Z\{(k_{AZ},\{m\}_{k_{AB}})\}_{k_{AZ}} \qquad L1.1 \arrow[d, "send_{ZA}(\{(k_{AZ},\{m\}_{k_{AB}})\}_{k_{AZ}})"'] \\ 5. \qquad KB_5 := KB_4 \cup K_A\{(k_{AZ},\{m\}_{k_{AB}})\}_{k_{AZ}} \end{tikzcd}$$

$$K_A(k_{AZ},\{m\}_{k_{AB}}) \qquad L1.2$$

$$K_A\{m\}_{k_{AB}} \qquad L1.3$$

$$K_Am \qquad L1.2$$

$$\begin{tikzcd} K_A\{(k_{AZ},\{m\}_{k_{AZ}})\}_{k_{AZ}} \qquad L1.1 \arrow[d, "send_{AZ}(\{(k_{AZ},\{m\}_{k_{AZ}})\}_{k_{AZ}})"'] \\ 6. \qquad KB_6 := KB_5 \cup K_Z\{(k_{AZ},\{m\}_{k_{AZ}})\}_{k_{AZ}} \end{tikzcd}$$

$$K_Z(k_{AZ},\{m\}_{k_{AZ}}) \qquad L1.2$$

$$K_Z\{m\}_{k_{AZ}} \qquad L1.3$$

$$K_Zm \qquad L1.2$$

*Intruder Z knows M*

# Chapter 4

# Strips Dolev-Yao

One interesting aspect of our work, is that a model checker was developed. We decided to use the Strips system, since it is a new approach to be explored in this particular line of work. It allows all the actions included in the language.

With this, we transcripted our logic into the Strips language. We managed to also include some additional actions in our system which are only at a meta-level. In this case, however, some restrictions were added to avoid some possible branches which are not interesting for our analysis.

## 4.1 Language

In this section we present our Strips Dolev-Yao language. It has propositions, that represent the agents' knowledge and the actions already happened. Also, the protocol actions that are formed by specific preconditions and effects. This actions express possibilities in communication. We have send, receive, intercept, encryption (enc), decryption (dec), pair composition (conc) and pair decomposition (dconc). In our system only "add" effects are modelled without negation in proposition, since there's no loss of knowledge.

**Propositions:**

- $knows(x, i)$
- $sent(i, x_1, x_2)$
- $intercept(x_3, i, x_1, x_2)$
- $received(i, x_2, x_1)$

**Actions:**

send$(i, x_1, x_2)$

Precondition: $knows(x_1, i)$
Effect: $\mathit{sent}(i, x_1, x_2) \ \wedge \ \neg\ \mathit{intercepted}(x_3, i, x_1, x_2) \ \wedge \ \neg\ \mathit{received}(i, x_2, x_1)$

receive$(i, x_2, x_1)$
Precondition: $\mathit{sent}(i, x_1, x_2) \ \wedge \ \neg\ \mathit{intercepted}(x_3, i, x_1, x_2)$
Effect: $+\ (knows(x_2, i) \ \wedge \ \mathit{received}(i, x_2, x_1))$
$-\ (\neg\ \mathit{received}(i, x_2, x_1))$

intercept$(x_3, i, x_1, x_2)$
Precondition: $\mathit{sent}(i, x_1, x_2) \ \wedge \ \neg\ \mathit{received}(i, x_2, x_1)$
Effect: $+\ (knows(x_3, i) \ \wedge \ \mathit{intercepted}(x_3, i, x_1, x_2))$
$-\ (\neg\ \mathit{intercepted}(x_3, i, x_1, x_2))$

enc$(x, m, k_x)$
Precondition: $knows(x, m) \ \wedge \ knows(x, k_x)$
Effect: $knows(x, \{m\}_{k_x})$

dec$(x, \{m\}_{k_x})$
Precondition: $knows(x, \{m\}_{k_x}) \ \wedge \ knows(x, k_x)$
Effect: $knows(x, m)$

conc$(x, m, n)$
Precondition: $knows(x, m) \ \wedge \ knows(x, n)$
Effect: $knows(x, (m, n))$

dconc$(x, (m, n))$
Precondition: $knows(x, (m, n))$
Effect: $knows(x, m) \ \wedge \ knows(x, n)$

## 4.2 Examples

In this section, we translate some of the previous examples into the Strips language.

**Example 9** *Returning to the first Dolev and Yao's example. Agent $A$ sends a message $M$ to agent $B$. However a intruder $Z$ and forwards it to $B$.*

$$\begin{tikzcd}[row sep=small]
\mathit{knows}(A,m),\mathit{knows}(A,k_{AB}),\mathit{knows}(B,k_{AB}),\mathit{knows}(B,k_{BZ}),\mathit{knows}(Z,k_{BZ}) \arrow[d, "\mathit{enc}(A,m,k_{AB})"] \\
+\mathit{knows}(A,\{m\}_{k_{AB}}) \arrow[d, "\mathit{send}(\{m\}_{k_{AB}},A,B)"] \\
+\mathit{sent}(\{m\}_{k_{AB}},A,B) \\
\\
+\neg\mathit{intercepted}(Z,\{m\}_{k_{AB}},A,B) \\
\\
+\neg\mathit{received}(\{m\}_{k_{AB}},B,A) \arrow[d, "\mathit{intercept}(Z,\{m\}_{k_{AB}},A,B)"] \\
+\mathit{knows}(Z,\{m\}_{k_{AB}}) \\
\\
+\mathit{intercepted}(Z,\{m\}_{k_{AB}},A,B) \\
\\
-\neg\mathit{intercepted}(Z,\{m\}_{k_{AB}},A,B) \arrow[d, "\mathit{send}(\{m\}_{k_{AB}},Z,B)"] \\
+\mathit{sent}(\{m\}_{k_{AB}},Z,B) \\
\\
+\neg\mathit{intercepted}(A,\{m\}_{k_{AB}},Z,B) \\
\\
+\neg\mathit{received}(\{m\}_{k_{AB}},B,Z) \arrow[d, "\mathit{receive}(\{m\}_{k_{AB}},B,Z)"] \\
+\mathit{knows}(B,\{m\}_{k_{AB}}) \\
\\
+\mathit{received}(\{m\}_{k_{AB}},B,Z) \\
\\
-\neg\mathit{received}(\{m\}_{k_{AB}},B,Z) \arrow[d, "\mathit{dec}(B,\{m\}_{k_{AB}})"] \\
+\mathit{knows}(B,m) \arrow[d, "\mathit{enc}(B,m,k_{BZ})"] \\
+\mathit{knows}(B,\{m\}_{k_{BZ}}) \arrow[d, "\mathit{send}(\{m\}_{k_{BZ}},B,Z)"] \\
+\mathit{sent}(\{m\}_{k_{BZ}},B,Z) \\
\\
+\neg\mathit{intercepted}(A,\{m\}_{k_{BZ}},B,Z)
\end{tikzcd}$$

$$\begin{tikzcd}
+\neg received(\{m\}_{k_{BZ}}, Z, B) \arrow[d, "receive(\{m\}_{k_{BZ}}{,}Z{,}B)"'] \\
+knows(Z, \{m\}_{k_{BZ}})
\end{tikzcd}$$

$$+received(\{m\}_{k_{BZ}}, Z, B)$$

$$\begin{tikzcd}
-\neg received(\{m\}_{k_{BZ}}, Z, B) \arrow[d, "dec(Z{,}\{m\}_{k_{BZ}})"'] \\
+knows(Z, m)
\end{tikzcd}$$

**Example 10** *Returning to the second Dolev and Yao's example. A sends a message* $Mk_{AB}$ *to B, and B replies to Z with the same message encrypted with the key that was encoded with the message.*

$$\begin{tikzcd}
knows(A, m), knows(A, k_{AB}), knows(B, k_{AB}), knows(B, k_{BZ}), knows(Z, k_{BZ}) \arrow[d, "conc(A{,}m{,}k_{AB})"'] \\
+knows(A, (m, k_{AB})) \arrow[d, "enc(A{,}(m{,}k_{AB}){,}k_{AB})"'] \\
+knows(A, \{(m, k_{AB})\}_{k_{AB}}) \arrow[d, "send(\{(m{,}k_{AB})\}_{k_{AB}}{,}A{,}B)"'] \\
+sent(\{(m, k_{AB})\}_{k_{AB}}, A, B)
\end{tikzcd}$$

$$+\neg intercepted(Z, \{(m, k_{AB})\}_{k_{AB}}, A, B)$$

$$\begin{tikzcd}
+\neg received(\{(m, k_{AB})\}_{k_{AB}}, B, A) \arrow[d, "intercept(Z{,}\{(m{,}k_{AB})\}_{k_{AB}}{,}A{,}B)"'] \\
+knows(Z, \{(m, k_{AB})\}_{k_{AB}})
\end{tikzcd}$$

$$+intercepted(Z, \{(m, k_{AB})\}_{k_{AB}}, A, B)$$

$$\begin{tikzcd}
-\neg intercepted(Z, \{(m, k_{AB})\}_{k_{AB}}, A, B) \arrow[d, "send(\{(m{,}k_{AB})\}_{k_{AB}}{,}Z{,}B)"'] \\
+sent(\{((m, k_{AB})\}_{k_{AB}}, Z, B)
\end{tikzcd}$$

$$+\neg intercepted(A, \{(m, k_{AB})\}_{k_{AB}}, Z, B)$$

$$\begin{tikzcd}
+\neg received(\{(m,k_{AB})\}_{k_{AB}},B,Z) \arrow[d, "receive(\{(m,k_{AB})\}_{k_{AB}},B,Z)"'] \\
+knows(B,\{(m,k_{AB})\}_{k_{AB}})
\end{tikzcd}$$

$$+received(\{(m,k_{AB})\}_{k_{AB}},B,Z)$$

$$\begin{tikzcd}
-\neg received(\{(m,k_{AB})\}_{k_{AB}},B,Z) \arrow[d, "dec(B,\{(m,k_{AB})\}_{k_{AB}})"'] \\
+knows(B,(m,k_{AB}) \arrow[d, "dconc(B,(m,k_{AB}))"'] \\
+knows(B,m))
\end{tikzcd}$$

$$\begin{tikzcd}
+knows(B,K_{AB}) \arrow[d, "conc(B,m,k_{AB})"'] \\
+knows(B,(m,k_{AB})) \arrow[d, "enc(B,(m,k_{AB}),k_{AB})"'] \\
+knows(B,\{(m,k_{AB})\}_{k_{AB}}) \arrow[d, "send(\{(m,k_{AB})\}_{k_{AB}},B,Z)"'] \\
+sent(\{(m,k_{AB})\}_{k_{AB}},B,Z)
\end{tikzcd}$$

$$+\neg intercepted(A,\{(m,k_{AB})\}_{k_{AB}},B,Z)$$

$$\begin{tikzcd}
+\neg received(\{(m,k_{AB})\}_{k_{AB}},Z,B) \arrow[d, "receive(\{(m,k_{AB})\}_{k_{AB}},Z,B)"'] \\
+knows(Z,\{(m,k_{AB})\}_{k_{AB}})
\end{tikzcd}$$

$$+received(\{(m,k_{AB})\}_{k_{AB}},Z,B)$$

$$-\neg received(\{(m,k_{AB})\}_{k_{AB}},Z,B)$$

**Example 11** *Returning to the third Dolev and Yao's example. A sends a message* $(\{M\}_{k_{AB}},k_{AB})$ *to B and B replies, but Z intercepts the message and starts communicating directly with A.*

$$\begin{tikzcd}
knows(A,m), knows(A,k_{AB}), knows(A,k_{AZ}), knows(B,k_{AB}), knows(Z,k_{AZ}) \arrow[d, "{enc(A,m,k_{AB})}"'] \\
+knows(A,\{m\}_{k_{AB}}) \arrow[d, "{conc(A,\{m\}_{k_{AB}},k_{AB})}"'] \\
+knows(A,(\{m\}_{k_{AB}},k_{AB})) \arrow[d, "{enc(A,(\{m\}_{k_{AB}},k_{AB}),k_{AB})}"'] \\
+knows(A,\{(\{m\}_{k_{AB}},k_{AB})\}_{k_{AB}}) \arrow[d, "{send(\{(\{m\}_{k_{AB}},k_{AB})\}_{k_{AB}},A,B)}"'] \\
+sent(\{(\{m\}_{k_{AB}},k_{AB})\}_{k_{AB}},A,B)
\end{tikzcd}$$

$$+\neg intercepted(Z,\{(\{m\}_{k_{AB}},k_{AB})\}_{k_{AB}},A,B)$$

$$\begin{tikzcd}
+\neg received(\{(\{m\}_{k_{AB}},k_{AB})\}_{k_{AB}},B,A) \arrow[d, "{receive(\{(\{m\}_{k_{AB}},k_{AB})\}_{k_{AB}},B,Z)}"'] \\
+knows(B,\{(\{m\}_{k_{AB}},k_{AB})\}_{k_{AB}})
\end{tikzcd}$$

$$+received(\{(\{m\}_{k_{AB}},k_{AB})\}_{k_{AB}},B,A)$$

$$\begin{tikzcd}
-\neg received(\{(\{m\}_{k_{AB}},k_{AB})\}_{k_{AB}},B,A) \arrow[d, "{dec(B,\{\{m\}_{k_{AB}},k_{AB}\}_{k_{AB}})}"'] \\
+knows(B,(\{m\}_{k_{AB}},k_{AB})) \arrow[d, "{dconc(B,(\{m\}_{k_{AB}},k_{AB}))}"'] \\
+knows(B,\{m\}_{k_{AB}}))
\end{tikzcd}$$

$$\begin{tikzcd}
+knows(B,K_{AB}) \arrow[d, "{dec(B,\{\{m\}_{k_{AB}})}"'] \\
+knows(B,m) \arrow[d, "{enc(B,m,k_{AB})}"'] \\
+knows(B,\{m\}_{k_{AB}}) \arrow[d, "{conc(B,\{m\}_{k_{AB}},k_{AB})}"'] \\
+knows(A,(\{m\}_{k_{AB}},k_{AB})) \arrow[d, "{enc(B,(\{m\}_{k_{AB}},k_{AB}),k_{AB})}"'] \\
+knows(B,\{(\{m\}_{k_{AB}},k_{AB})\}_{k_{AB}}) \arrow[d, "{send(\{(\{m\}_{k_{AB}},k_{AB})\}_{k_{AB}},B,A)}"'] \\
+sent(\{(\{m\}_{k_{AB}},k_{AB})\}_{k_{AB}},B,A)
\end{tikzcd}$$

$$+\neg intercepted(Z,\{(\{m\}_{k_{AB}},k_{AB})\}_{k_{AB}},B,A)$$

$$\begin{tikzcd}
+\neg \mathit{received}(\{(\{m\}_{k_{AB}}, k_{AB})\}_{k_{AB}}, A, B) \arrow[d, "{\mathit{intercept}(Z,\{(\{m\}_{k_{AB}},k_{AB})\}_{k_{AB}}B,A)}"'] \\
+\mathit{knows}(Z, \{(\{m\}_{k_{AB}}, k_{AB})\}_{k_{AB}})
\end{tikzcd}$$

$$+\mathit{intercepted}(Z, \{(\{m\}_{k_{AB}}, k_{AB})\}_{k_{AB}}, B, A)$$

$$\begin{tikzcd}
-\neg \mathit{intercepted}(Z, \{(\{m\}_{k_{AB}}, k_{AB})\}_{k_{AB}}, B, A) \arrow[d, "{\mathit{conc}(Z,\{(\{m\}_{k_{AB}},k_{AB})\}_{k_{AB}},k_{AZ})}"'] \\
+\mathit{knows}(Z, (\{(\{m\}_{k_{AB}}, k_{AB})\}_{k_{AB}}, k_{AZ})) \arrow[d, "{\mathit{enc}(Z,(\{(\{m\}_{k_{AB}},k_{AZ})\}_{k_{AB}},k_{AZ}),k_{AZ})}"'] \\
+\mathit{knows}(Z, \{(\{(\{m\}_{k_{AB}}, k_{AB})\}_{k_{AB}}, k_{AZ})\}_{k_{AZ}}) \arrow[d, "{\mathit{send}(\{(\{(\{m\}_{k_{AB}},k_{AB})\}_{k_{AB}},k_{AZ})\}_{k_{AZ}},Z,A)}"'] \\
+\mathit{sent}(\{(\{(\{m\}_{k_{AB}}, k_{AB})\}_{k_{AB}}, k_{AZ})\}_{k_{AZ}}, Z, A)
\end{tikzcd}$$

$$+\neg \mathit{intercepted}(B, \{(\{(\{m\}_{k_{AB}}, k_{AB})\}_{k_{AB}}, k_{AZ})\}_{k_{AZ}}, Z, A)$$

$$\begin{tikzcd}
+\neg \mathit{received}(\{(\{(\{m\}_{k_{AB}}, k_{AB})\}_{k_{AB}}, k_{AZ})\}_{k_{AZ}}, A, Z) \arrow[d, "{\mathit{receive}(\{(\{(\{m\}_{k_{AB}},k_{AB})\}_{k_{AB}},k_{AZ})\}_{k_{AZ}},A,Z)}"'] \\
+\mathit{knows}(A, \{(\{(\{m\}_{k_{AB}}, k_{AB})\}_{k_{AB}}, k_{AZ})\}_{k_{AZ}})
\end{tikzcd}$$

$$+\mathit{received}(\{(\{(\{m\}_{k_{AB}}, k_{AB})\}_{k_{AB}}, k_{AZ})\}_{k_{AZ}}, A, Z)$$

$$\begin{tikzcd}
-\neg \mathit{received}(\{(\{(\{m\}_{k_{AB}}, k_{AB})\}_{k_{AB}}, k_{AZ})\}_{k_{AZ}}, A, Z) \arrow[d, "{\mathit{dec}(A,\{(\{(\{m\}_{k_{AB}},k_{AB})\}_{k_{AB}},k_{AZ})\}_{k_{AZ}}}"'] \\
+\mathit{knows}(A, (\{(\{m\}_{k_{AB}}, k_{AB})\}_{k_{AB}}, k_{AZ})) \arrow[d, "{\mathit{dconc}(A,(\{(\{m\}_{k_{AB}},k_{AB})\}_{k_{AB}},k_{AZ}))}"'] \\
+\mathit{knows}(A, \{(\{m\}_{k_{AB}}, k_{AB})\}_{k_{AB}}))
\end{tikzcd}$$

$$\begin{tikzcd}
+\mathit{knows}(A, K_{AZ}) \arrow[d, "{\mathit{dec}(A,\{(\{m\}_{k_{AB}},k_{AB})\}_{k_{AB}}}"'] \\
+\mathit{knows}(A, (\{m\}_{k_{AB}}, k_{AB})) \arrow[d, "{\mathit{enc}(A,(\{m\}_{k_{AB}},k_{AB}),k_{AZ})}"'] \\
+\mathit{knows}(A, \{(\{m\}_{k_{AB}}, k_{AB})\}_{k_{AZ}}) \arrow[d, "{\mathit{conc}(A,\{(\{m\}_{k_{AB}},k_{AB})\}_{k_{AZ}},k_{AZ})}"'] \\
+\mathit{knows}(A, (\{(\{m\}_{k_{AB}}, k_{AB})\}_{k_{AZ}}, k_{AZ})) \arrow[d, "{\mathit{enc}(A,(\{(\{m\}_{k_{AB}},k_{AB})\}_{k_{AZ}},k_{AZ}),k_{AZ})}"'] \\
+\mathit{knows}(A, \{(\{(\{m\}_{k_{AB}}, k_{AB})\}_{k_{AZ}}, k_{AZ})\}_{k_{AZ}}) \arrow[d, "{\mathit{send}(\{(\{(\{m\}_{k_{AB}},k_{AB})\}_{k_{AZ}},k_{AZ})\}_{k_{AZ}},A,Z)}"'] \\
+\mathit{sent}(\{(\{(\{m\}_{k_{AB}}, k_{AB})\}_{k_{AZ}}, k_{AZ})\}_{k_{AZ}}, A, Z)
\end{tikzcd}$$

$$+\neg intercepted(B, \{(\{(\{m\}_{k_{AB}}, k_{AB})\}_{k_{AZ}}, k_{AZ})\}_{k_{AZ}}, A, Z)$$

$$\begin{array}{c}
+\neg received(\{(\{(\{m\}_{k_{AB}}, k_{AB})\}_{k_{AZ}}, k_{AZ})\}_{k_{AZ}}, Z, A) \\
{\scriptstyle receive(\{(\{(\{m\}_{k_{AB}},k_{AB})\}_{k_{AZ}},k_{AZ})\}_{k_{AZ}},Z,A)}\Big\downarrow \\
+knows(Z, \{(\{(\{m\}_{k_{AB}}, k_{AB})\}_{k_{AZ}}, k_{AZ})\}_{k_{AZ}})
\end{array}$$

$$+received(\{(\{(\{m\}_{k_{AB}}, k_{AB})\}_{k_{AZ}}, k_{AZ})\}_{k_{AZ}}, Z, A)$$

$$\begin{array}{c}
-\neg received(\{(\{(\{m\}_{k_{AB}}, k_{AB})\}_{k_{AZ}}, k_{AZ})\}_{k_{AZ}}, Z, A) \\
{\scriptstyle dec(Z,\{(\{(\{m\}_{k_{AB}},k_{AB})\}_{k_{AZ}},k_{AZ})\}_{k_{AZ}}}\Big\downarrow \\
+knows(Z, (\{(\{m\}_{k_{AB}}, k_{AB})\}_{k_{AZ}}, k_{AZ})) \\
{\scriptstyle dconc(Z,(\{(\{m\}_{k_{AB}},k_{AB})\}_{k_{AZ}},k_{AZ}))}\Big\downarrow \\
+knows(Z, \{(\{m\}_{k_{AB}}, k_{AB})\}_{k_{AZ}})
\end{array}$$

$$\begin{array}{c}
+knows(Z, K_{AZ}) \\
{\scriptstyle dec(Z,\{(\{m\}_{k_{AB}},k_{AB})\}_{k_{AZ}})}\Big\downarrow \\
+knows(Z, (\{m\}_{k_{AB}}, k_{AB})) \\
{\scriptstyle dconc(Z,(\{m\}_{k_{AB}},k_{AB}))}\Big\downarrow \\
+knows(Z, \{m\}_{k_{AB}})
\end{array}$$

$$\begin{array}{c}
+knows(Z, K_{AB}) \\
{\scriptstyle conc(Z,\{m\}_{k_{AB}},k_{AZ})}\Big\downarrow \\
+knows(Z, (\{m\}_{k_{AB}}, k_{AZ})) \\
{\scriptstyle enc(Z,(\{m\}_{k_{AB}},k_{AZ}),k_{AZ})}\Big\downarrow \\
+knows(Z, \{(\{m\}_{k_{AB}}, k_{AZ})\}_{k_{AZ}}) \\
{\scriptstyle send(\{(\{m\}_{k_{AB}},k_{AZ})\}_{k_{AZ}},Z,A)}\Big\downarrow \\
+sent(\{(\{m\}_{k_{AB}}, k_{AZ})\}_{k_{AZ}}, Z, A)
\end{array}$$

$$+\neg intercepted(B, \{(\{m\}_{k_{AB}}, k_{AZ})\}_{k_{AZ}}, Z, A)$$

$$\begin{array}{c}
+\neg received(\{(\{m\}_{k_{AB}}, k_{AZ})\}_{k_{AZ}}, A, Z) \\
{\scriptstyle receive(\{(\{m\}_{k_{AB}},k_{AZ})\}_{k_{AZ}},A,Z)}\Big\downarrow \\
+knows(A, \{(\{m\}_{k_{AB}}, k_{AZ})\}_{k_{AZ}})
\end{array}$$

$$+received(\{(\{m\}_{k_{AB}}, k_{AZ})\}_{k_{AZ}}, A, Z)$$

$$\begin{tikzcd}
-\neg received(\{(\{m\}_{k_{AB}}, k_{AZ})\}_{k_{AZ}}, A, Z) \arrow[d, swap, "dec(A{,}\{(\{m\}_{k_{AB}}{,}k_{AZ})\}_{k_{AZ}})"] \\
+knows(A, (\{m\}_{k_{AB}}, k_{AZ})) \arrow[d, swap, "dconc(A{,}(\{m\}_{k_{AB}}{,}k_{AZ}))"] \\
+knows(A, \{m\}_{k_{AB}})
\end{tikzcd}$$

$$\begin{tikzcd}
+knows(A, K_{AZ}) \arrow[d, swap, "dec(A{,}\{m\}_{k_{AB}})"] \\
+knows(A, m) \arrow[d, swap, "enc(A{,}m{,}k_{AZ})"] \\
+knows(A, \{m\}_{k_{AZ}}) \arrow[d, swap, "conc(A{,}\{m\}_{k_{AZ}}{,}k_{AZ})"] \\
+knows(A, (\{m\}_{k_{AZ}}, k_{AZ})) \arrow[d, swap, "enc(A{,}(\{m\}_{k_{AZ}}{,}k_{AZ}){,}k_{AZ})"] \\
+knows(A, \{(\{m\}_{k_{AZ}}, k_{AZ})\}_{k_{AZ}}) \arrow[d, swap, "send(\{(\{m\}_{k_{AZ}}{,}k_{AZ})\}_{k_{AZ}}{,}A{,}Z)"] \\
+sent(\{(\{m\}_{k_{AZ}}, k_{AZ})\}_{k_{AZ}}, A, Z)
\end{tikzcd}$$

$$+\neg intercepted(B, \{(\{m\}_{k_{AZ}}, k_{AZ})\}_{k_{AZ}}, A, Z)$$

$$\begin{tikzcd}
+\neg received(\{(\{m\}_{k_{AZ}}, k_{AZ})\}_{k_{AZ}}, Z, A) \arrow[d, swap, "receive(\{(\{m\}_{k_{AZ}}{,}k_{AZ})\}_{k_{AZ}}{,}Z{,}A)"] \\
+knows(Z, \{(\{m\}_{k_{AZ}}, k_{AZ})\}_{k_{AZ}})
\end{tikzcd}$$

$$+received(\{(\{m\}_{k_{AZ}}, k_{AZ})\}_{k_{AZ}}, Z, A)$$

$$\begin{tikzcd}
-\neg received(\{(\{m\}_{k_{AZ}}, k_{AZ})\}_{k_{AZ}}, Z, A) \arrow[d, swap, "dec(Z{,}\{(\{m\}_{k_{AZ}}{,}k_{AZ})\}_{k_{AZ}})"] \\
+knows(Z, (\{m\}_{k_{AZ}}, k_{AZ})) \arrow[d, swap, "dconc(Z{,}(\{m\}_{k_{AZ}}{,}k_{AZ}))"] \\
+knows(Z, \{m\}_{k_{AZ}})
\end{tikzcd}$$

$$\begin{tikzcd}
+knows(Z, K_{AZ}) \arrow[d, swap, "dec(Z{,}\{m\}_{k_{AZ}})"] \\
+knows(Z, m)
\end{tikzcd}$$

# Chapter 5

# Planner

The language we decided to use to implement our Strips Dolev-Yao model checker was Python 2.

In this process we faced some obstacles, since standard Strips considers every possibility, whereas for our analysis only certain paths are really interesting. These other paths are acceptable if you only look at an action level, but they do not make sense during the protocol execution. For instance, it's not normal for an agent to send a message directly to the intruder in the first step.

And at the action construction level, other difficulties arised due to protocol variety. Since protocol restrictions have to be considered in the definitions of the actions, for each different protocol different conditions have to be a part of these actions' implementations.

## 5.1 Implementation

The code is divided into several parts. We begin with the main function. This fragment receives all information, classes and other functions, and joins all of the aspects of the work, returning the final result.

Then we have the predicates. These are a set of classes, with a class for each predicate existing in the system. They are a part of the effects and the preconditions of the actions.

And finally the actions themselves, which are another set of classes implementing all possible actions of the protocol. Since each protocol has different requirements, we also have to adjust certain conditions in the code.

Now, we show some parts of the code, with examples of a predicate and an action.

Listing 5.1: The implementation of predicate knows

```
class Knows(object):
        def __init__(self, x, info):
                self.x = x
                self.info = info

        def listar(self):
                if (type(self.info) is Key) or
                   (type(self.info) is Encoded) :
                        return ['knows', self.x,
                           self.info.listar()]
                else:
                        return ['knows', self.x,
                           self.info]
```

Listing 5.2: The implementation of action send

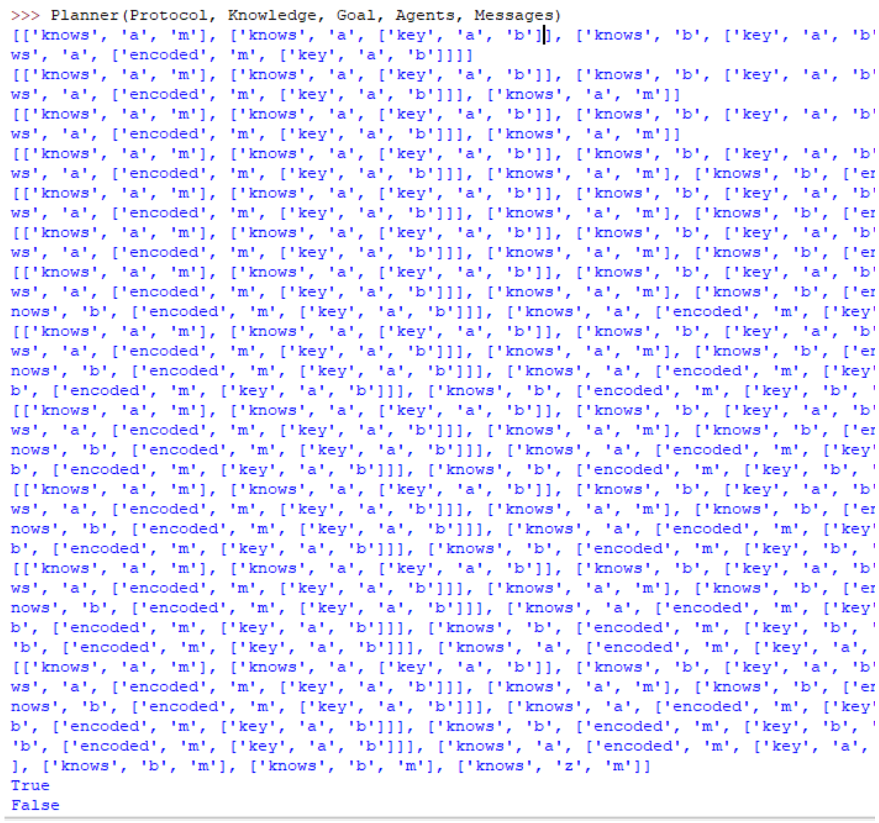
```
class Send(object):
 def precond(self):
  global agents
  global messages
  global knowledge
  global actions

  for agent1 in agents:
   for agent2 in agents:
    if agent1 != agent2:
     for agent3 in agents:
      for agent4 in agents:
       for message in messages:
        key = Key(agent3, agent4)
        enc = Encoded(message, key)
        if Knows(agent1, enc).listar() in knowledge and
           not(Sent(enc, agent1, agent2).listar() in
           actions):
         if ((agent2 == enc.key.x1) or (agent2 ==
            enc.key.x2)) or (Received(enc, agent1,
            agent2).listar() in actions):
          return [True, Sent(enc, agent1,
             agent2).listar()]
  return [False]

 def action(self):
  global agents
  global messages
  global knowledge
  global actions

  for agent1 in agents:
   for agent2 in agents:
    if agent1 != agent2:
     for agent3 in agents:
      for agent4 in agents:
       for message in messages:
        key = Key(agent3, agent4)
        enc = Encoded(message, key)
        if Knows(agent1, enc).listar() in knowledge:
         if ((agent2 == enc.key.x1) or (agent2 ==
            enc.key.x2)) or (Received(enc, agent1,
            agent2).listar() in actions):
          actions.append(Sent( enc, agent1,
             agent2).listar())
```

## 5.2 Execution

The results shown after an execution are composed of the evolution of the base knowledge, and booleans which are true when a goal was reached in the path, and false otherwise.

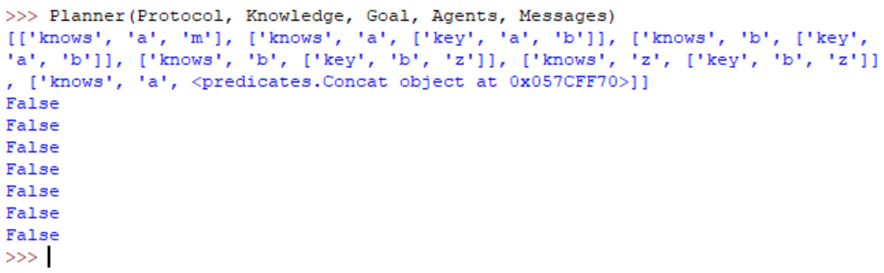
If a true value is returned, it means that it is possible to achieve a goal in one of the paths, and therefore the protocol is not secure.

Figures 5.1 and 5.2 show the results received after running the examples 2 and 3, respectively.



```
>>> Planner(Protocol, Knowledge, Goal, Agents, Messages)
[['knows', 'a', 'm'], ['knows', 'a', ['key', 'a', 'b']], ['knows', 'b', ['key', 'a', 'b'
ws', 'a', ['encoded', 'm', ['key', 'a', 'b']]]]
[['knows', 'a', 'm'], ['knows', 'a', ['key', 'a', 'b']], ['knows', 'b', ['key', 'a', 'b'
ws', 'a', ['encoded', 'm', ['key', 'a', 'b']]], ['knows', 'a', 'm']]
[['knows', 'a', 'm'], ['knows', 'a', ['key', 'a', 'b']], ['knows', 'b', ['key', 'a', 'b'
ws', 'a', ['encoded', 'm', ['key', 'a', 'b']]], ['knows', 'a', 'm']]
[['knows', 'a', 'm'], ['knows', 'a', ['key', 'a', 'b']], ['knows', 'b', ['key', 'a', 'b'
ws', 'a', ['encoded', 'm', ['key', 'a', 'b']]], ['knows', 'a', 'm'], ['knows', 'b', ['er
[['knows', 'a', 'm'], ['knows', 'a', ['key', 'a', 'b']], ['knows', 'b', ['key', 'a', 'b'
ws', 'a', ['encoded', 'm', ['key', 'a', 'b']]], ['knows', 'a', 'm'], ['knows', 'b', ['er
[['knows', 'a', 'm'], ['knows', 'a', ['key', 'a', 'b']], ['knows', 'b', ['key', 'a', 'b'
ws', 'a', ['encoded', 'm', ['key', 'a', 'b']]], ['knows', 'a', 'm'], ['knows', 'b', ['er
[['knows', 'a', 'm'], ['knows', 'a', ['key', 'a', 'b']], ['knows', 'b', ['key', 'a', 'b'
ws', 'a', ['encoded', 'm', ['key', 'a', 'b']]], ['knows', 'a', 'm'], ['knows', 'b', ['er
nows', 'b', ['encoded', 'm', ['key', 'a', 'b']]], ['knows', 'a', ['encoded', 'm', ['key'
[['knows', 'a', 'm'], ['knows', 'a', ['key', 'a', 'b']], ['knows', 'b', ['key', 'a', 'b'
ws', 'a', ['encoded', 'm', ['key', 'a', 'b']]], ['knows', 'a', 'm'], ['knows', 'b', ['er
nows', 'b', ['encoded', 'm', ['key', 'a', 'b']]], ['knows', 'a', ['encoded', 'm', ['key'
b', ['encoded', 'm', ['key', 'a', 'b']]], ['knows', 'b', ['encoded', 'm', ['key', 'b', '
[['knows', 'a', 'm'], ['knows', 'a', ['key', 'a', 'b']], ['knows', 'b', ['key', 'a', 'b'
ws', 'a', ['encoded', 'm', ['key', 'a', 'b']]], ['knows', 'a', 'm'], ['knows', 'b', ['er
nows', 'b', ['encoded', 'm', ['key', 'a', 'b']]], ['knows', 'a', ['encoded', 'm', ['key'
b', ['encoded', 'm', ['key', 'a', 'b']]], ['knows', 'b', ['encoded', 'm', ['key', 'b', '
[['knows', 'a', 'm'], ['knows', 'a', ['key', 'a', 'b']], ['knows', 'b', ['key', 'a', 'b'
ws', 'a', ['encoded', 'm', ['key', 'a', 'b']]], ['knows', 'a', 'm'], ['knows', 'b', ['er
nows', 'b', ['encoded', 'm', ['key', 'a', 'b']]], ['knows', 'a', ['encoded', 'm', ['key'
b', ['encoded', 'm', ['key', 'a', 'b']]], ['knows', 'b', ['encoded', 'm', ['key', 'b', '
[['knows', 'a', 'm'], ['knows', 'a', ['key', 'a', 'b']], ['knows', 'b', ['key', 'a', 'b'
ws', 'a', ['encoded', 'm', ['key', 'a', 'b']]], ['knows', 'a', 'm'], ['knows', 'b', ['er
nows', 'b', ['encoded', 'm', ['key', 'a', 'b']]], ['knows', 'a', ['encoded', 'm', ['key'
b', ['encoded', 'm', ['key', 'a', 'b']]], ['knows', 'b', ['encoded', 'm', ['key', 'b', '
'b', ['encoded', 'm', ['key', 'a', 'b']]], ['knows', 'a', ['encoded', 'm', ['key', 'a',
[['knows', 'a', 'm'], ['knows', 'a', ['key', 'a', 'b']], ['knows', 'b', ['key', 'a', 'b'
ws', 'a', ['encoded', 'm', ['key', 'a', 'b']]], ['knows', 'a', 'm'], ['knows', 'b', ['er
nows', 'b', ['encoded', 'm', ['key', 'a', 'b']]], ['knows', 'a', ['encoded', 'm', ['key'
b', ['encoded', 'm', ['key', 'a', 'b']]], ['knows', 'b', ['encoded', 'm', ['key', 'b', '
'b', ['encoded', 'm', ['key', 'a', 'b']]], ['knows', 'a', ['encoded', 'm', ['key', 'a',
], ['knows', 'b', 'm'], ['knows', 'b', 'm'], ['knows', 'z', 'm']]
True
False
```

Figure 5.1: Execution of the first Dolev and Yao example



```
>>> Planner(Protocol, Knowledge, Goal, Agents, Messages)
[['knows', 'a', 'm'], ['knows', 'a', ['key', 'a', 'b']], ['knows', 'b', ['key',
'a', 'b']], ['knows', 'b', ['key', 'b', 'z']], ['knows', 'z', ['key', 'b', 'z']]
, ['knows', 'a', <predicates.Concat object at 0x057CFF70>]]
False
False
False
False
False
False
False
>>>
```

Figure 5.2: Execution of the second Dolev and Yao example

# Chapter 6

# Final remarks and future works

In this work, we have defined our extension to the traditional multi-agent epistemic logic, and then transcripted our logic into the Strips language. We have done so, so that in the next step our model checker could be described and implemented.

The results obtained via our implementation were sound with respect to the logic definitions and rules. At first, our goal was to extend the logic language itself with actions, but since our focus lies on model checking, we have quickly realized that a Strips language would be more suitable for this kind of application.

## 6.1 Future works

We have outlined some other ways in which our "add" actions can be further developed in the future. In dynamic epistemic logics, it is easier to see the changes in kripke models that result from actions happening. It's also possible to see the changes graphically.

Another system to be considered is propositional dynamic logic. The main issue with this language is writing down the actual steps taken. While it's reasonably easy to implement, it's hard to keep track of the evolution on a step-by-step basis. Therefore, a possible future work extension is to find a way to make this kind of system more usable in practice.

Beyond the actions models, we outlined an extension based on Simon Kramer's article [18]. In his work, he makes a distinction between two types of knowledge, named *de re* and *de dicto*. The Knowledge *de dicto* is the standard knowledge of epistemic logic and represents *knows that it is the case that*. Also, the knowledge *de re* denotes that the agent *knows the content of* some information.

Another possibility to extend this work is to improve the model checker implementation, by allowing restriction inputs at a higher level language, and have it automatically translate it to serve as input for the algorithm.

# Bibliography

[1] DOLEV, D., YAO, A. C. “On the Security of Public Key Protocols”, *Information Theory, IEEE Transactions on*, v. 29, n. 2, pp. 198–208, 1983.

[2] BURROWS, M., ABADI, M., NEEDHAM, R. “A Logic of Authentication”, *ACM Transactions on Computer Systems*, v. 8, n. 1, pp. 18–36, 1990.

[3] ABADI, M., ROGAWAY, P. “Reconciling Two Views of Cryptography (The Computational Soundness of Formal Encryption)*”, *Journal of Cryptology*, v. 15, n. 2, pp. 103–127, 2002.

[4] ABADI, M., GORDON, A. D. “A Calculus for Cryptographic Protocols: The Spi Calculus”. In: *Proceedings of the 4th ACM Conference on Computer and Communications Security*, pp. 36–47. ACM, 1997.

[5] VON WRIGHT, G. H. *An Essay in Modal Logic.* Amsterdam: North-Holland Pub. Co., 1951.

[6] HINTIKKA, J. *Knowledge and Belief: An Introduction to the Logic of the Two Notions.* Ithaca, N. Y., Cornell University Press, 1962.

[7] VAN DITMARSCH, H., VAN DER HOEK, W., KOOI, B. *Dynamic Epistemic Logic.* Synthese Library Series, volume 337. The Netherland, Springer, 2008.

[8] FAGIN, R., HALPERN, J. Y., MOSES, Y. *Reasoning about knowledge.* Cambridge, Massachusetts, MIT Press, 1995.

[9] BLACKBURN, P., DE RIJKE, M., VENEMA, Y. *Modal Logic.* UK, Cambridge University Press, 2001.

[10] BLACKBURN, P., BENTHEM, J. F. A. K. V., WOLTER, F. *Handbook of Modal Logic, Volume 3 (Studies in Logic and Practical Reasoning).* New York, NY, USA, Elsevier Science Inc., 2006. ISBN: 0444516905.

[11] DIFFIE, W., HELLMAN, M. “New Directions in Cryptography”, *IEEE Transactions on Information Theory*, v. 22, n. 6, pp. 644–654, 1976.

[12] RIVEST, R. L., SHAMIR, A., ADLEMAN, L. “A Method for Obtaining Digital Signatures and Public-key Cryptosystems”, *Commun. ACM*, v. 21, pp. 120–126, 1978.

[13] LUGER, G. F. *Artificial Intelligence: Structures and Strategies for Complex Problem Solving.* 6th ed. USA, Addison-Wesley Publishing Company, 2008. ISBN: 0321545893, 9780321545893.

[14] RUSSELL, S. J., NORVIG, P. *Artificial intelligence - a modern approach, 2nd Edition.* Prentice Hall series in artificial intelligence. Prentice Hall, 2003.

[15] FIKES, R. E., NILSSON, N. J. “STRIPS: A New Approach to the Application of Theorem Proving to Problem Solving”. In: *Proceedings of the 2Nd International Joint Conference on Artificial Intelligence*, IJCAI’71, pp. 608–620, San Francisco, CA, USA, 1971. Morgan Kaufmann Publishers Inc.

[16] FIKES, R. E., NILLSON, N. J., COCOSCO, C. A. “A Review of ”STRIPS: A New Approach to the Application of Theorem Proving to Problem Solving by R.E. Fikes, N.J. Nillson, 1971””. 1998.

[17] BENEVIDES, M. R. F., FERNANDEZ, L. C. F., OLIVEIRA, A. C. C. M. “Epistemic Logic Based on Dolev-Yao Model”. In: *Anais - XXXVII Congresso da Sociedade Brasileira de Computação - ETC*, pp. 67–70. Sociedade Brasileira de Computação, 2017.

[18] KRAMER, S. “Cryptographic Protocol Logic: Satisfaction for (Timed) Dolev-Yao Cryptography”, *Journal of Logic and Algebraic Programming*, v. 77, n. 1–2, 2008. `http://dx.doi.org/10.1016/j.jlap.2008.05.005`.

# Appendix A

# Protocols

In this appendix, we present two protocols used as example in the original BAN Logic paper [2]. We used our logic to authenticate them too. First, we show the protocol, and then the analysis with our system.

## A.1 The Kerberos Protocol's analysis

### A.1.1 The protocol

$$\begin{array}{ll} Message\,1. & A \rightarrow S:\ A, B; \\ Message\,2. & S \rightarrow A:\ \{T_s,\ L,\ K_{ab},\ B,\ \{T_s,\ L,\ K_{ab},\ A\}_{K_{bs}}\}_{K_{as}}; \\ Message\,3. & A \rightarrow B:\ \{T_s,\ L,\ K_{ab},\ A\}_{K_{bs}},\ \{A,\ T_a\}_{K_{ab}}; \\ Message\,4. & B \rightarrow A:\ \{T_a + 1\}_{K_{ab}}. \end{array}$$

### A.1.2 The analysis

0. $$KB_0 = \{K_A A, K_A B, K_A k_{AS}, K_A T_A, K_B k_{BS},$$

$$K_B T_B, K_Z k_{zs}, K_Z T_Z, K_Z Z\} \qquad \textit{initial knowledge}$$

$$\Big\downarrow{\scriptstyle send_{SA}(\{(T_S, L, k_{AB}, B, \{(T_S, L, k_{AB}, A)\}_{k_{BS}})\}_{k_{AS}})}$$

1. $$KB_1 := KB_0 \cup K_A\{(T_S, L, k_{AB}, B, \{T_S, L, k_{AB}, A\}_{k_{BS}})\}_{k_{AS}}$$

$$K_A(T_S, L, k_{AB}, B, \{T_S, L, k_{AB}, A\}_{k_{BS}}) \qquad L1.2$$

$$K_A\{T_S, L, k_{AB}, A\}_{k_{BS}} \qquad L1.3$$

$$K_A(A, T_A) \qquad L1.3$$

$$K_A\{(A, T_A)\}_{k_{AB}} \qquad L1.1$$

$$\Big\downarrow{\scriptstyle send_{AB}(\{(T_S,L,k_{AB},A)\}_{k_{BS}},\{(A,T_A)\}_{k_{AB}})}$$

$$---$$

$$\Big\downarrow{\scriptstyle Z\ intercepts}$$

2. $KB_2 := KB_1 \cup K_Z(\{(T_S, L, k_{AB}, A)\}_{k_{BS}} \wedge \{(T_A, A)\}_{k_{AB}})$

**For the next step, we have three possibilities:**

*First, when the intruder $Z$ intercepts the message, he sends it to agent $B$ without modifications:*

2. $KB_2 := KB_1 \cup K_Z(\{(T_S, L, k_{AB}, A)\}_{k_{BS}} \wedge \{(T_A, A)\}_{k_{AB}})$

$$\Big\downarrow{\scriptstyle send_{ZB}(\{(T_S,L,k_{AB},A)\}_{k_{BS}},\{(A,T_A)\}_{k_{AB}})}$$

3. $KB_3 := KB_2 \cup K_B(\{(T_S, L, k_{AB}, A)\}_{k_{BS}} \wedge \{(T_A, A)\}_{k_{AB}})$

$$K_B(T_S, L, k_{AB}, A) \qquad L1.2$$

$$K_B(T_A, A) \qquad L1.2$$

$$K_B T_A \qquad L1.3$$

$$K_B\{T_A + 1\}_{k_{AB}} \qquad L1.1$$

$$\Big\downarrow{\scriptstyle send_{BZ}(\{T_A+1\}_{k_{AB}})}$$

4. $KB_4 := KB_3 \cup K_Z\{T_A + 1\}_{k_{AB}}$

$$KB_4 \not\vdash K_Z T_A + 1$$

Intruder $Z$ does not know $T_A + 1$

*Second, when the intruder $Z$ intercepts the message, he sends it to agent $B$ while changing the first part of the message. The intruder uses a part of communication that he could require to server $S$:*

$$\begin{tikzcd}[row sep=small] 2. \quad KB_2 := KB_1 \cup K_Z(\{(T_S, L, k_{AB}, A)\}_{k_{BS}} \wedge \{(T_A, A)\}_{k_{AB}}) \arrow[d, "send_{ZB}(\{(T_S,L',k_{ZB},Z)\}_{k_{BS}},\{(A,T_A)\}_{k_{AB}})"'] \\ 3. \quad KB_3 := KB_2 \cup K_B(\{(T_S, L', k_{ZB}, Z)\}_{k_{BS}} \wedge \{(T_A, A)\}_{k_{AB}}) \end{tikzcd}$$

$$K_B(T_S, L', k_{ZB}, Z) \qquad L1.2$$

$$KB_3 \nvdash K_B k_{AB}$$

$$KB_3 \nvdash K_B T_A \wedge A$$

The communications do not continue. Intruder $Z$ does not know the contents of these communications.

*Third, when the intruder $Z$ intercepts the message, he sends it to agent $B$ while changing the second part of the message. The intruder uses a shared key required to server $S$:*

$$\begin{tikzcd}[row sep=small] 2. \quad KB_2 := KB_1 \cup K_Z(\{(T_S, L, k_{AB}, A)\}_{k_{BS}} \wedge \{(T_A, A)\}_{k_{AB}}) \arrow[d, "send_{ZB}(\{(T_S,L,k_{AB},A)\}_{k_{BS}},\{(Z,T_Z)\}_{k_{ZB}})"'] \\ 3. \quad KB_3 := KB_2 \cup K_B(\{(T_S, L, k_{AB}, A)\}_{k_{BS}} \wedge \{(T_Z, Z)\}_{k_{ZB}}) \end{tikzcd}$$

$$K_B(T_S, L, k_{AB}, A) \qquad L1.2$$

$$KB_3 \nvdash K_B k_{ZB}$$

$$KB_3 \nvdash K_B T_Z \wedge Z$$

The communications do not continue. Intruder $Z$ does not know the contents of these communications.

## A.2 The Andrew Secure RPC Hanshake's analysis

### A.2.1 The protocol

$Message\,1.\quad A \rightarrow B:\ A, \{N_a\}_{K_{ab}}.$
$Message\,2.\quad B \rightarrow A:\ \{N_a+1, N_b\}_{k_{ab}}.$
$Message\,3.\quad A \rightarrow B:\ \{N_b+1\}_{k_{ab}}.$
$Message\,4.\quad B \rightarrow A:\ \{K'_{ab}, N'_b\}_{k_{ab}}.$

### A.2.2 The analysis

0. $KB_0 = \{K_A k_{AZ}, K_A k_{AB}, K_Z k_{AZ}, K_Z k_{BZ},$

$K_B k_{AB}, K_B k_{BZ}, K_A N_A\}$ *initial knowledge*

$$\begin{tikzcd} KB_0 \vdash K_A\{N_A\}_{k_{AB}} \arrow[d, "send_{AB}(\{N_A\}_{k_{AB}})"'] \\ KB_1 := KB_0 \cup K_B\{N_A\}_{k_{AB}} \end{tikzcd}$$ $L1.1$

1.

$K_B N_A$ $L1.1$

$$\begin{tikzcd} K_B N_B \arrow[d, "send_{BA}(\{(N_A+1, N_B)\}_{k_{AB}})"'] \\ KB_2 := KB_1 \cup K_A(\{(N_A+1, N_B)\}_{k_{AB}}) \end{tikzcd}$$

2.

$K_A(N_A+1, N_B)$ $L1.2$

$K_A N_B$ $ax.\ 8$

$$\begin{tikzcd} K_A\{N_B+1\}_{k_{AB}} \arrow[d, "send_{AB}(\{(N_B+1)\}_{k_{ab}})"'] \\ --- \arrow[d, "Z\ intercepts"'] \\ KB_3 := KB_2 \cup K_Z\{N_B+1\}_{k_{AB}} \arrow[d, "send_{ZB}(\{(N_B+1)\}_{k_{ab}})"'] \\ KB_4 := KB_3 \cup K_B\{N_B+1\}_{k_{AB}} \end{tikzcd}$$

$L1.1$ (for $K_A\{N_B+1\}_{k_{AB}}$)

3. $KB_3 := KB_2 \cup K_Z\{N_B+1\}_{k_{AB}}$

4. $KB_4 := KB_3 \cup K_B\{N_B+1\}_{k_{AB}}$

$$K_B N_B + 1$$ $L1.2$

$$K_B k'_{AB}$$

$$K_B N'_B$$

$$\begin{tikzcd} K_B\{(k'_{AB}, N'_B)\}_{k_{ZB}} \arrow[d, "send_{ZB}(\{(k'_{AB},N'_B)\}_{k_{ZB}})"'] \\ KB_5 := KB_4 \cup K_Z\{(k'_{AB}, N'_B)\}_{k_{ZB}} \end{tikzcd}$$

$L1.3, L1.1$ (for $K_B\{(k'_{AB}, N'_B)\}_{k_{ZB}}$)

5. $KB_5 := KB_4 \cup K_Z\{(k'_{AB}, N'_B)\}_{k_{ZB}}$

$$K_Z k'_{AB}$$ $L1.2, L1.3$

$$K_Z N'_B$$ $L1.2, L1.3$

$$\begin{tikzcd} K_Z\{(k'_{AB}, N'_B)\}_{k_{AZ}} \arrow[d, "send_{ZA}(\{(k'_{AB},N'_B)\}_{k_{AZ}})"'] \\ KB_7 := KB_6 \cup K_A\{(k'_{AB}, N'_B)\}_{k_{AZ}} \end{tikzcd}$$

$L1.3, L1.1$ (for $K_Z\{(k'_{AB}, N'_B)\}_{k_{AZ}}$)

6. $KB_7 := KB_6 \cup K_A\{(k'_{AB}, N'_B)\}_{k_{AZ}}$

$$K_A k'_{AB}$$ $L1.2, L1.3$

$$K_A N'_B$$ $L1.2, L1.3$

*Intruder Z can decrypt messages encrypted with* $k'_{AB}$.

# Appendix B

# Model Checking Source Code

Here we present the core of the implementation, and the main elements of examples 1 and 2. The complete code is available on Github. (`https://github.com/annaccmo/StripsDolevYao`)

Listing B.1: The implementation of planner

```
def Planner(Protocol, Knowledge, Goal, Agents, Messages):
         '''
         Planner(Protocol, Knowledge, Goal, Agents,
            messages)

          Protocol: actions possibles in protocol.
          Knowledge: initial knowledge.
          Goal:
          Agents: every agents in protocol, including
             the intruder.
          Messages: messages send in protocol.
         '''

         global protocol
         global knowledge
         global actions
         global goal
         global agents
         global messages

         actions = []
         goal = Goal
         agents = Agents
```

```
messages = Messages

l = len(Protocol)
i = 0
while i < l:
    j = i
    knowledge = Knowledge
    protocol = []
    while j < l:
        protocol.append(Protocol[j])
        j+=1
    j = 0
    while j < i:
        protocol.append(Protocol[j])
        j+=1

    stop = False

    while not(stop):
            stop = True
            for action in protocol:
                    a = action.precond()
                    if a[0]:
                            if not(a[1] in
                               actions):
                                    action.action()
                                    print
                                       knowledge
                                    stop = False

                    if goal in knowledge:
                           print 'True'
                           i = l
                           stop = True
                           break

    print 'False'
    i+=1
```

Listing B.2: First example of Dolev-Yao

```
#Example 1 of Dolev-Yao

from dyel_strips3 import *

#Agents

Agents = ['a','b','z']

#Mensage

Messages = ['m']

#Protocol

Protocol = [encode, decode, send, receive, intercept]

#Initial Knowledge

kab = Key('a', 'b').listar()
kbz = Key('b', 'z').listar()

Knowledge = [Knows('a', 'm').listar(), Knows('a',
   kab).listar(), Knows('b', kab).listar(), Knows('b',
   kbz).listar(), Knows('z', kbz).listar()]

#Goal

Goal = Knows('z', 'm').listar()
```

Listing B.3: Second example of Dolev-Yao

```
#Exemplo 2 do Dolev-Yao

from dyel_strips3 import *



#Agents
```

```
Agents = ['a','b','z']

#Mensage

Messages = ['m']

#Protocol

Protocol = [encode2, decode2, concatenate, deconcat,
   send2, receive2, intercept2]

#Initial Knowledge

kab = Key('a', 'b').listar()
kbz = Key('b', 'z').listar()

Knowledge = [Knows('a', 'm').listar(), Knows('a',
   kab).listar(), Knows('b', kab).listar(), Knows('b',
   kbz).listar(), Knows('z', kbz).listar()]

#Goal

Goal = Knows('z', 'm').listar()
```