Computational aspects of infinite automata simulation and closure system related issues

Karima Ennaoui

Numéro d'ordre : D. U :
E D S P I C :

Université Clermont-Auvergne
Ecole Doctorale
Sciences Pour l'Ingénieur de Clermont-Ferrand

Thèse
présentée par

Karima Ennaoui

pour obtenir le grade de

DOCTEUR D'UNIVERSITÉ

SPÉCIALITÉ : INFORMATIQUE

COMPUTATIONAL ASPECTS OF INFINITE AUTOMATA SIMULATION AND CLOSURE SYSTEM RELATED ISSUES

Soutenue publiquement le 28 Septembre 2017 devant le jury :

Prof. Arnaud Durand	Rapporteur et Examinateur
Prof. Marianne Huchard	Rapporteur et Examinateur
Dr. Karell Bertet	Examinateur
Dr. Arnaud Mary	Examinateur
Prof. Jean-Marc Petit	Examinateur
Prof. Lhouari Nourine	Directeur de thèse
Prof. Farouk Toumani	Co-encadrant de thèse

This thesis is dedicated to my family;
to my wonderful, inspiring and caring parents,
to Mohamed, my pillar of strength and sharp life-coach,
to my beautiful sister Siham, my true ultimate friend,
to Omar, my refreshing breeze on the hottest days.

Acknowledgment

My first thanks go to my advisors Pr. Lhouari Nourine and Pr. Farouk Toumani for their advice throughout the course of this experience and for giving me the opportunity to endeavor in various research topics.

Thank you to Prof Arnaud Durand and Prof. Marianne Huchard for taking the time to review my thesis report and providing useful and caring remarks and suggestions to further improve it. My thanks go as well to Pr. Jean-Marc Petit, Dr. Karell Bertet and Dr. Arnaud Mary for their valuable participation in my thesis committee. I owe many thanks as well to Khaled Maafa for his effort and participation in the course of this thesis, specially in the closure system's maximal extension issue.

I'm grateful to Beatrice Bourdieu, Sévrine Migignac, Susan Arbon-Leahy and Prof. Vincent Barra who have given me real support during my years of study in Clermont-ferrand.

Thanks to my friends Jing-yu Wang, Ibtissam Elhassouni, Fadoua Khanboubi, Sahar Bsaybes, Rajae Chahboune, Salah-eddine Elkhadiri, Bachar Rouly and Rabie Ajib for their support and patience during this roller coaster journey. My thanks are also due to the Baimiks for being my family in Clermont-ferrand.

Finally, thanks to everyone who has provided assistance or discussion in the subject matter and whose name was not listed.

Résumé

La thèse est consacrée à des problématiques d'algorithmique et de complexité sur deux sujets.

Le premier sujet s'intéresse à la composition des services web, une question cruciale dans l'intégration flexible et l'interopérabilité d'applications autonomes, hétérogènes et distribuées. En considèrent une description comportementale des services-web, ce problème a été réduit à la simulation d'un automate par le produit fermé d'un ensemble d'automates. La thèse étudie dans sa première partie la complexité de ce problème en considérant deux paramètres : le nombre des instances considéré de chaque service et la présence des états hybrides : état à la fois intermédiaire et final dans un automate.

Quand le nombre des instances de chaque services est borné, le problème est exptime-complet. Nous définissons également le cas particulier où le nombre des instances considéré en paralléle de chaque service est borné, et prouvons que le problème devient exptime-complet également.

Finalement, on étudie l'impact de la présence des états hybrides dans les services composants. Nous prouvons que la composition devient exptime-complet quand ce nombre est borné par 0, 1 et 2.

Le second sujet porte sur les systèmes de fermeture et s'intéresse au calcul de l'extension maximale d'un système de fermeture ainsi qu'à l'énumération des clefs candidates d'une base implicative.

L'extension maximale d'un système de fermeture est le plus grand système de fermeture (dans le sens d'inclusion) qui le contient et qui ne change pas la borne sup de ses éléments. Nous donnons un algorithme incrémental polynomial qui génère cette extension avec une relation binaire comme input et output.

Pour le problème d'énumération des clefs candidates, la notion de key-ideal est définie, en prouvant que leur énumération est équivalente à l'énumération des clefs candidates. Ensuite, on donne un algorithme qui permet de générer les key-ideal

minimaux en temps incrémental polynomial et les key-ideal non minimaux en délai polynomial.

Mots clé : service web, comportement de service web, composition de service web, simulation des automates, complexité, Exptime-complet, théorie d'ordre, système de fermeture, treillis, extension maximale, bases de données, base d'implications, relation binaire, Clé candidate, énumération, délai polynomial, délai incrémental

Abstract

In the first part of this thesis, we are interested in some complexity aspects of the web services composition problem. It is a crucial question in the interoperability and integration of autonomous web service applications. One of the related issues is deciding whether a web service can be composed out of an existing repository of web services. This question was reduced to the theoretical issue of automata simulation, when each service is described by its business protocol.

We define web services composition, its platform and the theoretical issue behind, namely state machines simulation. In the general case of the problem, it is a question of simulation between a finite state machine and an infinite one : the product closure machine. The complexity of the issue relies mainly on two parameters; the number of considered instances and the presence of the so-called hybrid states (states that are both intermediate and final) in the composing machine. When the number of instances of each web service is bounded, we prove that the problem is exptime-complete. This subproblem resembles the well-studied case where each web service can be called at most once in the composition process. Then, we define the particular case where the number of instances used simultaneously in the composition is bounded and prove it exptime-complete. Finally, we investigate the issue of hybrid states. We study its behavior and prove that if their number is bounded by either 0, 1 or 2 then the problem is exptime-complete.

The second part concerns closure systems and two related issues: Maximal extension of a closure system and Candidate keys enumeration. We start by establishing the correspondence between certain representations of closure systems in the literature: lattices, implicational bases and binary relations.

Then, we investigate the problem of a closure system's maximal extension. The complexity of this issue depends on the encoding of the closure system. The most interesting case is when the input is a lattice's join-irreducible and meet-irreducible sets families, equivalently a binary relation, then we give an incremental polynomial algorithm that computes the lattice's maximal extension.

Finally, the last chapter raises the question of candidate keys enumeration problem. It concerns generating all candidate keys of a given implicational base. We introduce the notion of key-ideal set: an ideal of a candidate key in an order defined through the implicational base. We demonstrate that candidate keys enumeration is as hard as the enumeration of a subset of key-ideal sets that we call minimal key-ideal sets. Knowing that the number of the latter can be significantly smaller than the number of candidate keys, we give an efficient key-ideal sets enumeration algorithm, that enumerates all non-minimal key-ideal sets in a polynomial delay from minimal key-ideal sets that are generated in incremental polynomial time.

Key words: web service, web service behaviour, web service composition, automata simulation, complexity, product closure of automata, asynchronous product, Exptime-complete, order theory, closure system, lattice, maximal extension, database, implicational base, candidate key, binary relation, enumeration, polynomial delay, incremental delay.

Contents

List of Figures

Part I

General Introduction

This thesis contains two independent parts. The first part is a complexity study of web services composition problem, while the second concerns the closure system structure and some related computational issues.

Web services is a computing solution that facilitates integration among autonomous and distributed applications over a network. It is based on an architecture built around three principal roles: a service provider, a service requester and a register [5].

Web services composition (WSC) raises from the situation where none of the existing services can satisfy a client request. This leads to questioning if a collaboration of the existing services can achieve the requested task [28]. This is what we call web services composition.

When the service is described by its behavior through Finite State Machines (FSM), namely Business protocol based composition, [28] proves that the composition issue is equivalent to simulation of a finite state machine by the infinite product of a state machine called **Product Closure State Machine**. In case of unbounded instances, this problem has been proved decidable with an Ackermanian function as upper bound in [28]. Their proof is based on Dickson lemma, and hence cannot be exploited to derive tighter upper bounds. An Expspace-hard lower bound is given by Lasota [35]. The source of complexity derived from the analysis of the algorithm given in [28] is related to the presence of the so-called hybrid states[1] in the components and loops in the target: if the target FSM is loop free, the WSC problem becomes NP-complete and when the components are hybrid state free the problem is proven Exptime.

In the first chapter of part I, we give the general context of the problem and its motivation by defining the concept of web services, web services modeling and web services composition. In the second chapter, we consider additional parameters related to bounded/unbounded web services composition. When the number of instances is bounded, we prove that the problem is Exptime-Complete [23]. We then consider the web service composition when the number of hybrid states is bounded by an integer k. We show that this problem is Exptime-Complete for $k = 0$, $k = 1$ [23] and $k = 2$.

In the second part, we begin by defining **closure systems** as a natural modeling structure that is related to many known and well studied structures: **lattices**, **implicational bases** and **formal contexts**. A closure system is a subset family $\mathbb{F}$ over a set $X \in \mathbb{F}$ and for every pair in $\mathbb{F}$, their intersection is in $\mathbb{F}$.

We investigate two enumeration issues related to closure systems. We begin

[1]Hybrid states of an FSM are final states with outgoing transitions and correspond to unbounded places in Petri net terminology.

with the problem of computing a closure system's maximal extension. It raises the question of computing the maximal closure system (inclusion wise) that contains the initial one, while preserving upper bounds. Adaricheva and Nation [2] have simplified the construction given in [3] and gave an exact formula for the largest extension of any closure system. It was however observed in [47] that the direct use of the characterization of the largest extension given in [2] leads to an exponential time algorithm for building the largest extension of a closure space. Hence, we give a polynomial incremental algorithm that computes the lattice's maximal extension.

Finally, we focus on the problem of candidate keys enumeration of an implicational base. This problem has several applications (such as formal context analysis [36], systems security [44]). A candidate key of an implicational base, also known as minimal generator in lattice or FCA terminologies, is a minimal subset of attributes that identifies uniquely every tuple of the relation. A candidate key of an implicational base can be computed in linear time. However, the best known algorithm that lists all candidate keys is given in [37] by Lucchesi and Osborn with a polynomial incremental complexity, but using exponential space. Saiedian and Spencer [41], introduce the notion of attribute graph of a set of functional dependencies and show that candidate keys are union of candidate keys of strongly connected components of the attribute graph.

We define a key-ideal set, an ideal of a candidate key in an order defined through Σ. We demonstrate that candidate keys enumeration is as hard as the enumeration of a subset of key-ideal sets that we call minimal key-ideal sets. Assuming that the number of the latter can be significantly smaller than the number of candidate keys, we give an efficient key-ideal sets enumeration algorithm, that enumerates all non-minimal key-ideal sets in a polynomial delay from minimal key-ideal sets that are enumerated in incremental polynomial time [22].

Part II

Web Services Composition

Chapter 1

Introduction

1.1 Context

Web Service [6] is an evolving computing paradigm that tends to become a technology of choice to facilitate integration among autonomous and distributed applications over a network. Its service oriented architecture is based on three main roles: service provider, service requester and service registry [33]. It provides a conceptual description of these roles and their interactions.

One main web services purpose is to deal with integration challenges such as service description, service discovery, service composition, service quality and others [34].

A formal description of a web service considers either one or both of two aspects: the profile and the behavior [14]. The first one refers to the functional aspect of a service: message exchange and data impact. The latter however deals with the external behavior of these services and describes possible scenarios of their interactions. A description of a web service's behavior is also called a service business protocol. Transition systems are a widely used format of service business protocols.

Through literature, different models have been used to represent web service business protocols. Transition systems formalism is widely adopted in this context to model *statefull* applications exposed as web services. A transition system's model is usually used to describe discrete systems, by considering a set of states and a set of transitions representing their interactions. It is an abstract model that covers other popular formalism such as Finite State Machines and Petri Nets. In this context, states represent the different phases that a service may go through while transitions represent "abstract" activities that a service can perform [16, 7, 8].

Web Services Composition (WSC) raises from the situation where none of the

existing services can satisfy a client request. The idea is to find out, computationally, if the target service can be composed out of the existing ones and eventually compute it. This automatic approach of composition simplifies the development of software by reusing existing components and offers capabilities to customize complex systems built on the fly [28].

Decomposition depends as well on the chosen model: profile or behavior or both. We focus here on business protocol based composition, described through finite state machines. Complexity analysis of this problem was first considered by Muscholl *et al.* [38], under the implicit assumption that at most one *instance* of each available service can be used in a composition, where it is shown Exptime-Complete. In fact, [38] shows that when business protocols are described by means of Finite State Machines, the composition issue can then be formalized as the problem of deciding whether there exists a simulation relation between the target protocol and the asynchronous product of the available ones.

This setting has been extended in [28] to the case where the number of instances that can be used in the synthesis is unbounded. Web Services Composition is formalized in this latter case as a simulation problem between a Finite State Machine and an Infinite State Machine, called Product Closure State Machine, that is able to compute the asynchronous product closure of a Finite State Machine.

Asynchronous product of Finite State Machines (and Product Closure State Machines) is a subclass of Basic Parallel Processes [18] , the class of communication free petri nets: every transition has at most one input place. Simulation of Finite State Machines by Basic Parallel Processes was proven Expspace-hard by Lasota [35] and 2-Exptime-hard in [19].

In case of unbounded instances, the WSC problem has been proved decidable with an Ackermanian function as upper bound in [28]. The proof of [28] is based on Dickson lemma, and hence cannot be exploited to derive tighter upper bounds. An Expspace-hard lower bound is given by Lasota [35]. The source of complexity derived from the analysis of the algorithm given in [28] is related to the presence of hybrid states in the components and loops in the target: if the target FSM is loop free, the WSC problem becomes NP-complete and when the components are hybrid state free the problem is proven Exptime.

1.2 Contributions

We consider additional parameters related to bounded/unbounded web services composition. We consider as inputs an FSM M (the target protocol) and a set of FSMs $\mathcal{R}$ (the protocols of the available services) and we investigate the complexity of test-

ing simulation between M and the shuffle closure of $\mathcal{R}$, represented as a PCSM [28]. More precisely, we study the complexity of the following problems:

1. $WSC(M, \mathcal{R})$: The problem of composing M using an unbounded number of instances of $\mathcal{R}$.

2. $BC(M, \mathcal{R}, k)$: The problem of composing M using at most k instances of each FSM in $\mathcal{R}$.

3. $UCHS(M, \mathcal{R}, k)$: The problem of composing M using an unbounded number of instances of $\mathcal{R}$, with the number of hybrid states in $\mathcal{R}$ is bounded by k.

Note that the upper bound complexity of $WSC(M, \mathcal{R})$ given in [28] is a an exptime tower driven from the Dickson lemma. We define in the second chapter certain sub-cases of WSC where we bound the number of hybrid states in $\mathcal{R}$ by $0, 1$ and 2 and prove that the problem can be solved in these cases in exponential time.

Table 1.1 displays known and new complexity results regarding the WSC problem.

M	Acyclic FSM	general FSM
$BC(M, \mathcal{R}, k)$	NP-complete [28]	Exptime-complete
$BC(M, \mathcal{R}, 1)$		Exptime-complete [38]
$WSC(M, \mathcal{R})$	NP-complete [28]	Decidable [28]
$UCHS(M, \mathcal{R}, 0)$		Exptime-complete [23]
$UCHS(M, \mathcal{R}, 1)$		Exptime-complete [23]
$UCHS(M, \mathcal{R}, 2)$		Exptime-complete

Table 1.1: Complexity of Web services composition problem.

1.3 Outline

The first part of this thesis is organized as follows. The second chapter introduces the concept of web services as a computing solution founded around the service oriented architecture proposed by W3C in order to challenge certain application integration issues. It goes on defining different Web service's modeling approaches and forms. Then, it defines web services composition issue, in its context, singling out business protocol based composition.

The third chapter begins with preliminaries section that recalls some basic definitions about state machines, simulation and order theory. Section 3.2 investigates

the problem of bounded web services composition and proves that it is Exptime-Complete. In section 3.3, we consider the web service composition when the number of hybrid states is bounded. We show that this problem is Exptime-Complete for $k = 0$, $k = 1$ and $k = 2$.

Chapter 2

Web Services

2.1 Web Services Formalism

2.1.1 Definition

Web Service [6] is an evolving computing paradigm that tends to become a technology of choice to facilitate inter-operation among autonomous and distributed applications over a network. It has changed the way software applications are designed and implemented, aiming for a more rapid, independent, standardized solutions that can be loosely composed in order to fulfill more complex tasks.

Web services propose a computation approach, based on a service oriented architecture. In this context, a service refers to a computational entity capable of running autonomously a task, platform-independent and inter-operable in the sense that it can be dynamically discovered and loosely composed with other services, in order to perform more complex tasks.

The service oriented architecture proposed by W3C does not impose any implementation restrictions. In fact, it describes a set of functional components and their interactions, in a conceptual level (Figure 2.1). It is based upon the interactions between three roles [33]:

- **Service provider:** From a business perspective, this is the entity (person or organization) that owns the service. From an architectural perspective, this is the platform that hosts access to the service.

- **Service requester:** From a business perspective, this is the business entity that requires certain functions to be satisfied. From an architectural perspective, this is the application that is looking for and invoking or initiating an in-

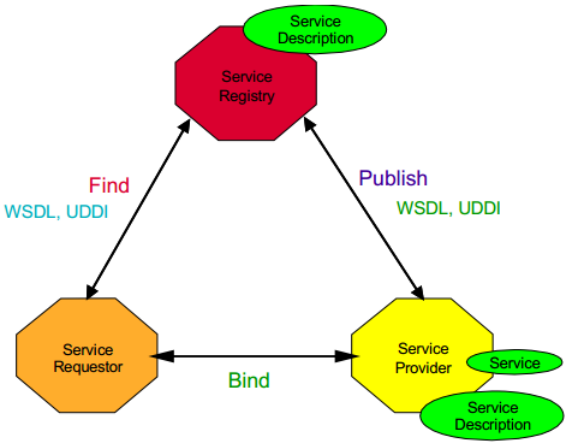


Figure 2.1: *Web Services architecture.*

teraction with a service. The service requester role can be played by a browser driven by a person or a program without a user interface, for example another Web service.

- **Service registry:** This is a searchable registry of service descriptions where service providers publish their service descriptions. Service requesters find services and obtain binding information (in the service descriptions) for services during development for static binding or during execution for dynamic binding.

However, the basic service oriented architecture does not necessarily address in a sufficient manner certain key issues faced while implementing or using service oriented computing solutions such as:

Service description must be formulated in a standardized, multi-levelled and precised manner in order to guarantee the interoperability and collaboration of different entities, increasing recursively the utility of the web service.

Service Discovery refers to the issue of locating easily available services, then helping in the process of selecting the one that meets the requested functional needs and eventually calling and executing it in an acceptable time. The typical way of addressing these issues is providing services descriptions stored in a standardized directories that can be browsed in order to identify the most relevant services by using dynamic binding techniques. During the discovery process, other question might be raised such as advertisement techniques that the service providers are constantly interested in developing and the service negotiation process where the provider and the requester (the client) enter in, in order to establish sort of an agreement.

Since web services are based on the idea of integrating multiple software components from diverse environments into one process, **Service security** can be an issue as well. Such a distributed computing activity needs a solution that considers, instead of applications, services and end-to-end processes, as well as authentication issues of users, their roles and authorities in internal and external functionalities and their privileges in common resources. Tracking users activities is also necessary in order to manage unauthorized operations.

An other related issue is **Service composition**. In fact, web services are mostly seen as a promising solution to deal with integration issues such as application description and composition. When none of the existing services can perform a certain task individually, two questions are raised in this context. First, there is the verification question: Can they do it together? If the answer is yes, the second one is: How can they do it? The answer to this starts in the early steps of the implementation process of the web service; In fact, the correct attitude while developing it is to predict its collaboration with other unspecified components. Hence, it should be self-described, in a standardized manner, as explained in the next W3C definition:

Web service is a software system designed to support inter-operable machine-to-machine interaction over a network. It has an interface described in a machine-processable format (specifically WSDL). Other systems interact with the Web service in a manner prescribed by its description using HTTP with an XML serialization in conjunction with other Web-related standards.

Another way of predicting these collaborations is to agree on both the semantics and the mechanics of the message exchange. The latter is established by a two level description of the web service in question. In order for it to be adaptable to any sort of agents, this description ought to be standardized and machine processable, defining basic details of the service's interface. On the semantic level, an agreement is established between the service provider and requester, framing the goals and possible outcomes of their cooperation. This contract can be implicit in certain cases or explicitly written in any chosen language.

Other related issues are **Services quality** which determines their usability and utility, **Service Interaction** regarding the infrastructures and tools that enable services interaction.This is by no means an exhaustive list.

2.1.2 Web Services Modeling

A formal description of a web service usually considers two aspects: the web service profile and the web service behavior. On one hand, the profile stands here for the functional aspect of a service. It deals with the message exchanges between its

operations and considers the impact of and on data during their execution.

The behavior, on the other hand, describes the interaction between services, by giving all acceptable scenarios of message exchanges and operation execution between them, i.e its external behavior. It is also known as the web service business protocol. It is *statefull* representation of the internal logic and private details of the services implementation. It specifies how an entity initiates a dialog or responds if invoked by other agents. This level of internal description is not mentioned in communication protocols; They are more focused on the external behavior of web services while interacting with each other. We give next an example illustrating the business protocol concept.

Example 1. *Consider a service that manages the access to a catalog. It allows to check the catalog (**CheckCatalog**) and choose an item (**SelectItem**). The first thing a client can do is checking the catalog. Then he can either leave or choose an item. Once an item is chosen, he can either go back to the catalog display and choose another one, or end the process. Obviously, in order to choose an item we must select the catalog first. In other words, a valid conversation would be : (CheckCatalog, SelectItem, CheckCatalog), but not (CheckCatalog, SelectItem, SelectItem). In figure 2.2, we describe the behavior (business protocol) of this service in an informal way.*

*The second service verifies the availability of an order. It starts with checking its availability (**CheckAvailability**). Then it is either confirmed (**ConfirmOrder**) or canceled (**CancelOrder**). The valid conversations in this service are: (CheckAvailability, ConfirmOrder) or (CheckAvailability, CancelOrder), but not (CheckAvailability). We give an informal description of this service in figure 2.3, the same way as for the previous service in figure 2.2.*

Through literature, different models have been used to represent web service business protocols. Transition systems formalism is widely adopted in this context to model *statefull* applications exposed as web services. A transition system's model is usually used to describe discrete systems, by considering a set of states and a set of transitions representing their interactions. It is an abstract model that covers other popular formalisms such as Finite State Machines and Petri Nets. In this context, states represent the different phases that a service may go through while transitions represent "abstract" activities that a service can perform [16, 7, 8]. For instance, we give in the next example a finite state machine representation of two web services.

Example 2. *We go back to the web services in example 1. We give in figure 2.4.a the finite state machine that describes the behavior of the web service managing the access to a catalog. This behavior is described informally in figure 2.2.*

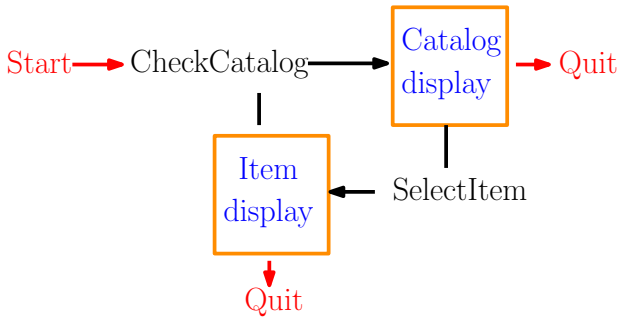


Figure 2.2: *An informal description of the behavior of a service managing a catalog*

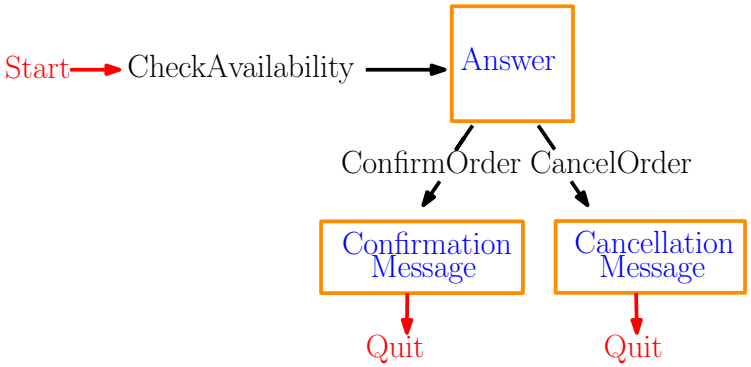
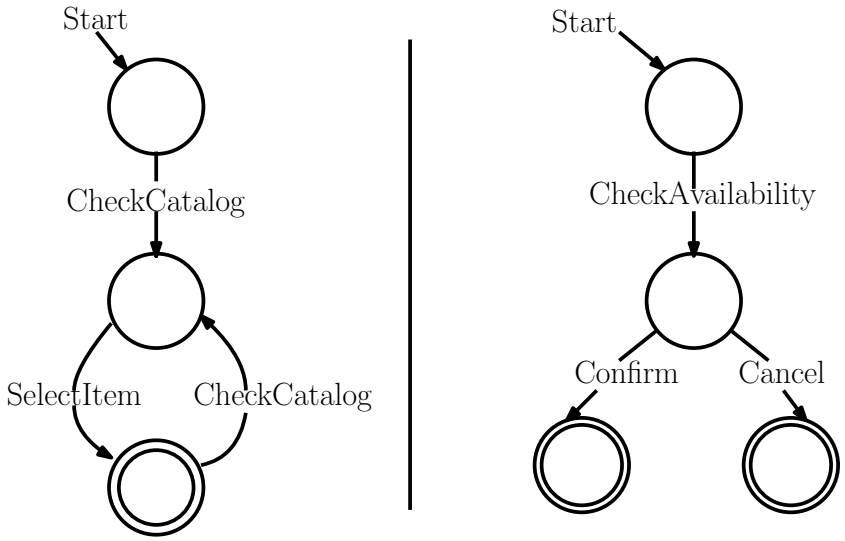


Figure 2.3: *An informal description of a verification service's behavior*



a. Service managing a catalog's access b. Service of checking an order's availability

Figure 2.4: *Finite state machines describing Services Business Protocols*

In figure 2.4.b, we give a finite state machine representation of the behavior of a service that verifies an order's availability (Figure 2.3).

The beginning of each service's execution process is indicated by an unlabeled entering arrow to a state. A state is a possible end of this process if it is double circled.

Needless to say that the interface along with its various descriptions must allow the service's functionalities to be used independently of the hardware, software platform or the programming language chosen. This allows and encourages Web Services-based applications to be loosely coupled, specially with new ideas aiming their construction as the current project.

2.2 Web Services Composition Problem

2.2.1 Definition

The issue of web services composition raises from the situation where none of the existing services can satisfy a client request. The idea is to find out, computationally,

if the target service can be composed out of the existing ones and eventually compute it. This automatic approach of composition simplifies the development of software by reusing existing components and offers capabilities to customize complex systems built on the fly [28].

There has been some work aiming a manual web services composition, based on a low level description models (Microsoft's BizTalk [1]). A human agent takes in charge the complex task of orchestrating the synthesis and the results are not always accurate.

The automatic Web Services Synthesis has been motivated by the continuous work done by the business world to develop a number of standards and formalize the specification of Web services, their flow composition and execution [43]. In addition to that, the Semantic Web community focuses on reasoning about web resources by explicitly declaring their preconditions and effects with terms precisely defined in ontologies [43].

The dimensions of the decomposition process depend on the description provided of the services and their conceptual level. A syntactic description provides a standardized machine processable representation of the allowed interactions between services. It focuses mainly on the way a service behaves and less about the aim of its operations. A semantic representation, on the other hand, is about the purpose and consequences of these operations. It is more flexible than the syntactic one, and can be presented in different shapes: explicit or implicit, machine processable or human.

Decomposition depends as well on the chosen model: profile or behavior or both. The profile is more about data management: consequences of the services operations on the database and the influence of the database's state on non deterministic decisions in the web service. The behavior is more about a contract between different entities, describing for each one of them the messages that they can receive or send to other services. It is an internal description from the web service's point of view, but external for each of the composing services considered a black box in this level of description.

There is a wide interest in computing web services composition, which motivated a lot of effort and progress. Several models have been considered

However, the difficulty of automatic web services composition is still an issue due to several reasons [42]. For instance, the bigger the services repository is, the better chances are to compose new ones, the more interesting this computing solution becomes. However, this makes the search, identification and selection process more difficult. Another example is the changing aspect of web services. They are updated on the fly, hence the composition needs to keep up with that at a runtime.

[1] http://www.microsoft.com/france/serveur-cloud/biztalk/

2.2.2 Business Protocol based Composition

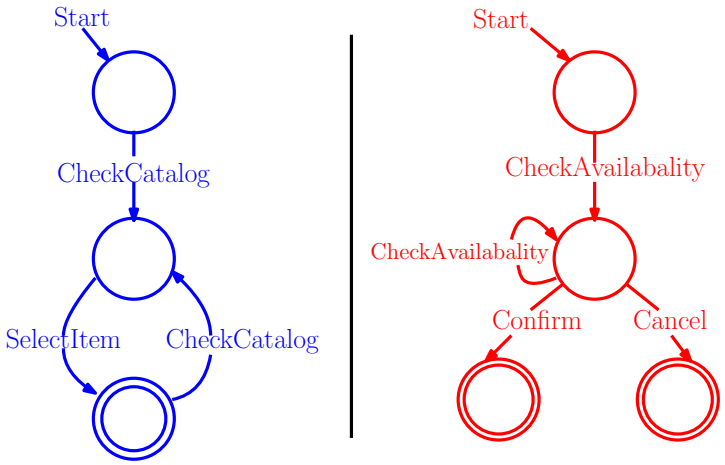
This section deals with composing web services described by their business protocol. The formalism considered is finite state machines, where each state indicates a phase of the service's execution, and a transition refers to an activity. An end-to-end process begins with the unique initial state and halts with a final one. This corresponds to the Roman model studied in [10]. [28] gives a generic definition of protocol synthesis problem using this formalism and reduces it to a simulation issue between transition systems. This will be explained in thorough details in the next chapter.

Needless to say that such description omits the database effect on the synthesis considered in the Colombo model [4][9]. Their work considers alongside with the behavioral aspect, the message exchange formalism and the activities consequences on the real world.

A business protocol presents every valid conversation made by an end-to-end execution of the corresponding web service. Business protocol based composition is interested in synthesizing every valid conversation C of the targeted service with a number of an end-to-end executions of the available services. A delegator is a finite state machine equivalent to the targeted service's business protocol, where each transition is assigned to a composing service. After the decision problem interrogating the synthesis possibility is positively answered, a delegator should be computed.

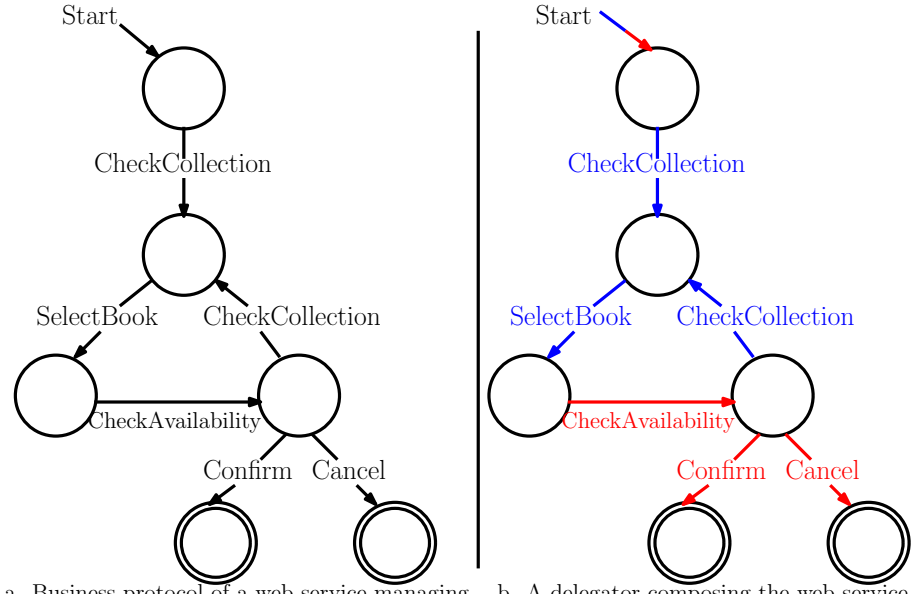
Example 3. *We want to compose a web service capable of managing long-distance renting from a library. It allows a member to check the library's collection (**CheckCollection**), choose a book (**SelectBook**), verify its availability (**CheckAvailability**) then either confirm the rent (**Confirm**), cancel it (**Cancel**) or go back to the library's collection (**CheckCollection**). Figure 2.6.a is a finite state machine describing its business protocol.*
We consider the two web services described in figure 2.5 as our repository in order to compose the targeted service.
Such composition is possible since we are able to construct, in figure 2.5.b, a delegator assigning each transition of the targeted service into an end-to-end process of either the service in 2.5.a managing access to a catalog (marked in blue) or the one in 2.5.b verifying the availability of an order (marked in red). The catalog here represents the library's collection and the order is a chosen book.

Each activity of a web service can be engaged in different conversations simultaneously. This means that, while composing, each web service can be instantiated any number of times. This number makes a huge difference in the problem's complexity. The simplest version is to allow the use of each service of the repository once. In [8], they consider the simplifying hypothesis that the number of instances involved of an



a. Service managing a catalog's access b. Service of checking an order's availabality

Figure 2.5: *Finite state machines describing the available web services*



a. Business protocol of a web service managing the renting service of a library b. A delegator composing the web service

Figure 2.6: *Finite state machine describing the targeted web service*

available protocol is bounded by a fixed constant k. Note that this k-bounded instance protocol synthesis problem can be reduced *w.l.o.g* to the simplest case where k=1 [28]. However, the most realistic assumption would be to take in consideration that each available protocol can be called as many times as possible. In this case, [28] proves that the problem of composition is decidable. This is done by reducing the decision problem of a web service composition by a repository, where each service can be used an unbounded number of time, to a problem of simulation between a finite state machine and an infinite state machine.

In the next chapter, we focus on this reduction as well as introduce other complexity parameters that can indicate the difficulty of the web services composition problem.

Chapter 3

Web Services Composition Complexity

3.1 Preliminaries

Service business protocols are formally described in this context as FSMs. We recall below the definition of such machines.

Definition 1. ***(Finite State Machine (FSM))***
*A State Machine (**SM**) M is a tuple $M = (\Sigma_M, Q_M, F_M, q_M^0, \delta_M)$, where: Σ_M is a finite alphabet, Q_M is a set of states, $\delta_M \subseteq Q_M \times \Sigma_M \times Q_M$ is a set of labeled transitions, $F_M \subseteq Q_M$ is a set of final states, and $q_M^0 \in Q_M$ is the initial state. If Q_M is finite then M is called a Finite State Machine (FSM).*

Moreover, a state $q \in Q_M$ is called: **accessible**, if there exists a path from the initial state to q; **co-accessible**, if there exists a path from q to a final state; **intermediate**, if $q \notin F_M$ and $\exists p_1, p_2 \in Q_M$, $s.t$ $(p_1, a, q) \in \delta_M$ and $(q, b, p_2) \in \delta_M$, we denote by $I(M)$ the set of intermediate states of M; **hybrid**, if $q \in F_M$, $q \neq q_M^0$ and there exist at least one transition $(q, b, p) \in \delta_M$, with $p \in Q_M$ and $b \in \Sigma$, the set of hybrid states is denoted $H(M)$ and **terminal**, if $q \in F_M$ and is not hybrid.

We consider here only FSMs where all states are both accessible and co-accessible. We define the **norm of a state** q as the finite length of the shortest path from q to a final state. **The norm of an FSM** M, noted *norm(M)*, is the maximal norm of its states.

k-Iterated Product Machine (k-IPM) and Product State Machine (PCSM)
We start by defining the asynchronous product and union operations on FSMs:

Definition 2. ***(Asynchronous product and Union of two FSMs)*** *Let $M = (\Sigma_M, Q_M, F_M, q_M^0, \delta_M)$ and $M' = (\Sigma_{M'}, Q_{M'}, F_{M'}, q_{M'}^0, \delta_{M'})$ be two state machines. We have :*

- *The* ***shuffle or asynchronous product*** *of M and M', denoted $M \times M'$, is a state machine $(\Sigma_M \cup \Sigma_{M'}, Q_M \times Q_{M'}, F_M \times F_{M'}, (q_M^0, q_{M'}^0), \lambda)$ where the transition function λ is defined as follows: $\lambda = \{((q, q'), a, (q_1, q_1')) : ((q, a, q_1) \in \delta_M$ and $q' = q_1')$ or $((q', a, q_1') \in \delta_{M'}$ and $q = q_1)\}$.*

  For $\{M_1, ..., M_k\}$ a set of state machines where $k \geq 3$, we define reciprocally $M_1 \times .... \times M_k$ as the state machine $((M_1 \times ... \times M_{k-1}) \times M_k)$.

- *The* ***union*** *of M and M', denoted $M \cup M'$, is the state machine $(\Sigma_M \cup \Sigma_{M'} \cup \{\epsilon\}$, $Q_M \cup Q_{M'} \cup \{q_0\}$, $F_M \cup F_{M'}$, q_0, $\delta_M \cup \delta_{M'} \cup \{(q_0, \epsilon, q_M^0), (q_0, \epsilon, q_{M'}^0)\})$.*

For a set of available FSMs $\mathcal{R} = \{M_1, ...M_m\}$, we consider a compact structure that abstracts all possible executions that can be produced using the components of $\mathcal{R}$. First, we begin by the simple case where each M_j can be used only once:

Definition 3. ***(Union of asynchronous products of FSMs set)*** *The asynchronous product of all the subsets elements of FSMs repository $\mathcal{R} = \{M_1....M_m\}$ is the FSM: $\odot(\mathcal{R}) = \bigcup_{\{i_1,...,i_j\} \subseteq \{1,...,m\}} (M_{i_1} \times ... \times M_{i_j})$.*

Second, we consider the case where the number of copies of each $M_j \in \mathcal{R}$ is bounded by an integer k:

Definition 4. ***(k-iterated product of FSMs set $\mathcal{R}$)*** *The* ***k-iterated product of*** *$\mathcal{R}$ is defined by $\mathcal{R}^{\otimes_k} = \mathcal{R}^{\otimes_{k-1}} \times \odot(\mathcal{R})$ with $\mathcal{R}^{\otimes_1} = \odot(\mathcal{R})$.*

Finally, we consider the general case where the number of instances of each $M_j \in \mathcal{R}$ is unbounded. This corresponds to the product closure of $\mathcal{R}$ [28]:

Definition 5. ***(Product closure of FSMs set)*** *The* ***product closure*** *of $\mathcal{R}$, noted $\mathcal{R}^{\otimes}$, is defined as: $\mathcal{R}^{\otimes} = \bigcup_{i=0}^{+\infty} \mathcal{R}^{\otimes_i}$.*

The **P**roduct **C**losure **S**tate **M**achine (**PCSM**) of $\mathcal{R}$, defined in [28] and proven equivalent to $\mathcal{R}^{\otimes}$, is the SM with unbounded number of tokens stacked at the beginning in the initial states in $\mathcal{R}$. Then, the instantaneous description of a PCSM gives the number of tokens (instances) at each state in $\mathcal{R}$ that the PCSM currently underlies. This description is called a configuration of $\mathcal{R}^{\otimes}$ and every componenet of the configuration is called a witness of its corresponding state. We omit from this description the initial states (source:infinite number of tokens) and terminal states (sink:terminated instances) and represent only intermediate and hybrid states.

A configuration in $\mathcal{R}^{\otimes}$ is called final if all witnesses that correspond to intermediate states (not final) are null. We define a path of a PCSM as a sequence of transitions between configurations in $\mathcal{R}^{\otimes}$ and a path's length as the number of its transitions. For a configuration $c \in \mathcal{R}^{\otimes}$, $norm(c)$ denotes the length of the shortest path from c to a final configuration.

Example 4. *Figure 3.1 illustrates the execution of the sequence "abca" by the PCSM of the FSM set* $\{M, M'\}$ *in figure 3.1-(a).* M *and* M' *contain one intermediate state* q_1 *and two hybrid states* q_2 *and* q_5*. Therefore, figure 3.1-(b) depicts a part of the PCSM* $\{M, M'\}^{\otimes}$ *with triplets as configurations where integers witness respectively the number of tokens in* q_1*,* q_2 *and* q_5*. For each configuration c in figure 3.1-(b), we associate an instant t (or several instants) during the execution when c describes the PCSM. At the beginning* $(t = 0)$*,* $\{M, M'\}^{\otimes}$*'s instantaneous description is* $(0, 0, 0)$*, interpreting an empty stack in every state of* M *and* M'*, except the initial states* q_0 *and* q'_0 *with an infinite number of tokens (figure 3.1-(c)). To execute the transition* (q_0, a, q_1)*, a token is moved from* q_0 *to* q_1 *in figure 3.1-(d), corresponding to the configuration* $(1, 0, 0)$ *in instant* $t = 1$*. In* $t = 2$*, the executed transition* (q'_0, b, q_4) *corresponds to moving a token from the initial state* q'_0 *to a terminal one* q_4 *(figure 3.1-(e)). Since the instantaneous description does not consider neither initial states nor terminal ones, then the configuration stays the same as the previous instant. Notice that this move corresponds to both creating and terminating an instance of the FSM* M'*. Then, the transition* (q'_0, c, q_5) *is executed by moving a token from* q'_0 *to the hybrid state* q_5*. This creates a new instance implying, in this case, an increase in the number of simultaneously used instances in the execution. This is depicted in figure 3.1-(f). Finally, a token is moved from the state* q_1 *to* q_2 *in figure 3.1-(g), in order to execute the transition* (q_1, a, q_2)*. It changes* $\{M, M'\}^{\otimes}$*'s instantaneous description in* $t = 4$ *into* $(0, 1, 1)$ *which is a final configuration (i.e* $(0, 1, 1) \in F_C$*) since all tokens in the PCSM are in final states (either hybrid or terminal).*

Formally, we define the PCSM of $\mathcal{R}$ as the SM $(\Sigma_{\mathcal{R}}, \mathcal{C}_{\mathcal{R}^{\otimes}}, F_{\mathcal{C}}, c_0, \Phi_{\mathcal{R}^{\otimes}})$, where:

1. $\Sigma_{\mathcal{R}} = \bigcup_{M_j \in \mathcal{R}} \Sigma_{M_j}$;
2. $\mathcal{C}_{\mathcal{R}^{\otimes}}$ is the set of states (also called configurations of $\mathcal{R}^{\otimes}$). $\mathcal{C}_{\mathcal{R}^{\otimes}} \subset \mathbb{N}^n$, with: $n = n_I(\mathcal{R}) + n_H(\mathcal{R})$ with: $n_I(\mathcal{R}) = \Sigma_{M_j \in \mathcal{R}} |I(M_j)|$ and $n_H(\mathcal{R}) = \Sigma_{M_j \in \mathcal{R}} |H(M_j)|$. For each configuration c, $c[m]$ (the m^{th} component of c) is called a witness of the unique state $q_m \in Q_{M_j}$. Note that:
   - q_m is an intermediate state, if $1 \leq m \leq n_I(\mathcal{R})$;
   - q_m is an hybrid state, if $n_I(\mathcal{R}) + 1 \leq m \leq n$.

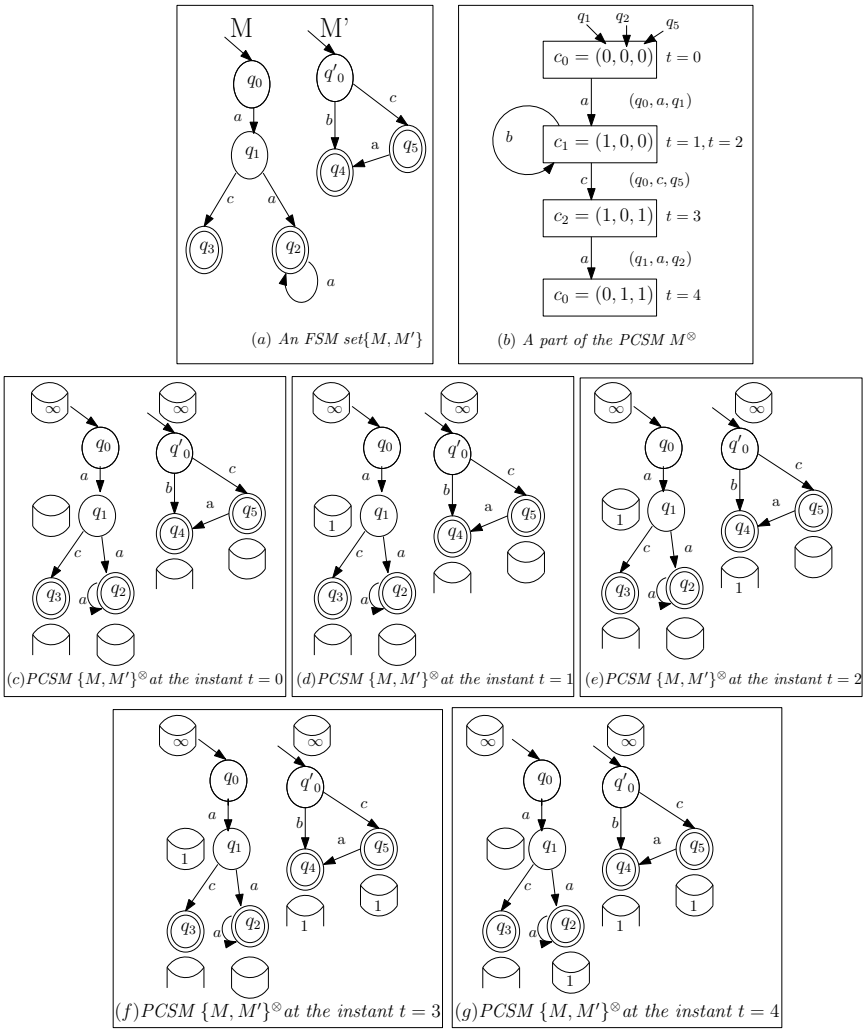


Figure 3.1: An example of execution of a sequence using a PCSM.

In an abuse of notation, we use $c[m]$ and $c[q_m]$ interchangeably.

3. $F_{\mathcal{C}}$ is the set of final states. $F_{\mathcal{C}} = \{c \in \mathcal{C}_{\mathcal{R}^{\otimes}} | c[m] = 0,$ *for each:* $1 \leq m \leq n_I(\mathcal{R})\}$;

4. $c_0 = \{0\}^n$ is the initial state of $\mathcal{R}^{\otimes}$;

5. $\Phi_{\mathcal{R}^{\otimes}} \subseteq \mathcal{C}_{\mathcal{R}^{\otimes}} \times \Sigma_{\mathcal{R}} \times \mathcal{C}_{\mathcal{R}^{\otimes}}$ is the set of transitions. we have $(c_1, a, c_2) \in \Phi_{\mathcal{R}^{\otimes}}$ if

and only if:

- there exists $(q_0, a, q) \in Q_{M_j}$, such that: q_0 is the initial state of M_j and $c_2[q] = c_1[q] + 1$ and $c_2[p'] = c_1[p']$ for each $p' \neq q$. Or
- there exists $(p, a, q) \in Q_{M_j}$, such that: p is not the initial state of M_j, q is either a terminal state or the initial state of M_j, $c_2[p] = c_1[p] - 1$ and $c_2[p'] = c_1[p']$ for each $p' \neq p$. Or
- there exists $(p, a, q) \in Q_{M_j}$, such that: neither p nor q is the initial state of M_j, $c_2[p] = c_1[p] - 1$, $c_2[q] = c_1[q] + 1$ and $c_2[p'] = c_1[p']$ for each $p' \neq p, q$.

We recall below the definition of the simulation preorder between two SMs.

Definition 6. ***(Simulation)***
Let $M = (\Sigma_M, Q_M, F_M, q^0_M, \delta_M)$ and $N = (\Sigma_N, Q_N, F_N, q^0_N, \delta_N)$ be two SMs. A state $p \in Q_M$ is simulated by a state $q \in Q_N$, denoted $p \ll_{(M,N)} q$ ($p \ll q$ when M and N are understood from context), if and only if the following two conditions hold:

1. *$\forall a \in \Sigma_M$ and $\forall p' \in Q_M$ such that $(p, a, p') \in \delta_M$, there exists $(q, a, q') \in \delta_N$ such that $p' \ll q'$, and*
2. *if $p \in F_M$, then $q \in F_N$.*

M is simulated by N, denoted $M \ll N$, if and only if the initial state of N simulates the initial state of M, i.e. *$q^0_M \ll q^0_N$.*

Observe that, by definition, each transition of a PCSM can at most increase or decrease a configuration component by 1. In addition, if a configuration is final then all intermediate states witnesses are equal to 0. Therefore, given a set of FSMs $\mathcal{R}$ and $c \in \mathcal{C}_{\mathcal{R}^\otimes}$, we have $\Sigma_{q \in \bigcup_{M_i \in \mathcal{R}} I(M_i)} c[q] \leq norm(c)$. Moreover, since final states can only be simulated by final ones, then for M an FSM and $p \in Q_M$, if $p \ll c$ then $norm(c) \leq norm(p)$. Hence, we are able to derive the following property.

Property 1. ***(Intermediate witnesses bound) [28]*** *For $c \in \mathcal{C}_{\mathcal{R}^\otimes}$ and $p \in Q_M$, if $p \ll c$ then $\Sigma_{q \in I(\mathcal{R})} c[q] \leq norm(p)$, where $I(\mathcal{R}) = \bigcup_{M_i \in \mathcal{R}} I(M_i)$.*

We denote $\mathcal{C}^M_{\mathcal{R}^\otimes} = \{c \in \mathcal{C}_{\mathcal{R}^\otimes} | \Sigma_{q \in I(\mathcal{R})} c[q] \leq norm(M)\}$.

In [28], the WSC problem in the unbounded case is reduced to simulation test between an FSM and a PCSM and proven to be decidable. The termination of the algorithm given in [28] is proven using the following property:

Property 2. ***(configuration cover) [28]*** *Let c and c' be two configurations of $\mathcal{R}^{\otimes}$, such that: $c[m] = c'[m]$, $m \in [1, n_I(\mathcal{R})]$ and $c[m] \leq c'[m]$, $m \in [n_I(\mathcal{R}) + 1, n]$. if $q \ll c$, where q is a state of a SM M, then $q \ll c'$.*

We say that c' covers c, denoted $c \lhd c'$.

We introduce below the algorithm of [28], focusing the presentation on the structure of its execution tree.

Definition 7. ***(Simulation Tree of an FSM by a PCSM)***
We call a simulation tree $T_{sim}(M, \mathcal{R}^{\otimes}) = (V, v_0, E)$ with:

- *$v_0 = (q_M^0, c_0)$ is the root of the tree;*
- *$V \subset Q_M \times \mathcal{C}_{\mathcal{R}^{\otimes}}^M$ is the set of nodes;*
- *If $(q, c) \in V$ and q is final in M then so is c in $\mathcal{R}^{\otimes}$;*
- *$E \subset V \times V$ is the set of the tree's edges. $\forall e = ((p, c), (q, d)) \in E : \exists a \in \Sigma_M$ s.t $(p, a, q) \in \delta_M$ and $(c, a, d) \in \Phi_{\mathcal{R}^{\otimes}}$.*
- *$v = (p, c) \in V$ is a leaf in $T_{sim}(M, \mathcal{R}^{\otimes})$ iff p is terminal in A or there exists an ancestor $(p, c') \in V$ of v in $T_{sim}(M, \mathcal{R}^{\otimes})$ such that $c \lhd c'$.*

Example 5. *Figure 3.2-(c) is an example of a simulation tree, verifying if the initial state s_0 of the FSM A (figure 3.2-(a)) is simulated by the initial configuration $c_0 = (0, 0, 0)$ of the PCSM $\{M, M'\}^{\otimes}$ (figure 3.2-(b)). A branch is terminated with success when a terminal state of A is reached and paired with a final configuration (all intermediate witnesses are null), or when a configuration of $\{M, M'\}^{\otimes}$ that covers one of its predecessors is reached and paired with the same state of A. In this case, the simulation tree proves that $A \ll \{M, M'\}^{\otimes}$.*

In the next section, we shall bound the size of this tree in the case of bounded WSC problem (*i.e.*, when the number of web services instances allowed to be used in the simulation is bounded by a parameter k).

3.2 Bounded Composition

3.2.1 Composition with bounded number of instances

We call a *bounded* WSC problem, a service composition problem where the number of copies of each web service in the repository $\mathcal{R}$ used to compose the target M is bounded a priori by an integer k. This problem is formally stated as follows.

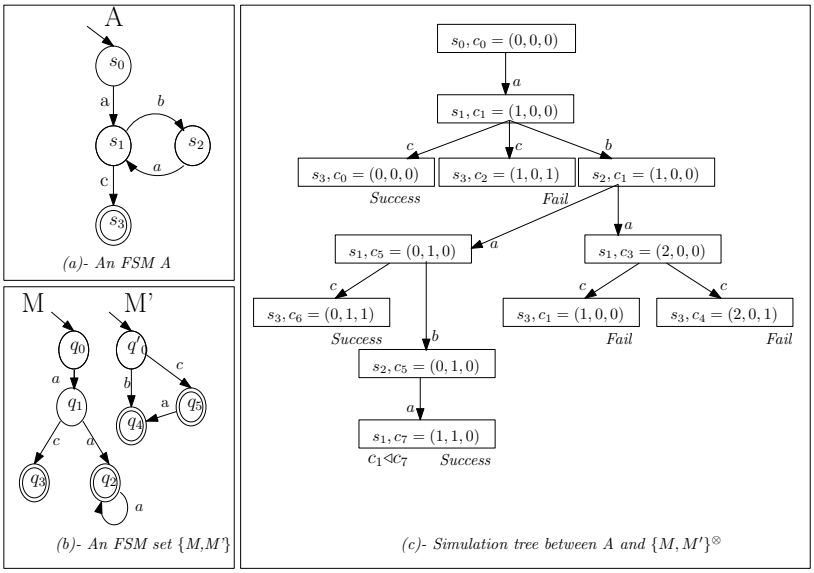


Figure 3.2: An example of a simulation tree.

Problem 1. ***Bounded Composition*** $BC(M, \mathcal{R}, k)$
Input : $\mathcal{R}$ *a set of FSMs;* M *a target FSM;* k *an integer.*
Question : $M \ll \mathcal{R}^{\otimes_k}$ *?*

The particular case $BC(M, \mathcal{R}, 1)$ has been investigated by Muscholl and Walukiewicz [38] where it is shown to be Exptime-Complete. We shall prove in this section that $BC(M, \mathcal{R}, k)$ is also Exptime-Complete. We point out that the straightforward reduction of $BC(M, \mathcal{R}, k)$ to $BC(M, \mathcal{R}, 1)$, obtained by duplicating k times each service of $\mathcal{R}$, is not polynomial in the input size, since k may be large, and hence cannot be used to achieve our goal.

The parameter k drops the infinite aspect and reduces the search space. In this case, a loop in M can only be simulated by loops in $\mathcal{R}$.

For example, one can observe that, in figure 3.3, M is not simulated by $\{R_1, R_3\}^{\otimes^k}$ for every $k \in \mathbb{N}$. This is because when we repeat the loop in M $(k+1)$ times, there is no corresponding execution in $\{R_1, R_3\}^{\otimes^k}$. However, we have $M \ll \{R_1, R_2\}^{\otimes^k}$, for any $k \geq 1$.

In the following, we give an upper bound of the number of states that might appear in $\mathcal{R}^{\otimes_k}$, with $k \in \mathbb{N}$.

Lemma 8. *Let* $\mathcal{R}$ *be a set of FSM and* k *is an integer. The number of states in* $\mathcal{R}^{\otimes_k}$*, noted* $|\mathcal{C}_{\mathcal{R}^{\otimes_k}}|$*, is bounded by* $|\{c \in \mathbb{N}^n \ s.t \ c[i] \leq k, i \in [1,n]\}| = O(2^{nlogk})$*, where*

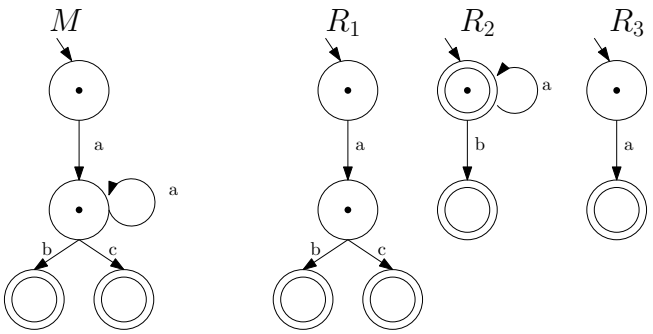


Figure 3.3: An example of bounded composition problem instances.

$n = n_I(\mathcal{R}) + n_H(\mathcal{R})$.

Proof:
Notice that $\mathcal{R}^{\otimes} = (\bigcup_{i=0}^{k} \mathcal{R}^{\otimes_i}) \cup (\bigcup_{i=k+1}^{+\infty} \mathcal{R}^{\otimes_i})$.

In fact, the states in $\bigcup_{i=0}^{k} \mathcal{R}^{\otimes_i}$ correspond to the PCSM's configurations subset $\{c \in \mathcal{C}_{\mathcal{R}^{\otimes_k}} \mid 0 \leq c[i] \leq k, i \in [1,n]\}$. Hence, the number of states of $\bigcup_{i=0}^{k} \mathcal{R}^{\otimes_i}$ is bounded by $\underbrace{(k+1) \times \ldots \times (k+1)}_{\text{n times}} = 2^{nlog(k+1)}$. □

This lemma reduces the search space to an exponential size and leads to the following theorem.

Theorem 9. *$BC(M, \mathcal{R}, k)$ is Exptime-Complete*

Proof: `Exptime.` To show that $BC(M, \mathcal{R}, k)$ is Exptime, we bound the size of the simulation tree. A node of the simulation tree corresponds to (q, c) where q is a state of M and c a configuration of $\mathcal{R}^{\otimes_k}$. According to Lemma 8, the number of PCSM's configurations is bounded by k^n. So the number of nodes in the simulation tree is at most $|Q_M| \times k^n = 2^{nlog(k)+log(|Q_M|)}$. And since each node does not have to be visited more than once, then the complexity is in Exptime.

`Exptime-Hardness.` It can be deducted directly from the Exptime-Hardness of the particular case $BC(M, \mathcal{R}, 1)$ [38]. □

3.2.2 Composition with bounded number of parallel instances

We consider a new parameter in service web composition that bounds the number of communications in parallel between the target and the services, i.e. the number

of live services executions is bounded, but the number of instances is not. It appears that the web services composition with unbounded instances and bounded parallel instances is Exptime-Complete.

To do so, we limit the configurations of the PCSM $\mathcal{R}^{\otimes}$ to configurations where the number of waiting instances is bounded by k. Indeed, when we need to use a new instance in $\Phi_{\mathcal{R}^{\otimes}}$, we check if $\sum_{i=1}^{n} c[i] \geq k$. If so, we decrease $c[j]$ for some $j \in [n_I(\mathcal{R}) + 1, n_I(\mathcal{R}) + n_H(\mathcal{R})]$, i.e. we finish an instance that is waiting in an hybrid state. Let us denote by $\mathcal{R}^{\otimes_{k,p}}$ the obtained state machine.

Problem 2. ***Bounded Parallel Instances Composition*** *($PBC(M, \mathcal{R}, k)$)*
Input : $\mathcal{R}$ *a set of FSMs;*
M *a target FSM.*
k *an integer, bounding the number of parallel instances of* $\mathcal{R}$*'s components used simultaneously in the simulation.*
Question : $M \ll \mathcal{R}^{\otimes_{k,p}}$ *?*

Note that $PBC(M, \mathcal{R}, k)$ can use an unbounded number of instances but only k instances in parallel.

Theorem 10. *$PBC(M, \mathcal{R}, k)$ is Exptime-complete.*

Proof: First we show that $PBC(M, \mathcal{R}, k)$ is Exptime. Clearly the entry of any configuration is bounded by k (hybrid states are included) and therefore we can check simulation in Exptime, since the depth of the simulation tree is bounded by k^n (see Lemma 8).

To show the Exptime-hardness, we reduce the unbounded composition without hybrid states to the problem $PBC(M, \mathcal{R}, k)$. In fact, the number of tokens in intermediate states of $\mathcal{R}$ is bounded by $norm(M)$ (property 1). Hence, when $\mathcal{R}$ is hybrid state free, the number of instances that can be used in the simulation is bounded by $norm(M)$. In other words, it corresponds to $PBC(M, \mathcal{R}, norm(M))$. And since unbounded composition without hybrid states is Exptime-hard **??**, then $PBC(M, \mathcal{R}, k)$ is Exptime-hard, and therefore Exptime-complete. □

For k a constant, we obtain the following.

Corollary 11. *$PBC(M, \mathcal{R}, k)$ is polynomial, for each fixed integer k.*

Proof: First of all, let us consider for every configuration c of $\mathcal{R}^{\otimes_{k,p}}$, a new component $c[n+1] = k - (\sum_{i=1}^{n} c[i])$, with $n = n_I(\mathcal{R}) + n_H(\mathcal{R})$.

For every configuration c in $\mathcal{R}^{\otimes_{k,p}}$, the non-empty witnesses $\{c[i] > 0, 0 \leq i \leq n\}$ correspond to a partition of k elements (instances) into a sequence of j non empty

subsets, for $j = |\{c[i] > 0, 1 \leq i \leq n\}| \leq k$. Note that j is in fact inferior to $min(k, n)$, but since k is a constant then it is more interesting to keep it as a lower bound of j.

For every $j \leq k$, the number of labeled partitions of k elements into a sequence of j non empty subsets is $j! \times \{^k_j\}$, where $\{^k_j\}$ is a Stirling number of the second kind [1]. Hence, the number of configurations in $\mathcal{R}^{\otimes_{k,p}}$ that have j non-empty witnesses is bounded by $C^j_n \times j! \times \{^k_j\}$. Notice that $C^j_n = e^{\frac{n...\times(n-j+1)}{j!}}$ is in the order of $O(n^j)$.

We conclude that the number of configurations in $\mathcal{R}^{\otimes_{k,p}}$ is bounded by $\sum_{j=1}^{k} C^j_n \times j! \times \{^k_j\} \in O(n^k)$.

Finally, by applying the simulation algorithm in [29], $PBC(M, \mathcal{R}, k)$ can be decided in $O(m_v.m_e)$, where $m_v = |Q_M| + |\mathcal{C}_{\mathcal{R}^\otimes}|$ and $m_e \leq |Q_M|^2 + |\mathcal{C}_{\mathcal{R}^\otimes}|^2$ are respectively the number of edges and transitions in M and $\mathcal{R}^{\otimes_{k,p}}$. □

Another factor of complexity of the WSC problem is the number of hybrid states in the available services. We investigate next the effect of this parameter on the complexity of the WSC problem.

3.3 Bounded number of hybrid states

The presence of hybrid states is a source of complexity in a WSC problem. As mentioned before, the size of intermediate states witnesses in configurations of $\mathcal{R}^\otimes$ used to simulate M is bounded by $norm(M)$. We are however unable to provide a similar bound for the number of hybrid states witnesses.

Figure 3.4 is an example of simulation between an FSM M and a PCSM $\mathcal{R}^\otimes$. The FSMs in $\mathcal{R}$ contain two hybrid states (state 1 and 2) and no intermediate state. Hence, a configuration of $\mathcal{R}^\otimes$ is a pair of integers witnessing the number of tokens in state 1 and state 2. The example illustrates the different roles that an hybrid state of $\mathcal{R}$ can play to simulate a state of M. Indeed an hybrid state of $\mathcal{R}$, can be used as: (i) a terminal state, e.g., when testing whether $q_5 \ll (1, 1)$, we can consider the second hybrid state of $\mathcal{R}$ as a terminal state and terminate the test, or (ii) an intermediate state, e.g., when testing whether $q_2 \ll (1, 1)$, the second hybrid state of $\mathcal{R}$ here plays the role of intermediate state, or (iii) both a terminal and an intermediate state, e.g., when testing whether $q_1 \ll (1, 0)$, a transition of $\Phi_{\mathcal{R}^\otimes}$ labeled by $(b, (-1, 0))$ only appears in one branch in the simulation tree $\mathcal{T}_{sim}(M, \mathcal{R}^\otimes)$. Hence, the first hybrid state of $\mathcal{R}^\otimes$ is considered intermediate in one branch and terminal in the other, or a hybrid state, e.g., when it is used to simulate an hybrid state of $H(M)$.

We consider in the following the problem defined below.

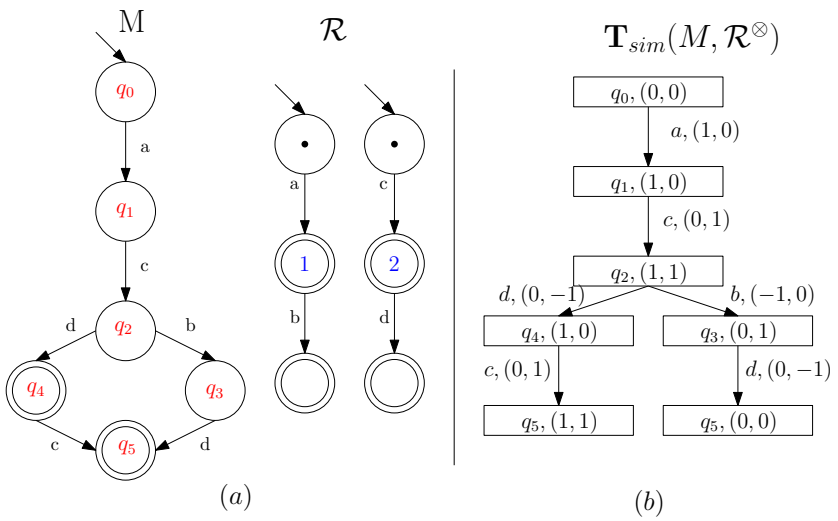


Figure 3.4: Example of the simulation tree

Problem 3. ***Unbounded Composition With limited number of Hybrid States*** $UCHS(M, \mathcal{R}, k)$

Input : k an integer; $\mathcal{R}$ a set of FSMs, containing at most k hybrid states; M a target FSM.
Question : $M \ll \mathcal{R}^{\otimes}$?

We point out that $UCHS(M, \mathcal{R}, k+1)$ is harder then $UCHS(M, \mathcal{R}, k)$. In the sequel, we progressively investigate the complexity of $UCHS(M, \mathcal{R}, k)$ problem for $k = 0$, then for $k = 1$ and finally for $k = 2$.

3.3.1 Case of composition without hybrid states

In this section, we are interested in the problem $UCHS(M, \mathcal{R}, 0)$. We first give a polynomial transformation, denoted $\mathcal{K}$, which is used to reduce $BC(M, \mathcal{R}, 1)$ to $UCHS(N, \mathcal{R}', 0)$. This transformation provides a mean to bound the number of instances used to prove simulation.

Definition 12. ***Transformation*** $\mathcal{K}$. *For an FSM $M = (\Sigma_M, Q_M, F_M, q_0^M, \delta_M)$ and a set of FSMs $\mathcal{R} = \{M_1, ..., M_m\}$, we define $\mathcal{K}(M, \mathcal{R}) = (N, \mathcal{R}' = \{N_1, .., N_m\})$ where:*

1. *Each N_i is built based on M_i, by adding a letter t_i to its alphabet, a final state f_i and a transition set $\{(q_0^{M_i}, t_i, f_i)\} \cup \{(q, t_i, f_i) | q \in F_{M_i}\}$. All final states of M_i become intermediate in N_i.*

2. N is defined as:

- $\Sigma_N = \Sigma_M \cup \{t_i | 1 \leq i \leq m\}$;
- $Q_N = Q_M \cup \{r_i | 1 \leq i \leq m\}$;
- $F_N = \{r_m\}$;
- $\delta_N = \delta_M \cup \{(q, t_1, r_1) | q \in F_M\} \cup \{(r_i, t_{i+1}, r_{i+1}) | 1 \leq i < m\}$.

Figure 3.5 illustrates an example of this transformation. We prove later in proposition 14 that $\mathcal{K}$ defines a polynomial reduction of $BC(M, \mathcal{R}, 1)$ to $UCHS(N, \mathcal{R}', 0)$. In fact, the intuition behind this reduction is based on two points:

- By adding the sequence of letters $t_1, ..., t_m$ at the end of every execution accepted by N and adding t_i at the end of every execution accepted by $N_i \in \mathcal{R}'$, we ensure that even in an unbounded instances simulation, we can not use more than one instance of every N_i in order to simulate N.

- The construction of $\mathcal{R}'$ verifies that every hybrid state in $M_i \in \mathcal{R}$ becomes intermediate in N_i, while keeping its dual role: either terminate the execution by adding the letter t_i to the execution of N_i and reaching the terminal state f_i, or keep the execution in the same way as M_i.

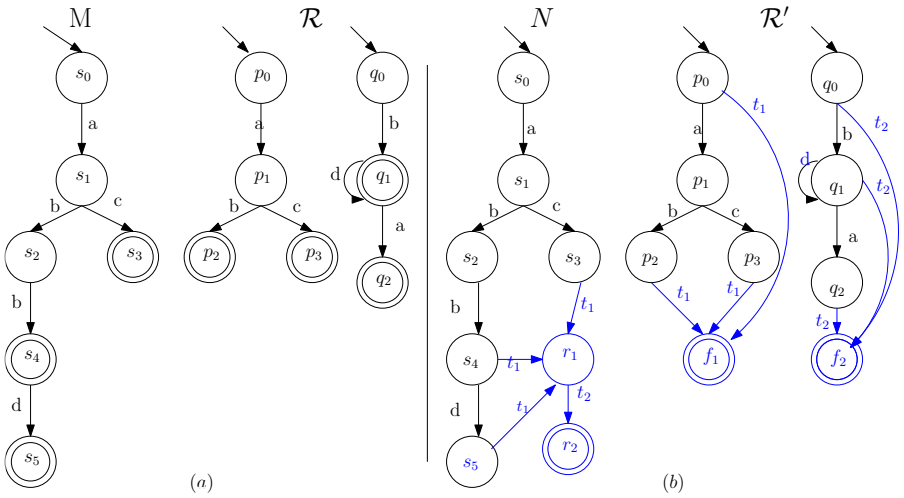


Figure 3.5: *An example of transformation $\mathcal{K}$*

The following propositions show that the transformation $\mathcal{K}$ preserves the simulation preorder.

Proposition 13. *Let M be an FSM, $\mathcal{R} = \{M_1, ..., M_m\}$ be a set of FSMs and $\mathcal{K}(M, \mathcal{R}) = (N, \mathcal{R}' = \{N_1, .., N_m\})$. For p and q two states of respectively M and $\mathcal{R}^{\otimes_1}$, we have: $p \ll_{(M,(\mathcal{R})^{\otimes_1})} q$ iff $p \ll_{(N,(\mathcal{R}')^{\otimes_1})} q$.*

Proof: Let p be a state of M and q be a state $\mathcal{R}^{\otimes_1}$ verifying $p \ll_{(M,(\mathcal{R})^{\otimes_1})} q$.

By construction of $\mathcal{K}(M, \mathcal{R})$, *if* p is terminal in M *then* $p \ll_{(N,(\mathcal{R}')^{\otimes_1})} q$.

We suppose next that for every $(p, a, p') \in \delta_M$ and $(q, a, q') \in \delta_{\mathcal{R}^{\otimes_1}}$, *if* $p' \ll_{(M,\mathcal{R}^{\otimes_1})} q'$, *then* $p' \ll_{(N,(\mathcal{R}')^{\otimes_1})} q'$. We prove, under this hypothesis, that $p \ll_{(N,(\mathcal{R}')^{\otimes_1})} q$.

For each $(p, a, p') \in \delta_N$, we have:

- *if* $a \in \Sigma_M$, *then* there exists $(q, a, q') \in \delta_{\mathcal{R}^{\otimes_1}} \subseteq \delta_{(\mathcal{R}')^{\otimes_1}}$ such that $p' \ll_{(N,(\mathcal{R}')^{\otimes_1})} q'$.

- else $a = t_1$, $p' = r_1$ and q is a product of final states of $\mathcal{R}$. therefore, there exists $(q, t_1, q') \in \delta_{(\mathcal{R}')^{\otimes_1}}$ such that $q' = (f_1, {q'}_{i_1}, ..., {q'}_{i_l})$ where ${q'}_{i_j}$ is final in $\mathcal{R}$ such that $p' \ll_{(N,(\mathcal{R}')^{\otimes_1})} q'$.

We conclude that if $p \ll_{(M,\mathcal{R}^{\otimes_1})} q$ then $p \ll_{(N,(\mathcal{R}')^{\otimes_1})} q$.

Reciprocally, we have $(p, a, p') \in \delta_N$ (respectively $\delta_{(\mathcal{R}')^{\otimes_1}}$) and $a \notin \{t_i | 1 \leq i \leq m\}$ iff $(p, a, p') \in \delta_M$ (respectively $\delta_{\mathcal{R}^{\otimes_1}}$). In addition, the definition of $\mathcal{K}$ ensures that if p is final in M and $p \ll_{(N,(\mathcal{R}')^{\otimes_1})} q$ then q is final in $\mathcal{R}^{\otimes_1}$. Hence if $p \ll_{(N,(\mathcal{R}')^{\otimes_1})} q$ then $p \ll_{(M,\mathcal{R}^{\otimes_1})} q$. □

In particular, we take p as the initial state of M and q the initial state of $\mathcal{R}^{\otimes_1}$. This implies that:

Proposition 14. *Let M be an FSM, $\mathcal{R} = \{M_1, ..., M_m\}$ be a set of FSMs and $\mathcal{K}(M, \mathcal{R}) = (N, \mathcal{R}' = \{N_1, .., N_m\})$. We have: $M \ll \mathcal{R}^{\otimes_1}$ iff $N \ll (\mathcal{R}')^{\otimes}$.*

Proof: We have $N \ll (\mathcal{R}')^{\otimes_1}$ if and only if $N \ll (\mathcal{R}')^{\otimes}$. Indeed, each path that starts from the initial state to a final one in N contains exactly one transition labeled by t_i, for each $i \in [1, m]$ and a similar path in each N_i contains exactly one transition labeled by t_i. □

Hence, $\mathcal{K}$ is a polynomial reduction of $BC(M, \mathcal{R}, 1)$ problem to the UCHS problem. This enables to derive the following result.

Theorem 15. *$UCHS(M, \mathcal{R}, 0)$ problem is Exptime-complete.*

Proof: According to proposition 14, the $\mathcal{K}$ transformation reduces $BC(M, \mathcal{R}, 1)$ to $UCHS(M, \mathcal{R}, 0)$ in polynomial time. Thus $UCHS(M, \mathcal{R}, 0)$ is Exptime-hard. Since it is also proven Exptime in [28], then $UCHS(M, \mathcal{R}, 0)$ is Exptime-complete. □

3.3.2 Case of composition with one hybrid state

We consider the problem $UCHS(M, \mathcal{R}, 1)$ where M is an FSM and $\mathcal{R}$ a set of FSMs containing at most one hybrid state ($n_H(\mathcal{R}) \leq 1$). We denote $k_0 = |Q_M|.2^{n_I(\mathcal{R}).log(norm(M))}$. Two nodes (q, c) and (q', c') in a simulation tree are called comparable if $q = q'$ and either $c \lhd c'$ or $c' \lhd c$. The nodes (q, c) and (q', c') are said incomparable otherwise.

Property 3. *Let $\mathcal{R}$ be a set of FSMs containing at most one hybrid state. Two configurations of $\mathcal{R}^{\otimes}$ are comparable by the cover relation, if and only if they have exactly the same intermediate witnesses.*

Proof: According to property 2, for c, c' two configurations in $\mathcal{R}^{\otimes}$ we have $c \lhd c'$ *iff*:

1. c and c' have the same intermediate witnesses; and
2. for every hybrid witness $c[h]$, we have: $c[h] \leq c'[h]$.

In the current case, we consider that $\mathcal{R}$ has at most one hybrid witness. Hence, for any pair of configurations of $\mathcal{R}^{\otimes}$, condition 2 is verified.

We conclude that for every two configurations c, c' in $\mathcal{R}^{\otimes}$, $c \lhd c'$ *iff* c and c' have the same intermediate witnesses. □

Property 4. *Let S be a set of nodes of $\mathcal{T}_{sim}(M, \mathcal{R}^{\otimes})$ that are pairwise incomparable, then $|S| \leq k_0$.*

Proof: In configurations considered in $\mathcal{T}_{sim}(M, \mathcal{R}^{\otimes})$, intermediate witnesses are bounded by $norm(M)$ (property 1). Therefore and according to property 3, the number of incomparable configurations considered in $\mathcal{T}_{sim}(M, \mathcal{R}^{\otimes})$ is at most $2^{n_I(\mathcal{R}).log(norm(M))}$. Since $S \subset Q_M \times \mathcal{C}_{\mathcal{R}^{\otimes}}$, then $|S| \leq k_0$. □

Proposition 16. *If $n_H(\mathcal{R}) = 1$, then for each $(q, c) \in \mathcal{T}_{sim}(M, \mathcal{R}^{\otimes})$, the witness $c[h]$ of the unique hybrid state in $\mathcal{R}$ is bounded by ${k_0}^2 + k_0$.*

Proof: Let P be a path in $\mathcal{T}_{sim}(M, \mathcal{R}^{\otimes})$ and $S = (v_n = (q_n, c_n))_{n \in \mathbb{N}}$ be the sequence of nodes in P where each $v_i \in S$ is the i^{th} node met in P that is comparable to exactly one of its predecessors $v = (q_i, c)$ in P. Note that:

1. Each v_i and v_j in S are incomparable;

2. If a node $v \in P$ is comparable to more than one ot its predecessors in P, then $v \notin S$;

3. Nodes in S are indexed in the order of their appearance of P.

If $S = \emptyset$, then all nodes of P are not comparable. The size of P is then bounded by k_0 (property 4), therefore, $c[h] \leq k_0$ for each (q, c) in P.

We suppose next that $S \neq \emptyset$ and take $S = (v_1, ..., v_k)$, $k \in \mathbb{N}$. We prove recursively that for each $n \in [1, k]$, $c_n[h] \leq n.k_0$.

For $n = 1$, all predecessors of v_1 in P form a set of nodes that are incomparable pairwise. Hence, $c_1[h] \leq k_0$ (property 4).
For $1 < n < k$, we suppose that $c_n[h] \leq n.k_0$. Each node $v = (q, c)$ strictly between v_n and v_{n+1} in P, v is not in S, therefore either:

1. v is comparable to a node v_i with $i \in [1, n]$. In this case, $c[h] < c_i[h] \leq n.k_0$ (otherwise $c_i \lhd c$, thus by definition of $\mathcal{T}_{sim}(M, \mathcal{R}^{\otimes})$, v should be a leaf).

2. v is incomparable to all its predecessors. The number of such nodes in P is strictly bounded by $k_0 - 1$ (because they should all be incomparable to each other and to v_n and v_{n+1}). And since transitions displacements is in $\{-1, 0, 1\}^h$, then we have $c[h] < n.k_0 + k_0 - 1$.

We conclude from above that for every $v = (q, c)$ between v_n and v_{n+1} in P, $c[h] \leq n.k_0 + k_0 - 1$ (supposing w.l.o.g that $k_0 \geq 1$). Thus, $c_{n+1}[h] \leq n.k_0 + k_0 = (n+1).k_0$.

Once we reach the last node v_k in S, all its possible successors in P are either: comparable to a node $v_i \in S$ with $c[h] < c_i[h]$, or incomparable to any of its predecessors in P or it is the leaf of P.

Finally, since $k < k_0$ (because S is a sequence of incomparable nodes), we conclude that each node of P is in $Q_M \times ([1, norm(A)]^{n_I(\mathcal{R})} \times [1, {k_0}^2 + k_0])$. $\square$

Lemma 17. *$UCHS(M, \mathcal{R}, 1)$ is in Exptime.*

Proof: To show that $UCHS(M, \mathcal{R}, 1)$ is Exptime, we bound the size of the simulation tree. A node of the simulation tree corresponds to (q, c) where q is a state of M and c a configuration of $\mathcal{R}^{\otimes}$ that verifies, according to proposition 16, the following:

- $c[h] \leq {k_0}^2$ where $c[h]$ is the witness of the unique hybrid state in $\mathcal{R}$;
- $c[i] \leq norm(M)$ where $c[i]$ is a witness of an intermediate state in $\mathcal{R}$.

Hence, the number of nodes in the simulation tree is bounded by

$$\underbrace{|Q_M|.norm(M)^{n_I(\mathcal{R})}}_{k_0}.\ (k_0^2 + k_0) = O(k_0^3)$$

And since deciding simulation only requires to visit a node once, then the complexity is in Exptime.

□

To prove the Exptime-hardness of the problem, we recall that $UCHS(M, \mathcal{R}, 0)$ is Exptime-hard (theorem 15) and that $UCHS(M, \mathcal{R}, 1)$ is harder than $UCHS(M, \mathcal{R}, 0)$.

Theorem 18. *$UCHS(M, \mathcal{R}, 1)$ is Exptime-complete.*

3.3.3 Case of composition with two hybrid states

In this section, we consider the problem of unbounded composition of web services with at most 2 hybrid states in $\mathcal{R}$, i.e $UCHS(M, \mathcal{R}, 2)$.

Our approach is based on reducing the simulation problem to the Z-reachability issue [13, 17].

Interestingly, the simulation verification has been reduced in [40] to a two players game in a directed graph $(V_{att}, V_{def}, \delta, v_0)$, such that $V = V_{att} \cup V_{def}$ is the set of vertices with $V_{att} \subseteq Q_M \times Q_N$ and $V_{def} \subseteq Q_M \times Q_N \times \Sigma_M$, $\delta \subseteq (V_{att} \times V_{def}) \cup (V_{def} \times V_{att})$ is the edge set verifying:
-for $(q, p) \in V_{att}$ and $(q, a, q') \in \delta_M$, we have $((q, p), (q', p, a)) \in \delta$; and
-for $(q, p, a) \in V_{def}$ and $(p, a, p') \in \delta_N$, we have $((q, p, a), (q, p')) \in \delta$.

The game is played by an attacker and a defender. It starts by putting a token in $v_0 = (q_M^0, q_N^0) \in V_{att}$, then the players move it along the edges of the graph. If the token is on a vertex $v \in V_{att}$ then the attacker moves it, otherwise it is the defender's turn.

A strategy of a player $x \in \{a, d\}$ is a function $S : V^*.V_x \mapsto V$, where $V^*.V_x$ denotes all sequences of vertices in V that end with a vertex in V_x and $S(v_0, ..., v_k) = v_{k+1}$ implies that $(v_k, v_{k+1}) \in \delta$. In each different play, a player x adapts a strategy that decides his moves.

The defender wins every infinite play. Otherwise, the first player who cannot move loses. M is simulated by N **iff** the defender has a winning strategy regardless of his opponent's strategy.

The Z-reachability game, on the other hand, is played on a finite weighted graph $(V_\Box, V_\Diamond, E, v_0, e_0, k)$ by 2 players $\Box$ and $\Diamond$. A play begins by placing a token in $v_0 \in V_\Box$, then the players move it along the graph's edges $E \subseteq V \times V \times \{-1, 0, 1\}^k$, with $V = V_\Box \cup V_\Diamond$ and $k \in \mathbb{N}$. If the token is in vertex $v \in V_\Box$ then $\Box$ moves it, otherwise his opponent does. The play is winning for $\Diamond$ if the components of the sum of the weights of the edges traversed plus $e_0 \in \mathbb{N}^k$ are strictly above $(0, .., 0) \in \mathbb{N}^k$ during the whole play, otherwise $\Box$ wins. if the play is finite, then the first player who cannot move loses. We define here a strategy of a player $x \in \{\Box, \Diamond\}$ like in the simulation game. A player wins the Z-reachability game if he has a strategy that ensures winning, whatever his opponent's strategy is. Chaloupka proves in [17] that a 2-dimensional Z-Reachability problem (for k=2) can be solved in $O(|V|^{17})$.

Considering an instance of the problem $UCHS(M, \mathcal{R}, k)$, we build next an equivalent k-dimensional Z-Reachability game of an exponential size.

Theorem 19. *There exist an algorithm that can solve $UCHS(M, \mathcal{R}, 2)$ in $O((norm(M)^n \times |Q_M| \times |\Sigma_M|)^{17})$, with n is the number of states in $\mathcal{R}$.*

Proof: Considering the simulation game $(V_{att}, V_{def}, \delta, v_0)$ associated to M and $\mathcal{R}^\otimes$, note that the only known upper bound of $|V_{att} \cup V_{def}|$ is Ackermanian. However the set $\mathcal{C}_I = \{(c[1], ..., c[n_I(\mathcal{R})])|c \in \mathcal{C}^M_{\mathcal{R}^\otimes}\}$ has an exponential size (Property 1).
Hence, we consider the weighted graph $(V_\Box, V_\Diamond, E, w_0, e_0, k)$ with: $V_\Box \subseteq Q_M \times \mathcal{C}_I$; $V_\Diamond \subseteq Q_M \times \mathcal{C}_I \times \Sigma_M$; $E \subseteq V \times V \times \{-1, 0, 1\}^k$, with $V = V_\Box \cup V_\Diamond$, $k = n_H(\mathcal{R})$ and:
-for $(q, c) \in V_\Box$ and $(q, a, q') \in \delta_M$, we have $((q, p), (q', p, a), (0, ..., 0)) \in \delta$;
-for $(q, c, a) \in V_\Diamond$ and $(d, a, d') \in \Phi_{\mathcal{R}^\otimes}$ with c and d have the same intermediate components $(c[i] = d[i], i \in [1, n_I(\mathcal{R})])$, we have $((q, c, a), (q, c')) \in \delta$ with $c'[i] = d'[i], i \in [1, n_I(\mathcal{R})]$.
and $w_0 = (q^0_M, (0, ..., 0)) \in V_\Box$ and $e_0 = (1, .., 1) \in \mathbb{N}^k$.

We consider two mappings:
Let $f : V_{def} \cup V_{att} \mapsto V$ be defined for $q, c, a \in Q_M, \mathcal{C}^M_{\mathcal{R}^\otimes}, \Sigma_M$ as: $f(q, c) = (q, c')$ and $f(q, c, a) = (q, c', a)$ with $c'[i] = c[i]$ for $i \in [1, n_I(\mathcal{R})]$.

Let $g : V^* \mapsto V_d \cup V_{att}$ be defined for $q, c, a \in Q_M, \mathcal{C}^M_{\mathcal{R}^\otimes}, \Sigma_M$ and $w_0, ..., w_l \in V$ as: $g(w_0, ..., w_l, w_{l+1} = (q, c)) = (q, c')$ or $g(w_0, ..., w_l, w_{l+1} = (q, c, a)) = (q, c', a)$ and for each $i \in [1, n_I(\mathcal{R})]$ and $j \in [1, n_H(\mathcal{R})]$, $c'[i] = c[i]$ and $c'[n_I(\mathcal{R}) + j]$ is equal to the j^{th} component of the sum of weights of the edges traversed in the path $\{w_0, ..., w_{l+1}\}$.

Let $S_\Diamond$ be a winning strategy of $\Diamond$ in the Z-Reachability game $(V_\Box, V_\Diamond, E, v_0, e_0, k)$. We build next a winning strategy S_d for the defender in the simulation game $(V_{att}, V_{def}, \delta, v_0)$. Let $v_0, ..., v_l \in V_{att} \cup V_{def}$ be a path in the simulation game and $w_i = f(v_i)$ for each $i \in [1, l]$ and $w_{l+1} = S_\Diamond(w_0, ..., w_l)$. we take $S_d(v_0, ..., v_l) = g(w_0, .., w_{l+1})$ because by construction we have $(v_l, g(w_0, .., w_{l+1})) \in \delta$. Hence, if $S_\Diamond$ is the winner, then so is the defender.

Reciprocally, we take S_d a winning strategy of the defender in the simulation game and we build $S_\Diamond$, a winning strategy of $\Diamond$ in the Z-Reachability game. Let $w_0, ..., w_l \in V$ be a path in the Z-Reachability with the sum of weights of the edges traversed in the path $\{w_0, ..., w_l\}$ is superior to $(0, ..., 0)$. Considering $v_i = g(w_0, ..., w_i)$ for each $i \in [1, l]$ and $v_{l+1} = S_d(v_0, ..., v_l)$, we take $S_\Diamond(w_0, ..., w_l) = f(v_{l+1})$.

Hence we conclude that there is simulation between M and $\mathcal{R}^\otimes$ **iff** $\Diamond$ wins the Z-Reachability game. Since for k=2, this is decided in $O(|V|^{17})$ [17] and $|V| \leq norm(M)^{n_I(\mathcal{R})} \times |Q_M| \times |\Sigma_M|$, we conclude the result. □

We conclude in the next corollary the Exptime-completeness of $UCHS(M, \mathcal{R}, 2)$.

Corollary 20. *$UCHS(M, \mathcal{R}, 2)$ is Exptime-complete.*

Proof: First, $UCHS(M, \mathcal{R}, 2)$ is Exptime according to theorem 19. Second, it is harder then $UCHS(M, \mathcal{R}, 0)$ which is Exptime-hard (theorem 15). Hence, $UCHS(M, \mathcal{R}, 2)$ is Exptime-complete. □

Chapter 4

Conclusion

4.1 Summary

In this work, we investigated the complexity of web services composition problem. We have considered two parameters that are source of complexity of the web services composition problem. We have shown that among the considered problems, several instances remain Exptime-complete when one of these parameters is bounded.

When final states are considered, this problem is related to reachability issue where we raise the question of achieving a stable configuration in each execution path. This is mainly, as we see the problem, its source of difficulty. Hybrid states convey an obstacle as well, because of their non-decidable role in the execution: they are both intermediate states that can be recalled later and final states that define a definite execution's termination.

4.2 Perspectives

It remains an open question to identify the complexity of $UCHS(M, \mathcal{R}, k)$ for any $k \in \mathbb{N}$; [13] proves in the context of Z-Reachability that the problem is k-Exptime. This complexity is quite far from the known lower bound (Exptime). This may open some new tracks in defining upper bound complexity for a restricted k-dimensional Vector Addition Systems with States (VASS) classes.

It is also interesting to improve the polynomial complexity given for k=2 in [17] and/or give a simpler algorithm that can eventually be extended to the general case.

Left to say that the relation between this problem and the reachability issue convinced us that finding the complexity of the general problem requires a collaborative and long study of the issue.

Part III

Closure systems and related issues

Chapter 5

Introduction

5.1 Context

An efficient process to solve a problem has to begin with a modeling step. It is the phase where the question and all its considered constraints are put into a formal analyzable format, whether it is a plain text, a graph or a mathematical equation...or any other conceptual model.

The appearance and development of logic in the twentieth century, arising from the work of Gödel and Tarski [30, 32], gave a big boost to scientific research. It gave a formal platform of problem modeling that questions valid reasoning. Along with the electronic and computer science revolution, this took problem resolution process into a whole other level. It is however still not enough for certain problem classes that still demand an unpractical time of response. Hence the challenge of developing algorithms and getting a better understanding of mathematical structures in order to enhance the complexity of such problems or prove that it is not possible.

Partial orders [20] are one of the mathematical modelings that permeates our everyday life to such an extent that we take it for granted. Many important properties of an ordered set are related to the existence of certain upper bounds and lower bounds of its subsets. When such an upper bound and a lower bound exist for every pair of an order set, we are in the presence of a lattice. A popular example is natural numbers ordered by divisibility; the upper bound of two numbers is their least common multiple and the lower bound is their greatest common divisor.

Closure system is a natural modeling structure that has applications, to name a few, in topology, algebra and logic [11]. It is basically a subset family $\mathbb{F}$ on a set X verifying that X is in $\mathbb{F}$ and for every pair in $\mathbb{F}$ their intersection is in $\mathbb{F}$. Ordered by inclusion, every closure system forms a lattice. Reciprocally, every lattice

arises as a closure system on some set. Another representation of closure systems is closure operators [11], implicational bases [45] and binary relations [26]. The choice of representation is mainly dependent on the application and the problem considered (Database, Datamining, Game theory, Logic...).

5.2 Contributions and outline

Chapter 6 is an opening of this part, where we define closure systems and related modeling concepts: closure operator, lattice, implicational bases and binary relations. We establish the correspondence between these objects.

Chapter 7 investigates the problem of computing a closure system's maximal extension. We demonstrate how such an extension can be computed depending on the input closure system's encoding.

When the input is the implicit closure system's sets, we give a description of the maximal extension's closed sets.

When the input is an implicational base Σ, we give a new simple proof that erasing all unitary implications from Σ defines a base of the maximal extension.

When the input is lattice's join and meet-irreducible sets, equivalently a binary relation, then we give an incremental polynomial algorithm that computes the lattice's maximal extension.

Chapter 8 concerns the problem of candidate keys enumeration of an implicational base. We define a key-ideal set, an ideal of a candidate key in an order defined through Σ. We demonstrate that if we can enumerate a subset of key-ideal sets, that we call minimal key-ideal sets, in polynomial delay with a polynomial space, then we can enumerate candidate keys in polynomial delay and space as well. Knowing that the number of the latter can be significantly smaller than the number of candidate keys, we give an efficient key-ideal sets enumeration algorithm, that enumerates all minimal key-ideal sets in incremental polynomial time and then enumerates all non-minimal ones in a polynomial delay.

Chapter 6

Closure Systems and Representations

This chapter defines closure systems and some of its representations. We show how closure systems, closure operators, lattices, implicational bases and binary relations are all strongly related. The following definitions and notations will be considered all along the rest of the manuscript. Note also that **all considered objects are finite**.

6.1 Closure system

Closure system refers to a collection of subsets over a finite set X, closed by intersection and containing X. This mathematical structure appears in the litterature under different names, such as convexity space [15] and Moore family [11].

Definition 21. *Closure System*
$\mathbb{F}$ is called a closure system over a non-empty finite set X if and only if it is a subset of 2^X verifying:

- *X is in $\mathbb{F}$; and*
- *for every S, S' in $\mathbb{F}$, $S \cap S'$ is in $\mathbb{F}$.*

Each subset in $\mathbb{F}$ is called an $\mathbb{F}$-closed set, or simply closed set when $\mathbb{F}$ is understood from the context. The complementary of a closed set is called feasible set.

6.2 Closure operator

Definition 22. *Closure Operator*
A closure operator Φ on a set X is a function $\Phi : 2^X \mapsto 2^X$ verifying for every subset S of X:

- *$\Phi(S)$ contains S (extensive); and*
- *if S' is a subset of S, then $\Phi(S')$ is a subset of $\Phi(S)$ (increasing); and*
- *$\Phi(\Phi(S)) = \Phi(S)$ (idempotent).*

To highlight the strong correspondence between closure systems and closure operators, we describe next the classic bijection between both objects. On one hand, with every closure system $\mathbb{F}$, we associate the following closure operator:

$$\Phi_{\mathbb{F}} : 2^X \mapsto 2^X, \quad S \mapsto \Phi_{\mathbb{F}}(S) = \bigcap \{S' \in \mathbb{F} | S \subseteq S'\}$$

On the other hand, for every closure operator Φ, the corresponding closure system is the family of subsets $\mathbb{F} = \{\Phi(S)|S \subseteq X\}$.

6.3 Lattice

We begin by defining formally a partial order:

Definition 23. *Partial Order*
A binary relation $\leqslant$ on a set X is a partial order if it is:

- *reflexive: for every x in X, $x \leqslant x$; and*
- *antisymmetric: for x, y in X, if $x \leqslant y$ and $y \leqslant x$ then $x = y$; and*
- *transitive: for x, y, z in X, if $x \leqslant y$ and $y \leqslant z$ then $x \leqslant z$.*

Partially ordered set $P = (X, \leqslant)$ is a set X together with a partial order $\leqslant$.

When an order relation $\leq$ over a set X verifies that for every x and y in X, we have either $x \leq y$ or $y \leq x$, then $\leq$ is called a total order. We call the lexicographical order derived from $\leq$, the order defined over 2^X that converts subsets of S into increasingly ordered sequences and defines such sequence $(x_1, ..., x_i)$ of X lexicographically smaller than another sequence $(y_1, ..., y_j)$ of X if $(x_1, ..., x_i) \subseteq (y_1, ..., y_j)$ or $x_l \leq y_l$, for the first l where x_l and y_l differ.

Definition 24. *Upper Bound, Lower Bound*
Consider a partially ordered set $P = (X, \leqslant)$ and S a subset of X. The upper bound (i.e. supremum or join) of S in P is the least element in X, if exists, that is greater or equal to all elements of S. We define dually the lower bound (i.e. infimum, or meet) of S in P.

We denote $x_1 \vee ... \vee x_i$ (resp. $x_1 \wedge ... \wedge x_i$) or $\bigvee S$ (resp. $\bigwedge S$) an upper bound (resp. lower bound) of $S = \{x_1, ..., x_i\}$.

Definition 25. *Cover relation*
Consider a partially ordered set $P = (X, \leqslant)$ and two elements x, y in X, we say that x is covered by y and note $x \prec y$ if $x < y$ and for every z in X, $x < z \leqslant y$ implies that $y = z$.

Definition 26. *Ideal and filter*
Consider a partially ordered set $P = (X, \leqslant)$.

- *an ideal of P is a subset I of X such that if $x \in I$ and $y \leqslant x$ then $y \in I$. We denote, for every subset S of X, $\downarrow_{\leqslant} S$ the smallest ideal containing S, noted $\downarrow S$ when $\leqslant$is understood from the context. For every ideal I of P, $Max(I)$ is the set $\{x \in I \mid y \in I$ imply $x \not< y\}$.*

- *a filter of P is a subset F of X such that $x \in F$ and $x \leqslant y$ imply $y \in F$. We denote, for every subset S of X, $\uparrow_{\leqslant} S$ the smallest filter containing S, noted $\uparrow S$ when $\leqslant$is understood from the context.*

Definition 27. *Join-semilattice, Meet-semilattice, Lattice*

Consider a partially ordered set $\mathbb{L} = (X, \leqslant)$ with $X \neq \emptyset$. $\mathbb{L}$ is called a join-semilattice (resp. a meet-semilattice) if each two element-subset $\{x, y\}$ of X has a join (resp. a meet) in $\mathbb{L}$. $\mathbb{L}$ is a lattice if it is both a join-semilattice and a meet-semilattice.

A join-sublattice (resp. meet-sublattice) of $\mathbb{L}$ is a lattice $\mathbb{L}' = (X' \subseteq X, \leqslant')$ verifying for every pair of attributes in X', their join (resp. meet) in $\mathbb{L}$ is the same in $\mathbb{L}'$.

We recall the arrow relations introduced in [46]. Let $\mathbb{L} = (X, \leqslant)$ be a lattice and $x, y \in X$. Then:

1. $x \swarrow_{\mathbb{L}} y$ (noted $x \swarrow y$ when $\mathbb{L}$ is understood from the context), if and only if x is minimal in $\mathbb{L} \setminus \downarrow y$.

2. $x \nearrow_{\mathbb{L}} y$ (noted $x \nearrow y$ when $\mathbb{L}$ is understood from the context), if and only if y is maximal in $\mathbb{L} \setminus \uparrow x$.

3. $x \swarrow\!\!\!\nearrow_{\mathbb{L}} y$ (noted $x \swarrow\!\!\!\nearrow y$ when $\mathbb{L}$ is understood from the context), if and only if $x \swarrow_{\mathbb{L}} y$ and $x \nearrow_{\mathbb{L}} y$.

Definition 28. *Join-irreducible, Meet-irreducible*

Consider a lattice $\mathbb{L} = (X, \leqslant)$. An element j in X is called join-irreducible if for each x, y in X, $x \vee y = j$ implies either $x = j$ or $y = j$.

Dually, m in X is called meet-irreducible if for each x, y in X, $x \wedge y = m$ implies either $x = m$ or $y = m$.

$\mathbb{J}_{\mathbb{L}}$ (resp. $\mathbb{M}_{\mathbb{L}}$) refers to the set of all join-irreducibles (resp. meet-irreducibles) in $\mathbb{L}$.

Every finite lattice has one greatest element called the *top* and denoted $\top$ and one least element called *bottom* and denoted $\bot$. The elements covered by $\top$ are

called *co-atoms* and those that cover $\bot$ are called *atoms*. Note that every atom is join-irreducible and every co-atom is meet-irreducible.

Lattices and closure systems are two strongly connected concepts. In fact, for every closure system $\mathbb{F}$ on a set X, the partially ordered set $(\mathbb{F}, \subseteq)$ is a lattice.

In an abuse of notation, we may refer in this document to the lattice $(\mathbb{F}, \subseteq)$ by its corresponding closure system $\mathbb{F}$. $\mathbb{J}_{\mathbb{F}}$ and $\mathbb{M}_{\mathbb{F}}$ refer to join-irreducible sets and meet-irreducible sets of the lattice $(\mathbb{F}, \subseteq)$.

Reciprocally, a lattice $\mathbb{L} = (\mathbb{L}, \leqslant)$ is isomorphic to both closure systems defined over its join-irreducible sets and its meet-irreducible sets as follows:

- $\mathbb{F}_J(\mathbb{L}) = \{J(x), x \in \mathbb{L}\}$, where $J(x) = \{j \in \mathbb{J}_{\mathbb{L}} \mid j \leq x\}$.
- $\mathbb{F}_M(\mathbb{L}) = \{M(x), x \in \mathbb{L}\}$, where $M(x) = \{m \in \mathbb{M}_{\mathbb{L}} \mid x \leq m\}$.

When all join-irreducible sets of a lattice $\mathbb{L}$ are singletons, then $\mathbb{L}$ (and its corresponding closure system $\mathbb{F}_J(\mathbb{L})$) is called atomistic.

Definition 29. *Join-semidistributive and meet-semidistributive Lattice*

A lattice $\mathbb{L}$ is called join-semidistributive, if $x \vee y = x \vee z$ implies that $x \vee y = x \vee (y \wedge z)$ for every x, y and z in $\mathbb{L}$. Dually, we call $\mathbb{L}$ meet-semidistributive if $x \wedge y = x \wedge z$ implies that $x \wedge y = x \wedge (y \vee z)$.

6.4 Implicational base

Definition 30. *Implicational Base or Definite Horn CNF*

Let X be a finite set of elements. An implicational base Σ *over X is a set of pairs (L, R) in $2^X \times 2^X$. Each pair is called a* functional dependency *or an* implication *and denoted $L \to R$. L and R are respectively called premise and conclusion. If $|L| = 1$ (L is a singleton), then the implication $L \to R$ is called unitary.*

To an implicational base Σ, we associate a closure operator Φ (and consequently a closure system) where for every subset of elements S, $\Phi(S)$ is the smallest subset of elements S^Σ containing S and verifying for every $L \to R$ in Σ: if $L \subseteq S^\Sigma$ then $R \subseteq S^\Sigma$. We say that Σ satisfies a functional dependency $L \to R$ if and only if R is a subset of L^Σ.

A closure system, however, can correspond to multiple equivalent implicational bases. The number of dependencies in these bases differ and can be exponential in the size of the closure system's representation (for example meet-irreducible set).

Two bases are said equivalent, if they define the same closure system. A base Σ is called *minimum* when $|\Sigma| \leq |\Sigma'|$ for every Σ' equivalent to Σ.

Note that for x and y two different elements in X, if $x \to y$ and $y \to x$, then x and y are said equivalent. In such case, the corresponding lattices $(\mathbb{F}, \subseteq)$ and $(\{F \setminus \{x\} | F \in \mathbb{F}\}, \subseteq)$ are isomorphic. We suppose *w.l.o.g* that if $x \in \Phi(\{y\})$ then $y \notin \Phi(\{x\})$.

6.5 Binary relations and coloured posets

Definition 31. *Binary Relation*

A binary relation, called Formal Context in FCA terminology, over two sets X and M is a subset of the Cartesian product $X \times M$. We identify a binary relation $\mathbb{C}$ by the triplet (X, M, I), where X is an object set, M is an attribute set, and $I \subseteq X \times M$ is called incidence expressing the correspondence between objects and attributes.

A binary relation can also be seen as a binary matrix where every row (resp. column) corresponds to an object (resp. an attribute). It is a way of representing lattices, and therefore closure systems.

Given a binary relation $\mathbb{C} = (X, M, I)$, we define :

$$f : 2^X \to 2^M, f(S) = \{m \in M | \forall x \in S, (x, m) \in I\}$$

$$g : 2^M \to 2^X, g(S') = \{x \in X | \forall m \in S', (x, m) \in I\}$$

The composed functions $f \circ g$ and $f \circ g$ are in fact closure operators. We associate to $\mathbb{C}$ the concept lattice $(\{(S, S') | g(S) = S' \text{ and } f(S') = S,\ S \subseteq M,\ S' \subseteq X\}, \preccurlyeq)$, where the order $\preccurlyeq$ is formalized by: $(S_1, S_1') \preccurlyeq (S_2, S_2')$ if and only if $S_1 \subseteq S_2$.

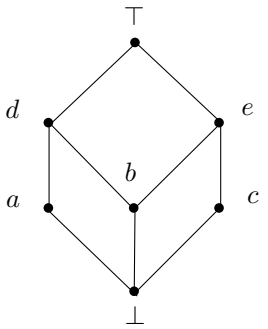
Reciprocally, a lattice $\mathbb{L} = (X, \leqslant)$ is isomorphic to the concept lattice of the binary relation $(\mathbb{J}_\mathbb{L}, \mathbb{M}_\mathbb{L}, \leqslant)$. Note that $\mathbb{L}$ can be constructed efficiently from the binary relation $(\mathbb{J}_\mathbb{L}, \mathbb{M}_\mathbb{L}, \leqslant)$ [39, 12, 24].

Another interpretation of such binary relations was given in [27], using coloured partially ordered sets (*c-poset*).

Definition 32. *C-poset*

A c-poset is a 4-tuple $(X, \leqslant, M, \gamma)$, where $(X, \leqslant)$ is a partially ordered set, M is a colours set and γ is a set colouring of the elements of X by subsets of M.

For a c-poset $\mathbb{P}_c = (X, \leqslant, M, \gamma)$ and any subset S of X, we define $\gamma(S) = \bigcup_{x \in S} \gamma(x)$. An ideal colour set of $\mathbb{P}_c$ is a set $I_c \subseteq M$ such that there is an ideal I of $(X, \leqslant)$ with $I_c = \gamma(I)$. Denote by $\mathbb{F}(\mathbb{P}_c)$ the set of all ideal colour sets of $\mathbb{P}_c$.



	a	c	d	e
a	1	0	1	0
b	0	0	1	1
c	0	1	0	1

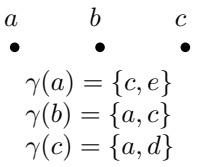


Figure 6.1: *A lattice example* $\mathbb{L}$ *and its associated binary relation* $Bip(\mathbb{L})$ *and c-poset* $\mathbb{P}_c(\mathbb{L})$.

For every $I_c = \gamma(I)$ and ${I_c}' = \gamma(I')$) in $\mathbb{F}(\mathbb{P}_c)$, $I_c \cup {I_c}' = \gamma(I \cup I')$ is in $\mathbb{F}(\mathbb{P}_c)$, since $I \cup I'$ is an ideal in $(X,\leqslant)$. Thus $\overline{\mathbb{F}}(\mathbb{P}_c) = \{M \setminus I_c \mid I_c \in \mathbb{F}(\mathbb{P}_c)\}$ is a closure system (i.e $\mathbb{F}(\mathbb{P}_c)$ is a co-closure system).

We associate with a lattice $\mathbb{L} = (X,\leqslant)$ the c-poset $(\mathbb{J}_\mathbb{L},\leqslant,\mathbb{M}_\mathbb{L},\gamma)$ where γ is a set colouring of the elements of $\mathbb{J}_\mathbb{L}$ by subsets of $\mathbb{M}_\mathbb{L}$ defined by:

$$\gamma : \mathbb{J}_\mathbb{L} \to 2^{\mathbb{M}_\mathbb{L}}, \quad \gamma(j) = \{m \in \mathbb{M}_\mathbb{L} \mid \quad j \swarrow m\}$$

In fact, the mapping φ from $\mathbb{L}$ to $(\overline{\mathbb{F}}(\mathbb{J}_\mathbb{L},\leqslant,\mathbb{M}_\mathbb{L},\gamma),\subseteq)$ is bijective:

$$\varphi : \mathbb{L} \to (\overline{\mathbb{F}}(\mathbb{J}_\mathbb{L},\leqslant,\mathbb{M}_\mathbb{L},\gamma),\subseteq), \text{ where } \varphi(a) = \{m \in \mathbb{M}_\mathbb{L}, a \not\leqslant m\} \tag{6.1}$$

Example 6 is an example of a closure system and its different representations.

Example 6. *We consider the closure system* $\mathbb{F} = \{\emptyset, \{a\}, \{b\}, \{c\}, \{a,b\}, \{b,c\}, \{a,b,c\}\}$. *Its closure operator is* Φ *defined over* $2^{\{a,b,c\}}$ *as:* $\Phi(\{a,c\}) = \{a,b,c\}$ *and* $\Phi(S) = S$ *for every other subset* S *of* $\{a,b,c\}$.

Figure 6.1 displays a lattice $\mathbb{L} = (\{\bot, a, b, c, d, e, \top\},\leqslant)$ *isomorphic to* $(\mathbb{F},\subseteq)$. *Its join and meet-irreducible sets are respectively* $\mathbb{J} = \{a,b,c\}$ *and* $\mathbb{M} = \{a,c,d,e\}$. $\mathbb{P}_c(\mathbb{L})$ *is the c-poset* $(\mathbb{J},\leqslant,\mathbb{M},\gamma)$ *that corresponds to* $\mathbb{L}$, *where* $(\mathbb{J},\leqslant)$ *is an antichain and* $\gamma(a) = \{c,e\}$, $\gamma(b) = \{a,c\}$ *and* $\gamma(c) = \{a,d\}$. *We also give in figure 6.1* $Bip(\mathbb{L})$ *the binary relation* $(\mathbb{J},\mathbb{M},\leqslant)$ *associated to* $\mathbb{L}$.

$\Sigma = \{ac \to b\}$ *is a minimum implicational base that corresponds to* $\mathbb{F}$.

Chapter 7

Closure Systems Extension

We give in this chapter an efficient algorithm for building the largest extension of a closure system. We show that the given algorithm gives additional insight on the structure of the largest extension by studying meet-irreducible sets transformation through the process.

7.1 Introduction

An extension of a closure system on a finite set X is a closure system on the same set X containing the given one as a join-sublattice.

Extension of closure systems has been the subject of several research work. Among them is [3] where Adaricheva, Gorbunov and Tumanov show that the largest meet-semidistributive extension of a meet-semidistributive lattice is in fact a convex geometry.

Adaricheva and Nation [2] have simplified the construction given in [3] and gave an exact formula for the largest extension of any closure system. It was however observed in [47] that the direct use of the characterization of the largest extension given in [2] leads to an exponential time algorithm for building the largest extension of a closure space, since one has to check a condition for every subset of X. This fact can hinder applications needing the manipulation of large closure systems such as those arising in knowledge space theory [21]. Thus the main motivation of this chapter is to give efficient algorithms for building the largest extension of a closure system.

We begin by pointing out the extension relation considered in this chapter:

Definition 33. *Let $\mathbb{F}$ and $\mathbb{F}'$ be two closure systems over a set of elements X and Φ and Φ' respectively their corresponding closure operators. We say that $\mathbb{F}'$ (resp.*

Φ') *is an extension of* $\mathbb{F}$ *(resp.* Φ*) if and only if* $\mathbb{F}'$ *contains* $\mathbb{F}$ *and for every family* $\{S_1, ..., S_k\}$ *of* Φ*-closed subsets of* X, $\Phi(S_1 \cup ... \cup S_k) = \Phi'(S_1 \cup ... \cup S_k)$.

In [25], it is proven that for every closure system $\mathbb{F}$, there exists a unique maximal extension, denoted $\mathbb{F}_{max}$, that contains any extension of $\mathbb{F}$.

We consider all along this chapter a closure operator Φ over a finite set X, $\mathbb{F}$ its corresponding closure system, $\mathbb{M}$ its meet-irreducible sets family and U_Φ the elements subset $\{x \in X | \Phi(x) \neq \{x\}\}$, i.e U_Φ corresponds to the elements in X that are not atoms in $(\mathbb{F}, \subseteq)$.

Let Σ_Φ be an implicational base corresponding to the closure operator Φ and verifying:

For every non-unitary implication $L \to R$ in Σ_Φ, L and R are Σ_p-closed, where $\Sigma_p = \{x \to \Phi(x) | x \in U_\Phi\}$.

We recall that a subset S of X is called Φ-closed when $\Phi(S) = S$. For an element $x \in X$, $\Phi^*(x)$ denotes the set $\Phi(x) \setminus \{x\}$. Notice that for x in U_Φ, $\Phi^*(x)$ is always Φ-closed since there is no equivalent elements in X. Moreover, $\Phi^*(x)$ is the only Φ-closed set covered by $\Phi(x)$.

Example 7. *Let us consider the closure system* $\mathbb{F} = \{\emptyset, \{a\}, \{b\}, \{ac\}, \{ad\}, \{abcd\}\}$ *(see figure 7.1.1). Let* Φ *be the closure operator corresponding to* $\mathbb{F}$*. Observe how for* $\Phi(\{c\}) = \{a, c\}$, $\Phi^*(\{c\}) = \{a\}$ *is the only* Φ*-closed set covered by* $\Phi(\{c\})$*. Similarly, we have for* $\Phi(\{d\}) = \{a, d\}$, $\Phi^*(\{d\}) = \{a\}$ *is the only* Φ*-closed set covered by* $\Phi(\{c\})$.

7.2 Maximal closure system extension

In this section, we give a description of a closure system's maximal extension. Our strategy is to drop at each step the implication $x \to \Phi(x)$ from Σ_Φ, where x is in U_Φ. We prove that this defines an extension of the original closure system. Then, once all unitary implications are dropped, we prove in theorem 35 that the resulting atomistic closure system is the maximal extension.

Theorem 34. *Let* Φ *be a closure operator over a finite set* X, $\mathbb{F}$ *be its corresponding closure system and* x *be an element in* U_Φ*. We consider that* $\Delta_x(\mathbb{F}) = \{\Delta_x(F) = F \cup \{x\} | F$ *is* Φ*-closed and* F *does not contain* $\Phi^*(x)\}$ *and that* $\mathbb{F}_x$ *denotes the subset* $\mathbb{F} \cup \Delta_x(\mathbb{F})$ *. Then:*

1. $\mathbb{F}_x$ *is a closure system. Let* Φ_x *be the corresponding closure operator.*

2. Φ_x *is an extension of* Φ.

3. Σ_{Φ_x} *is equivalent to* $\Sigma_\Phi \setminus \{x \to \Phi(x)\}$.

Proof:

1- Let F_1 and F_2 be two sets in $\mathbb{F}_x$, we prove that $F_1 \cap F_2$ is in $\mathbb{F}_x$. If F_1 and F_2 are Φ-closed, then $F_1 \cap F_2$ is in $\mathbb{F} \subseteq \mathbb{F}_x$. Else if none of them is Φ-closed, then $F_i = \Delta_x(E_i)$ where E_i is equal to $F_i \setminus \{x\}$ and is Φ-closed, for $i = 1, 2$. Hence, $F_1 \cap F_2 = \Delta_x(E_1 \cap E_2)$. Else if F_1 is Φ-closed but not F_2, then $E_2 = F_2 \setminus \{x\}$ is Φ-closed and either:

- $F_1 \cap F_2 = F_1 \cap E_2$, hence $F_1 \cap F_2$ is in $\mathbb{F}_x$; Or
- $F_1 \cap F_2 = (F_1 \cap E_2) \cup \{x\}$ and since $\Phi^*(x) \not\subseteq E_2$ (see definition of $\Delta_x()\mathbb{F}$) then $F_1 \cap E_2$ does not contain $\Phi^*(x)$ either. Hence, $\Delta_x(F_1 \cap E_2) = (F_1 \cap E_2) \cup \{x\} = F_1 \cap F_2$ is in $\mathbb{F}_x$.

2- We prove that for F_1 and F_2 two sets in $\mathbb{F}$, we have $\Phi_x(F_1 \cup F_2) = \Phi(F_1 \cup F_2)$. We recall that the closure of a subset of elements F is the smallest set in the closure system containing F. Since $\mathbb{F}$ is a subset of $\mathbb{F}_x$, this means that proving $\Phi_x(F_1 \cup F_2) = \Phi(F_1 \cup F_2)$ is equivalent to proving that $\Phi_x(F_1 \cup F_2)$ is Φ-closed.

- If $\Phi^*(x)$ is a subset of F_1 or F_2 (or both), then $\Phi^*(x)$ is a subset of $\Phi_x(F_1 \cup F_2)$. Therefore $\Phi_x(F_1 \cup F_2)$ is in $\mathbb{F}$ (see definition of Δ_x). Else
- If $\Phi^*(x)$ is not a subset of F_1 and F_2, then F_1 and F_2 do not contain x. We suppose that $F = \Phi_x(F_1 \cup F_2)$ is not Φ-closed. Hence, $F \setminus \{x\}$ is in $\mathbb{F} \subseteq \mathbb{F}_x$ and contains $F_1 \cup F_2$. This implies that $\Phi_x(F_1 \cup F_2) \subseteq F \setminus \{x\}$ (by definition of a closure operator). However this contradicts the fact that $F \setminus \{x\} \subset F = \Phi_x(F_1 \cup F_2)$. Hence, $F = \Phi_x(F_1 \cup F_2)$ is Φ-closed.

3- We prove that $\mathbb{F}_x$ corresponds to the closure system of $\Sigma_{\Phi_x} = \Sigma_\Phi \setminus \{x \to \Phi(x)\}$. First, let F be in $\mathbb{F}_x$. We prove next that F is Σ_{Φ_x}-closed. We have:

- either F is Σ_Φ-closed. And since Σ_Φ contains Σ_{Φ_x} then F is also Σ_{Φ_x}-closed;
- or $F = F' \cup \{x\}$ where F' is in $\mathbb{F}$ and $\Phi^*(x) \not\subseteq F'$. Let $L \to R$ be in Σ_{Φ_x} verifying $L \subseteq F$. To prove that R is a subset of F, we distinguish two cases:
  - $x \notin L$: then we have $L \subseteq F \setminus \{x\} = F'$. And since F' is Σ_Φ-closed then R is a subset of F' and therefore of $R \subseteq F$. Thus, F is Σ_{Φ_x}-closed

– $x \in L$: since $L \to R$ is in $\Sigma_{\Phi_x} = \Sigma_\Phi \setminus \{x \to \Phi(x)\}$ and $x \in L$, then $L \neq \{x\}$. Therefore $L \to R$ is non-unitary implication. We recall that for every non unitary implication $L' \to R'$ in Σ_Φ, L' and R' are supposed to be Σ_p-closed where Σ_p is the set of unitary implications in Σ_Φ. This implies that L is closed under unitary implications Σ_p of Σ_Φ, in particular $x \to \Phi^*(x)$. Thus, $\Phi^*(x)$ is a subset of L, which contradicts the fact that $\Phi^*(x) \nsubseteq F$ and $L \subseteq F$.

Second, let F be a Σ_{Φ_x}-closed subset of X. We prove that F is either in $\mathbb{F}$ or in $\Delta_x(\mathbb{F})$. We distinguish the following cases:

- $x \notin F$: In this case, for every $L \to R$ in Σ_Φ verifying $L \subseteq F$, x is not in L. Hence, $L \to R$ is in Σ_{Φ_x} and therefore R is a subset of F. This means that F is Σ_Φ-closed.

- $\Phi^*(x) \subseteq F$ and $x \in F$: Let $L \to R$ be in Σ_Φ verifying $L \subseteq F$. If $L = \{x\}$, then $R \subseteq \Phi^*(x)$ is a subset of F. Else, $L \to R$ is in Σ_{Φ_x}. Since F is Σ_{Φ_x}-closed, then R is a subset of F. This means that F is in $\mathbb{F}$.

- $\Phi^*(x) \nsubseteq F$ and $x \in F$: In this case we prove that F is in $\Delta_x(\mathbb{F})$. To do so, we consider $F' = F \setminus \{x\}$ and prove that $F = \Delta_x(F')$, equivalent to F' is in $\mathbb{F}$ and $\Phi^*(x) \nsubseteq F'$.

  Let $L \to R$ be in Σ_Φ such that L is a subset of F'. Since $L \subseteq F'$ and $x \notin F'$, then x is not in L. This implies that $L \to R$ is in Σ_{Φ_x}. And since $L \subseteq F' \subseteq F$ and F is Σ_{Φ_x}-closed, then R is a subset of F.Moreover, we recall that for every non unitary implication $L' \to R'$ in Σ_Φ, L' and R' are supposed to be closed under unitary implications set Σ_p of Σ_Φ. Hence, R is Σ_p-closed. This implies that if $x \in R$ then $\Phi^*(x) \subseteq R \subseteq F$ and this contradicts the fact that $\Phi^*(x)$ is not a subset of F. Therefore, we conclude by contradiction that $x \notin R$. Thus, we have $R \subseteq F \setminus \{x\}$, i.e $R \subseteq F'$. To sum up, for every $L \to R$ in Σ_Φ verifying $L \subseteq F'$, we have $R \subseteq F'$.Thus, F' is in $\mathbb{F}$ and does not contain $\Phi^*(x)$. Hence $F = F' \cup \{x\}$ is in $\Delta_x(\mathbb{F})$ (by definition of Δ_x).

We conclude that F is in $\mathbb{F}_x$ if and only if it is Σ_{Φ_x}-closed. □

Let $U_\Phi = \{x_1, ..., x_k\}$. We define the closure operator $\Phi_{x_1,...,x_k}$ corresponding to $\Sigma_{\Phi_{x_1,...,x_k}} = \Sigma_\Phi \setminus \Sigma_p$ where we drop every unitary functional dependency $x_i \to \Phi(x_i)$ from Σ_Φ, at each step shown in the previous theorem. Note that for every x, x' in U_Φ, we have $\Phi_x(x') = \Phi(x')$, since unitary implications where $L = \{x'\}$ are the same in Σ_{Φ_x} as in Σ_Φ.

Let $\mathbb{F}_{x_1,...,x_k}$ be the closure system defined by $\Phi_{x_1,...,x_k}$. Since $\mathbb{F}_{x_1,...,x_k}$ is atomistic, then we could use the results of [25] in order to prove that $\Phi_{x_1,...,x_k}$ corresponds to the maximal extension. However, we choose to give a full proof in the next theorem.

Theorem 35. *Let Φ be a closure operator over a finite set X, $\mathbb{F}$ be its corresponding closure system and $U_\Phi = \{x_1, ..., x_k\}$. Then $\mathbb{F}_{x_1,...,x_k}$ is the maximal extension $\mathbb{F}_{max}$ of $\mathbb{F}$.*

Proof: First, note that a closure systems extension is a transitive relation: an extension of an extension of $\mathbb{F}$ is an extension of $\mathbb{F}$. Hence, we can deduce from theorem 34 that $\mathbb{F}_{x_1,...,x_k}$ is an extension of $\mathbb{F}$. Thus, $\mathbb{F}_{max}$ contains $\mathbb{F}_{x_1,...,x_k}$, since it is the unique maximal extension of $\mathbb{F}$. Second, we prove that $\mathbb{F}_{max}$ is a subset of

$\mathbb{F}_{x_1,...,x_k}$.

Let us suppose that $\mathbb{F}_{max}$ is not a subset of $\mathbb{F}_{x_1,...,x_k}$ and let F be in $\mathbb{F}_{max}$ and not in $\mathbb{F}_{x_1,...,x_k}$. Then, there exists $L \to R$ in $\Sigma_{\Phi_{x_1,...,x_k}}$ verifying that $L \subseteq F$ and $R \nsubseteq F$. Thus, $L \to R$ is not satisfied by $\Sigma_{\Phi_{max}}$. We prove that $\Phi(L) \neq \Phi_{max}(L)$. Since $L \to R$ is in $\Sigma_{\Phi_{x_1,...,x_k}} \subseteq \Sigma_\Phi$, then $R \subseteq \Phi(L)$. However, since $\Sigma_{\Phi_{max}}$ does not satisfy $L \to R$, then $R \nsubseteq \Phi_{max}(L)$. We conclude that $\Phi(L) \neq \Phi_{max}(L)$.

We prove next that L is the union of Σ_Φ-closed sets: The fact that $L \to R$ is in $\Sigma_{\Phi_{x_1,...,x_k}}$ implies that $L \to R$ is in Σ_Φ and that $L \to R$ is non unitary (by definition of $\Sigma_{\Phi_{x_1,...,x_k}}$). And since for every non unitary implication in Σ_Φ, its left side is Σ_p-closed, then $L = A_1 \cup ... \cup A_k$ where $A_i = \Phi(a_i)$, for $a_i \in L$.

From all above, we conclude that there exists a family $\mathcal{S} = \{A_1, ..., A_k\}$ of Σ_Φ-closed sets verifying $\Phi(A_1 \cup ... \cup A_k) \neq \Phi_{max}(A_1 \cup ... \cup A_k)$. This contradicts the fact that Φ_{max} is an extension of Φ. Hence, $\mathbb{F}_{max}$ is indeed a subset of $\mathbb{F}_{x_1,...,x_k}$. We conclude from all above that $\mathbb{F}_{max} = \mathbb{F}_{x_1,...,x_k}$. □

Example 8. *Figure 7.1 shows an example of a closure system's maximal extension. We consider $\mathbb{F} = \{\emptyset, a, b, ac, ad, abcd\}$ illustrated in figure 7.1.1.*

Figure 7.1.2 and figure 7.1.3 show the two steps of theorem 34. The resulting closure system in figure 7.1.4 is the maximal extension, according to theorem 35.

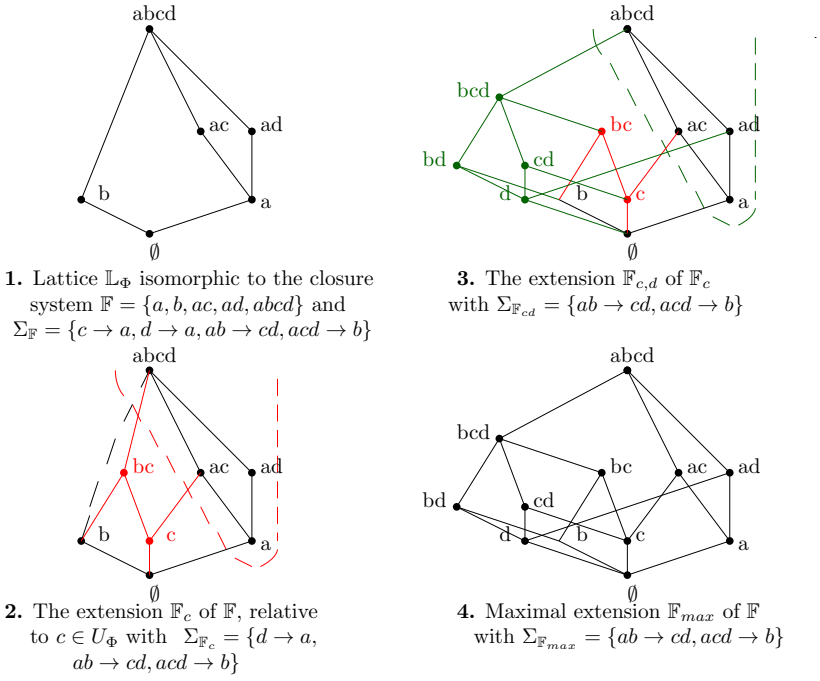


Figure 7.1: *Application of theorem 34 to complete a closure system's maximal extension.*

7.3 Meet-irreducibles of maximal closure system extension

For every x in U_Φ, the construction process of Φ_x's closure system, defined in theorem 34, implies that Φ and Φ_x have the same number of join-irreducible sets. We can not however say the same thing about meet-irreducible sets. We give in this section an upper and lower bound of $|\mathbb{M}_x|$, the number of meet-irreducible sets in $\mathbb{F}_x$.

Let $\mathbb{L}$be the lattice $(\mathbb{F}, \subseteq)$. We define a partition $(\mathbb{M}_1, \mathbb{M}_2, \mathbb{M}_3, \mathbb{M}_4, \mathbb{M}_5)$ of $\mathbb{M}$ (figure 7.2) where:

- $\mathbb{M}_1(x)$ are meet-irreducible sets that contain x;
- $\mathbb{M}_2(x)$ are meet-irreducible sets that contain $\Phi^*(x)$ but not x;
- $\mathbb{M}_3(x)$ is the set $\{F \in \mathbb{M}$ such that $\Phi^*(x) \nearrow_\mathbb{L} F$ and $\Phi(x) \nearrow_\mathbb{L} F\}$;
- $\mathbb{M}_4(x)$ is the set $\{F \in \mathbb{M}$ such that $\Phi^*(x) \nearrow_\mathbb{L} F$ and $\Phi(x) \not\nearrow_\mathbb{L} F\}$;

- $\mathbb{M}_5(x)$ are meet-irreducible sets that do not contain $\Phi^*(x)$ and are not in $M_3(x) \cup M_4(x)$.

We refer to $\mathbb{M}_i(x)$ by $\mathbb{M}_i$, when x is understood from the context.

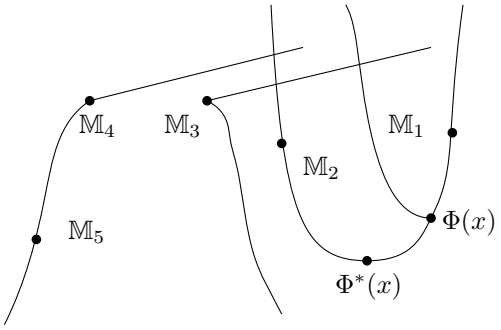


Figure 7.2: *Partition of meet-irreducible sets of Φ relative to $x \in U_\Phi$*

The following properties describe meet-irreducible sets of the extension Φ_x as defined in theorem 34.

Property 5. *Let F be meet-irreducible in $\mathbb{F}$, then:*

1. *If F is in $\mathbb{M}_1 \cup \mathbb{M}_2$, then F is meet-irreducible in $\mathbb{F}_x$.*
2. *If F is in $\mathbb{M}_4 \cup \mathbb{M}_5$, then $\Delta_x(F)$ is meet-irreducible in $\mathbb{F}_x$ and F is not meet-irreducible in $\mathbb{F}_x$.*
3. *If F is in $\mathbb{M}_3$, then F and $\Delta_x(F)$ are meet-irreducible in $\mathbb{F}_x$. In addition, $\{x\} \swarrow\!\!\nearrow_{\Phi_x} F$.*

Proof:

1- F is in $\mathbb{M}_1 \cup \mathbb{M}_2$ implies that F contains $\Phi^*(x)$. This implies that all covers of F in $\mathbb{F}_x$ contain $\Phi^*(x)$ and therefore they are Φ-closed. We conclude that F is meet-irreducible in $\mathbb{F}_x$.

2- For F in $\mathcal{M}_4$ or $\mathcal{M}_5$, F has a unique cover F' in $\mathbb{F}$ that does not contain x. In $\mathbb{F}$, F is also covered by $\Delta_x(F)$ and $\Delta_x(F) \nsubseteq F'$ and $F \subseteq F'$. We conclude that F is not meet-irreducible in $\mathbb{F}_x$.

3- Let F be in $\mathbb{M}_3$. First, we point out that, by definition of $\mathbb{F}_x$, F does not contain $\Phi^*(x)$. Since $\Phi(x) \nearrow F$ then for every closed set F' in $\mathbb{F}$, F' contains F implies that $x \in F'$, and therefore $\Delta_x(F) \subseteq F'$. Hence, F is meet-irreducible in $\mathbb{F}_x$ and $\Delta_x(F)$ is its unique cover. We prove next that $\Delta_x(F)$ is meet-irreducible. We suppose that $\Delta_x(F) = F_1 \cap F_2$, with F_1 and F_2 in $\mathbb{F}_x$. If F_1 and F_2 are both Φ-closed, then $F_1 \cap F_2$ is also Φ-closed. However, $\Delta_x(F) = F_1 \cap F_2$ is not Φ-closed. Hence the contradiction.

Else, either F_1 or F_2 (or both) are in $\Delta_x(\mathbb{F})$. Let F_1 be in $\Delta_x(\mathbb{F})$. Hence, $F_1 \setminus \{x\}$ is in $\mathbb{F}$, does not contain $\Phi^*(x)$ and contains F. This contradicts the fact that $\Phi^*(x) \nearrow F$.

We conclude that $\Delta_x(F)$ is meet-irreducible.

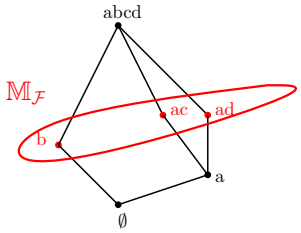
In addition, $\{x\}$ is an atom and $x \notin F$. Hence, $\{x\} \swarrow F$. Moreover, F is meet-irreducible in $\mathbb{F}_x$ and its only cover $\Delta_x(F)$ contains x. Therefore, $\{x\} \nearrow F$. We conclude that $\{x\} \swarrow\!\!\!\nearrow F$. $\square$

Example 9. *In figure 7.3, we reconsider the closure system in example 8* $\mathbb{F} = \{\emptyset, \{a\}, \{b\}, \{a, c\}, \{a, d\}, \{a, b, c, d\}\}$ *and its extension* $\mathbb{F}_c = \{\emptyset, \{a\}, \{b\}, \{c\}, \{a, c\}, \{a, d\}, \{b, c\}, \{a, b, c, d\}\}$ *with* $c \to a \in \Sigma_{\mathbb{F}}$.

Figure 7.3.1 distinguishes meet-irreducible sets of $\mathbb{F}$; $\mathbb{M}_{\mathbb{F}} = \{\{b\}, \{a, c\}, \{a, d\}\}$. *Following the decomposition of* $\mathbb{M}_{\mathbb{F}}$ *defined in 7.2 for* $x = c$, *we have (column 1 and 2 in figure 7.3.2):*

- $\{a, c\} \in \mathbb{M}_1$ *because* $c \in \{a, c\}$;
- $\{a, d\} \in \mathbb{M}_2$ *because* $\phi^*(c) = a \in \{a, c\}$ *and* $c \notin \{a, d\}$;
- $\{b\} \in \mathbb{M}_3$ *because* $\Phi^*(c) \nearrow_{\mathbb{L}} \{b\}$ *and* $\Phi(c) \nearrow_{\mathbb{L}} \{b\}$.

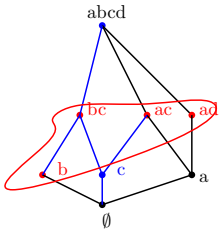
Property 5 dictates then that $\{a, c\}$ *and* $\{a, d\}$ *stay meet-irreducible sets in the extension* $\mathbb{F}_c$ *and both* $\{b\}$ *and its image* $\Delta_c(\{b\}) = \{b, c\}$ *are meet-irreducible sets in* $\mathbb{F}_c$ *(figure 7.3.3).*



1. The closure system $\mathbb{F} = \{\emptyset, \{a\}, \{b\}, \{a,c\}, \{a,d\}, \{a,b,c,d\}\}$ with $c \to a \in \Sigma_{\mathbb{F}}$

$F \in \mathbb{M}_{\mathcal{F}}$	$\mathbb{M}_i, x = c$	$F \in \mathbb{M}_{\mathcal{F}_c}$?	$F \cup \{c\} \in \mathbb{M}_{\mathcal{F}_c}$?
$\{b\}$	$\mathbb{M}_3$	yes	yes
$\{a,c\}$	$\mathbb{M}_1$	yes	—
$\{a,d\}$	$\mathbb{M}_2$	yes	—

2. Table of the correspondence between meet-irreducible sets of $\mathcal{F}$ and meet irreducible sets of $\mathcal{F}_c$



3. $\mathbb{F}$'s extension $\mathbb{F}_c = \{\emptyset, \{a\}, \{b\}, \{c\}, \{a,c\}, \{a,d\}, \{b,c\}, \{a,b,c,d\}\}$ with $\Sigma_{\mathbb{F}_c} = \Sigma_{\mathbb{F}} \setminus \{c \to a\}$

Figure 7.3: *Example of the correspondence between meet-irreducible sets of a closure system $\mathbb{F}$ and meet-irreducible sets of its extension $\mathbb{F}_x$.*

Property 5 gives a lower bound of the number of meet irreducible sets in $\mathbb{F}_x$ in corollary 36.

Corollary 36. *The number of meet irreducible sets in $\mathbb{F}_x$ is at least $|\mathbb{M}| + |\mathbb{M}_3|$.*

Proof: This is a direct result of properties 5. In fact, for every meet-irreducible F in $\mathbb{F}$, either F or $\Delta_x(F)$ is in $\mathbb{M}_x$ or both (for $F \in \mathbb{M}_3$). □

Notice that if $S \in \Delta_x(\mathbb{F})$ is a new meet-irreducible in $\mathbb{F}_x$, then $S \setminus \{x\}$ has at most a unique cover in $\mathbb{F} \setminus (\uparrow \Phi^*(x))$. This does not mean necessarily that S is meet-irreducible in $\mathbb{F}$.

Let $\mathbb{M}_6$ be the set $\{F \in \mathbb{F} \setminus (\uparrow \Phi^*(x))$ such that F has exactly one cover in $\mathbb{F} \setminus (\uparrow \Phi^*(x))$ and F is not in $\mathbb{M}\}$.

Theorem 37. *Let F be meet-irreducible in $\mathbb{F}_x$. we have either:*

1. *F is in $\mathbb{M}_1 \cup \mathbb{M}_2 \cup \mathbb{M}_3$; Or*
2. *F is not Φ-closed and $F \setminus \{x\}$ is in $\mathbb{M}_3 \cup \mathbb{M}_4 \cup \mathbb{M}_5 \cup \mathbb{M}_6$.*

Proof: Let F be meet-irreducible in $\mathbb{F}_x$. We distinguish 3 cases:

If F is in $\uparrow \Phi^*(x)$: In this case, F has exactly the same covers in $\mathbb{F}$ and in $\mathbb{F}_x$. Hence, if F is meet-irreducible in $\mathbb{F}_x$ then F is meet-irreducible in $\mathbb{F}$, implying that F is in $\mathbb{M}_1 \cup \mathbb{M}_2$.

If F is in $\mathbb{F} \setminus \uparrow \Phi^*(x)$: In this case, F is covered by $\Delta_x(F)$. Hence, if F is meet-irreducible in $\mathbb{F}_x$ then all covers of F in $\mathbb{F}$ contain x, implying that F is in $\mathbb{M}_3$.

If $F = \Delta_x(F')$: In this case, F is covered by $\Delta_x(F'')$, for every F'' that covers F' in $\mathbb{F}\setminus \uparrow \Phi^*(x)$. Hence, if F is meet-irreducible in $\mathbb{F}_x$ then F has at most one cover in $\mathbb{F}\setminus \uparrow \Phi^*(x)$, implying that F is in $\mathbb{M}_3 \cup \mathbb{M}_4 \cup \mathbb{M}_5 \cup \mathbb{M}_6$. $\square$

Next, we give properties for an element in $\mathbb{M}_6$ which satisfies that $\Delta_x(F)$ is meet-irreducible in $\mathbb{F}_x$. The next property helps in giving an upper bound of the number of meet-irreducible sets in $\mathbb{F}_x$, by describing them as the join of meet-irreducible sets of $\mathbb{F}$.

Property 6. *For every F in $\mathbb{M}_6$, if $\Delta_x(F)$ is meet-irreducible in $\mathbb{F}_x$, then there exists $F' \in \mathbb{M}_3 \cup \mathbb{M}_4$ and $F'' \in \mathbb{M}_1 \cup \mathbb{M}_2$ such that $F = F' \cap F''$.*

Proof: Let F_1 be the unique cover of F in $\mathbb{F} \setminus (\uparrow \Phi^*(x))$ and $F_2 = F \vee \Phi^*(x)$ be the cover of F in $\uparrow \Phi^*(x)$.

First, since $\Phi^*(x) \nsubseteq F_1$ then there exists F' in $\mathbb{M}_3 \cup \mathbb{M}_4$ that contains F_1.

Second, considering that F_2 is not a subset of F_1 and it contains $\Phi^*(x)$, then there exists F'' in $\mathbb{M}_1 \cup \mathbb{M}_2$ such that F'' contains F_2 and $F_1 \nsubseteq F''$.

To sum up, F' is in $\mathbb{M}_3 \cup \mathbb{M}_4$ and F'' is in $\mathbb{M}_1 \cup \mathbb{M}_2$ and $F' \cap F''$ is in $\mathbb{F}$ and it contains F. However, we have: F' does not contain $\Phi^*(x) \subseteq F_2$, hence $F_2 \nsubseteq F' \cap F''$. And F'' does not contain F_1, hence $F_1 \nsubseteq F' \cap F''$. Therefore, $F \subseteq F' \cap F''$ implies that $F = F' \cap F''$. $\square$

The following property gives a necessary and sufficient condition that helps decide if a subset M in $\mathbb{M}_6$ correspond to a meet-irreducible sets in $\mathbb{F}_x$ or not.

Property 7. *Let F be in $\mathbb{M}_6$. $\Delta_x(F)$ is meet-irreducible in $\mathbb{F}_x$ if and only if $\Phi(F \cup \{x\}) = \Phi(F' \cup \{x\})$, where F' is the cover of F in $\mathbb{F}\setminus \uparrow \Phi^*(x)$.*

Proof: First of all, we prove the following proposition:

$$\textit{For } F \textit{ in } \mathbb{F}\textit{, if } F \textit{ does not contain } \Phi^*(x)$$

$$\textit{then } \Phi(F \cup \{x\}) \textit{ is equal to } \Phi_x(\Delta_x(F) \cup \Phi(x)). \tag{7.1}$$

Notice that $\Phi(F \cup \{x\}) = \Phi(F \cup \Phi(x))$.

We have $\Delta_x(F) = F \cup \{x\}$ and $x \in \Phi(x)$. Hence, $F \cup \Phi(x) = \Delta_x(F) \cup \Phi(x)$. Therefore, $\Phi_x(F \cup \Phi(x)) = \Phi_x(\Delta_x(F) \cup \Phi(x))$. And since Φ_x is an extension of Φ and F and $\Phi(x)$ are Φ-closed, then $\Phi_x(F \cup \Phi(x)) = \Phi(F \cup \Phi(x))$.

We conclude that $\Phi(F \cup \{x\}) = \Phi(F \cup \Phi(x)) = \Phi_x(\Delta_x(F) \cup \Phi(x))$.

Let F be in $\mathbb{M}_6$ and F' be its unique cover in $\mathbb{F}\setminus \uparrow \Phi^*(x)$. First, We prove that: $\Delta_x(F)$ is meet-irreducible $\Rightarrow \Phi(F \cup \{x\}) = \Phi(F' \cup \{x\})$.

If $\Delta_x(F)$ is meet-irreducible then $\Delta_x(F')$ is its unique cover. This implies that every Φ_x-closed set that contains $\Delta_x(F)$, contains also $\Delta_x(F')$. In particular, $\Delta_x(F) \subseteq \Phi_x(\Delta_x(F) \cup \Phi(x))$ implies that $\Delta_x(F') \subseteq \Phi_x(\Delta_x(F) \cup \Phi(x))$. Hence, $\Phi_x(\Delta_x(F') \cup \Phi(x)) \subseteq \Phi_x(\Delta_x(F) \cup \Phi(x))$. Moreover, we have $\Delta_x(F) \subseteq \Delta_x(F')$ and therefore $\Phi_x(\Delta_x(F) \cup \Phi(x)) \subseteq \Phi_x(\Delta_x(F') \cup \Phi(x))$. We conclude that $\Phi_x(\Delta_x(F) \cup \Phi(x)) = \Phi_x(\Delta_x(F') \cup \Phi(x))$. And according to 7.1, we have $\Phi_x(\Delta_x(F) \cup \Phi(x)) = \Phi(F \cup \{x\})$ and $\Phi_x(\Delta_x(F') \cup \Phi(x)) = \Phi(F' \cup \{x\})$.

Thus, we have $\Phi(F \cup \{x\}) = \Phi(F' \cup \{x\})$. Second, we prove that $\Delta_x(F)$ is

meet-irreducible in $\mathbb{F}_x \Leftarrow \Phi(F \cup \{x\}) = \Phi(F' \cup \{x\})$.

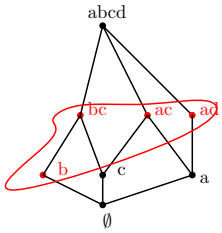
We suppose that $\Phi(F \cup \{x\}) = \Phi(F' \cup \{x\})$. This implies that $\Phi_x(\Delta_x(F) \cup \Phi(x)) = \Phi_x(\Delta_x(F') \cup \Phi(x))$ (according to 7.1). Thus, since $\Delta_x(F') \subseteq \Phi_x(\Delta_x(F') \cup \Phi(x))$ then $\Delta_x(F')$ is a subset of $\Phi_x(\Delta_x(F) \cup \Phi(x))$.

Since $\Phi(x) = \{x\} \cup \Phi^*(x)$ and $x \in \Delta_x(F)$, then $\Phi_x(\Delta_x(F) \cup \Phi(x)) = \Phi_x(\Delta_x(F) \cup \Phi^*(x))$. Therefore $\Delta_x(F')$ is a subset of $\Phi_x(\Delta_x(F) \cup \Phi^*(x))$.

To sum up, $\Delta_x(F')$ is the only cover of $\Delta_x(F)$ in $\mathbb{F}_x \setminus \uparrow \Phi^*(x)$ and $\Delta_x(F)$ is a subset of $\Phi_x(\Delta_x(F) \cup \Phi^*(x))$. We conclude that $\Delta_x(F')$ is the only cover of $\Delta_x(F)$ in $\mathbb{F}_x$. □

Example 10. *Let us consider in figure 7.4, the closure system* $\mathbb{F}_c = \{\emptyset, \{a\}, \{b\}, \{c\}, \{a, c\}, \{a, d\}, \{b, c\}, \{a, b, c, d\}\}$ *from the previous example 7.3 and its extension* $\mathbb{F}_{cd} = \{\emptyset, \{a\}, \{b\}, \{c\}, \{d\}\ \{a, c\}, \{a, d\}, \{b, c\}, \{b, d\},\ \{b, c, d\}, \{a, b, c, d\}\}$, *with* $d \to a \in \Sigma_{\mathbb{F}_c}$.

Figure 7.4.1 distinguishes meet-irreducible sets of $\mathbb{F}_c$; $\mathbb{M}_{\mathbb{F}_c} = \{\{b\},\ \{a, c\}, \{a, d\}, \{b, c\}\}$. *Figure 7.4.2 depicts the correspondence between meet irreducible sets of* $\mathbb{F}_c$ *and* $\mathbb{F}_{cd}$. *Then, table in figure 7.4.3 distinguishes closed sets in* $\mathbb{M}_6$ *using the property 6. In this case* $\mathbb{M}_6 = \{\{c\}\}$. *Then, we use property 7 to verify that* $\Delta_d(\{c\})$ *is meet-irreducible in* $\mathbb{F}_{cd}$: *since* $\Phi(\{c\} \cup \{d\}) = \{a, b, c, d\}$ *is equal to* $\Phi(\{b, c\} \cup \{d\}) = \{a, b, c, d\}$ *where* $\{b, c\}$ *is* $\{c\}$*'s cover that does not contain* $\Phi^*(\{d\})$, *then* $\Delta_d(\{c\}) = \{c, d\}$ *is meet-irreducible in* $\mathbb{F}_{cd}$.



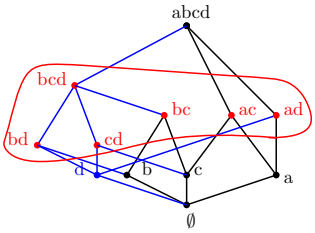
1. Lattice of the closure system $\mathbb{F}_c = \{\emptyset, \{a\}, \{b\}, \{c\}, \{a,c\}, \{a,d\}, \{b,c\}, \{a,b,c,d\}\}$ with $d \to a \in \Sigma_{\mathbb{F}_c}$

$F \in \mathbb{M}_{\mathcal{F}_c}$	$\mathbb{M}_i, x = d$	$F \in \mathbb{M}_{\mathcal{F}_{cd}}$?	$F \cup \{c\} \in \mathbb{M}_{\mathcal{F}_{cd}}$?
$\{b\}$	$\mathbb{M}_5$	yes	no
$\{a,c\}$	$\mathbb{M}_2$	yes	—
$\{a,d\}$	$\mathbb{M}_1$	yes	—
$\{b,c\}$	$\mathbb{M}_3$	yes	yes

2. Table of the correspondence between meet-irreducible sets of $\mathcal{F}_c$ and meet irreducible sets of $\mathcal{F}_{cd}$

$F \in \mathbb{M}_1 \cup \mathbb{M}_2$	$F' \in \mathbb{M}_3 \cup \mathbb{M}_4$	$F \wedge F' \in \mathbb{M}_6$	$F \wedge F' \in \mathbb{M}_{\mathcal{F}_{cd}}$?
$\{a,c\}$	$\{b,c\}$	$\{c\}$	yes
$\{a,d\}$	$\{b,c\}$	—	no

3. Table of meet-irreducible sets of $\mathcal{F}_{cd}$ that do not correspond to meet-irreducible sets of $\mathcal{F}_c$.



4. Lattice of $\mathbb{F}_c$'s extension $\mathbb{F}_{cd} = \{\emptyset, \{a\}, \{b\}, \{c\}, \{d\}, \{a,c\}, \{a,d\}, \{b,c\}, \{c,d\}, \{b,d\}, \{b,c,d\}, \{a,b,c,d\}\}$ with $\Sigma_{\mathbb{F}_{cd}} = \Sigma_{\mathbb{F}_c} \setminus \{d \to a\}$

Figure 7.4: *Example of meet-irreducible sets of a closure system and its extension.*

From theorem 37 and property 6, we give in the next corollary an upper bound of the number of meet irreducible sets in $\mathbb{F}_x$.

Corollary 38. *The number of meet irreducible sets in* $\mathbb{F}_x$ *is bounded by* $|\mathbb{M}| + |\mathbb{M}_3| + (|\mathbb{M}_1| + |\mathbb{M}_2|) \times (|\mathbb{M}_3| + |\mathbb{M}_4|)$.

7.4 Computation of the maximal extension

Given the set of meet-irreducible sets $\mathbb{M}$ of a closure system $\mathbb{F}$ over a set X, we propose an algorithm to compute the set of meet-irreducible sets $\mathbb{M}_{max}$ of $\mathbb{F}_{max}$ the maximal extension of $\mathbb{F}$.

At each step, the algorithm takes an element $x \in X$ verifying that its closure in $\mathbb{F}$ is different than $\{x\}$ and apply theorem 34 to compute $\mathbb{M}_x$ the meet-irreducible sets of the extension $\mathbb{F}_x$, based on $\mathbb{M}$.

EXTENSION calls a subroutine, named CLOSURE, to determine the smallest closed set containing an element S, for S a subset of X. This function calculates basically the intersection of all meet-irreducible sets that contain S.

The function EXPAND generates the extension corresponding to an element x, of a closure operator defined by its meet-irreducible sets $\mathbb{M}$. Its output is $\mathbb{M}_x$, meet-irreducible sets of $\mathbb{F}_x$. The algorithm starts by adding all subsets in $\mathbb{M}$ that stay meet-irreducible in the extension. Then, it adds subsets F in $\Delta_x(\mathbb{M})$ that are in $\mathbb{M}_x$, as well as meet-irreducible sets that verify the properties 6 and 7.

Algorithm 1: EXTENSION$(X, \mathbb{M})$

Output: Meet-irreducible sets $\mathbb{M}_{max}$ of the maximal extension

begin

  $U_\Phi = \{x \in X |$ CLOSURE$(x) \neq \{x\}\}$;

  for $x \in U_\Phi$ **do**

    $\mathbb{M}$ =EXPAND$(x, \mathbb{M})$;

Theorem 37 and property 6 prove that $\mathbb{M}_x$, as generated by EXPAND, covers all meet-irreducible sets of the extension. For every M in $\mathbb{M}$, either M is meet-irreducible in $\mathbb{F}_x$ (algorithm 2, line 3), or $M \cup \{x\}$ is meet-irreducible in $\mathbb{F}_x$ (algorithm 2, line 1) or both (algorithm 2, line 2). This depends on where M belongs in the partition $\{\mathbb{M}_1, \mathbb{M}_2, \mathbb{M}_3, \mathbb{M}_4, \mathbb{M}_5\}$ of $\mathbb{M}$ as defined in the previous section (see 7.2).

In addition, $\mathbb{M}_x$ contains meet-irreducible sets of the form $F \cup \{x\}$ where F is an intersection of a pair (M_1, M_2) in $\mathbb{M}_1 \cup \mathbb{M}_2$ and $\mathbb{M}_2 \cup \mathbb{M}_3$ (algorithm 2, line 4). In order to verify which of these intersections is indeed meet-irreducible in $\mathbb{F}_x$, we use the sufficient and necessary condition in property 7 (algorithm 2, line 5), which is easier to confirm than verifying if $F \cup \{x\}$ is meet-irreducible by comparing it to the intersection of all other sets in $\mathbb{M}_x$.

We conclude the correctness of the algorithm.

Theorem 39. EXPAND $(X, \mathbb{M})$ *computes the extension* $\mathbb{M}_x$ *of the closure system of* $\mathbb{M}$*, for every* x *in* U_Φ*.*

Theorem 40. EXTENSION $(X, \mathbb{M})$ *computes the maximal extension of the closure system* $\mathbb{F}$ *in incremental-polynomial time.*

Proof: According to theorem 35, $\mathbb{F}_{x_1,...,x_k}$ is the maximal extension $\mathbb{F}_{max}$ of $\mathbb{F}$, where $U_\Phi = \{x_1, ..., x_k\}$. Hence by applying the function EXPAND$(x_i, \mathbb{M})$ where $\mathbb{M}$ corresponds to meet-irreducible sets of the extension $\mathbb{F}_{x_1,...,x_i}$, for every $x_i \in U_\Phi$, the resulting meet-irreducible sets family corresponds to the maximal extension of the input.

For every extension $\mathbb{F}_x$, EXPAND generates its meet irreducible sets in polynomial time of the order $O(|\mathbb{M}|^3.|X|)$, in function of the input size of the current execution.

Hence, EXTENSION runs in an incremental polynomial time in order to generate all meet irreducible sets of the maximal extension. Its complexity is of the order $O(|X|^2.|\mathbb{M}_{max}|^3)$.

□

Generating all meet-irreducible sets of the maximal extension is an enumeration problem. Another enumeration question that this problem may inspire, is the relation between generating candidate keys of a closure system and candidate keys of its maximal extension. Generally, is there an impact of the existence of unitary implications on the problem of enumerating candidate keys? We investigate this question in the next chapter and give an efficient algorithm that generates candidate keys based on this point of view.

Algorithm 2: EXPAND$(x, \mathbb{M})$

Output: Meet-irreducible sets of the extension $\mathbb{M}_x$

begin

$\mathbb{M}_x = \emptyset$;
$\mathbb{M}_{1\cup2} = \emptyset$;
$\mathbb{M}_{3\cup4} = \emptyset$;
$F_x =$CLOSURE(x);
$F_x^* = F_x \setminus \{x\}$;
for $M \in \mathbb{M}$ **do**
  if $F_x^* \not\subseteq M$ **then**
1 $\mathbb{M}_x = \mathbb{M}_x \cup \{M \cup \{x\}\}$;
    if *($F_x^* \nearrow_\Phi M$)* **then**
      $\mathbb{M}_{3\cup4} = \mathbb{M}_{3\cup4} \cup \{M\}$;
      if *($F_x \nearrow_\Phi M$)* **then**
2 $\mathbb{M}_x = \mathbb{M}_x \cup \{M\}$;
  else
    $\mathbb{M}_{1\cup2} = \mathbb{M}_{1\cup2} \cup \{M\}$;
3 $\mathbb{M}_x = \mathbb{M}_x \cup \{M\}$;
for $(M_1, M_2) \in \mathbb{M}_{3\cup4} \times \mathbb{M}_{1\cup2}$ **do**
4 $F = M_1 \cap M_2$;
  $F' = X$;
  for $M \in \mathbb{M}_{3\cup4}$ **do**
    if $F \subseteq M$ **then**
      $F' = F' \cap M$;
5 **if** CLOSURE$(F \cup \{x\})$=CLOSURE$(F' \cup \{x\})$ **then**
    $\mathbb{M}_x = \mathbb{M}_x \cup \{F \cup \{x\}\}$;

RETURN $\mathbb{M}_x$

Chapter 8

Candidate Keys Enumeration of an Implicational Base

We explore in this chapter the problem of candidate keys enumeration. We begin by giving some related definitions in the preliminaries section. Then, we explore, in the second section, the algorithm proposed by Lucchesi and Osborn through the structure of candidate keys graph. We improve its complexity by defining a total order over the attributes set. Third section defines, inter alia, the concept of key-ideal sets formally and distinguishes a subset of minimal key-ideal sets. Then, we prove our main contribution of this chapter stating that if there is a polynomial delay and space algorithm enumerating all minimal key-ideal sets, then there is one for candidate keys enumeration. We give, as well, algorithms that generate minimal key-ideal sets in polynomial incremental time, non minimal key-ideal sets in polynomial delay and an algorithm that can generate all key-ideal sets in polynomial delay if the number of minimal key-ideal sets is polynomial in the input size.

8.1 Preliminaries

For an implicational base Σ over an element set X, we suppose, without loss of generality, that $|R| = 1$ for every functional dependency $L \to R$ in Σ.

A candidate key of an implicational base, also known as minimal generator in lattice or FCA terminologies, is a minimal subset of elements that identifies uniquely every tuple of the relation, in database terminology.

Definition 41. *K is a key of an implicational base Σ over X, if its closure K^Σ is exactly X. K is a* candidate key *if no subset of K is also a key of Σ.*

We denote by $\mathbb{K}(\Sigma, X)$ the set of all candidate keys of Σ over X (also called minimal keys in the literature), noted $\mathbb{K}(\Sigma)$ when X is understood from the context.

Enumeration problems, in a computational context, raise the question of determining the set of all solutions of a certain problem associated to a polynomial predicate P. This is different than counting problems that searches only for the number of all possible solutions.

The complexity of such problems often depends not only on the difficulty of the associated decision problem and the input size, but also on the number of these solutions and the ability to avoid regenerating a solution multiple times; An easy enumeration problem could be exponential in the input size, if the number of solutions is exponential. Hence, complexity classes of such problems differ than the usual decision problems; various complexity classes of enumeration problems are introduced in [31].

- **Output polynomial**: an algorithm enumerating all solutions in polynomial time in the order of P and the number of solutions.
- **Polynomial delay**: (1) the time to compute the first solution, (2) the time between outputting any two solutions, and (3) the time to detect that no further solution exists, are all polynomial in the order of P.
- **Incremental Polynomial time** : the time to compute the next solution and for detecting that no further solution exists is polynomial in the order of both P and of number of the already computed solutions.

Certain algorithms demand stocking every generated solution in order to avoid redundancy. This requires an exponential space when the number of solutions is exponential. In certain cases, this exponential space can be reduced into polynomial space by defining a parent relation between solutions that avoids redundancy without revisiting at each step the solutions set, which is the case in the non-minimal key-ideal sets enumeration problem.

8.2 Candidate keys enumerations

The difficulty of enumerating candidate keys depends on the input of the problem which can be a relation or a relation schema, i.e. an implicational base.

When the input is a relation schema, Lucchesi and Osborn [37] gave a incremental polynomial algorithm to list all candidate keys, but using exponential space. Saiedian and Spencer [41], introduce the notion of element graph of a set of functional

dependencies and show that candidate keys are union of candidate keys of strongly connected components of the element graph.

When the input is given by a relation, it is an open question if there is an output-polynomial time algorithm for candidate keys enumeration.

The candidate keys enumeration problem considered in this chapter is as follows:

Keys Enumeration (K-ENUM)

Input: An implicational base Σ over X.

Output: The set of all candidate keys $\mathbb{K}(\Sigma)$.

Lucchesi and Osborn's algorithm, in [37], solves this problem for an input Σ in $O(|\Sigma|.|\mathbb{K}(\Sigma)|^2)$ elementary operations. They start by initializing a keys set $\mathbb{K}$ with a random candidate key, then generate new ones using functional dependencies from Σ. Each time a new candidate key is generated, $\mathbb{K}$ is revisited to avoid redundancy. It is an incremental polynomial algorithm with complexity $O(|X|.|\Sigma|.|\mathbb{K}(\Sigma)|(|\mathbb{K}(\Sigma)|+|X|))$. This algorithm is based on the following theorem:

Theorem 42. *[37] Let $\mathbb{K} \subseteq \mathbb{K}(\Sigma)$ be a non-empty subset of candidate keys. If $\mathbb{K} \neq \mathbb{K}(\Sigma)$, then $\mathbb{K}$ contains a key K and Σ contains a functional dependency (L, a) such that $L \cup K \setminus \{a\}$ includes a candidate key from $\mathbb{K}(\Sigma) \setminus \mathbb{K}$.*

Lucchesi and Osborn's algorithm can also be seen as a Depth-first search of the super graph of candidate keys. We propose next a formal definition of this graph.

Definition 43. *We define the **directed transition graph** of candidate keys of an implicational base Σ on X, noted $G(\Sigma) = (\mathbb{K}(\Sigma), \mathcal{T})$ where vertices are candidate keys of Σ and (K, K') belongs to $\mathcal{T}$ if there is a dependency $(L, a) \in \Sigma$ with $K' \subseteq (L \cup K \setminus a)$. We label the arc (K, K') with the dependency (L, a).*

Theorem 42 implies that for every two candidate keys K and K', K is reachable from K' in the candidate keys graph. Hence the next corollary.

Corollary 44. *$G(\Sigma)$ is strongly connected.*

We describe next a new implementation of Lucchesi and Osborn Algorithm by improving its time complexity to $O(|X|.|\Sigma|.|\mathbb{K}(\Sigma)|)$.

We consider the lexicographical order derived from the total order $(X = \{a_1, ..., a_n\}, <)$ where $a_1 < a_2 < ... < a_n$. We consider a key S of Σ. Algorithm ALPHA-KEY(S, Σ) computes the lexicographically **last** candidate key contained in S.

Algorithm 3: ALPHA-KEY(S, Σ)

Output: the last candidate key $K \subseteq S$ in the considered lexicographical order

begin
- Let $S = \{s_1, ..., s_n\}$ with $s_1 < ... < s_n$;
- $K = S$;
- **for** $i = 1, n$ **do**
  - **if** $(K \setminus \{s_i\})^{\Sigma} = X$ **then**
    - $K = K \setminus \{s_i\}$;
- Return K;

Algorithm 4: ALL-KEYS(Σ)

Output: All keys of Σ

begin
- $K = \{$ALPHA-KEY$(X, \Sigma)\}$;
- $Mark(K) = unvisited$;
- $\mathbb{K} = \{K\}$;
- **while** *there is a unmarked* $K \in \mathbb{K}$ **do**
  - $Mark(K) = visited$;
  - **for** $L \rightarrow a \in \Sigma$ **do**
    - $S = L \cup (K \setminus \{a\})$;
    - $K' =$ALPHA-KEY(S, Σ);
    - **if** $K' \notin \mathbb{K}$ **then**
      - $Mark(K') = unvisited$;
      - $\mathbb{K} = \mathbb{K} \cup \{K'\}$;

Theorem 45. *Algorithm* ALL-KEYS *outputs all keys in incremental polynomial time with a time complexity of the order* $O(|\mathbb{K}(\Sigma)|^2.|\Sigma| + |\mathbb{K}(\Sigma)|.|\Sigma|^2.|X|)$.

Proof: The complexity C_α of the algorithm ALPHA-KEY is of the order $O(|\Sigma|.|X|)$. For every candidate key in $\mathbb{K}(\Sigma)$ and every implication in Σ, we apply ALL-KEYS and browse $\mathbb{K}(\Sigma)$ in order to verify that the new solution is new. Hence, the complexity of ALL-KEYS is of the order $O(|\mathbb{K}(\Sigma)|.|\Sigma|.(C_\alpha + |\mathbb{K}(\Sigma)|)) = O(|\mathbb{K}(\Sigma)|^2.|\Sigma| + |\mathbb{K}(\Sigma)|.|\Sigma|^2.|X|)$. Moreover, it takes $O(|\mathbb{K}(\Sigma)|.|\Sigma| + |\Sigma|^2.|X|)$ time in order to generate a new solution. Hence, this enumeration algorithm is incremental polynomial. □

Example 11. *We consider a relational database schema with an element set* $\{a, b, c, d, e, f\}$

and functional dependencies as follows: $\Sigma = \{c \to b, b \to a, f \to e, e \to d, bd \to a, bd \to c, bd \to e, bd \to f, ae \to b, ae \to c, ae \to d, ae \to f\}$.

ALL-KEYS(Σ) *searches the candidate keys graph in figure 8.1. A transition from a candidate key* K *to another* K' *in this graph means that for a functional dependency* $L \to a$ *in* Σ *with* a *in* K, K' *is the last candidate key, in the alphabetic order, included in* $K \setminus \{a\} \cup L$.

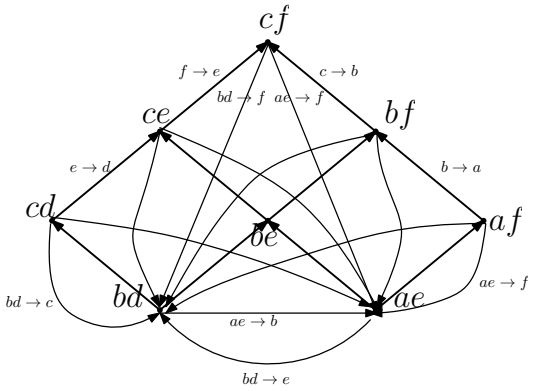


Figure 8.1: *Candidate keys graph*

The resulting execution tree of ALL-KEYS(Σ) *algorithm is given in figure 8.2. All enumerated keys set must be revisited at each step in order to avoid redundancy.*

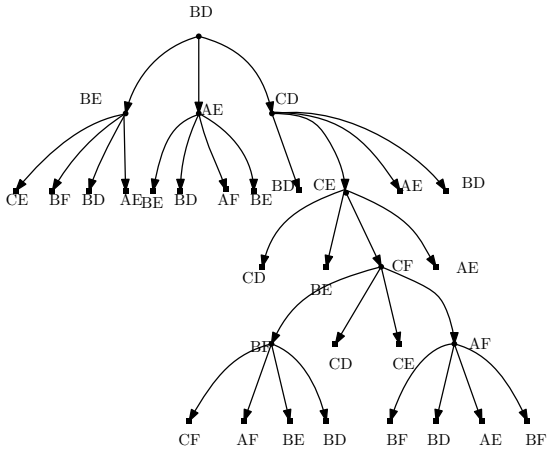


Figure 8.2: *Candidate keys enumeration using Lucchesi and Osborn's algorithm*

8.3 Key-ideal sets enumeration

This section defines the concept of key-ideal sets. Then, we prove that if there is a polynomial delay algorithm enumerating all minimal key-ideal sets, then there is one for candidate keys enumeration.

Let Σ be an implicational base over an element set X. We begin by defining an order over X, induced by a subset of Σ. We consider the set of unitary functional dependencies $\Sigma_P = \{L \to R \in \Sigma \mid |L| = 1\}$ and $G(\Sigma_P)$ the graph which vertices are the elements in X and there is an arc from an element a to an element b, if $b \in a^{\Sigma_P}$. Σ_P is said to be acyclic if $G(\Sigma_P)$ is acyclic.

Lemma 46. *If Σ_P is not acyclic, then: A subset $K \subseteq X$ is a candidate key if and only if for every $a \in K$ and b in the same strongly connected component of $G(\Sigma_P)$, we have $(K \setminus \{a\}) \cup \{b\}$ a candidate key.*

Proof: Let $K \subset X$ be a candidate key, $a \in K$ and $b \in X$ in a same strongly connected component of Σ_P with a. We have $a \in b^{\Sigma_P}$ and $b \in a^{\Sigma_P}$. This implies that $K^{\Sigma} = ((K \setminus \{a\}) \cup \{b\})^{\Sigma}$. Hence, $((K \setminus \{a\}) \cup \{b\})$ is a key.

Suppose that $S \subset ((K \setminus \{a\}) \cup \{b\})$ is a key. Either S or $((S \setminus \{b\}) \cup \{a\})$ is a subset of K and they are both keys. This contradicts the fact that K is a candidate key. Hence $((K \setminus \{a\}) \cup \{b\})$ is a candidate key. □

Whenever Σ_P is not acyclic, we consider the implicational base Σ' over X', where X' is composed by choosing one element from each strongly connected component and Σ' is obtained from Σ by replacing elements of each connected component by the one chosen in X'.

Example 12. *Consider the implicational base $\Sigma = \{a \to b, b \to a, b \to c, bd \to e\}$ on the set $X = \{a, b, c, d, e\}$. The graph $G(\Sigma_P)$ is not acyclic as shown in Figure 8.3. $\{a, b\}$ and $\{c\}$ are the strongly connected components in $G(\Sigma_P)$. The obtained implicational base is $\Sigma' = \{a \to c, ad \to e\}$.*

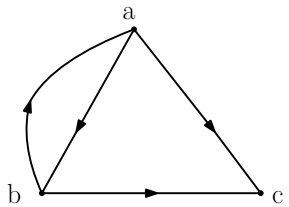


Figure 8.3: *Unary functional dependencies with a cycle*

We suppose in this chapter, without loss of generality, that Σ_P is acyclic and consider the partial order $P(X, \Sigma) = (X, \leq)$ defined by Σ_p over X, where $a \leq b$ if and only if $a \in b^{\Sigma_p}$. We define $pred(a) = \{b \in X \mid \nexists c \in X$, such that $a < c < b\}$.

Definition 47. *Key-ideal set, Minimal key-ideal set*

An ideal $I \subseteq X$ in $P(X, \Sigma)$ is called a key-ideal if its maximal elements set $Max(I)$ is a candidate key. $\mathcal{I}^{\mathbb{K}}$ refers to $P(X, \Sigma)$'s key-ideals sets.

A key-ideal set $I \in \mathcal{I}^{\mathbb{K}}$ is called minimal if no subset of I is a key-ideal set. The set of all minimal key-ideal sets of Σ are denoted by $\mathcal{I}^{\mathbb{K}}_{min}$. We denote by $P(\mathcal{I}^{\mathbb{K}}, \subseteq)$ the set of all key-ideal sets ordered under set inclusion.

Note that every key-ideal set of Σ is also a key of Σ, but not necessarily a candidate key. The notion of minimal key-ideals sets seems important for candidate keys enumeration, since the number of key-ideal sets may be exponential either in the size of minimal key-ideal sets. Consider $X = \{a_1, a_2, ..., a_{2p-1}, a_{2p}\}$ for some integer p and $\Sigma = \{a_1...a_p \rightarrow X\} \cup \{a_{p+i} \rightarrow a_i, 1 \leq i \leq p\}$. Then there is a unique minimal key-ideal set $I = \{a_1, ..., a_p\}$, but there is 2^p key-ideal sets.

In addition to the previously defined candidate keys enumeration problem K-Enum, we consider the following enumeration problems:

Key-Ideals Enumeration (KI-Enum)

Input: An implicational base Σ on a set X.

Output: The set of all key-ideals sets $\mathcal{I}^{\mathbb{K}}$ of Σ.

Minimal Key-Ideals Enumeration (MinKI-Enum)

Input: An implicational base Σ on a set X.

Output: The set of all minimal key-ideals sets $\mathcal{I}^{\mathbb{K}}_{min}$ of Σ.

There is a one-to-one mapping between $\mathcal{I}^{\mathbb{K}}$ and $\mathbb{K}$ ($K = Max(I)$ and $I = K^{\Sigma_P}$). Thus problems K-ENUM and KI-ENUM are linearly equivalent since the size of a key-ideal is bounded by $|X|$.

Proposition 48 demonstrates how any key-ideal set that is not minimal, can be generated through others by adding one element and its closure in Σ_p. This defines the idea of the transition function (proposition 49) of a forest (a set of trees) that covers all key-ideal sets and where roots set is exactly $\mathcal{I}^{\mathbb{K}}_{min}$ (theorem 50). Then, we prove that knowing $\mathcal{I}^{\mathbb{K}}_{min}$, the generation of this forest can be done in polynomial delay.

Proposition 48. *Let $I \in \mathcal{I}^{\mathbb{K}} \setminus \mathcal{I}^{\mathbb{K}}_{min}$. There exists $a \in Max(I)$ and $I' \in \mathcal{I}^{\mathbb{K}}$ such that $I = I' \cup \{a\}^{\Sigma_P}$ and I covers I' in $P(\mathcal{I}^{\mathbb{K}}, \subseteq)$.*

Proof: First, since I is not a minimal key-ideal then there exists at least one $a \in Max(I)$ such that $I \setminus \{a\}$ is a key. Hence $Max(I \setminus \{a\})$ is also a key.

We consider $K \subseteq Max(I \setminus \{a\})$ a candidate key and $I' = K^{\Sigma_P}$ its corresponding key-ideal set.

Second, we prove by contradiction that $Max(I) \setminus \{a\}$ is a subset of K. If we suppose the opposite, then for b in $Max(I) \setminus (\{a\} \cup K)$, we have K is a subset of $(Max(I) \setminus \{b\})^{\Sigma_P}$. Hence, $Max(I) \setminus \{b\}$ is a key, which contradicts the fact that $Max(I)$ is a candidate key. We conclude from it that $I = K^{\Sigma_P} \cup \{a\}^{\Sigma_P} = I' \cup \{a\}^{\Sigma_P}$.

Finally, we show that I covers I' in $P(\mathcal{I}^{\mathbb{K}}, \subseteq)$. For every ideal set $I'' \subseteq X$ verifying that $I' \subset I'' \subset I$, we have $K \subset Max(I'')$. This means that $Max(I'')$ is not a candidate key and therefore, $I'' \notin \mathcal{I}^{\mathbb{K}}$. $\square$

Proposition 48 ensures the existence of I' but not its uniqueness. Therefore, we distinguish in the next proposition one of these key-ideal sets as the parent of I, using a lexicographical total order $(\leq_{lex}, X)$.

Proposition 49. *For every $I \in \mathcal{I}^{\mathbb{K}} \setminus \mathcal{I}^{\mathbb{K}}_{min}$, there exists a unique key-ideal $I' = Parent(I) \in \mathcal{I}^{\mathbb{K}}$, satisfying the following:*

1. *I covers I' in $P(\mathcal{I}^{\mathbb{K}}, \subseteq)$.*
2. *$I = I' \cup \{a\}^{\Sigma_P}$, where a is the lexicographically largest element in $Max(I)$ such that $I \setminus \{a\}$ is a key.*
3. *$Max(I')$ is the lexicographically largest candidate key included in $Max(I \setminus \{a\})$.*

Proof: Let $I \in \mathcal{I}^{\mathbb{K}} \setminus \mathcal{I}^{\mathbb{K}}_{min}$, $S = \{a \in Max(I) | I \setminus \{a\}$ is a key$\}$ and $\mathcal{I}_a = \{K \in \mathbb{K} | I = K^{\Sigma_p} \cup \{a\}$ and I covers $K^{\Sigma_p}\}$ for some $a \in S$. Since $I \notin \mathcal{I}^{\mathbb{K}}_{min}$ then S is not empty and according to Proposition 48, there exists at least one element $a' \in S$ such that $\mathcal{I}_{a'} \neq \emptyset$.

We consider a as the lexicographically largest element in S with $\mathcal{I}_a \neq \emptyset$ and I' is the key-ideal set corresponding to the lexicographically largest key in $\mathcal{I}_a$. □

For every $I \in \mathcal{I}^{\mathbb{K}} \setminus \mathcal{I}^{\mathbb{K}}_{min}$, recall that $Parent(I)$ denotes the unique ideal satisfying properties of Proposition 49, and it is empty if $I \in \mathcal{I}^{\mathbb{K}}_{min}$. Now we define $G(\Sigma) = (\mathcal{I}^{\mathbb{K}}, E)$ as a directed graph of key-ideal sets where $(I', I) \in E$ if and only if $Parent(I) = I'$.

Theorem 50. *$G(\Sigma) = (\mathcal{I}^{\mathbb{K}}, E)$ is a covering forest of $P(\mathcal{I}^{\mathbb{K}}, \subseteq)$, with roots key-ideal sets in $\mathcal{I}^{\mathbb{K}}_{min}$.*

Proof: First, key-ideal sets in $\mathcal{I}^{\mathbb{K}}_{min}$ have no parent and thus $\mathcal{I}^{\mathbb{K}}_{min}$ are roots. According to Proposition 49, for every $I \in \mathcal{I}^{\mathbb{K}} \setminus \mathcal{I}^{\mathbb{K}}_{min}$, there exists a unique $(Parent(I), I)$ in E. Moreover, the size of I is strictly greater than the size of $Parent(I)$. Thus for any $I \in \mathcal{I}^{\mathbb{K}}$, there exists a sequence $I_0 \subset ... \subset I_k = I$ of key-ideal sets such that $I_0 \in \mathcal{I}^{\mathbb{K}}_{min}$ and $I_{j-1} = Parent(I_j)$, $1 \leq j \leq k$ with $a_j \in X \setminus I_{j-1}$. □

We conclude in theorem 51 the main result of this section.

Theorem 51. *If there is a polynomial delay and space algorithm to enumerate $\mathcal{I}^{\mathbb{K}}_{min}$, then there is one to enumerate all candidate keys in polynomial delay and space.*

Proof: According to theorem 50, the set $\mathcal{I}^{\mathbb{K}}_{min}$ correspond to roots of the forest of all key-ideal sets. Let $I_1, ..., I_j$ be the sequence of key-ideal sets in $\mathcal{I}^{\mathbb{K}}_{min}$ already found in polynomial delay. We show that the enumeration of remaining key-ideal sets in the tree rooted at I_j can be enumerated in polynomial delay and space.

We apply a depth first search algorithm at I_j. At each step I, we compute the children of I using Proposition 49, i.e. the set $\{I \cup \{a\}^{\Sigma_P} \mid a \in X \setminus I, Parent(I \cup \{a\}^{\Sigma_P}) = I\}$. Since checking conditions of Proposition 49 can be done in polynomial time, the total cost at each node is bounded by a polynomial. Moreover the depth of the tree is bounded by the size of X. Thus to find the next key-ideal is bounded by a polynomial in the two cases, i.e. the case when the next belongs to the tree or the case where the next is the key-ideal $I_{j+1} \in \mathcal{I}^{\mathbb{K}}_{min}$.

The dfs algorithm uses a polynomial space since the depth of the tree is bounded by the size of X. Thus the total space used is polynomial, since the enumeration of $\mathcal{I}^{\mathbb{K}}_{min}$. □

In the following we give an algorithm to enumerate all minimal key-ideal sets in incremental polynomial time and for each minimal key-ideal found, we apply theorem 50 to enumerate the key-ideal sets in the associated tree.

Lucchesi and Osborn give in [37] an algorithm that enumerates candidate keys of an implicational base Σ. It starts with a random candidate key, then generate new ones using functional dependencies from Σ: substitute an element a from the current key with a subset L such that $L \rightarrow a$ is in Σ, then minimize following a certain order. Each time a new candidate key is generated, the set of already enumerated keys is revisited to ensure that there is no redundancy. It is an incremental polynomial algorithm with complexity $O(|X|.|\Sigma|.|\mathbb{K}|(|\mathbb{K}+|X|))$ which can be easily improved to $O(|X|^2.|\Sigma|.|\mathbb{K}|)$ by using a trie or lexicographical tree to store candidate keys. We prove in theorem 52 that we can enumerate only minimal key-ideal sets in the same way as this algorithm.

Theorem 52. *Let $\mathcal{I}$ be a non-empty subset of $\mathcal{I}^{\mathbb{K}}_{min}$. $\mathcal{I} \neq \mathcal{I}^{\mathbb{K}}_{min}$ if and only if $\mathcal{I}$ contains a minimal key-ideal set I and Σ contains an implication $L \rightarrow a$ such that $(L \cup I \setminus \{a\}^{\Sigma_P})^{\Sigma_P}$ does not include any key-ideal set in $\mathcal{I}$.*

Proof: We first prove that the stated condition is sufficient to have $\mathcal{I} \neq \mathcal{I}^{\mathbb{K}}_{min}$. First, since $L \rightarrow a$ is in Σ, then $(L \cup I \cup \{a\}^{\Sigma_P}) \subseteq (L \cup I \setminus \{a\}^{\Sigma_P})^{\Sigma}$. Moreover $(L \cup I \cup \{a\}^{\Sigma_P})$ is a key because it contains a key-ideal set I, hence $(L \cup I \setminus \{a\}^{\Sigma_P})$ is also a key. This means that $(L \cup I \setminus \{a\}^{\Sigma_P})^{\Sigma_P}$ is a key and an ideal, thus it contains a minimal key-ideal set that does not figure in $\mathcal{I}$ according to the condition.

To prove that the condition is necessary, we assume that there exists a minimal key ideal set $I' \notin \mathcal{I}$. Let S be an ideal in $P(X, \Sigma)$ verifying: (1) S includes I' and (2) for every I'' in $\mathcal{I}$, S does not include I'' and (3) for every element $a \notin S$, $S \cup \{a\}$ is either not an ideal, or it contains a key-ideal set in $\mathcal{I}$. Note that S is a proper subset of X and a key.

Since $S \neq S^{\Sigma} = X$, then Σ contains $L \rightarrow a$ with L included in S and $a \notin S$. Let $b = min_{P(X,\Sigma)}\{b' \in X \setminus S \mid b' \leq a\}$. Then we have $S \cup \{b\}$ is an ideal in $P(X, \Sigma)$. Hence according to (3), $S \cup \{b\}$ contains a key-ideal set I in $\mathcal{I}$.

Now we prove that $(L \cup I \setminus \{a\}^{\Sigma_P})^{\Sigma_P}$ is a subset of S. First, $I \setminus b$ is included in S by construction. Second, b is in $\{a\}^{\Sigma_P}$. Third, S includes L. We conclude that $L \cup I \setminus \{a\}^{\Sigma_P}$ is included in S. And since S is an ideal then the closure over Σ_p is also in S. Therefore, $(L \cup I \setminus \{a\}^{\Sigma_P})^{\Sigma_P}$ does not include any key-ideal set from $\mathcal{I}$. □

Hence, the next algorithm ALL-MIN-KEY-IDEALS generates all minimal key-ideal sets, using mainly the same strategy as the algorithm in [37]. The only difference is

while minimizing: instead of searching for a subset candidate key, we minimize in a certain order until reaching a minimal key-ideal set.

Theorem 53. *Algorithm* ALL-MIN-KEY-IDEALS *outputs all minimal key-ideal sets in incremental polynomial time, i.e.* $O(|X|.\ |\Sigma|.\ |\mathcal{I}^{\mathbb{K}}_{min}|)$.

Algorithm 5: ALL-MIN-KEY-IDEALS(Σ)

```
Output: All minimal key-ideal sets of Σ
begin
    I =ALPHA-MINIMAL-KEY-IDEAL(X, Σ);
    ℐ = {I};
    Output(I);
    Mark(I) = unmarked;
    while there is an unmarked I ∈ ℐ do
        Mark(I) = marked;
        for L → a ∈ Σ do
            S = L^{Σ_P} ∪ (I \ a^{Σ_P});
            I' =ALPHA-MINIMAL-KEY-IDEAL(S, Σ);
            if I' ∉ ℐ then
                Mark(I') = unmarked;
                ℐ = ℐ ∪ {I'};
    return ℐ;
```

ALL-MIN-KEY-IDEALS calls the routine ALPHA-MINIMAL -KEY-IDEAL to determine the lexicographically **last** minimal key-ideal set of Σ that is a subset of a specified key S. It requires $O(|\Sigma|.|X|^2)$.

The chosen lexicographical order $(X, \leq_{lex})$ verifies that if $a \leq b$ in $P(X, \Sigma)$ then $b \leq_{lex} a$. This is convenient to the structure of the algorithm, in order to verify that it generates the lexicographically last minimal key-ideal set in a subset S by erasing elements from it in an increasing lexicographical order. We also assume for every element subset denoted next $\{a_1, ..., a_n\}$, that $a_i \leq_{lex} a_j$ for $i \leq j$.

Algorithm 6: ALPHA-MINIMAL-KEY-IDEAL($S = \{a_1, ..., a_n\}, \Sigma, X$)

Output: A minimal key-ideal $I \subseteq S$

begin

  if *S is not an ideal* ***or*** $S^{\Sigma} \neq X$ **then**

    Return NULL;

  for $i = 1, n$ **do**

    if $a_i \in Max(S)$ *and* $(S \setminus \{a_i\})^{\Sigma} = X$ **then**

      $S = S \setminus \{a_i\}$;

  Return S;

For every new minimal key-ideal set I generated by ALL-MIN-KEY-IDEALS, the algorithm KEY-IDEAL-SETS can traverse a tree of non minimal key-ideal sets using the transition function PARENT. What makes this approach interesting is that the algorithm KEY-IDEAL-SETS takes a polynomial delay and space to generate these non-minimal key-ideal sets.

Algorithm 7: KEY-IDEAL-SETS(I, Σ)

Output: key-ideal sets $\mathcal{I}$ all containing I

begin

  $(X \setminus I) = \{a_1, ..., a_n\}$;

  $S = X \setminus I$;

  $\mathcal{I} = \{I\}$;

  $i = n$;

  while $i > 0$ **do**

    $I' = I \cup \{a_i\}^{\Sigma_P}$;

    if $I =$PARENT(I', Σ) **then**

      $\mathcal{I} = \mathcal{I} \cup$KEY-IDEAL-SETS$(I', \Sigma)$;

    $i = i - 1$;

  Return $\mathcal{I}$;

Algorithm 8: PARENT(I, Σ)

Output: Key-ideal set $I' \subseteq I$ verifying property 49

begin

$Max(I) = \{a_1, ..., a_n\}$;

$i = n$;

while $i > 0$ **do**

if I *is an ideal and* $(I \setminus \{a_i\})^{\Sigma} = X$ **then**

$K = Max(I \setminus \{a_i\}) = \{a'_1, ..., a'_l\}$;

for $j = 1, l$ **do**

if $(K \setminus \{a'_j\})^{\Sigma} = X$ **then**

$K = K \setminus \{a'_j\}$;

return K^{Σ_p};

$i = i - 1$;

According to theorem 50, every key-ideal set is in exactly one tree generated by KEY-IDEAL-SETS. Hence, if we apply this algorithm each time we find a new minimal key-ideal set, then this enumerates all key-ideal sets and therefore all candidate keys. Since KEY-IDEAL-SETS runs in polynomial space and time in the size of the input, then the following algorithm ALL-KEY-IDEALS stays in incremental polynomial time.

Theorem 54. *Algorithm* ALL-KEY-IDEALS *outputs all key-ideal sets in incremental polynomial time. In particular, it enumerates all minimal key-ideal sets in* $O(|X|^2.|\Sigma|.|\mathcal{I}^{\mathbb{K}}_{min}|)$. *And given* $\mathcal{I}^{\mathbb{K}}_{min}$, *it generates all key-ideal sets that are not minimal in polynomial delay, using* $O(|X|.|\mathcal{I}^{\mathbb{K}}_{min}|)$ *space memory.*

Algorithm 9: ALL-KEY-IDEALS(Σ)

Output: All key-ideal sets of (Σ)

begin

$I =$ALPHA-MINIMAL-KEY-IDEAL(X, Σ);

$\mathcal{I}_{min} = \{I\}$;

$\mathcal{I} =$KEY-IDEAL-SETS(I, Σ);

$Mark(I) = unmarked$;

while *there is an unmarked* $I \in \mathcal{I}_{min}$ **do**

$Mark(I) = marked$;

$\mathcal{I} = \mathcal{I} \cup$KEY-IDEAL-SETS$(I, \Sigma)$;

for $L \to a \in \Sigma$ **do**

$S = L^{\Sigma_P} \cup (I \setminus a^{\Sigma_P})$;

$I' =$ALPHA-MINIMAL-KEY-IDEAL(S, Σ);

if $I' \notin \mathcal{I}_{min}$ **then**

$Mark(I') = unmarked$;

$\mathcal{I}_{min} = \mathcal{I}_{min} \cup \{I'\}$;

return $\mathcal{I}$;

Chapter 9

Conclusion

9.1 Summary

This part concerns closure systems and related structures: implicational bases, binary relations and lattices. We have investigated two problems: enumeration of candidate keys of an implicational base and generation of closure system's maximal extension.

We begin by dealing with the generation of a closure system's maximal extension. We gave an incremental polynomial algorithm that enumerates all meet-irreducible sets of the maximal extension. We gave as well some properties that describe the maximal extension.

Then, we proposed a new approach to enumerate candidate keys of an implicational base by defining a key-ideal set, an ideal of a candidate key in an order defined by Σ. We demonstrated that candidate keys enumeration is as hard as the enumeration of a subset of key-ideal sets that we call minimal key-ideal sets. And since candidate keys can be exponentially more numerous than minimal key-ideal sets, we gave an efficient key-ideal sets enumeration algorithm, that enumerates all non-minimal key-ideal sets in a polynomial delay and minimal key-ideal sets in incremental polynomial time.

9.2 Perspectives

Several questions exist to follow further research directions based on the present work. The existence of a polynomial delay algorithm that can generate the maximal extension is still an open question. It is also interesting to identify lattice properties that could parametrize the complexity of the problem, for example k-semi distributivity

of lattices.

As for candidate keys enumeration, our framework can also be improved using the idea in [41] where they consider acyclicity of Σ. Our work can be applied to each strongly connected component of Σ. It can also be applied to minimal generators in FCA terminology. Identification and complexity characterization of different implicational base classes can help as well to achieve efficient algorithms and new complexity bounds. The existence of a polynomial delay and space algorithm that enumerates all candidate keys is also an open question.

A common perspective between both problematics, is to describe candidate keys of closure system based on its extension, and vice versa.

Bibliography

[1] Milton Abramowitz and Irene A. Stegun. *Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables*. Dover, New York, ninth dover printing, tenth gpo printing edition, 1964.

[2] K. V. Adaricheva and J. B. Nation. Largest extension of a finite convex geometry. *algebra universalis*, 52(2):185–195, 2004.

[3] K.V. Adaricheva, V.A. Gorbunov, and V.I. Tumanov. Join-semidistributive lattices and convex geometries. *Advances in Mathematics*, 173(1):1 – 49, 2003.

[4] Lakhdar Akroun, Boualem Benatallah, Lhouari Nourine, and Farouk Toumani. Decidability and complexity of simulation preorder for data-centric web services. In *International Conference on Service-Oriented Computing*, pages 535–542. Springer, 2014.

[5] Gustavo Alonso, Fabio Casati, Harumi Kuno, and Vijay Machiraju. *Web Services*, pages 123–149. Springer Berlin Heidelberg, Berlin, Heidelberg, 2004.

[6] GuVery stavo Alonso, Fabio Casati, Harumi Kuno, and Vijay Machiraju. *Web Services: Concepts, Architectures and Applications*. Springer, 2010.

[7] B. Benatallah, F. Casati, and F. Toumani. Web Service Conversation Modeling: A Cornerstone for E-Business Automation. *IEEE Internet Computing*, 08(1):46–54, 2004.

[8] Daniela Berardi, Diego Calvanese, Giuseppe De Giacomo, Richard Hull, and Massimo Mecella. Automatic composition of transition-based semantic web services with messaging. In *Very Large Data Bases Conference VLDB*, pages 613–624. VLDB Endowment, 2005.

[9] Daniela Berardi, Diego Calvanese, Giuseppe De Giacomo, Richard Hull, and Massimo Mecella. Automatic composition of web services in colombo. In *Italian Symposium on Advanced Database Systems SEBD*, pages 8–15, 2005.

[10] Daniela Berardi, Diego Calvanese, Giuseppe De Giacomo, Maurizio Lenzerini, and Massimo Mecella. Automatic composition of e-services that export their behavior. In *International Conference on Service-Oriented Computing*, volume 2910, pages 43–58. Springer, 2003.

[11] Garrett Birkhoff. *Lattice theory*, volume 25. American Mathematical Society New York, 1948.

[12] Jean-Paul Bordat. Calcul pratique du treillis de galois d'une correspondance. *Mathématiques et Sciences humaines*, 96:31–47, 1986.

[13] Tomáš Brázdil, Petr Jančar, and Antonín Kučera. Reachability games on extended vector addition systems with states. In *Automata, Languages and Programming*, pages 478–489. Springer, 2010.

[14] Antonio Brogi and Sara Corfini. Behaviour-aware discovery of web service compositions. *International Journal of Web Services Research*, 4(3):1, 2007.

[15] V.W Bryant and R.J Webster. Convexity spaces. i. the basic properties. *Journal of Mathematical Analysis and Applications*, 37(1):206 – 213, 1972.

[16] T. Bultan, X. Fu, R. Hull, and J. Su. Conversation specification: a new approach to design and analysis of e-service composition. In *WWW'03*. ACM, 2003.

[17] Jakub Chaloupka. Z-reachability problem for games on 2-dimensional vector addition systems with states is in p. In *Reachability Problems*, pages 104–119. Springer, 2010.

[18] Søren Christensen. Decidability and decomposition in process algebras. *PhD thesis*, 1993.

[19] Jean-Baptiste Courtois and Sylvain Schmitz. Alternating vector addition systems with states. In *Mathematical Foundations of Computer Science 2014*, pages 220–231. Springer, 2014.

[20] B. A. Davey and H. A. Priestley. *Introduction to lattices and order*. Cambridge University Press, second edition, 1991.

[21] Jean-Paul Doignon and Jean-Claude Falmagne. Spaces for the assessment of knowledge. *International Journal of Man-Machine Studies*, 23(2):175 – 196, 1985.

[22] Karima Ennaoui and Lhouari Nourine. Hybrid algorithms for candidate keys enumeration for a relational schema. In *Conférence sur la Gestion de Données - Principes, Technologies et Applications BDA*, 2016.

[23] Karima Ennaoui, Lhouari Nourine, and Farouk Toumani. Complexity aspects of web services composition. In *International Workshop on Petri Nets and Software Engineering*, pages 85–104, 2017.

[24] B. Ganter. Two basic algorithms in concept analysis. Technical report, No 831, Technische Hoschule Darmstadt, 1984.

[25] Bernhard Ganter and Heiko Reppe. Base points, non-unit implications, and convex geometries. In *International Conference on Formal Concept Analysis*, pages 210–220. Springer, 2007.

[26] W. Geyer. The generalized doubling construction and formal concept analysis. *Algebra universalis*, 32:341–367, 1994.

[27] Michel Habib and Lhouari Nourine. A representation theorem for lattice via set-colored posets. Technical report, www.isima.fr/ nourine/Documents, 2012.

[28] Ramy Ragab Hassen, Lhouari Nourine, and Farouk Toumani. Protocol-based web service composition. In *International Conference on Service-Oriented Computing*, pages 38–53, 2008.

[29] Monika Rauch Henzinger, Thomas A Henzinger, and Peter W Kopke. Computing simulations on finite and infinite graphs. In *Annual Symposium on Foundations of Computer Science*, pages 453–462. IEEE, 1995.

[30] Wilfrid Hodges. Tarski's truth definitions. In Edward N. Zalta, editor, *The Stanford Encyclopedia of Philosophy*. Metaphysics Research Lab, Stanford University, fall 2014 edition, 2014.

[31] David S. Johnson, Mihalis Yannakakis, and Christos H. Papadimitriou. On generating all maximal independent sets. *Information Processing Letters*, 27(3):119 – 123, 1988.

[32] Juliette Kennedy. Kurt gödel. In Edward N. Zalta, editor, *The Stanford Encyclopedia of Philosophy*. Metaphysics Research Lab, Stanford University, winter 2016 edition, 2016.

[33] Heather Kreger et al. Web services conceptual architecture (wsca 1.0). *IBM Software Group*, 5:6–7, 2001.

[34] Kyriakos Kritikos, Barbara Pernici, Pierluigi Plebani, Cinzia Cappiello, Marco Comuzzi, Salima Benrernou, Ivona Brandic, Attila Kertész, Michael Parkin, and Manuel Carro. A survey on service quality description. *ACM Computing Surveys (CSUR)*, 46(1):1, 2013.

[35] Slawomir Lasota. Expspace lower bounds for the simulation preorder between a communication-free petri net and a finite-state system. *Inf. Process. Lett.*, 109(15):850–855, 2009.

[36] Stéphane Lopes, Jean-Marc Petit, and Lotfi Lakhal. Functional and approximate dependency mining: database and FCA points of view. *Journal of Experimental & Theoretical Artificial Intelligence*, 14(2-3):93–114, April 2002.

[37] Cláudio L. Lucchesi and Sylvia L. Osborn. Candidate keys for relations. *Journal of Computer and System Sciences*, 17(2):270 – 279, 1978.

[38] Anca Muscholl and Igor Walukiewicz. A lower bound on web services composition. *Logical Methods in Computer Science*, 4(2), 2008.

[39] Lhouari Nourine and Olivier Raynaud. A fast algorithm for building lattices. *Inf. Process. Lett.*, 71(5-6):199–204, 1999.

[40] Ramy Ragab Hassen. *Automatic composition of prototocol-based Web services*. PhD thesis, Clermont-Ferrand 2, 2009.

[41] H. Saiedian and T. Spencer. An efficient algorithm to compute the candidate keys of a relational database schema. *The Computer Journal*, 39(2):124–132, 1996.

[42] Quan Z. Sheng, Xiaoqiang Qiao, Athanasios V. Vasilakos, Claudia Szabo, Scott Bourne, and Xiaofei Xu. Web services composition: A decade's overview. *Information Sciences*, 280:218 – 238, 2014.

[43] Biplav Srivastava and Jana Koehler. Web service composition-current solutions and open problems. In *ICAPS 2003 workshop on Planning for Web Services*, volume 35, pages 28–35, 2003.

[44] Nicolas Veyrat-Charvillon, Benoît Gérard, Mathieu Renauld, and François-Xavier Standaert. An optimal key enumeration algorithm and its application to side-channel attacks. In *International Conference on Selected Areas in Cryptography*, pages 390–406. Springer, 2012.

[45] Marcel Wild. The joy of implications, aka pure horn formulas: Mainly a survey. *Theoretical Computer Science*, 658, Part B:264 – 292, 2017.

[46] Rudolf Wille. Subdirect decomposition of concept lattices. *algebra universalis*, 17(1):275–287, 1983.

[47] Hiyori Yoshikawa, Hiroshi Hirai, and Kazuhisa Makino. A representation of antimatroids by horn rules and its application to educational systems. *Journal of Mathematical Psychology*, 2016.