Toward a complexity classification of CSP through kernel width

Florian Richoux

Toward a complexity classification of CSP through kernel width

Florian Richoux

Thèse de Doctorat
dirigée par Miki Hermann
effectuée au
Laboratoire d'Informatique de l'École Polytechnique

Soutenue publiquement le 12 novembre 2009

Jury

Manuel	BODIRSKY	
Nadia	CREIGNOU	Rapporteur
Víctor	DALMAU	Rapporteur
Gilles	DOWEK	
Arnaud	DURAND	
Miki	HERMANN	Directeur
Jaroslav	NEŠETŘIL	Rapporteur
Gustav	NORDH	

Remerciements

Les remerciements constituent toujours un passage délicat, car c'est à travers ces quelques lignes que l'on tente d'exprimer toute notre gratitude aux personnes qui nous ont entourés, avec qui on a été en contact, qui nous ont soutenus, parfois même supportés, durant ces trois années de thèse. Délicat aussi parce que l'on a constamment l'angoisse d'oublier quelqu'un en écrivant ces lignes.

Tout d'abord, je remercie chaque membre du jury d'avoir bien accepté d'en faire partie, en particulier les rapporteurs : Jarik, Nadia et Víctor. Je regrette profondément qu'Arnaud n'est pas été retenu comme rapporteur de mon mémoire, et m'excuse auprès de lui pour le quiproquo que cela a entraîné. De même, je remercie Frank Nielsen qui était d'accord pour faire parti du jury mais qui a eu malheureusement un contre-temps de dernière minute. Ainsi, je suis redevable à Gilles qui a bien voulu le remplacer, bien que cette thèse s'inscrit dans un domaine différent du sien.

Merci à mes collègues et amis que j'ai fréquentés tous les jours, ou presque, au laboratoire d'informatique de Polytechnique, et qui donne à ce laboratoire son côté vivant et convivial. Je n'ai pas connu d'autre labo où l'ambiance était aussi sympathique et agréable. En particulier, je pense à mes collègues doctorants ou anciens doctorants : Alexis, Antoine, Carlos, David, Ivan, Luca, Nicolas, Olivier, Romain, Sylvain, Thang, Uli, Vincent et Vivek pour le temps passé ensemble, dans le même bureau, autour d'un café, d'une bière, d'un jeu de carte, à discuter science, politique, et tout autre sujet nous permettant de refaire le monde. Aux membres de mon équipe, Algopt : Christoph, David, Philippe, Vincent et Yann, pour les blagues nulles mais qui me font rire quand même, pour l'argent amassé lors des soirées poker qui m'a permis d'aller faire les courses le lendemain, et bien sûr pour les discutions scientifiques parfois bien animées et leur aide indiscutable durant ces trois années de thèse. Merci aux ITA : Corinne, James et Matthieu pour avoir été toujours là pour m'aider, me renseigner et parfois même faire tout ceci en prenant en compte les contraintes pas toujours évidentes que je leur donnais. Merci au corps enseignant avec lequel j'ai travaillé, notamment Fabrice, Gilles, Jean-Christophe, Olivier et Xavier, et un merci spécial pour Thomas qui m'a beaucoup soutenu, et bien au delà du simple cadre de l'enseignement.

Durant cette thèse, j'ai eu la chance de travailler avec des chercheurs dont les connaissances, la rapidité de compréhension des choses et l'intuition mathématique forcent le respect, et j'ai pu apprendre beaucoup auprès d'eux. Je pense notamment à Gustav et Manuel avec qui j'ai travaillé sur les sujets traités respectivement par les chapitres 4 et 3 de ce mémoire de thèse.

Participer avec eux à ces projets de recherche a été et continue d'être une expérience aussi enrichissante qu'excitante. Bien entendu, je pense aussi à mon principal collègue et directeur de thèse Miki dont l'étendue des connaissances scientifiques m'étonnera toujours, et qui a su durant cette thèse me guider vers des questions passionnantes qui aboutissent aujourd'hui à la constitution de ce mémoire.

Mais qu'aurais-je pu produire sans l'attention et le soutien de mon entourage ? Ainsi, je remercie mes parents, qui m'ont toujours soutenu dans mes choix, quels qu'ils soient, et m'ont laissé une grande liberté me permettant de satisfaire ma curiosité et d'arriver là où je suis aujourd'hui. Merci également à mes sœurs, Sandra et Sophie, pour leur gentillesse et leur attention, chacune à leur manière. Je remercie aussi toute ma famille de ne pas avoir été trop insistant pour que je leur explique mon domaine de recherche, problème assez épineux lorsque l'on est informaticien théoricien et qui n'est pas toujours facile à contourner.

Nous sommes tous également entourés d'amis, piliers indispensables pour soutenir les épreuves de la vie. Je pense à mes amis d'Orléans, ceux de longue date, voire de très longue date comme Sébastien que je connais depuis l'âge de... deux ans ! Des amis qui viennent du collège, du lycée, de la fac, des amis comme on en a pas beaucoup et sur qui vous pouvez toujours compter. Les plus anciens : Alexandre, ma "petite sœur" Alice, Charles, Émilie, Olivier et Flora, les Sébastiens, Virginie et enfin Yves. Les camarades de prépa : JF, John et Manu. Les camarades de fac : Adrien et Laetitia, MC et Fabien, Seb et Ketty. Shifty l'inclassable ! Les camarades de la Cellule Orléanaise : Arnaud, Clément et Fouad. Tous ont été là, à mes côtés, dans les périodes difficiles traversées pendant cette thèse. C'est quelque chose que je n'oublierai jamais.

Au delà d'Orléans, je tiens à remercier mes amis que j'ai connus sur Marseille et que j'ai eu la chance de continuer à voir sur Paris. Nicolas et Sara-Jane dont j'apprécierai toujours le flegme et la vivacité, Vanessa qui a été et restera quelqu'un d'important pour moi, et bien entendu Laurent, personne que l'on gagne à connaître autant sur le plan scientifique qu'amical et humain. Lui aussi fut présent à des moments cruciaux de ma thèse, et je l'en remercie.

Merci à mes amis parisiens : Gaël pour nos conversations profondes et enrichissantes sur des sujets bien différents, même si j'avais parfois du mal à le suivre, mes adorables voisins Ambre et Loïc que j'ai eu le plaisir de connaître et fréquenter en dépis de l'anonymat de rigueur à Paris, les membres de mon club de karate à l'X et de ju-jitsu sur Paris, malheureusement trop nombreux pour que je les nomme ici.

Merci à mes camarades de X'Doc auprès de qui j'ai passé deux années formidables, et dont le dynamisme et l'enthousiasme étaient communicatifs. Merci en particulier à l'équipe 2007 : Andreea, Cécile, Franck, Gaëlle, Gwenaël, Michel, Liêm, Pierre et Pierrick. J'ai passé avec eux des moments exceptionnels et inoubliables. Merci également à l'équipe 2008 : Anirban, Antoine, David, Éric, Larbi, Mahsa et Thomas, avec qui j'ai passé ce grand moment qu'était l'organisation de la venue de Jorge Cham à l'école.

Merci également à Barney Stinson pour être aussi awesome, et à Jamie Steven, chercheur astrophysicien à qui je dois le style de présentation de ce mémoire.

Enfin, merci à ma petite Mari pour être là, tout simplement, et pour tout ce qu'elle m'apporte. Elle me rappelle que la science est une chose très importante dans ma vie, mais pas un tout. À elle, merci.

Pour terminer, j'aimerais tout particulièrement remercier les professeurs Siva Anantharaman et Gérard Ferrand de l'Université d'Orléans qui m'ont fait découvrir par hasard ce domaine qu'est l'informatique théorique en licence, et qui a entraîné cette passion pour cette science qui ne m'a plus lâché par la suite, et la professeure Nadia Creignou de l'Université de la Méditerranée, à qui je dois beaucoup et sans qui cette thèse à Polytechnique n'aurait pas été possible.

私の最愛の人

Contents

Computer Science is no more about computers than astronomy is about telescopes.

Edsger Dijkstra (1930–2002)

1 Introduction

It is difficult to determine borders between Theoretical Computer Science (TCS) and Mathematics. So far no clear separations have been made, and everybody has its own way to express what TCS is. From my point of view, this science belongs to Mathematics and could be defined as follow: *Theoretical Computer Science is the science which studies and treats mathematical problems in an automatic way.* My opinion is that this science is completely independent from these machines that we call computers: they are consequences of TCS applications. The computer scientist Edsger Dijkstra said "*Computer Science is no more about computers than astronomy is about telescopes*". This sentence is deeply true: computers are just tools for TCS, but not studied objects.

Can TCS results be always applied to computers or other concrete purposes? Hopefully no. Things which interest me the most in TCS are of theoretical aspect. Another sentence I completely agree from a famous scientist is the following one, from the physicist Richard Feynman: "*Physics is like sex: sure, it may give some practical results, but that's not why we do it*". This is right for all theoretical sciences, like TCS. My vision of Mathematics and theory in general is close to Hardy's one. Indeed, the mathematician Godfrey Hardy found that what is beautiful in Mathematics is its "uselessness", that is, when theories in Mathematics do not find concrete applications in the material world, by opposition of the "idea world" as he certainly would call it himself: "*The mathematician's patterns, like the painter's or the poet's must be beautiful; the ideas, like the colours or the words must fit together in a harmonious way. Beauty is the first test: there is no permanent place in the world for ugly Mathematics. [...] A science is said to be useful if its development tends to accentuate the existing inequalities in the distribution of wealth, or more directly promotes the destruction of human life. [...] I have never done anything 'useful'. No discovery of mine has made, or is likely to make, directly or indirectly, for good or ill, the least difference to*

the amenity of the world... Judged by all practical standards, the value of my mathematical life is nil; and outside mathematics it is trivial anyhow. I have just one chance of escaping a verdict of complete triviality, that I may be judged to have created something worth creating. And that I have created something is undeniable: the question is about its value" in "A Mathematician's Apology" (1940). However, theory is of course useful in the sense that it always finds applications somewhere, for theory itself for instance, and sometimes for some applications. Again, Hardy said "*pure Mathematics is on the whole distinctly more useful than applied. For what is useful above all is technique, and mathematical technique is taught mainly through pure Mathematics*". I think that, *unfortunately*, every theoretical result finds one day directly or indirectly a way to be applied concretely.

Theoretical Computer Science contains many branches such as algorithms, combinatorics, logic, proof theory, ... The field for which we have a special interest in this thesis is computational complexity. Intuitively, what is it? It is the field where we study intrinsic difficulty of solving mechanically a given problem. Roughly speaking, we distinguish easy problems, also named tractable problems, from hard ones, named intractable problems. For instance, imagine a maze for mouses starting by a unique path which meet an intersection at a certain point: the path is divided in two ways (the left and right way). Both paths will also be divided in two ways, and so on. We obtain a maze corresponding to what we call a binary tree. Each path ends either by an exit or a dead-end. Place a mouse at the beginning of such a maze: the mouse has no smarter way than to try each path before finding the exit (if it exists). Thus, to know if there exists an exist in a maze with n intersections the mouse has to run in 2^n different paths in the worth case (that is, if the mouse is unlucky). This problem of deciding if a given maze contains an exit or not is consider as a difficult one (for a mouse) since the number of tries is exponential in the size of the maze. Assume now we have an overtrained mouse which is able to understand and to follow instructions like "At the first intersection take left, then left, then right, then left, ...". If we put this mouse at the starting line of the maze with a valid sequence of instructions (valid in the sense that the number of given instructions corresponds exactly to the number of intersections the mouse will cross), thus the mouse will be easily able to decide if the given path leads to an exit or not: the mouse has just to cross $\log n$ intersections (that is, the height of the binary tree). This problem is consider as an easy one (for everybody) since the number of tries is a polynomial in the size of the maze (here, even a logarithm).

In this thesis, we focus on complexity of mathematical problems called

Constraint Satisfaction Problems, or CSP for short. Again, we illustrate this notion with an example. A classical one is the scheduling of classes in a high school: at the beginning of each academic year, one has to establish a schedule which takes into account some constraints. For instance, a class cannot have two lessons at the same moment, as well as a professor cannot be in two classrooms at the same moment, or two professors in the same classroom, etc. Finding a coherent scheduling which verifies all constraints of this type is a Constraint Satisfaction Problem. This scheduling problem, as well as CSP problems in general, are very difficult problems.

However, we can consider the parametric version of the CSP problem, denoted CSP(S) where the parameter S is a set of relations which represent solutions of the constraints we are allowed to use to built an instance of our problem. There exist some sets of relations which implies the tractability of CSP(S), such as sets of Horn relations for instance. These parameters are what we call "islands of tractability", lost in an ocean of intractability. One can wonder if there exists parameters leading to intermediate situations in between tractability and intractability of the considered problem, that is, if there exists a parameter S such that CSP(S) is neither in P nor NP-complete, that is, tractable or intractable. If the answer to this question is "no", then one can wonder if there exists a way to split the set of all parameters into two parts, one leading to tractable problems, the other one to intractable problems, such that the borderline is clear and computable. In the 70's, Schaefer answered these questions in [Sch78], showing the full classification of the complexity of CSP(S) over a Boolean domain thanks to a Dichotomy Theorem, telling that CSP(S) is either in P or NP-complete depending a simple dichotomy criterion on S.

The goal of research in complexity of CSP is to extend Schaefer's result to other cardinalities of a fixed domain, or to prove that such borderline does not exist. So far, known results in this field exhibit a borderline between tractable and intractable problems, leading to the conjecture that such a borderline always exists. Indeed, Feder and Vardi conjectured in [FV98] the existence of a Dichotomy Theorem for the problem CSP(S) for every finite domains, that is, considered a finite domain D and a set S of relations on D, the problem CSP(S) is either in P or is NP-complete. We will now tell why an extension of Schaefer's theorem over higher cardinality of domain is such a difficult problem.

Schaefer did not realize he could use a very powerful tool to prove his theorem: "*The work presented here is similar in spirit to the classification by Post [...] but the generation operations are quite different, and to the best of our knowledge, none of the particulars of Post's proof carry over to this*

work" in [Sch78]. Indeed, we know today that the Post lattice is a very useful tool and elegant way to prove the Schaefer's Dichotomy Theorem. However, before introducing the Post lattice, we need to present the connection between the computational complexity of CSP and universal algebra.

One of the most exploited way to study the complexity of CSP is toward tools brought by universal algebra. The link between the complexity of CSP and universal algebra has been revealed by Jeavons *et al.* in papers [JC95], [JCG97] and [Jea98]. In their papers, they have shown that if a set S of relations is closed under functions, named polymorphisms, it always leads to the tractability or the intractability of CSP(S), and this, for every finite domain. Moreover, Jeavons *et al.* proved that, given a set S of relations, the set of polymorphisms of S form an algebraic structure named clone (even if it is a slight abuse of language, we can denoted it as "the clone of S"). The set of clones in a Boolean domain ordered by inclusion give us the Post lattice, which is a countable infinite lattice fully characterized by Post in [Pos41]. Thanks to this lattice, we can rewrite Schaefer's Dichotomy Theorem with a characterization considering only polymorphisms of the parameter S. Indeed, even if this lattice is (countable) infinite, results from Jeavons *et al.* permit us to deduce, for a given set S of relations, if CSP(S) is in P, then for all sets S' of relations such that the clone of S' contains the clone of S we have CSP(S') in P, and if CSP(S) is NP-complete, then for all sets S' of relations such that the clone of S' is contained in the clone of S we have CSP(S') NP-complete. A survey presenting the possibilities offered by the Post lattice can be found in [CV08].

Nowadays, research in this field strongly rely on universal algebra to characterize the complexity of CSP problem and its extensions or logical problems in general. The reader can find a collection of survey articles about the study of the complexity of such problems with the help of universal algebra in [CKV08]. Almost thirty years after Schaefer's Theorem, Bulatov proved in [Bul06a] a Dichotomy Theorem for the complexity of CSP(S) over a ternary domain. One can wonder why it took so much times to extend Schaefer's results. The principal reason is that a powerful tool likes Post lattice does not exist anymore. Indeed, Yanov and Muchnik proved in [YM59] that the number of these clones is uncountable for domains of cardinality higher than or equals to three, leading thus to an uncountable lattice. Techniques presented in the last paragraph are then no more available.

Since the complexity of CSP(S) over a finite domain (of arbitrary cardinality, or even of cardinality greater than three) are very difficult problems, one can avoid the difficulty due to the uncountability of the clone lattice by considering some restriction or extension of the original problem. Hence,

one way to make easier the problem is to consider a sub-problem allowing a countable lattice. For this, one can choose among others to enrich the formalism of CSP problems like in [Mar06] and [Mar08], or to force a certain structure on the parameter S like in [CCHS08] and [CHKS08], or to force a certain structure on the domain D like in [Sch78]. These two first ideas have been chosen as starting ideas of this thesis.

1.1 Outline of this thesis

Chapter 2 is the chapter of preliminaries where we formally define each mathematical notion studied or used in several chapter of this thesis. Notions which are specific to a chapter will be defined in the considered chapter.

Chapter 3 will present results obtained for Monotone Constraint Satisfaction Problems. In this chapter, we have chosen to enrich the formalism of CSP with the disjunction, that is, constraints of a formula are not anymore logically linked by conjunctions only but by a conjunction or a disjunction. This extension allowed us to have a countable and even finite clone lattice, which was the original argument to study Monotone CSP. Even if the lattice does not finally help us, we obtain a Dichotomy Theorem for the complexity of the parametric monotone CSP problem over every finite domains. This result has also been proved independently by Martin in the CoRR paper [Mar06]. It has been published in [HR09] and contributes to the study of model-checking of fragments of the first-order logic (see [Mar08]). We also proved a Dichotomy Theorem for Monotone CSP over countable infinite domains. This study has been published in [BHR09] and is in line the study of constraint problems over infinite domains (see for instance the survey [Bod08]).

Chapter 4 will present results obtained for co-Boolean CSP. We choose here to force the following structure to the parameter S: Let f be a unary function on a finite domain D with a co-domain of size two. The graph of f, denoted $f^\bullet$, is the relation $f^\bullet = \{(x, f(x)) \mid x \in D\}$. We force in this chapter the parameter S to be a set of graphs of unary functions. Thus, an instance of CSP(S) is a conjunction of constraints of the form $f(x) = y$. This restriction leads also to a countable (and again even finite) clone lattice. However, like for Chapter 3, this lattice was again too complicated to be efficiently used. This difficulty does not forbid us to prove a Dichotomy Theorem for the complexity of homogeneous co-Boolean CSP where all unary functions composing our constraints share the same Boolean co-domain. This theorem holds for every finite domain. We also present the beginning of a

proof for a conjectured Dichotomy Theorem for the complexity of co-Boolean CSP in general over a finite domain. These works have been done with Miki Hermann and Gustav Nordh and should be submitted soon. It is in line with the study of CSP over unary functions where an instance of this problem is a conjunction of constraints of the form $f(x) = y$ where f is a unary function (with no condition on the co-domain). This problem has been proved by [FMS04] to be as general (and then as difficult) as the general CSP problem, that is, there exists a satisfiable instance of the general CSP problem if and only if there exists a satisfiable instance of the CSP problem over unary functions.

Chapter 5 certainly contains the most important contributions in this thesis. It is in line with works of Feder *et al.* since it deals with the complexity of CSP(S) over unary functions. This chapter presents the method of kernel width, an innovating method which could be very useful to fully characterize the complexity of such problem or some of its sub-problems. It contains three main results (not with the same impact and importance). First, we show that, for the CSP(S) problem over unary functions on every finite domain, the linear kernel width implies the tractability of CSP(S). This is the first result available on every cardinality of finite domain using the kernel width. Moreover, this result was not known so far. Then, we present a Dichotomy Theorem of the complexity of the CSP(S) problem over unary functions on a ternary domain via a criterion based on the kernel width. Despite the fact that this result was already cover by Bulatov's result, this is the first Dichotomy Theorem proved via the kernel width method. Finally, we prove the NP-completeness of a specific homogeneous co-Boolean CSP(S) problem over every finite domain D if the kernel width is higher than or equal to the cardinality of D. Again, this result is cover by our result on homogeneous co-Boolean CSP in Chapter 4, but it is the first result of NP-completeness with a criterion based on the kernel width. This result leads us to the conjecture that, if this criterion is verified, the CSP(S) problem over unary functions is intractable.

2
Preliminaries

In this chapter, we formally define the most general used and studied notions in this thesis. Specific notions will be introduced at the beginning of each chapters.

2.1 Computational complexity

Computational complexity is the study of intrinsic complexity, or difficulty, of mathematical problems which can be treat automatically. We measure this difficulty regarding to needed resources. These resources can be time or space needed by a computational model to solve a given problem. Both time complexity and space complexity are measured regarding to the input size of the considered problem. Time complexity of a problem p is measured with the number of needed operations to answer p.

We introduce in this section usual complexity classes of decision problems which are L, P and NP. A complexity class is the class of problems which can be solved within a certain time or space complexity. A decision problem is a problem where the answer is either "yes" or "no".

Before introducing complexity classes, we need to formalize the notions of language and Turing machine. We also present the notion of polynomial-time many-one reduction.

Let p be a decision problem, and A the set of instances of p. We call *positive instance* of p, or *solution* of p, an instance of p which leads to the answer "yes", and we denote by $A^+ \subseteq A$ the set of positive instances of p. We say that A^+ is the *language* of the problem p. Thus, an instance x of p is a solution of p if $x \in A^+$.

In this thesis, we deal with decidable languages. A language is *decidable* if a Turing machine recognizes it, and a Turing machine is a model of computation which can be see as a machine with one or more infinite tapes where moves a read-write head which acts regarding to the current state of the

machine and the current read symbol on the tape. The fact that a Turing machine contains one or more tapes does not change its working or properties in the sense that a Turing machine with only one tape is as powerful as a Turing machine with several ones, that is, if a language L is recognized by a Turing machine with several tapes, then there exists a Turing machine with one tape which also recognize L. We give here a formal definition of the Turing machine.

Definition 2.1.1 *A* Turing Machine *is a 6-tuple* $(Q, \Sigma, \delta, q_s, q_a, q_r)$ *where*

- Q *is the finite set of states of the machine,*
- Σ *is a finite alphabet which does not contain the special blank character* $\sqcup$*,*
- δ *is the transition function defined as*

$$\delta \colon Q \times (\Sigma \cup \{\sqcup\}) \to Q \times (\Sigma \cup \{\sqcup\}) \times \{L, R\},$$

- $q_s \in Q$ *is the start state,*
- $q_a \in Q$ *is the accept state,*
- $q_r \in Q$ *is the reject state, with* $q_r \neq q_a$*.*

Let's us explain how a Turing machine works. Let M be a Turing machine with one tape. M receives an input x which is a finite string composed of elements from the given alphabet Σ. Usually, we write $x := x_0 x_1 \dots x_{n-1}$ for a n-long string with x_0, x_1, …, $x_{n-1} \in \Sigma$. At the beginning, x is written on the tape of M, the current state of M is q_s, and the read-write head of M is placed on the case containing the first character x_0 of x. All cases of the tape which are not filled up with the input x contain the blank character $\sqcup$. Let's attribute a number to each case of the tape as follow: the start case containing x_0 at the beginning will be named c_0, and next cases to its right will be denoted by c_1 (containing x_1), c_2 (containing x_2), and so on. Cases to the left of c_0 will be denoted by c_{-1}, c_{-2}, and so on. These cases contain $\sqcup$ at the beginning.

Now we can run the machine M. A run of M is a succession of steps, which is the application of the function δ to the couple (q, e) where q is the current state and e the character reads by the head of M. The function δ output a triple (q', e', d) where q' is a state in Q, not necessarily different from q, e' a character in $\Sigma \cup \{\sqcup\}$ written in the case c_i where the head is

positioned, and d the direction toward we move the head. If $d = L$ holds, we move the head one step to the left, so to the case c_{i-1}, and on the opposite if we have $d = R$ we move it one step to the right, that is to the case c_{i+1}. The current state of the machine is now q', and the character reads by the head is the one contained in the case c_{i-1} or c_{i+1}, depending where we have move the head. If the current state is q_a, then the machine "accepts", *i.e.* the input belongs to the language recognized by M, and if the current state is q_r, then the machine "rejects", so the input is not in the language recognized by the machine. Notice that we can have the situation where a run never reaches the state q_a nor q_r. Then we say that the machine M loops, that is, it never halts. A language recognized by a Turing machine which loops on certain inputs is called a *semi-decidable* language, since when the machine loops, we actually never know if the machine loops or take some times to finish its computation.

We say that a Turing machine M is *deterministic* if its transition function δ is deterministic, that is, for every couple (q, e) in $Q \times (\Sigma \cup \{\sqcup\})$, $\delta(q, e)$ outputs a unique element in $Q \times (\Sigma \cup \{\sqcup\}) \times \{L, R\}$. If there exist some couple (q, e) for which the values of $\delta(q, e)$ constitute a set $V \subseteq Q \times (\Sigma \cup \{\sqcup\}) \times \{L, R\}$ which is not a singleton, then we say that M is *non-deterministic*.

One fundamental notion in TCS which is constantly used in this thesis is the notion of reduction of problems. We especially use the polynomial-time many-one reduction (also know as Karp reduction) defined as follow.

Definition 2.1.2 *A language $\mathcal{A}$ is* polynomial-time many-one reducible *to a language $\mathcal{B}$ if there exists a polynomial-time computable function $f : \mathcal{A} \to \mathcal{B}$ such that, for every string ω, we have $\omega \in \mathcal{A}$ if and only if $f(\omega) \in \mathcal{B}$. This is denoted by $\mathcal{A} \leq_p \mathcal{B}$, and we say that f is a* polynomial-time many-one reduction.

Thus, a polynomial-time many-one reduction from a problem p_1 to a problem p_2, denoted by $p_1 \leq_p p_2$, is a polynomial-time computable function transforming an instance of p_1 into an instance of p_2 such that x is a positive instance of p_1 if and only if $f(x)$ is a positive instance of p_2. Remark that, for every instance x of p_1, we must have the size of $f(x)$ polynomially bounded by the size of x.

Moreover, we say that two problems p_1 and p_2 are *polynomial-time equivalent* (in this thesis, *equivalent* for short), denoted by $p_1 \equiv_p p_2$, if both $p_1 \leq_p p_2$ and $p_2 \leq_p p_1$ hold.

Another reduction used in this thesis is the *logspace reduction*. Rather than formally define it, we can modify Definition 2.1.2 as follow: there is a

logspace reduction from a language $\mathcal{A}$ to a language $\mathcal{B}$, denoted by $\mathcal{A} \leq_{\log} \mathcal{B}$, if the function f is logspace computable, that is, if the Turing machine which compute f only uses a logarithmic number of cases on its working tape regarding to the size of the input. Two problems p_1 and p_2 are *logspace equivalent*, denoted by $p_1 \equiv_{\log} p_2$, if both $p_1 \leq_{\log} p_2$ and $p_2 \leq_{\log} p_1$ hold.

The reader can find another formal definitions of Turing machine and polynomial-time many-one reduction in [Sip97] or [Pap94].

Before introducing complexity classes used in this document, we need to explain the *big-O* notation, also named asymptotic notation.

Notation 2.1.3 *Let f and g be two functions $f, g\colon \mathbb{N} \to \mathbb{N}$. We write $f(n) = O(g(n))$ if there exist two positive integers c and n_0 such that for every $n \geq n_0$, the equation $f(n) \leq c.g(n)$ holds. Thus, we say that $g(n)$ is an asymptotic upper bound of $f(n)$.*

Consider a Turing machine M which recognize a language L. Roughly speaking, there exist two ways to consider the time complexity of M: the *average-case analysis* where one consider the average number of steps M needs to recognize an element x of L, and the *worst-case analysis* where one consider the maximal number of steps M needs to recognize an element x of L, that is, x is an input leading to the longest finite computation of M. Thus, the goal of the worst-case analysis is to determine an asymptotic upper bound verified for every input recognized by the machine. This thesis only deals with worst-case analysis. Similarly, when we will speak about space complexity, we will consider the worst-case analysis.

Example 2.1.4 *Let $f\colon \mathbb{N} \to \mathbb{N}$ be the function describing the running time of a Turing machine M, that is, f takes as its input the size of the input x of M such that x leads to the longest finite run of M. We denote by n the size of x, i.e. we have $n = |x|$.*

Assume that $f(n)$ is defined as follow:

$$f(n) = 4n^7 + 5n^4 + 2n^3 + n + 2$$

Then, we have $f(n) = O(n^7)$. Indeed, taking the highest order term of the polynomial without the constant coefficient 4 is sufficient to obtain an asymptotic upper bound. Thus, we say that M runs in $O(n^7)$. ■

Class P is the class of problems which can be solve deterministically in polynomial-time, *i.e.*, there exists a deterministic Turing machine M where the maximal number of needed steps to solve it is a polynomial on the input

size n where n^k is the highest term, for some fixed k. Such problems are decidable in $O(n^k)$. Thus, if we denote by TIME(x) the class of problems deterministically decidable in $O(x)$, P can be defined as follow:

$$\mathrm{P} = \bigcup_k \mathrm{TIME}(n^k)$$

Very similarly, the class NP is the class of problems which can be solved non-deterministically in polynomial-time, *i.e.*, there exists a non-deterministic polynomial-time Turing machine solving the given problem.

Then, if we denote by NTIME(x) the class of problems non-deterministically decidable in $O(x)$, we can define the class NP from the same way we have defined P:

$$\mathrm{NP} = \bigcup_k \mathrm{NTIME}(n^k)$$

To give an intuition, the class NP is the class of problems which are difficult to answer, but for which it is easy to verify if a proposed answer is a solution or not.

It is clear that, given a problem p, there exists a non-deterministic polynomial-time Turing machine solving p if there exists a deterministic one. Therefore we have $\mathrm{P} \subseteq \mathrm{NP}$. The question to know if $\mathrm{NP} \subseteq \mathrm{P}$ is unknown and considered to be one of the most difficult problems in TCS[1].

A very important notion is the completeness of a complexity class. We often use this notion for the class NP, telling that a problem is NP-complete. Intuitively, an NP-complete problem p is a problem considered to be among the most difficult problems in NP since one can reduce every problem in NP to the problem p. More formally, a problem p is NP-complete if $p \in \mathrm{NP}$ and for every problem $p' \in \mathrm{NP}$ we have $p' \leq_p p$.

As least, we define the class L which is the class of problem where needed space to compute a problem for a deterministic Turing machine is within $\log n$ with n being the input size. Denoting by SPACE(x) the class of problems deterministically decidable in space $O(x)$, the class L can be defined as follow:

$$\mathrm{L} = \mathrm{SPACE}(\log n)$$

There exists a lot of computational classes in complexity theory, however all along this thesis we will only used these three classes, especially classes P and NP.

[1] The Clay Mathematics Institute had chosen in 2000 seven problems, that they called millennium problems, considered to be the most difficult open problems in Mathematics, which are: the Birch and Swinnerton-Dyer conjecture, the Hodge conjecture, the Navier-Stokes conjecture, the Poincaré conjecture (solved by Grisha Perelman in 2006), the Riemann hypothesis, the Yang-Mills theory and the P versus NP problem.

2.2 Constraint Satisfaction Problem

Constraint Satisfaction Problems (or CSP for short) constitute a common formalism to describe many algorithmic problems from combinatorics and graph theory, artificial intelligence, computational molecular biology, etc. CSP notion appears for the first time in 1974 in Montanari's article [Mon74], and is now present in many areas of computer science: CSPs have attracted a lot of attention, in particular in combinatorics, artificial intelligence, and finite model theory; we refer to the recent monograph with survey articles on this subject [CKV08].

Before introducing what is a CSP instance, we need the notion of constraint. A k-ary *constraint* is a function taking k variables in input and outputs either "true" of "false" when we assign a value to each variable. In logic, a constraint is named a predicate.

A CSP instance can be seen as a first-order formula defined inductively in the following way:

(1) TRUE and FALSE, respectively denoted by $\top$ and $\bot$, are formulas;

(2) a constraint $C(x_1, \ldots, x_k)$ is a formula;

(3) if φ_1 and φ_2 are formulas, then $\varphi_1 \wedge \varphi_2$ is a formula;

(4) finally if φ is a formula containing a free variable x then $\exists x \varphi$ is a formula.

Notice that in the literature, first-order formulas allowing conjunctions and existential quantification only are called *primitive positive formulas*.

We consider a CSP problem on a fixed domain D. The valuation of variables is a function I named *interpretation* such that we have $I \colon V \to D$, with V the set of variables. Thus, the formula $\varphi(x_1, \ldots, x_k)$ is satisfiable if there exists an interpretation I such that for each $x_j \in \{x_1, \ldots, x_k\}$, replacing x_j by $I(x_j)$ in φ give us a true sentence. We say that I is a valid interpretation for φ and write $I(x_1), \ldots, I(x_k) \models \varphi(x_1, \ldots, x_k)$.

Thus, we can formally define the Constraint Satisfaction Problem as follow:

Problem: CSP
Input: A formula $\varphi(\vec{x})$.
Question: Is the formula $\varphi(\vec{x})$ satisfiable?

CSPs are well-known to be NP-complete in general. However, one can find some sub-problems which are tractable, that is, in P. These "islands

of tractability" appear when we restrict the type of allowed constraints to build a CSP instance. It should be noticed that such islands of tractability also appear with some structural restrictions, like for instance when we force a structure on the domain: this leads among others to the Dichotomy Theorems of Schaefer on Boolean domains and Bulatov on ternary domains. The restriction of allowed constraints leads us to the parametric version of CSPs. Before formally defining parametric CSPs, we need to introduce the notion of relation.

A n-ary *relation* R on domain D is a subset of D^n. One can built a constraint $R(x_1, \ldots, x_n)$ built upon the relation R, such that R is the set of valid interpretations for $R(x_1, \ldots, x_n)$ (see Example 2.2.1). A tuple t of an n-ary relation R is then an element of D^n. We denote the i-th coordinate, or position, in t by $t[i]$.

Example 2.2.1 *On the Boolean domain* $D = \{0, 1\}$.

Take for example these three relations:

- $R_1 = \{(0,0); (1,1)\}$
- $R_2 = \{(0,1); (1,0)\}$
- $R_3 = \{0,1\}^3 \setminus \{(0,0,0); (1,1,1)\}$

Respectively, the corresponding constraints are the following:

- $= (x, y)$, *usually written "*$x = y$*",*
- $\neq (x, y)$, *usually written "*$x \neq y$*",*
- *and* $\text{NAE}(x, y, z)$ *which correspond to the ternary constraint "not-all-equal".*

Moreover, let the tuple $t \in R_2$ *such that* $t = (1, 0)$. *We have then* $t[0] = 1$ *and* $t[1] = 0$. ■

Thus, the parametric version of CSP is the problem to decide if there exists a valid interpretation for a given CSP instance built over a set S of relations, that is, where solutions of constraints correspond to a relation in S. We say that S is a *template* and we denote by $\text{CSP}(S)$ the parametric version of CSP over S.

Problem: $\text{CSP}(S)$
Input: A formula $\varphi(\vec{x})$ built over relations in S.
Question: Is the formula $\varphi(\vec{x})$ satisfiable?

Example 2.2.2 *Let D be the domain $\{0,1,2,3\}$, and let S be the set of relations $\{=,\neq,\leq\}$. Then, we can have for example the following instance of* $\text{CSP}(S)$*:*

$$\varphi(a,b,c,d) := (a=c) \wedge (a \neq b) \wedge (a \leq d) \wedge (b \leq d) \wedge (c \neq b)$$

An example of valid interpretation I for $\varphi(a,b)$ can be $I(a)=1$, $I(b)=2$, $I(c)=1$ and $I(d)=3$. ■

We call *S-formula* a formula where constraints are built upon relations in S.

From now on, we will focus exclusively on the study of parametric $\text{CSP}(S)$ problems, denoted as CSP for short in the following. Our goal is, given a framework, to detect which templates imply tractability and to prove, if possible, that all other templates imply intractability.

2.3 Universal algebra

One powerful way to study complexity of CSP goes through algebraic methods. Links between complexity of CSP and universal algebra are very well presented in many papers such as in [BCRV03] and [BCRV04], and in [JCP98], [BKJ00] and [BJK05]. Advantages of using universal algebra for our study are multiple: among others, it gives us a unified framework based on a rich algebraic theory to characterize sets of structures and classes of problems, links each efficient algorithm to a structural property such as the closure under certain polymorphisms and allow us to give hardness proof without reduction.

Let's denote the arity of a function f by $\operatorname{ar} f$ and the arity of a relation R by $\operatorname{ar} R$.

First, we introduce a notion for functions, which constitute fundamental mathematical objects in this thesis. Let f be a function of arity k on a fixed size domain D such that we have $f\colon D^k \to D$. The *range* of f, denoted by $\operatorname{ran} f$, is the set $\{f(x) \mid x \in D\}$. We write $f(A)$ for a subset $A \subseteq D$ to denote the set $\{f(a) \mid a \in A\}$.

Let R be an n-ary relation on a domain D. We say that a k-ary function $f\colon D^k \to D$ is a *polymorphism* of R if f applied component-wise to k tuples in R outputs a tuple in R. Formally, let R be a n-ary relation and let $t_1,\ldots,t_k$ be k tuples in R. Let f be a k-ary function such that, applied component-wise to tuples $t_1,\ldots,t_k$, it outputs a tuple t.

$$\begin{array}{cccccccc} & t_1 & t_2 & \dots & t_k & & t \\ & \shortparallel & \shortparallel & & \shortparallel & & \shortparallel \\ \mathrm{f}(& a_1^1 & a_1^2 & \dots & a_1^k &) = & a_1 \\ \mathrm{f}(& a_2^1 & a_2^2 & \dots & a_2^k &) = & a_2 \\ & \vdots & \vdots & & \vdots & & \vdots \\ \mathrm{f}(& a_n^1 & a_n^2 & \dots & a_n^k &) = & a_n \end{array}$$

If, for every k tuples $t_1, \dots, t_k$ in R, f outputs a tuple t which belongs to R, then we say that f is a polymorphism of R. In this thesis, we will often write $f(t_1, \dots, t_k)$ as a shortcut to express the fact that we apply f component-wise to tuples $t_1, \dots, t_k$. Reciprocally, we say that R is an *invariant* of f. By extension, we say that p is a polymorphism of a set S of relations if p is a polymorphism of each relation R in S. Reciprocally, we say that R is an invariant of a set F of functions if R is an invariant of each functions f in F. All along this thesis, we will say that a set S of relations is *closed under* a function f if f is a polymorphism of S.

The study of constraint satisfaction problems often uses the notion of clones and co-clones. A *clone*, or a *functional clone*, is a set of functions containing the identity and closed under composition. The smallest clone containing the functions F is denoted by $[F]$. Similarly, a *co-clone*, or a *relational clone*, is a set of relations containing the equality $eq = \{(d, d) \mid d \in D\}$ and closed under conjunction, variable identification and existential quantification. The smallest co-clone containing the set of relations S is denoted by $\langle S \rangle$.

We introduce now the Galois connection, which is constantly used to characterize complexity of CSP. Let X, Y be two arbitrary sets, and let $m_1 \colon X \to Y$ and $m_2 \colon Y \to X$ be two mappings such that we have:

- For all $x, y \in X$ such that $x \leq y$, $m_1(y) \leq m_1(x)$ holds,
- For all $x, y \in Y$ such that $x \leq y$, $m_2(y) \leq m_2(x)$ holds,
- For all $x \in X$ and $y \in Y$, $x \leq m_2(m_1(x))$ and $y \leq m_1(m_2(y))$ hold.

We say that m_1 and m_2 present a *Galois connection* between the ordered structures $(X, \leq)$ and $(Y, \leq)$.

Now; consider a function f. We denote by $\operatorname{Inv} f$ the set of invariants by f. Thus, observe that $\operatorname{Inv} f$ is a set of relation. For F a set of functions, we write $\operatorname{Inv} F$ the set of invariants by each function in F such that $\operatorname{Inv} F = \bigcap_{f \in F} \operatorname{Inv} f$. Similarly, given a relation R, we denote by $\operatorname{Pol} R$ the set of polymorphisms of R. Then, $\operatorname{Pol} R$ is a set of functions. For a set S of

relations, we denote by $\operatorname{Pol} S$ the set of polymorphisms of every relation in S. Thus, we have $\operatorname{Pol} S = \bigcap_{R \in S} \operatorname{Pol} R$. It is well-known in universal algebra that for every set S of relations and set F of functions, sets $\operatorname{Pol} S$ and $\operatorname{Inv} F$ form respectively a clone and a co-clone. Moreover, Pol and Inv present the good properties to form a Galois connection: consider $\mathcal{F}$ and $\mathcal{S}$ respectively the set of sets of functions and the set of sets of relations over a domain D. Thus, $\operatorname{Pol}\colon \mathcal{S} \to \mathcal{F}$ and $\operatorname{Inv}\colon \mathcal{F} \to \mathcal{S}$ form a Galois connection between structures $(\mathcal{F}, \subseteq)$ and $(\mathcal{S}, \subseteq)$. This implies that, the bigger the set S is, the smaller $\operatorname{Pol} S$ is, and vice versa. Thus, if a set S is closed under a polymorphism p, then every subset S' of S is also closed under p. In addition, following [Gei68] and [BKKR69] we have for every F and S the equations $[F] = \operatorname{Pol}\operatorname{Inv} F$ and $\langle S \rangle = \operatorname{Inv}\operatorname{Pol} S$.

For a fixed finite domain D of cardinality n, the set of clones and the set of co-clones ordered by inclusion, respectively $(\mathcal{F}, \subseteq)$ and $(\mathcal{S}, \subseteq)$, form two lattices. On a Boolean domain, the lattice of clones is called *Post lattice* since Emil Post described in his paper [Pos41] the full (infinite) list of Boolean clones and their inclusions. The reader can find a modern study of Post's results in [Lau06]. We can now explain how the universal algebra helps to determine the complexity of CSP.

The link between universal algebra and complexity of CSP is due to Jeavons *et al.* in the 90's (see [JC95], [JCG97], [Jea98], [JCP98] and [JCG99]). It is possible to determine the complexity of $\textsc{csp}(S)$ via algebraic properties thanks to the following propositions:

Proposition 2.3.1 *Let S_1 and S_2 be two finite sets of relations such that the inclusion $S_1 \subseteq \langle S_2 \rangle$ holds. Then we have the reduction $\textsc{csp}(S_1) \leq_p \textsc{csp}(S_2)$.*

This proposition allows us to conclude that, if $\textsc{csp}(S_2)$ is in P then every finite subset S_1 of $\langle S_2 \rangle$ gives a parameter which implies the polynomiality of $\textsc{csp}(S_1)$. On the opposite, if there exists a finite subset S_1 of $\langle S_2 \rangle$ such that $\textsc{csp}(S_1)$ is NP-complete then we can conclude that $\textsc{csp}(S_2)$ is also NP-complete. From this proposition, we have the following corollary.

Corollary 2.3.2 *Let S_1 and S_2 be two finite sets of relations such that $\langle S_1 \rangle = \langle S_2 \rangle$ holds. Then we have $\textsc{csp}(S_1) \equiv_p \textsc{csp}(S_2)$, that is, $\textsc{csp}(S_1)$ and $\textsc{csp}(S_2)$ are equivalent.*

Thus, this corollary allows us to focus on co-clones to study the complexity of CSP, since this complexity does not depend on a set of relations, but on the co-clone it generates.

Thanks to the Galois connection between clones and co-clones, we have the following proposition and corollary which allow us to work not on co-clones, but on clones to describe the complexity of CSP.

Proposition 2.3.3 *Let S_1 and S_2 be two finite sets of relations such that the inclusion* $\operatorname{Pol} S_2 \subseteq \operatorname{Pol} S_1$ *holds. Then we have the reduction* $\textsc{csp}(S_1) \leq_p \textsc{csp}(S_2)$.

Remark that the direction of the inclusion is $\operatorname{Pol} S_2 \subseteq \operatorname{Pol} S_1$. This is easily comprehensive since, due to the Galois connection, the bigger a set S of relations is, the smaller its set of polymorphisms $\operatorname{Pol} S$ is.

Corollary 2.3.4 *Let S_1 and S_2 be two finite sets of relations such that* $\operatorname{Pol} S_1 = \operatorname{Pol} S_2$ *holds. Then we have* $\textsc{csp}(S_1) \equiv_p \textsc{csp}(S_2)$.

The clone lattice is then helpful in the sense that, if we know the inclusion structure between clones and if we know the complexity of $\textsc{csp}(S)$ for a given S, we can conclude that:

- if $\textsc{csp}(S)$ is in P then, for each finite set S' of relations such that the clone $\operatorname{Pol} S'$ contains $\operatorname{Pol} S$, we have $\textsc{csp}(S')$ in P,
- and if $\textsc{csp}(S)$ is NP-complete then, for each finite set S' of relations such that the clone $\operatorname{Pol} S'$ is included in $\operatorname{Pol} S$, we have $\textsc{csp}(S')$ NP-complete.

I ask the reader to remember that what is most obvious may be most worthy of analysis. Fertile vistas may open out when commonplace facts are examined from a fresh point of view.

Lancelot Whyte (1896–1972)

3 Monotone CSP

The starting idea of this thesis was to study several aspects of CSP problems which avoid the uncountable clone lattice problem. Hence, the very first idea was to enrich the CSP formalism with the disjunction, that is, in addition of the conjunction, we allow the disjunction between constraints. This gives us problems named *Monotone* CSPs, or MCSP for short. Considering the problem MCSP(S) over a template S on a finite domain D, we known by Pöschel [Pös80] and [Pös04] and Krasner [Kra38] that such an extension implies the fact that every polymorphism of S is an endomorphism, that is, an unary function, allowing us to conclude that the clone lattice is now countable, and even finite, for every finite domain. CSPs with disjunction have been also considered from a different viewpoint using *disjunctive constraints* by Cohen, Jeavons, Jonsson and Koubarakis in [CJJK00], where disjunctive constraints have the form of the disjunction of two constraints.

Even if the study of the lattice revealed itself to be too complicated, our study of MCSP problems leads us to two Dichotomy Theorems. The first one is a Dichotomy Theorem for every finite domain cardinality shown in Section 3.1 and published in [HR09]. The second one is a Dichotomy Theorem for countable infinite domains, shown in Section 3.2 and published in [BHR09], which can be seen from two different points of view. Indeed, these instances of MCSP on a template S are both NP-hard and complete under polynomial-time many-one reduction for the class of reducible problems to CSP(S) under a non-deterministic polynomial-time many-one reduction.

The dichotomy theorem for finite domain cannot really be considered as something new since the result was implicitly known by the complexity part of the CSP community. Indeed, proof of this dichotomy has been made independently in [HR09] and in [Mar06]. However, from this study come up some new ideas around the kernel width notion which may be helpful to fully characterize the complexity of some CSP problems, or to give a sufficient property implying NP-completeness of CSP problems (see Conjecture 5.3.6)

3.1 Finite case

In this section, we study the complexity of parametric Monotone Constraint Satisfaction Problems MCSP(S) where MCSP is a generalization of CSP defined as follows: constraints are built upon relations in S, and we connect constraints each others by a conjunction or a disjunction. We present a dichotomy with simple conditions for the algorithmic problem MCSP(S) with S being a set of relations over a finite domain D.

A *formula* is defined inductively in the following way:

(1) TRUE and FALSE, respectively denoted by $\top$ and $\bot$, are formulas;

(2) a constraint $C(x_1, \ldots, x_k)$ built upon a k-ary relation R in S is a formula;

(3) if φ_1 and φ_2 are formulas, then $\varphi_1 \vee \varphi_2$ and $\varphi_1 \wedge \varphi_2$ are formulas;

(4) finally if φ is a formula containing a free variable x then $\exists x \varphi$ is a formula.

Notice that in this chapter, an S-formula will be a formula obtain as above and where constraints are built upon relations in S.

Remark 3.1.1 *We do not consider negation in our formulas because it brings us to an obvious generalization of the problem* CSP(S). *Indeed, Pöschel shows in his survey [Pös04] that, for the extension of* CSP *allowing logical connectors* $\{\vee, \wedge, \exists, \neq\}$ *and for every set* S *of relations consider as the template of this extended problem,* $\mathrm{Pol}\, S$ *is a set of functions containing permutations only (Theorem 2.3 in [Pös04] combined with results in [Pös80]). Thus, this generalization leads to* NP*-complete cases over every finite domain of cardinality greater than or equal to three, for every template* S.

We describe what is an interpretation in our context. Let S be a set of relations upon which we build our constraints. An *interpretation* $I \colon V \to D$ satisfies the constraint $C(x_1, \ldots, x_k)$ built upon $R \in S$, denoted by $I \vDash C(x_1, \ldots, x_k)$, if $(I(x_1), \ldots, I(x_k)) \in R$. It is extended to formulas in the following way:

(a) $I \vDash C_1(\vec{x}) \vee C_2(\vec{y})$ if $I \vDash C_1(\vec{x})$ or $I \vDash C_2(\vec{y})$,

(b) $I \vDash C_1(\vec{x}) \wedge C_2(\vec{y})$ if $I \vDash C_1(\vec{x})$ and $I \vDash C_2(\vec{y})$.

We note that for all formulas φ, we have $\top \vDash \varphi$ and $\bot \nvDash \varphi$. Thanks to the previous definition an of interpretation, now we can formally define the *Monotone Constraint Satisfaction Problem*.

Problem: MCSP(S)
Input: A formula $\varphi(\vec{x})$ built upon relations in S.
Question: Is the formula $\varphi(\vec{x})$ satisfiable?

Our study of Monotone Constraint Satisfaction Problems is made easier by the existence of a Galois connection among clones and co-clones. Pöschel showed in [Pös04] that applying disjunction on constraints implies that relations satisfying an instance of a MCSP problem are invariant under unary functions only. Polymorphisms of these relations are thus endomorphisms and we denote them by End instead of Pol. Let us denote respectively by $\mathcal{S}$ and $\mathcal{F}$ the sets of sets of relations and functions. Pöschel [Pös04] specifies that the mappings End: $\mathcal{S} \to \mathcal{F}$ and Inv: $\mathcal{F} \to \mathcal{S}$ present a Galois connection between the ordered structures $(\mathcal{S}, \subseteq)$ and $(\mathcal{F}, \subseteq)$. Moreover, for all sets of relations S and functions F, the identities $\operatorname{Inv}\operatorname{End} S = \langle S \rangle$ and $\operatorname{End}\operatorname{Inv} F = [F]$ hold. Pöschel points out that the sets $\langle S \rangle$ and $[F]$ are respectively a *weak Krasner algebra* and an *endomorphisms monoid*; however in this chapter, we will continue to name these structures co-clone and clone, respectively. In this scope we can consider the monotone constraint satisfaction problems as CSP defined upon weak Krasner algebras. The existence of the aforementioned Galois connection allows us to easily decide the complexity of monotone constraint satisfaction problems. The complexity analysis is based on the following result.

Proposition 3.1.2 *Let S_1 and S_2 be two finite non-empty sets of relations over the domain D, such that $eq \in S_1$ with eq the equality relation. The inclusion* $\operatorname{End} S_1 \subseteq \operatorname{End} S_2$ *implies* $\text{MCSP}(S_2) \leq_p \text{MCSP}(S_1)$.

Proof: From $\operatorname{End} S_1 \subseteq \operatorname{End} S_2$ follows $\langle S_1 \rangle = \operatorname{Inv}\operatorname{End} S_1 \supseteq \operatorname{Inv}\operatorname{End} S_2 = \langle S_2 \rangle$. Thus, every relation in $\langle S_2 \rangle$ can be obtained from relations in S_1, that is, every S_2-formula can be expressed by a S_1-formula. Since every co-clone (equivalent to a weak Krasner algebra) contains the equality relation eq and is closed under conjunction, disjunction, existential quantification and identification of variables, we immediately derive reductions

$$\text{MCSP}(S_2) \leq_p \text{MCSP}(S_2 \cup \{eq\}) \leq_p \text{MCSP}(S_1 \cup \{eq\})$$

Since $eq \in S_1$, we have $\text{MCSP}(S_1 \cup \{eq\}) = \text{MCSP}(S_1)$ and the result follows. □

According to Proposition 3.1.2, a complexity result proved for a set of relations S immediately extends to all relations in the co-clone $\langle S \rangle$, provided

that S contains the equality relation eq. Therefore from now on we always assume that the equality relation eq is included in every set S. Moreover, the presence of the quaternary relation

$$J = \{(x, y, z, w) \in D^4 \mid (x = y) \vee (z = w)\}$$

in the template S of a classical constraint satisfaction problem $\text{CSP}(S)$ transforms it to a monotone constraint satisfaction problem $\text{MCSP}(S)$, that is if $J \in S$ then $\text{CSP}(S) \equiv_p \text{MCSP}(S)$. We prove this through the next proposition.

Notice however that we can get rid of the equality relation in S, as describe in Section 3.2.

Proposition 3.1.3 *Let S be a set of relations on the domain D and consider the relation $J = \{(x, y, z, w) \in D^4 \mid (x = y) \vee (z = w)\}$. If $J \in S$ then* $\operatorname{Pol} S = \operatorname{End} S$.

Proof: Suppose that there exists a binary function $g \in \operatorname{Pol} S$ depending on both arguments, i.e., there exist values $a_0, a_1, a_2, b_0, b_1, b_2 \in D$ such that $a_0 \neq a_1$, $b_1 \neq b_2$, $g(a_0, a_2) \neq g(a_1, a_2)$ and $g(b_0, b_1) \neq g(b_0, b_2)$. Clearly, the vectors $j_1 = (a_0, a_1, b_0, b_0)$ and $j_2 = (a_2, a_2, b_1, b_2)$ belong to J, but the vector $(g(a_0, a_2),\ g(a_1, a_2),\ g(b_0, b_1),\ g(b_0, b_2))$ is absent from J. Therefore the function g cannot be a polymorphism of a set of relations containing J. From any function f of arity $\text{ar}(f) > 2$ we can always produce a binary function by variable identification. □

From the description of the Galois connections between weak Krasner algebras and endomorphisms monoids given above, it follows that a set S of relations where we have $\operatorname{Pol} S = \operatorname{End} S$ leads to a co-clone $\langle S \rangle$ closed under conjunction, disjunction, existential quantification and identification of variables. Thus we deduce for this result and Proposition 3.1.3 that $\text{CSP}(S) \equiv_p \text{MCSP}(S)$ holds if we have $J \in S$.

Since the number of endomorphisms over a finite domain is always finite, we are ensured to obtain a finite complexity characterization for MCSP.

We will exhibit a dichotomy of $\text{MCSP}(S)$ complexity characterized by a simple criterion. It is easier to prove this dichotomy for $\text{MCSP}(f(S))$ where f is a permutation keeping invariant the set S. We need the following proposition adapted from Jeavons [Jea98] showing that the problems $\text{MCSP}(S)$ and $\text{MCSP}(f(S))$ are logspace-equivalent.

Proposition 3.1.4 (Jeavons [Jea98]) *Let S be a set of relations on D and f be an unary function on D. Let $f(S) = \{f(R) \mid R \in S\}$. If each relation $R \in S$ is closed under f then* $\text{MCSP}(f(S))$ *is logspace-equivalent to* $\text{MCSP}(S)$.

Proof: Suppose that each relation $R \in S$ is closed under f. Let $\varphi(\vec{x})$ be an instance of $\text{MCSP}(f(S))$. We have a formula $\varphi(\vec{x})$ where constraints $C(x_1, \ldots, x_k)$ are constructed upon relations $R \in f(S)$. According to the hypothesis, all relations $R \in S$ are closed under f, *i.e.* the inclusion $f(R) \subseteq R$ holds for all $R \in S$. We deduce that $\varphi(\vec{x})$ is also an instance of $\text{MCSP}(S)$.

Take a formula $\varphi(\vec{x})$ being an instance of $\text{MCSP}(S)$. It can be transformed by a logspace reduction to an instance $\varphi'(\vec{x})$ of $\text{MCSP}(f(S))$ by replacing each constraint constructed upon a relation R by a constraint constructed upon $f(R)$. Moreover, because R is closed under f, we have $f(R) \subseteq R$ and we derive that all solutions of $\varphi'(\vec{x})$ are also solutions of $\varphi(\vec{x})$. Conversely, if h is a solution of $\varphi(\vec{x})$ then $f(h)$ is a solution of $\varphi'(\vec{x})$. Thus we have a logspace-equivalence among $\text{MCSP}(f(S))$ and $\text{MCSP}(S)$. □

We will study monotone constraint satisfaction problems $\text{MCSP}(S)$ through sets of functions F satisfying $\operatorname{Inv} F = S$. The following propositions prove that $\text{MCSP}(S)$ is polynomial if $[F]$ contains a constant function, and it is NP-complete otherwise.

Proposition 3.1.5 *Let F be a set of unary functions such that* $\operatorname{End} S = [F]$ *for a set of relations S. If $[F]$ contains a constant function then* $\text{MCSP}(S)$ *is polynomial.*

Proof: If the endomorphisms of S contain a constant function $f_d(x) = d$ for all $x \in D$, then for the set of relations $\langle S \rangle$ invariant under $[F]$, each relation $R \in \langle S \rangle$ contains a d-vector, *i.e.* a mapping of each variable to the value d. Therefore each instance of $\text{MCSP}(S)$ is satisfiable by a d-vector. □

Lemma 3.1.6 *Let S be a set of relations and F be a set of unary functions such that* $\operatorname{End} S = [F]$ *does not contain any constant functions. Let $f \in [F]$ be a function with the smallest cardinality of* $\operatorname{ran} f$. *Then* $\operatorname{End} f(S)$ *contains only permutations.*

Proof: Suppose there is a function $g \in \operatorname{End} f(S)$ not being a permutation. Necessarily g is not injective, *i.e.* the inclusion $\operatorname{ran} g \subsetneq \operatorname{ran} f$ holds. This is a contradiction with the fact that the cardinality of $\operatorname{ran} f$ is the smallest among all functions in $[F]$. □

Actually, if $f \in \operatorname{End} S$ such that $\operatorname{ran} f$ is one of the smallest cardinality among functions in $\operatorname{End} S$ ("one" of the smallest because one can have several different smallest ranges), $f(S)$ is called the core of S, and Lemma 3.1.6 is an already well-known result. However, we do not use a lot the notion of core in this chapter, that is why we do not introduce it here but in Chapter 4, where it reveals to be a fundamental notion.

Proposition 3.1.7 *Let F be a set of unary functions such that the clone $[F]$, equivalent to $\operatorname{End} S$ for a set of relations S, does not contain constant functions. Then* $\textsc{mcsp}(S)$ *is* NP*-complete.*

Proof: We adapt the proof of Proposition 5.6 from [Jea98]. Let $[F]$ be without constant functions. Let $f \in [F]$ be a function with minimal cardinality of its range $\operatorname{ran} f$. By Lemma 3.1.6, we know that $\operatorname{End} f(S)$ contains only permutations. Since $[F]$ does not contain constant functions, we also know that cardinality of $\operatorname{ran} f$ satisfies the condition $|\operatorname{ran} f| \geq 2$. We have to separate two cases.

If $|\operatorname{ran} f| = 2$, then we assume without loss of generality that $\operatorname{ran} f = \{0,1\}$. The set $\operatorname{End} f(S)$ contains only permutations on $\{0,1\}$. Let R_{NAE} be the relation $\{0,1\}^3 \setminus \{(0,0,0);(1,1,1)\}$. It is clear that relation R_{NAE} is closed under every permutations on $\{0,1\}$. Hence we have the inclusion $\operatorname{End} f(S) \subseteq \operatorname{End} R_{\text{NAE}}$. The relation R_{NAE} gives rise to the NOT-ALL-EQUAL-3SAT problem, known to be NP-complete. Indeed, it is well-known that $\textsc{csp}(S)$ is NP-complete if $R_{\text{NAE}} \in S$ (see [BCRV04] for instance). Since every instance of $\textsc{csp}(S)$ is obviously also an instance of $\textsc{mcsp}(S)$, we deduce that $\textsc{mcsp}(S)$ is also NP-complete. By Proposition 3.1.2 we conclude that $\textsc{mcsp}(f(S))$ is NP-complete.

Let now $|\operatorname{ran} f| \geq 3$. The set of relations $\operatorname{Inv}\operatorname{End}(f(S))$ is closed under permutations on $\operatorname{ran} f$. In particular, it contains the relation $Q \subseteq D^2$ where $Q = \{a_1, a_2, \ldots, a_k\}^2 \setminus \{(a_1,a_1);(a_2,a_2);\ldots;(a_k,a_k)\}$, such that we have $\operatorname{ran} f = \{a_1, a_2, \ldots, a_k\}$. The relation Q corresponds to valid valuations for all instances of the $|\operatorname{ran} f|$-coloring problem. This problem is known to be NP-complete since $|\operatorname{ran} f| \geq 3$. We conclude that $\textsc{mcsp}(f(S))$ is also NP-complete in this case.

We have seen in Proposition 3.1.4 that the problem $\textsc{mcsp}(f(S))$ is log-space-equivalent to $\textsc{mcsp}(S)$. Since we have one relation in $\operatorname{Inv}\operatorname{End}(f(S))$ leading to a NP-complete problem, we conclude that $\textsc{mcsp}(S)$ is NP-complete. □

From Propositions 3.1.5 and 3.1.7 we derive the main theorem of this section.

Theorem 3.1.8 *The monotone constraint satisfaction problem* MCSP(S) *is polynomial if the set* End S *contains a constant function. Otherwise, it is* NP*-complete.*

We have presented a very simple dichotomy criterion for the problem MCSP(S) on a finite domain D. To decide this condition, it is sufficient to check if the endomorphisms set End S, which must be finite, contains a constant function. The endomorphism set End S is always equal to the clone $[F]$ for some set of functions F.

3.2 Infinite case

In this section we study the computational complexity of MCSP(S) over countable infinite domains.

The class of constraint satisfaction problems over infinite domains is a rich class of problems; it can be shown that for every computational problem p there exists a set S of relations over a countable infinite domain such that CSP(S) is equivalent to the problem p under polynomial-time Turing reductions [BG08].

In this section, we show that the complexity classification for problems MCSP(S) over countable infinite domains can be reduced to the complexity classification for constraint satisfaction problems CSP(S).

As we showed in Section 3.1 for a finite domain D, the polynomial-time solvable cases of MCSP(S) are precisely those relational structures S where all relations in S contain the tuple $(a, \ldots, a)$ composed only from the element $a \in D$; in this case, MCSP(S) is called *a-valid*. Interestingly, this is no longer true for infinite domains.

Consider a countable infinite domain D and the relation $R_{\neq}$ on D such that $R_{\neq} := D^2 \setminus \{(d, d) \mid d \in D\}$. $R_{\neq}$ is clearly not a-valid. However, MCSP($\{R_{\neq}\}$) can be reduced to the Boolean formula evaluation problem (which is known to be in the complexity class L) as follows: let Φ be an instance of MCSP(S). Constraints in Φ of the form $x \neq y$ with two different variables x and y are replaced by *true*, and constraints of the form $x \neq x$ are replaced by *false*. The resulting Boolean formula is equivalent to true if and only if Φ is satisfiable.

The MCSP(S) problem over a finite or infinite domain can also be view as a model-checking problem of the existential positive first-order logic. A universal-algebraic study of the model-checking problem for finite structures S and various other syntactic restrictions of first-order logic (for instance positive first-order logic) can be found in [Mar08].

3.2.1 Result

We write $L \leq_m L'$ if there exists a deterministic polynomial-time many-one reduction from L to L'.

Definition 3.2.1 (from [LLS75]) *A problem A is* non-deterministic polynomial-time many-one reducible *to a problem B ($A \leq_{\mathrm{NP}} B$) if there is a non-deterministic polynomial-time Turing machine M such that $x \in A$ if and only if there exists $y \in B$ computed by M on input x. We denote by A_{NP} the smallest class that contains A and is closed under $\leq_{\mathrm{NP}}$.*

Observe that $\leq_{\mathrm{NP}}$ is transitive [LLS75]. To state the complexity classification for $\mathrm{MCSP}(S)$, we need the following concepts. First, we define the notion of unsatisfiable constraint.

Definition 3.2.2 *A constraint $C(x_1, \ldots, x_k)$ built upon a relation R is* unsatisfiable *if, for every interpretation I of the variables $x_1, \ldots, x_k$, we have $(I(x_1), \ldots, I(x_k)) \notin R$.*

Example 3.2.3 *Let R be the relation such that $R = D^2 \setminus \{(a, a) \mid a \in D\}$. Then the constraint $C(x, x)$ built upon R is unsatisfiable.* ■

Let Φ be an instance of $\mathrm{MCSP}(S)$. We construct a Boolean formula $F_S(\Phi)$ as follows. We first remove all existential quantifiers from Φ. Then we replace each unsatisfiable constraint in Φ of the form $C(x_1, \ldots, x_k)$ built upon the k-ary empty relation by *false*. All other constraints in Φ will be replaced by *true*. We write $F_S(\Phi)$ for the resulting Boolean formula. Note that if Φ is satisfiable then $F_S(\Phi)$ must be logically equivalent to *true*.

Definition 3.2.4 *We call a set S of relations* locally refutable *if the following holds: every instance Φ of $\mathrm{MCSP}(S)$ is satisfiable* if and only if *the Boolean formula $F_S(\Phi)$ (as described above) is logically equivalent to* true.

In Section 3.2.2, we will show the following result.

Theorem 3.2.5 *Let S be a set of relations on a countable infinite domain. If S is locally refutable then the problem $\mathrm{MCSP}(S)$ is in the complexity class* L. *Otherwise, $\mathrm{MCSP}(S)$ is complete for the class $\mathrm{CSP}(S)_{\mathrm{NP}}$ under polynomial-time many-one reductions.*

In particular, MCSP(S) is in P or is NP-hard (under deterministic polynomial-time many-one reductions). If S is a set of relations over a finite domain, then MCSP(S) is in P or NP-complete, because finite domain constraint satisfaction problems are clearly in NP. The observation that MCSP(S) is in P or NP-complete has previously been seen in Section 3.1. However, our proof remains the same for finite domains and is simpler than the proof of the previous section.

3.2.2 Proof

Before we prove Theorem 3.2.5, we start with the following simpler result.

Theorem 3.2.6 *Let S be a set of relations on a countable infinite domain. If S is locally refutable, then the problem* MCSP(S) *is in* L. *Otherwise,* MCSP(S) *is* NP*-hard (under polynomial-time many-one reductions).*

To show Theorem 3.2.6, we first prove the following lemma.

Lemma 3.2.7 *Let S be a set of relations on a countable infinite domain. If S is not locally refutable, then there are S-formulas ψ_0 and ψ_1 with the property that*

- *ψ_0 and ψ_1 are both satisfiable;*
- *$\psi_0 \wedge \psi_1$ is not satisfiable.*

Proof: Because S is not locally refutable, there is an unsatisfiable instance Φ of MCSP(S) such that the Boolean formula $F_S(\Phi)$ described above is logically equivalent to *true*. Among all formulas with this property, let Φ be the one that is of minimal length.

If Φ is of the form $\Phi_1 \vee \Phi_2$ then both Φ_1 and Φ_2 must be unsatisfiable, and one of the Boolean formulas $F_S(\Phi_1)$ or $F_S(\Phi_2)$ must be true; this contradicts the assumption that Φ is minimal.

If Φ is of the form $\Phi_1 \wedge \Phi_2$, then both $F_S(\Phi_1)$ and $F_S(\Phi_2)$ must be true. If both Φ_1 and Φ_2 are satisfiable, then we are done, because we have found two satisfiable S-formulas such that their conjunction is unsatisfiable. If Φ_1 or Φ_2 is unsatisfiable, say Φ_i is unsatisfiable for $i \in \{1, 2\}$, then this contradicts the assumption that Φ is minimal, because Φ_i is smaller than Φ, unsatisfiable and $F_S(\Phi_i)$ is true. If Φ is of the form $\exists x.\Phi'$ then this contradicts obviously the assumption that Φ is minimal. Note that Φ cannot be reduced to a single constraint, because in this case Φ is either unsatisfiable or $F_S(\Phi)$ is true (but not both). □

Proof of Theorem 3.2.6: If S is locally refutable, then MCSP(S) can be reduced to the positive Boolean formula evaluation problem, which is known to be in L. We only have to construct from an instance Φ of MCSP(S) a Boolean formula $F := F_S(\Phi)$ as described before Definition 3.2.4. Clearly, this construction can be performed with logarithmic workspace. We evaluate F, and reject if F is false, and accept otherwise.

If S is not locally refutable, we show NP-hardness of MCSP(S) by reduction from 3-SAT. Let I be a 3-SAT instance. We construct an instance Φ of MCSP(S) as follows. Let ψ_0 and ψ_1 be the S-formulas from Lemma 3.2.7 and assume they are d-ary. Let $v_1, \ldots, v_n$ be the Boolean variables in I. For each v_i we introduce d new variables $\vec{x}_i = x_i^1, \ldots, x_i^d$. Let Φ be the instance of MCSP(S) that contains the following conjuncts:

- For each $1 \leq i \leq n$, the formula $\psi_0(\vec{x}_i) \vee \psi_1(\vec{x}_i)$
- For each clause $l_1 \vee l_2 \vee l_3$ in I, the formula $\psi_{i_1}(\vec{x}_{j_1}) \vee \psi_{i_2}(\vec{x}_{j_2}) \vee \psi_{i_3}(\vec{x}_{j_3})$ where $i_p = 0$ if l_p equals $\neg x_{j_p}$ and $i_p = 1$ if l_p equals x_{j_p}, for all $p \in \{1, 2, 3\}$.

It is clear that Φ can be computed in deterministic polynomial time from I, and that Φ is satisfiable if and only if I is satisfiable. □

Note that, applied to finite domain, we obtain again the dichotomy from [HR09] and [Mar06], that is, MCSP(S) is in P if S is a-valid for an element $a \in D$ and NP-complete otherwise. We prove in the following proposition that, over a finite domain, S is locally refutable if and only if it is a-valid for an element a of the domain D.

Proposition 3.2.8 *Let D be a finite domain and S be a set of relations over D. S is locally refutable if and only if it is a-valid for some element $a \in D$.*

Proof: Let S be a-valid for an element $a \in D$. Then S is obviously locally refutable. Indeed, each relation R_i in S is a-valid, thus $F_S(C_i(\vec{x}))$ must be true, with $C_i(\vec{x})$ being the constraint built upon R_i, hence we have $F_S(\Phi)$ true for every formula Φ we can build from S.

Let S be locally refutable and assume that it is not a-valid for any $a \in D$. We split the proof into two parts.

Assume first that there exist two relations R_1 and R_2 in S, R_1 being a_1-valid and R_2 being a_2-valid, such that for all elements a in D we have $(a, a, \ldots, a) \notin R_1 \cap R_2$. Clearly, the two elements a_1 and a_2 from D must satisfy the condition $a_1 \neq a_2$. Let $C_i(x, x, \ldots, x)$ be a constraint built

upon the relation R_i, for $i = 1, 2$. Since each formula $C_i(x, x, \ldots, x)$ is a_i-valid, it is therefore locally refutable. However, the formula $C_1(x, x, \ldots, x) \wedge C_2(x, x, \ldots, x)$ cannot be locally refutable, since it is not a-valid for any $a \in D$, leading to a contradiction.

Assume now that there exists a relation R in S, which is not a-valid for any $a \in D$, i.e. such that for all elements $a \in D$ we have $(a, a, \ldots, a) \notin R$. Let $C(x_1, \ldots, x_k)$ be a constraint built upon the k-ary relation R. Consider D to be a domain of cardinality n, i.e. $|D| = n$. Let $x_1, \ldots, x_{kn}$ be $k.d$ distinct variables, and let's construct the formula φ such that φ is the conjunction of the constraint C over all combinations of k variables between $\{x_1, \ldots, x_{kd}\}$. By the pigeonhole principle, for each interpretation of φ, we must have at least k variables interpreted to the same value in D. Thus, the constraint C over these k variables must be false. This is a contradiction with the hypothesis that S is locally refutable.

We conclude that a set of relations S is locally refutable if and only if it is a-valid for some element $a \in D$.

□

Proof of Theorem 3.2.5: If S is locally refutable then the statement has been shown in Theorem 3.2.6. Suppose that S is not locally refutable. To show that MCSP(S) is contained in CSP(S)$_{\mathrm{NP}}$, we construct a non-deterministic Turing machine T which takes as input an instance Φ of MCSP(S), and which outputs an instance $T(\Phi)$ of CSP(S) as follows.

On input Φ the machine T proceeds recursively as follows:

- if Φ is of the form $\exists x.\varphi$ then return $\exists x.T(\varphi)$;
- if Φ is of the form $\varphi_1 \wedge \varphi_2$ then return $T(\varphi_1) \wedge T(\varphi_2)$;
- if Φ is of the form $\varphi_1 \vee \varphi_2$ then non-deterministically return either $T(\varphi_1)$ or $T(\varphi_2)$;
- if Φ is of the form $C(x_1, \ldots, x_k)$ then return $C(x_1, \ldots, x_k)$.

The output of T is an instance of CSP(S). It is clear that T has polynomial running time, and that Φ is satisfiable if and only if there exists a computation of T on Φ that computes a satisfiable instance of CSP(S).

We now show that MCSP(S) is hard for CSP(S)$_{\mathrm{NP}}$ under $\leq_m$-reductions. Let L be a problem with a non-deterministic polynomial-time many-one reduction to CSP(S), and let M be the non-deterministic Turing machine that computes the reduction. We have to construct a deterministic Turing machine M' that computes for any input string s in polynomial time in $|s|$

an instance Φ of MCSP(S) such that Φ is satisfiable if and only if there exists a computation of M on s that computes a satisfiable instance of CSP(S).

Say that the running time of M on s is in $O(|s|^e)$ for a constant e. Hence, there are constants s_0 and c such that for $|s| > s_0$ the running time of M and hence also the number of constraints in the input instance of CSP(S) produced by the reduction is bounded by $t := c\,|s|^e$. The non-deterministic computation of M can be viewed as a deterministic computation with access to non-deterministic advice bits as shown in [GJ79]. We also know that for $|s| > s_0$, the machine M can access at most t non-deterministic bits. If w is a sufficiently long bit-string, we write M_w for the deterministic Turing machine obtained from M by using the bits in w as the non-deterministic bits, and $M_w(s)$ for the instance of CSP(S) computed by M_w on input s.

If $|s| \leq s_0$, then M' returns $\exists x.(x = x)$ if there is an $w \in \{0,1\}^*$ such that $M_w(s)$ is a satisfiable instance of CSP(S), and M' returns $\exists \vec{x}.\psi_0(\vec{x}) \wedge \psi_1(\vec{x})$ otherwise (i.e., it returns a false instance of MCSP(S); ψ_0 and ψ_1 are defined in Lemma 3.2.7). Since s_0 is a fixed finite value, M' can perform these computations in constant time.

It is convenient to assume that S has just a single relation R. Indeed, remark we can always find a CSP which is deterministic polynomial-time equivalent and where the template is of the following form: if S corresponds to $\{R_1, \ldots, R_n\}$ where R_i has arity r_i and is not empty, then CSP(S) is equivalent to CSP($\{R_1 \times \cdots \times R_n\}$) where $R_1 \times \cdots \times R_n$ is the $\sum_{i=1}^{n} r_i$-ary relation defined as the Cartesian product of the relations $R_1, \ldots, R_n$. Similarly, MCSP(S) is equivalent to MCSP($\{R_1 \times \cdots \times R_n\}$). Let l be the arity of R. Then instances of CSP(S) with variables $x_1, \ldots, x_n$ can be encoded as sequences of numbers that are represented by binary strings of length $\lceil \log t \rceil$ as follows: The i-th number m in this sequence indicates that the $(((i-1) \bmod l)+1)$-st variable in the $(((i-1)\ div\ l)+1)$-st constraint is x_m.

For $|s| > s_0$, the sentence Φ computed by M' has the form

$$\exists x_1^1, \ldots, x_l^t.\ (\bigwedge_{i=1}^{t} C(x_1^i, \ldots, x_l^i) \wedge \Psi)\ . \qquad (3.1)$$

where $C(x_1^i, \ldots, x_l^i)$ is the constraint built upon R and Ψ is an S-formula defined below.

The **idea** is that *any* instance of CSP(S) computed by the machine M can be obtained by contracting variables in $\bigwedge_{i \leq t} C(x_1^i, \ldots, x_l^i)$. The way this is done is controlled by a Boolean formula that can be computed from the input s of M in polynomial time. The Boolean formula also contains Boolean variables for the non-deterministic advice bits of M. Each Boolean

variable v in the formula is simulated by a d-tuple $\vec{x}_v$ of variables in Ψ that is forced to satisfy $\psi_0(\vec{x}_v) \vee \psi_1(\vec{x}_v)$ (ψ_0 and ψ_1 are defined in Lemma 3.2.7), similarly as in the proof of Theorem 3.2.6, such that $v = 0$ corresponds to the unsatisfiability of $\psi_0(\vec{x}_v)$, and $v = 1$ corresponds to the satisfiability of $\psi_1(\vec{x}_v)$. The formula Φ will be such that there exists a computation of M that produces a satisfiable instance I of CSP(S) if and only if there exists an assignment to $x_1^1, \ldots, x_l^t$ that satisfies Ψ and such that $\bigwedge_{i \leq t} C(x_1^i, \ldots, x_l^i)$ is equivalent to I.

We now provide the details of the definition of the machine M' that computes Φ. We use a construction from the proof of Cook's theorem given in [GJ79]. In this proof, a computation of a non-deterministic Turing machine T accepting a language L is encoded by Boolean variables that represent the state and the position of the read-write head of T at time r, and the content of the tape at position j at time r. The tape content at time 0 consists of the input x, written at positions 1 through n, and the non-deterministic advice bit string w, written at positions -1 through $-|w|$. The proof in [GJ79] specifies a deterministic polynomial-time computable transformation f_L that computes for a given string s a SAT instance $f_L(s)$ such that there is an accepting computation of T on s if and only if there is a satisfying truth assignment for $f_L(s)$.

In our case, the machine M computes a reduction and thus computes an output string. Recall our binary representation of instances of the CSP: M writes on the output tape a sequence of numbers represented by binary strings of length $\lceil \log t \rceil$. It is straightforward to modify the transformation f_L given in the proof of Theorem 2.1 in [GJ79] to obtain for all positive integers a, a', b, b', c, c' where $a, a' \leq t$, $b, b' \leq l$, $c, c' \leq \lceil \log t \rceil$ a deterministic polynomial-time transformation $g_{a,a',b,b',c,c'}$ that computes for a given string s a SAT instance $g_{a,a',b,b',c,c'}(s)$ with distinguished variables $z_1, \ldots, z_t$ (for the non-deterministic bits in the computation of M) such that the following are equivalent:

- $g_{a,a',b,b',c,c'}(s)$ has a satisfying assignment where z_i is set to $w_i \in \{0, 1\}$ for $1 \leq i \leq t$;
- the c-th bit in the b-th variable of the a-th constraint in $M_w(s)$ equals the c'-th bit in the b'-th variable of the a'-th constraint in $M_w(s)$.

We use the transformations $g_{a,a',b,b',c,c'}$ to define M' as follows. The machine M' first computes the formulas $g_{a,a',b,b',c,c'}(s)$. For every Boolean variable v in these formulas we introduce a new conjunct $\varphi_0(\overline{x}_v) \vee \varphi_1(\overline{x}_v)$ where $\overline{x}_v$ is a d-tuple of fresh variables. Then, every positive literal $l = x_j$

in the original conjuncts of the formula is replaced by $\varphi_1(\overline{x}_j)$, and every negative literal $l = \neg x_j$ by $\varphi_0(\overline{x}_j)$. We then existentially quantify over all variables except for $x_{z_1}, \ldots, x_{z_t}$. Let $\psi_{a,a',b,b',c,c'}(s)$ denote the resulting existential positive formula. It is clear that the formula

$$\exists x_{z_1}, \ldots, x_{z_t}. \bigwedge_{a,a',b,b'} \left((\bigwedge_{c,c'} \psi_{a,a',b,b',c,c'}) \rightarrow x_b^a = x_{b'}^{a'} \right)$$

can be re-written in existential positive form Ψ without blow-up (we can replace implications $\alpha \rightarrow \beta$ by $\neg\alpha \vee \beta$, and then move the negation to the atomic level, where we can remove it by exchanging the role of φ_0 and φ_1), and hence Ψ can be computed by M' in polynomial time. The formula Ψ indeed has the properties required for the formula Ψ mentioned in Equation 3.1. □

3.3 Conclusion

In this chapter, we have present result obtained for Monotone Constraint Satisfaction Problems. The original idea was to enrich the formalism of CSP with the disjunction in order to allow a countable and even finite clone lattice. The lattice reveals itself to be in definitive not exploitable. However, we obtained a Dichotomy Theorem for the complexity of the parametric monotone CSP problem over every finite domains. We also proved a Dichotomy Theorem for Monotone CSP over countable infinite domains.

4

co-Boolean CSP

This chapters and the next one deal with the complexity of CSP where the template is a set of graphs of unary functions. Consider f to be a unary function. We call graph of f the relation defined as follows.

Definition 4.0.1 *The* graph $f^\bullet$ *of a unary function* f *is the binary relation where each tuple corresponds to the input and the output of* f*. Formally, we have*

$$f^\bullet = \{(x, f(x)) \mid x \in D\}$$

By extension, we denote by $F^\bullet$ the set of graphs of functions in F, that is $F^\bullet = \{f^\bullet \mid f \in F\}$. In this chapters and the next one, we will denote by F-formula a formula where constraints are built upon $F^\bullet$.

This chapter deals with *co-Boolean* CSP. A co-Boolean CSP is a problem CSP($F^\bullet$) where F is a set of unary functions with a co-domain of size at most two. We have made the distinction between homogeneous co-Boolean CSP (Section 4.3) where all functions in F share the same co-domain, and the general case (Section 4.4) where functions in F does not necessary share the same co-domain.

As well as Chapter 3, the starting idea here was to avoid the uncountable clone lattice problem, and as well as Chapter 3, we did not succeed to study efficiently the clone lattice of co-Boolean CSPs.

This clone lattice seemed however very convenient at the first look. Burris and Willard proved in their paper [BW87] that there is a finite number of clones of polymorphisms over graphs of unary functions. Unfortunately, even finite, the clone lattice grows exponentially regarding to the domain cardinality and is very hard to understand.

Nevertheless, it is possible to study the complexity of co-Boolean CSP problems without the help of the clone lattice. This chapter contains a Dichotomy Theorem for the homogeneous co-Boolean CSP problem, and an first approach for a Dichotomy Theorem for non-homogeneous co-Boolean CSP

verifying a property of non-duplication of the right hand side of relations. A dichotomy Theorem for non-homogeneous co-Boolean CSP in general remains an open question, but we have the intuition that the dichotomy criterion is the same as the homogeneous case.

It is important to stress that the study of co-Boolean CSP is a first step for the study of CSP over unary functions, that is, CSPs where each constraint is on the form $f(x) = y$ with f an unary function. A fundamental result from Feder, Madelaine and Stewart [FMS04] shows that for every template S, there exists a set F of two unary functions such that problems CSP(S) and CSP($F^\bullet$) are polynomial-time equivalent, that is, CSP over unary functions are as powerful as general CSP parametric problems. Graphs of unary functions give us a very structural template which is really convenient to work with.

4.1 Definitions

When studying the complexity of CSP($F^\bullet$) for a set of unary functions F it is convenient to represent $F^\bullet$ in a normal form. We define below the H-normal form.

Definition 4.1.1 *The* H-normal *form of the graphs of a set of unary functions $F = \{f_1, \ldots, f_k\}$ is the $(k+1)$-ary relation F^H which is the solution of formula $\varphi(x, y_1, \ldots, y_k)$ expressed as follow:*

$$\varphi(x, y_1, \ldots, y_k) := \bigwedge_{i \in \{1,\ldots,k\}} f_i(x) = y_i$$

It is very convenient to present the relation F^H as a matrix, where rows are tuples in F^H. Under this form, we intuitively see that the first column describes every element of the domain D and each other column corresponds to the description of a function in F. Speaking about F^H we call the first column the *left hand side* and the group of all other columns the *right hand side*.

Example 4.1.2 *Let $F = \{f_1, f_2\}$, with f_1 and f_2 two unary functions on D defined as follow:*

x	$f_1(x)$	$f_2(x)$
0	0	0
1	1	0
2	0	1
3	1	1

Thus, we have

$$F^H = \{(0,0,0);(1,1,0);(2,0,1);(3,1,1)\} = \begin{pmatrix} 0 & 0 & 0 \\ 1 & 1 & 0 \\ 2 & 0 & 1 \\ 3 & 1 & 1 \end{pmatrix}$$

and the right hand side of F^H, denoted F^H_{rhs}

$$F^H_{rhs} = \begin{pmatrix} 0 & 0 \\ 1 & 0 \\ 0 & 1 \\ 1 & 1 \end{pmatrix}$$

■

Let F be a set of unary functions on a domain D, and p be a polymorphism of F^H_{rhs}. Let's denote by $D_{rhs} \subseteq D$ the domain of F^H_{rhs}, that is, the set of elements present in F^H_{rhs}. We say that p' is an *extension* of p to F^H if:

- p' is a polymorphism of F^H,
- p and p' have the same arity k, and for every k-vector $\vec{x} \in D^k_{rhs}$, the equation $p'(\vec{x}) = p(\vec{x})$ holds,
- p' has the same properties as p. By property we mean the associativity, the commutativity, the conservativity and the idempotency, or the fact to be a constant operation, a majority operation, a minority operation, a semilattice operation, a 2-semilattice operation, a semi-projection or an essentially unary operation. For instance, if p is associative then p' is also associative; if p is a semi-projection then p' is also a semi-projection.

If such a function p' exists, we say that p is extended to F^H.

Another notion we will use all along this chapters and the next one is the notion of core of a set of relations.

Definition 4.1.3 *The* core *of a set S of relations, denoted S_c, is the subset of S such that every endomorphism on S_c is an automorphism.*

Notice that if a set S is a core, every unary polymorphism f of S is bijective, that is, f is a permutation on the domain D. Observe that to compute a core of a set S, you must take $f \in \operatorname{Pol} S$ to be one of the unary

polymorphisms of S with the smallest range and apply it on each relation in S. Thus, the set $f(S)$ is a core of S.

Now, we define some operations we will use very often in this chapter and in Chapter 5.

Definition 4.1.4 *We say that a n-ary operation f is* idempotent *if the following holds:*

$$\forall d \in D,\ f(d, d, \ldots, d) = d$$

Definition 4.1.5 *A binary operation f is* commutative *if we have:*

$$\forall a, b \in D,\ f(a, b) = f(b, a)$$

Definition 4.1.6 *A n-ary operation f is said to be* conservative *if we have the following equation:*

$$\forall a_1, a_2, \ldots, a_n \in D,\ f(a_1, a_2, \ldots, a_n) \in \{a_1, a_2, \ldots, a_n\}$$

From now on, we call *bcc* a binary commutative conservative operation.

Definition 4.1.7 *A binary operation f is* associative *if the following holds:*

$$\forall a, b, c \in D,\ f(a, f(b, c)) = f(f(a, b), c)$$

A binary associative commutative idempotent operation is called a *semi-lattice operation*. Sometimes, this operation is also called an ACI operation (ACI for Associative Commutative Idempotent).

Definition 4.1.8 *A* 2-semilattice operation *is a binary commutative idempotent operation s on a domain D such that we have $s(a, s(a, b)) = s(a, b)$ for all $a, b \in D$.*

Definition 4.1.9 *A n-ary operation p on a domain D is a* projection *if there exists an index $i \in [1, n]$ such that $p(a_1, \ldots, a_i, \ldots, a_n) = a_i$, for all $a_1, ldots, a_n$ in D.*

Definition 4.1.10 *A* semi-projection *s is a n-ary operation, with $n \geq 3$, such that s is not a projection and verifies the following property:*

$$\exists i \in \{1, \ldots, n\},\ \forall a_1, \ldots, a_n \in D,\ s(a_1, \ldots, a_n) = a_i \text{ if } |\{a_1, \ldots, a_n\}| < n$$

Definition 4.1.11 *An* essentially unary operation *e is a n-ary operation such that there exists an unary operation f and an index $i \in \{1, \ldots, n\}$ verifying*

$$\forall a_1, \ldots, a_n \in D,\ e(a_1, \ldots, a_n) = f(a_i)$$

Definition 4.1.12 *A* majority operation *majo and a* minority operation *mino are two ternary operations such that we have*

$$\forall a, b \in D,\ majo(a, a, b) = majo(a, b, a) = majo(b, a, a) = a$$

and

$$\forall a, b \in D,\ mino(a, a, b) = mino(a, b, a) = mino(b, a, a) = b$$

On three different values, a majority operation and a minority operation can output any value in D.

Definition 4.1.13 *Consider a linear order, that is a total order, on the domain D. A* maximum operation *max and a* minimum operation *min are two binary operations such that we have*

$$\forall a, b \in D,\ if\ a < b\ then\ max(a, b) = b.\ Otherwise\ max(a, b) = a$$

and

$$\forall a, b \in D,\ if\ a < b\ then\ min(a, b) = a.\ Otherwise\ min(a, b) = b$$

In order to have shorter notations, we will call *majority* a majority operation, *minority* a minority operation, *max* a maximum operation and *min* a minimum operation.

4.2 Generalities about sets of unary functions

In this section, we introduce some results which hold for every set F of unary functions. First, let's introduce the following well-known and fundamental result.

Theorem 4.2.1 *Let S be a set of relations. If S is closed under a majority, or a minority, or a max, or a min, or a semilattice operation then* CSP(S) *is in* P. *If S is closed under essentially unary operations only, then* CSP(S) *is* NP*-complete.*

Proof: See for instance [JCG97], [Jea98] and [BCRV04]. □

4.2.1 Polynomial-time reductions and equivalences

This section presents very useful polynomial-time reductions and equivalences between our studied problems.

Let F be a set of unary functions and let $\operatorname{Rel} F$ denote the set of solutions of every F-formula, that is, the set of solutions of every formula which can be built upon $F^\bullet$.

Lemma 4.2.2 *Let F and G be two sets of unary functions such that the inclusion $[F] \subseteq [G]$ holds. Then, $\textsc{csp}(F^\bullet) \leq_p \textsc{csp}(G^\bullet)$.*

Proof: Let F and G be two sets of unary functions such that $[F] \subseteq [G]$ holds. Hence, every functions that can be built by compositions with functions from F can also be built with functions from G. It means that every F-formula is also a G-formula, so we have the inclusion $\operatorname{Rel} F \subseteq \operatorname{Rel} G$.

By [Her08], we know that $\operatorname{Rel} F = \operatorname{Inv}\operatorname{Pol} F^\bullet$, for every set of functions F. Then, we have the inclusion $\operatorname{Inv}\operatorname{Pol} F^\bullet \subseteq \operatorname{Inv}\operatorname{Pol} G^\bullet$. Since the application of Pol reverses the inclusion, we have $\operatorname{Pol}\operatorname{Inv}\operatorname{Pol} F^\bullet \supseteq \operatorname{Pol}\operatorname{Inv}\operatorname{Pol} G^\bullet$. However we know that for every set of relations S, the equation $\operatorname{Inv}\operatorname{Pol} S = \langle S \rangle$ holds. Then, we have the equation $\operatorname{Pol}\operatorname{Inv}\operatorname{Pol} S = \operatorname{Pol}\langle S \rangle$, that is, $\operatorname{Pol}\operatorname{Inv}\operatorname{Pol} S = \operatorname{Pol} S$. Hence, we have the inclusion $\operatorname{Pol} F^\bullet \supseteq \operatorname{Pol} G^\bullet$.

By [Jea98] (Cor. 4.11), we immediately conclude that $\textsc{csp}(F^\bullet) \leq_p \textsc{csp}(G^\bullet)$ □

Proposition 4.2.3 *Let F be a set of unary functions. Problems $\textsc{csp}(F^\bullet)$ and $\textsc{csp}(F^H)$ are polynomial-time equivalent.*

Proof: By definition $F^H \in \langle F^\bullet \rangle$. To recover $f_i^\bullet$ from F^H, existentially quantify all columns of F^H except for the first and the $i+1$'th columns. Hence, it is clear that $\langle F^H \rangle = \langle F^\bullet \rangle$ and the result follows. □

Lemma 4.2.4 *Let F be a set of unary functions. If $\textsc{csp}(F^H_{rhs})$ is* NP*-complete then $\textsc{csp}(F^H)$ is* NP*-complete.*

Proof: This is obvious: by definition, we have $F^H_{rhs} \in \langle F^H \rangle$, thus the inclusion $\operatorname{Pol} F^H \subseteq \operatorname{Pol} F^H_{rhs}$ holds. Then, following [Jea98], we know that $\textsc{csp}(F^H_{rhs}) \leq_p \textsc{csp}(F^H)$ holds. We conclude immediately. □

4.2.2 Cores

The result obtained in this section is the following: to classify the complexity of $\textsc{csp}(F^\bullet)$ for any finite set F of unary functions, it is enough to classify the

complexity of $\text{CSP}(F^\bullet)$ for finite sets of unary functions containing all unary constant functions.

Given a graph of a unary function $f^\bullet$ on a domain D and another unary function $\pi\colon D \to D$, we denote by $\pi(f^\bullet)$ the relation $\{(\pi(d_i), \pi(d_j)) \mid (d_i, d_j) \in f^\bullet\}$. Similarly, given the graphs of a set F of unary functions we denote by $\pi(F^\bullet)$ the set of relations $\{\pi(f^\bullet) \mid f^\bullet \in F^\bullet\}$. Let $F_c^\bullet$ denote the core of $F^\bullet$. We begin by proving that, given a set F of unary functions, the core of the graphs of the functions in F is the graphs of a set G of unary functions, i.e., we have $G^\bullet = F_c^\bullet$. In other words, the core of the graphs of unary functions remains a set of graphs of unary functions.

Lemma 4.2.5 *Given a set F of unary functions, then there exists a set G of unary functions such that we have $F_c^\bullet = G^\bullet$. In other words, the core of a set of graphs of unary functions remains a set of graphs of unary functions.*

Proof: The proof is quite direct: assume $F^\bullet$ is not a core (otherwise we are done). Taking $p \in \operatorname{Pol} F^\bullet$ such that p has one of the smallest range among polymorphisms of $\operatorname{Pol} F^\bullet$, we obtain $p(F^\bullet)$ a core of $F^\bullet$, that is, we have $F_c^\bullet = p(F^\bullet)$. Thus, we must have $F_c^\bullet \subseteq F^\bullet$, which allows us to conclude.

□

Let Con_D denote the set of all unary constant functions over a domain D.

Lemma 4.2.6 *For every set F of unary functions on a domain D there is a set G of unary functions on domain $D_c \subseteq D$ such that $\text{CSP}(F^\bullet)$ is polynomial-time equivalent to $\text{CSP}(G^\bullet \cup Con_{D_c}^\bullet)$. More specifically, G can be chosen such that $G^\bullet$ is the core of $F^\bullet$ and D_c the domain of elements in the core of $F^\bullet$.*

Proof: The proof is direct thanks to Theorems 4.4 and 4.7 from [BJK05]. Theorem 4.4 shows that $\text{CSP}(F^\bullet)$ and $\text{CSP}(F_c^\bullet)$ are polynomial-time equivalent, and Theorem 4.7 shows that, in particular, problems $\text{CSP}(F_c^\bullet)$ and $\text{CSP}(F_c^\bullet \cup Con_{D_c}^\bullet)$ are also polynomial-time equivalent.

□

Notice this important fact: if F is a set of co-Boolean functions and there exists $d \in D$ such that for every function $f \in F$ we have $f(d) = d$, then the domain of the core of F collapse to become the singleton $\{d\}$. Thus, it is clear that this case is not interesting to study since it leads immediately to the tractability of $\text{CSP}(F^\bullet)$. This case is in fact verified if $F^\bullet$ is closed under a constant operation, a well-known case implying obviously the tractability.

Thus, in the Sections 4.3 and 4.4, we will consider that $F^\bullet$ is not closed under a constant operation, and then the core of $F^\bullet$ will not be simply reduced to a d-vector on a unary domain. This is also considered is the following remark.

Remark 4.2.7 *Let F be a set of unary functions on a domain D such that F contains at least two different constant functions f_1 and f_2 on D, such that f_1 always outputs a and f_2 always outputs b, with $a, b \in D$, $a \neq b$. Then F^H_{rhs} cannot be closed under any unary constant functions on D. Indeed, F^H_{rhs} contains two columns c_1 and c_2 corresponding to f_1 and f_2 which are only composed of a and b, respectively. However, to be closed under a constant function, F^H_{rhs} needs to contain a d-tuple, that is a tuple $(d, d, \ldots, d)$. This is impossible since each tuple in F^H_{rhs} contains a and b respectively at the c_1-th and c_2-th positions.*

4.3 Homogeneous case

In this section we assume F is a set of homogeneous co-Boolean functions, *i.e.*, all functions in F are unary functions and share the same co-domain of size two, say $\{0, 1\}$, if they are not constant functions. We recall that we consider $F^\bullet$ not to be closed under a constant operation since it is a trivial case of tractability. Thus, if we consider $F^\bullet$ to be a core on a domain D, this domain cannot be the trivial unary domain. Thanks to Lemma 4.2.6, we can assume that $F^\bullet$ is a core containing the graph of every constant function on D.

Lemma 4.3.1 *Let F be a set of homogeneous co-Boolean functions such that $F^\bullet$ is a core containing the graphs of every unary constant function on the domain D. If the right hand side F^H_{rhs} of F^H, viewed as a relation, is neither closed under majority, nor minority, nor max, nor min, then* CSP$(F^\bullet)$ *is* NP*-complete.*

Proof: Let $\varphi(x, \vec{y})$ be the formula for which F^H described the set of solutions. Recall that F^H_{rhs} is just the solutions of $\exists x \varphi(x, \vec{y})$. Since $F^\bullet$ is a core containing the graphs of every unary constant function on the domain D, F^H_{rhs} cannot be closed under any unary constant functions (see Remark 4.2.7). Now, if F^H_{rhs} is neither closed under majority, minority, max, nor min, then by Schaefer's result in [Sch78], CSP(F^H_{rhs}) is NP-complete. We conclude thanks to Lemma 4.2.4 and Proposition 4.2.3. □

So the cases which have to be analyzed are when F^H_{rhs} is closed under majority, minority, max, or min.

Lemma 4.3.2 *Let F be a set of homogeneous co-Boolean functions. If F^H_{rhs} is closed under a majority or a minority, then* $\text{CSP}(F^\bullet)$ *is in* P.

Proof: We only give the proof for majority since the minority case is completely analogous. Let F^H_{rhs} be of arity k, and be closed under a majority maj, that is, for every tuples $t_1, t_2, t_3 \in F^H_{rhs}$, $maj(t_1, t_2, t_3)$ outputs a tuple $t_4 \in F^H_{rhs}$. Recall that maj applied on $a, b, c \in D$ pairwise different can output everything, even an element d again different from a, b and c.

Let's t'_1, t'_2 and t'_3 corresponds to elements in the first column in F^H respectively at the same rows as t_1, t_2 and t_3, as shown in the following scheme where a bar separates the first column to F^H_{rhs}.

$$F^H = \left(\begin{array}{c|c} \vdots & \vdots \\ t'_1 & \textit{tuple } t_1 \\ \vdots & \vdots \\ t'_2 & \textit{tuple } t_2 \\ \vdots & \vdots \\ t'_3 & \textit{tuple } t_3 \\ \vdots & \vdots \end{array}\right)$$

Thus, there exists a majority maj' such that for all elements a, b, c in a same column of F^H_{rhs}, we have $maj'(a, b, c) = maj(a, b, c)$, and for t'_1, t'_2, t'_3 we have $maj'(t'_1, t'_2, t'_3) = t'_4$ with t'_4 the element in the first column of F^H at the same row as t_4. So, F^H is closed under maj'.

If F^H is closed under a majority or a minority we know by Theorem 4.2.1 that $\text{CSP}(F^H)$ is in P. By Lemma 4.2.4 and Proposition 4.2.3, we conclude that $\text{CSP}(F^\bullet)$ is in P. □

The remaining cases are when F^H_{rhs} is closed under max or min. First, we would like to define to what corresponds the point-wise order on two tuples. Let a, b two k-tuples on a binary domain. We have $a \leq_l b$ with $\leq_l$ the point-wise order if for all $i \in [0, (k-1)]$ we have $a[i] \leq b[i]$. In other words, $a \leq_l b$ holds if there is no $i \in [0, (k-1)]$ such that $a[i] = 1$ and $b[i] = 0$.

Lemma 4.3.3 *Let F be a set of homogeneous co-Boolean functions. If F^H_{rhs} is closed under max or min and the first two tuples $(0, a)$ and $(1, b)$ of F^H satisfy $a \leq_l b$ where $\leq_l$ is the point-wise order, then* $\text{CSP}(F^\bullet)$ *is in* P.

Proof: The proof is similar to the proof of Lemma 4.3.2, i.e., we extend the operation on F^H_{rhs} toward F^H, outputting the convenient element in the

first column. It gives us a max or min on a certain linear order on D (not necessary the canonical order).

By Theorem 4.2.1, we also known that F^H closed under max or min implies that $\text{CSP}(F^H)$ is in P. By Lemma 4.2.4 and Proposition 4.2.3, we have $\text{CSP}(F^\bullet)$ is in P. □

Finally we prove that other cases are NP-complete.

Lemma 4.3.4 *Let F be a set of homogeneous co-Boolean functions. If F^H_{rhs} is closed under max or min but neither closed under majority nor minority, and there is no point-wise order $\leq_l$ satisfying $a \leq_l b$ for the first two tuples $(0,a)$ and $(1,b)$, then $\text{CSP}(F^\bullet)$ is* NP*-complete.*

Proof: Note that F^H_{rhs} is not closed under both min and max since this would imply the closure of F^H_{rhs} under a majority. Note also that if we do not have $a \leq_l b$ then there must exist a column in F^H_{rhs} representing a function f_i such that it contains 1 and 0 respectively at the first and second row. Thus, if we project F^H on the first and the $(i+1)$-th column ($(i+1)$-th because the first column is column zero), we obtain a binary relation containing the two tuples $(0,1)$ and $(1,0)$. This relation is actually the graph of f_i, so it cannot contain neither the tuple $(0,0)$ nor the tuple $(1,1)$. Then, this relation is neither closed under max nor min. F^H_{rhs} is by assumption neither closed under majority nor minority. Hence, let F' be the set $F' = \{F^H_{rhs}, f_i^\bullet\}$. It is clear that we have $F' \subseteq \langle F^H \rangle$. Moreover, by Schaefer's theorem we know that $\text{CSP}(F')$ is NP-complete, then we deduce by Theorem 4.2.1, Lemma 4.2.4 and Proposition 4.2.3 that $\text{CSP}(F^H)$, and so $\text{CSP}(F^\bullet)$, are also NP-complete. □

Theorem 4.3.5 *Let F be a set of homogeneous co-Boolean functions. If $F^\bullet$ is closed under a constant operation then $\text{CSP}(F^\bullet)$ is obviously in* P*. If not, the problem $\text{CSP}(F^\bullet)$ is in* P *if F^H_{rhs} is closed under a majority operation, or a minority operation, or either a max or min operation such that the first two tuples $(0,a)$ and $(1,b)$ of F^H satisfy $a \leq_l b$ where $\leq_l$ is the point-wise order. Otherwise, $\text{CSP}(F^\bullet)$ is* NP*-complete.*

Proof: The proof is direct by Lemmata 4.3.1, 4.3.2, 4.3.3 and 4.3.4. □

4.4 General case

We consider in this part the problem $\text{CSP}(F^\bullet)$ where F is a set of non-homogeneous co-Boolean functions, that is, where functions in F do not

necessary share the same co-domain. Remember that we consider $F^\bullet$ not to be closed under a constant operation since it is a trivial case of tractability. Thus, if we consider $F^\bullet$ to be a core on a domain D, this domain cannot be the trivial unary domain. Before calling these function "co-Boolean functions", we denoted them "partition functions" since each function acts like a partitioning of the domain D. However, the denomination "partition functions" is already used by Bulatov and Grohe for a completely different notion (see [BG05]).

Nevertheless, a set of co-Boolean functions over D can be seen as a way to consider several partitions of D, like it is shown in Example 4.4.1. Thus, when studying different partitions of D at the same time, one can consider useless to have two elements a and b in D such that for every partitioning induced by a co-Boolean function in F, a and b are in the same partition. If we have this case, one could consider that a and b are similar elements in D. This case appears if, there are two elements a and b in D such that for every function f in F, the equation $f(a) = f(b)$ holds. To have a better representation, this is verified when two rows are identical in F^H_{rhs}; in the sequel, we will say that these rows are duplicated.

Example 4.4.1 *Let the domain $D = \{0, 1, 2, 3\}$, and f and g two co-Boolean functions defined as follow:*

x	$f(x)$	$g(x)$
0	0	3
1	1	3
2	0	1
3	1	3

Then one can considered that f and g make a partition of D: function f splits D into a 0- and a 1-part, respectively $\{x \mid f(x) = 0\} = \{0, 2\}$ and $\{x \mid f(x) = 1\} = \{1, 3\}$, and g into a 1- and a 3-part, respectively $\{x \mid g(x) = 1\} = \{2\}$ and $\{x \mid g(x) = 3\} = \{0, 1, 3\}$. ■

The study of such structure with no duplicated rows is not without interest since it represents the set of different partitions of D such that there are no similar elements each time in the same partition. Thus, we focus in this part of the study of the complexity of $\text{CSP}(F^\bullet)$ such that there is no duplicated rows in F^H_{rhs}.

So far however, we did not achieve to prove the following conjecture, whatever the problem $\text{CSP}(F^\bullet)$ is such that there is some duplicated rows in F^H_{rhs} or not.

Conjecture 4.4.2 *Let F be a set of co-Boolean functions. If $F^\bullet$ is closed under a constant operation then this obviously leads to the tractability of* $\mathrm{CSP}(F^\bullet)$. *If not, the problem* $\mathrm{CSP}(F^\bullet)$ *is in* P *if F^H_{rhs} is closed under a binary idempotent operation which is not a projection or under a majority or minority operation. Otherwise it is* NP*-complete.*

We can also present this conjecture in the same way as the "tractable algebras conjecture" (Conjecture 7.5 in [BJK05]), that is, our problem is NP-complete if $\mathrm{Pol}\, F^\bullet$ contains essentially unary operations only, otherwise it is polynomial.

To give the beginning of a proof of this conjecture, we will use the Rosenberg's list (Rosenberg's paper [Ros86] is difficult to find, but this list is presented in [Jea98] for instance).

Theorem 4.4.3 (Rosenberg's list, [Ros86]) *Let S be an arbitrary set of relations on a finite domain D. At least one of these conditions must hold:*

- $\mathrm{Pol}\, S$ *contains a constant operation,*
- $\mathrm{Pol}\, S$ *contains a binary idempotent operation which is not a projection,*
- $\mathrm{Pol}\, S$ *contains a majority,*
- $\mathrm{Pol}\, S$ *contains a minority,*
- $\mathrm{Pol}\, S$ *contains a semi-projection,*
- *or else* $\mathrm{Pol}\, S$ *is composed of essentially unary operations only.*

Thus, we will follow step by step this list and start to show what are the possible polymorphisms of F^H_{rhs}.

First, observe that we can easily lighten this list by getting rid of constant operations. Consider a set F of co-Boolean functions. We know by Lemma 4.2.6 that adding every constant function over D in F does not change the complexity of our problem $\mathrm{CSP}(F^\bullet)$. However, for a set F containing at least two constant operations, the relation F^H_{rhs} cannot be closed under any constant operation (see Remark 4.2.7).

From now on, we consider that a set F of functions will always contain every possible constant function on the considered domain D. However in order to keep a light notation, we will omit to explicitly write columns of F^H which represent constant functions in F.

Proposition 4.4.4 *Let F be a set of co-Boolean functions such that F^H_{rhs} is closed under a majority (respectively a minority). Then F^H is also closed under a majority (respectively a minority).*

Proof: Let F^H_{rhs} be closed under a majority m. It is easy to see that F^H is also closed under m since the first column in F^H is composed of every element of the domain D and are present in this column exactly once. Take any three rows r_1, r_2 and r_3 in F^H_{rhs}, and assume that m applied component-wise on these three rows outputs the row r_4. Now, extend m on rows (a, r_1), (b, r_2) and (c, r_3) in F^H corresponding to our three previous rows, and assume that m applied on it outputs (d, r_4). Since a, b, c are pairwise different, $m(a, b, c)$ can output any elements in D. Then, we choose $m(a, b, c) = d$.

The proof remains the same for minority. □

Proposition 4.4.5 *Let F be a set of co-Boolean functions. If there exists a binary idempotent commutative polymorphism of F^H_{rhs}, then there exists a binary commutative conservative (bcc) polymorphism of F^H_{rhs}.*

Proof: Let F be a set of co-Boolean functions and q be a binary idempotent commutative operation such that $q \in \operatorname{Pol} F^H_{rhs}$. Observe that for every function f in F, with $\operatorname{ran} f = \{a, b\}$, we must have $q(a, a) = a$ and $q(b, b) = b$ since q is idempotent and $q(a, b) = q(b, a) \in \{a, b\}$ since q is a polymorphism of F^H_{rhs}. Then, let q' be the bcc operation defined as follow: for all $f \in F$, we have $q'(x_f, y_f) = q(x_f, y_f)$ with $x_f, y_f \in \operatorname{ran} f$, and $q'(x, y) \in \{x, y\}$ for all sets $\{x, y\}$ which are not the range of a function f in F. We can see that q' is a polymorphism of F^H_{rhs}. □

The previous proposition shows we have a bcc in $\operatorname{Pol} F^H_{rhs}$ when we know that there is a binary idempotent commutative operation in $\operatorname{Pol} F^H_{rhs}$. This allow us to focus on bcc operation from now. The next proposition proves that a bcc polymorphism of F^H_{rhs} is actually a 2-semilattice operation.

Proposition 4.4.6 *Let F be a set of co-Boolean functions. Every bcc which is a polymorphism of F^H_{rhs} must be a 2-semilattice operation.*

Proof: Remember that a 2-semilattice operation is a binary commutative idempotent operation s on the domain D such that we have $s(a, s(a, b)) = s(a, b)$ for all $a, b \in D$.

Let q be a bcc. Assume we have $q(a, b) = a$ for some $a, b \in D$. Thus, we have $q(a, q(a, b)) = q(a, a) = a = q(a, b)$. Assume now that $q(a, b) = b$ holds for some $a, b \in D$. We have $q(a, q(a, b)) = q(a, b)$. Since q is commutative, we conclude that q is a 2-semilattice operation. □

In the sequel, we will often use the notion of a domain ordered regarding to a 2-semilattice operation.

Definition 4.4.7 *Let a, b be two elements of a domain D and s be a 2-semilattice operation on D. We write $a \leq_s b$ if we have $s(a, b) = b$. Notice that, since s is commutative, we have also $s(b, a) = b$. We say that the domain D is ordered* regarding to *a 2-semilattice operation s by considering the order $\leq_s$, linear or partial, over elements of D. We also say that $\leq_s$ is the order on D* induced by *s.*

Example 4.4.8 *Let D be the domain $D = \{0, 1, 2, 3\}$ and s_1, s_2 be two 2-semilattice operations on D defined as follow:*

s_1	0	1	2	3
0	0	1	0	1
1	1	1	1	1
2	0	1	2	1
3	1	1	1	3

s_2	0	1	2	3
0	0	0	0	3
1	0	1	1	3
2	0	1	2	3
3	3	3	3	3

Thus D is ordered regarding to s_1 by the partial order $(2 \leq_{s_1} 0 \leq_{s_1} 1; 3 \leq_{s_1} 1)$ and ordered regarding to s_2 by the linear order $2 \leq_{s_2} 1 \leq_{s_2} 0 \leq_{s_2} 3$. ■

Assume now that F^H_{rhs} is not closed under a majority or a minority, but is closed under a 2-semilattice operation which induces a linear or partial order on domain D.

We make a difference between 2-semilattice operations inducing a linear or partial order on domain D which verify the property of 2-semilattice isomorphism (introduced in Definition 4.4.11) and those which do not verify it.

Definition 4.4.9 *We call* functional semilattice *of a 2-semilattice operation s the semilattice of elements of the domain D regarding to the order induced by s.*

Definition 4.4.10 *Let F^H be the H-normal form of a set F of co-Boolean functions. We call* relational semilattice *of a 2-semilattice operation s on F^H_{rhs} the semilattice for which elements are tuples in F^H_{rhs} ordered regarding to a 2-semilattice operation s.*

Definition 4.4.11 *Let F^H be the H-normal form of a set F of co-Boolean functions and s be a 2-semilattice operation. Let also h be an isomorphism*

between the functional semilattice FS of s and the relational semilattice RS of s on F^H_{rhs}. We say that h is a connecting isomorphism *between FS and RS if h is such that, for every $d \in D$, we have $(d, h(d)) \in F^H$.*

Then, we can study these functional and relational semilattices and determine if there exist such a connecting isomorphism between them.

Example 4.4.12 *Let F be the following set of co-Boolean functions on $D = \{0, 1, 2, 3\}$:*

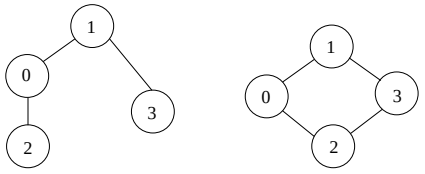
x	f	g	h
0	0	1	0
1	1	1	0
2	0	2	2
3	1	1	2

Let s_1, s_2 be two 2-semilattice operations on D defined as follow:

s_1	0	1	2	3
0	0	1	0	1
1	1	1	1	1
2	0	1	2	1
3	1	1	1	3

s_2	0	1	2	3
0	0	1	0	1
1	1	1	1	1
2	0	1	2	3
3	1	1	3	3

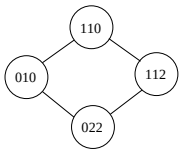
Thus, partial orders on D induce respectively by s_1 and s_2 are $(2 < 0 < 1; 3 < 1)$ and $(2 < 0, 3 < 1)$. Then, we have the following functional semilattices of s_1 and s_2, respectively:



Now, if we apply component-wise s_1 on each couple of tuples in F^H_{rhs}, except trivial couples (t, t) for $t \in F^H_{rhs}$, we obtain

$$\begin{aligned} s_1(010, 110) &= 110 \\ s_1(010, 022) &= 010 \\ s_1(010, 112) &= 110 \\ s_1(110, 022) &= 110 \\ s_1(110, 112) &= 110 \\ s_1(022, 112) &= 112 \end{aligned}$$

Observe that we obtain the same results by applying s_2 instead of s_1. We have the following relational semilattice s_1 (or s_2) on F^H_{rhs}:



It is easy to see that there exists a connecting isomorphism between this relational semilattice and the functional semilattice of s_2, but not with the functional semilattice of s_1. Actually, one can see there is even no isomorphisms between the relational semilattice above and the functional semilattice of s_1. ■

Proposition 4.4.13 *Let F be a set of co-Boolean functions. Let F^H_{rhs} be closed under a 2-semilattice operation s which induces a linear of partial order on domain D. The operation s is also a polymorphism of F^H if and only if there exists a connecting isomorphism h between the functional semilattice FS of s and the relational semilattice RS of s on F^H_{rhs}.*

Proof:
$\boxed{\Leftarrow}$: RS describes which row c in F^H_{rhs} is output when we apply s on rows a and b. FS describes the same structure but with elements of the first column in F^H. Let's a', b', c' be elements in D and a, b, c be tuples in F^H_{rhs}. Assume we have $s(a, b) = c$ and $s(a', b') = c'$ with $(a', a), (b', b) \in F^H$. If there exists a connecting isomorphism h between FS and RS, then we must have $(c', c) \in F^H$. Thus, s is a polymorphism of F^H.

$\boxed{\Rightarrow}$: Let s be a polymorphism of F^H. First, assume there exist only isomorphisms between FS and RS which are not connecting. Let h be such an isomorphism. Thus there must exists $a', b', c', d' \in D$ and $a, b, c, d \in F^H_{rhs}$, all pairwise different, with $(a', a), (b', b), (c', c), (d', d) \in F^H$ such that we have $s(a', b') = c'$ and $s(a, b) = d$. So it is clear that we have the equation $s((a', a), (b', b)) = (c', d)$. However, this is a contradiction with the fact that F^H is an H-normal form of a set F of co-Boolean functions since a different tuple starting by c' already exists in F^H.

Remark that the proof remains exactly the same if we consider that there exist no isomorphisms between FS and RS.

Hence, we can extend s to F^H, that is, if F^H_{rhs} is closed under s and there exists a connecting isomorphism between FS and RS, then F^H is also closed under s. □

Proposition 4.4.14 *Let F be a set of co-Boolean functions. A semi-projection which is a polymorphism of F^H_{rhs} cannot be extended to F^H.*

Proof: We recall the assumption that F^H_{rhs} does not contain any duplicated rows. Let F^H_{rhs} be closed under a semi-projection s. By definition, s should have an arity greater than or equals to 3, and $s(i,j,k)$ acts like a projection if and only if $|\{i,j,k\}| < 3$, with $i,j,k \in D$. However, since F^H_{rhs} is composed by co-Boolean functions only, for each triple (i,j,k) in a column we must verify the inequality $|\{i,j,k\}| < 3$. Then, s is always acting like a projection on F^H_{rhs}. Without loss of generality, assume we have $s(i,j,k) = i$ when $|\{i,j,k\}| < 3$ holds.

Try to extend s to F^H. Take three tuples a, b and c in F^H_{rhs}, and elements a', b' and c' pairwise different in the first column of F^H such that $(a',a), (b',b)$ and (c',c) are tuples in F^H. Since $s(a,b,c)$ must output the first tuple a, to extend s to F^H we must have $s(a',b',c') = a'$. However, remark that s is no longer a semi-projection since it is now always acting like a projection. □

It is now time to expose how results above help us to start a characterization of the complexity of co-Boolean CSP. As always in this chapter, this beginning of a dichotomy criteria does not consider the case where $F^\bullet$ is closed under a constant operation since it leads obviously to the tractability of $\text{CSP}(F^\bullet)$.

Theorem 4.4.15 *Let F be a set of co-Boolean functions. If the fact that $\operatorname{Pol} F^H_{rhs}$ containing a binary idempotent non-commutative operation or a binary idempotent commutative operation which is not a 2-semilattice operation implies the tractability of $\text{CSP}(F^H)$, then Conjecture 4.4.2 holds.*

Proof: We try to characterize the complexity of $\text{CSP}(F^\bullet)$ via some criteria around F^H_{rhs} only. Remember the Rosenberg's list (Theorem 4.4.3). Let's analyze point by point this list.

Let F be a set of co-Boolean functions on a domain D such that the matrix F^H_{rhs} does not contain any duplicated rows. Following Rosenberg's list, we know that $\operatorname{Pol} F^H_{rhs}$ must contain at least a constant operation, or a majority, or a minority, or a semi-projection, or a binary idempotent operation which is not a projection, or else $\operatorname{Pol} F^H_{rhs}$ is composed of essentially unary operations only.

However, since we consider that F contains each constant functions on D (Lemma 4.2.6), we cannot have a constant operation in $\operatorname{Pol} F^H_{rhs}$ (remember Remark 4.2.7).

If $\operatorname{Pol} F^H_{rhs}$ contains a majority or a minority, then by Proposition 4.4.4 we know that $\operatorname{Pol} F^H$ contains also the extension of this majority or minority. We can conclude by Lemma 4.2.4 and Proposition 4.2.3 that $\text{CSP}(F^\bullet)$ is in P.

If $\operatorname{Pol} F^H_{rhs}$ contains a semi-projection, we know thanks to Proposition 4.4.14 that such a semi-projection cannot be extended to F^H, that is, $\operatorname{Pol} F^H$ cannot contain a semi-projection. Thus, we are allowed to not consider this case since by the Rosenberg's list, $\operatorname{Pol} F^H_{rhs}$ must contain another operation which is not a semi-projection.

If $\operatorname{Pol} F^H_{rhs}$ contains essentially unary operations only, then it is well-known (see [JCG97] for instance) that $\textsc{csp}(F^H_{rhs})$ is NP-complete. We deduce then by Lemma 4.2.4 and Proposition 4.2.3 that $\textsc{csp}(F^\bullet)$ is NP-complete.

Hence, the last case to consider is when $\operatorname{Pol} F^H_{rhs}$ contains a binary idempotent operation which is not a projection. We consider first that this operation is commutative. If it is, then by Proposition 4.4.5 there exists a binary commutative conservative operation (bcc for short) which is a polymorphism of F^H_{rhs}. Moreover by Proposition 4.4.6, a bcc which is a polymorphism of F^H_{rhs} must be a 2-semilattice operation. Let's name this operation s. If s verifies the isomorphism property, that is, if there exists a connecting isomorphism between the functional semilattice of s and the relational semilattice of s on F^H_{rhs}, then by Proposition 4.4.13, there exists a way to extend s to F^H, that is, $\operatorname{Pol} F^H$ also contains a 2-semilattice operation. This allows us to conclude that $\textsc{csp}(F^\bullet)$ is in P, thanks to Lemma 4.2.4 and Proposition 4.2.3, and to [Bul06b] showing that a problem $\textsc{csp}(S)$ is in P if the set S of relations is closed under a 2-semilattice operation. If s does not verify the isomorphism property, then Proposition 4.4.13 shows that there is no way to extend s to F^H. Thus, as well as the case for semi-projection, $\operatorname{Pol} F^H_{rhs}$ must contains another kind of operation. Remark that this operation can be for instance a binary idempotent commutative operation f which is not a 2-semilattice operation.

Finally, if the fact that $\operatorname{Pol} F^H_{rhs}$ containing a binary idempotent non-commutative operation or a binary idempotent commutative operation which is not a 2-semilattice operation implies the tractability of $\textsc{csp}(F^H)$, then the problem $\textsc{csp}(F^\bullet)$ is in P if F^H_{rhs} is closed under a binary idempotent operation which is not a projection, or a majority operation, or a minority operation. Otherwise, F^H_{rhs} is closed under essentially unary operations only, and then $\textsc{csp}(F^\bullet)$ is NP-complete. □

We may have found a polynomial-time algorithm solving $\textsc{csp}(F^\bullet)$ if F^H_{rhs} is closed under a binary idempotent non-commutative operation, but unfortunately this result is too recent to be written in this thesis since we are not completely sure of it. Thus, notice that the closure under a binary idempotent non-commutative operation only is the last unknown case forbidding us to conclude this chapter by a Dichotomy Theorem.

4.5 Conclusion

Results presented in this chapter are first steps toward the study of the complexity of CSP over unary functions. We have proved a Dichotomy Theorem for the complexity of homogeneous co-Boolean CSP for every finite domain. As announced in the introduction of the chapter, the co-Boolean CSP problem is not definitely closed since we do not have a Dichotomy Theorem for non-homogeneous co-Boolean CSP with or without duplicated rows in F^H_{rhs}. Indeed, the case where the template is closed under a binary idempotent non-commutative operation or a binary idempotent commutative operation which is not a 2-semilattice operation only is not clear.

When you don't know where a road leads, it sure as hell will take you there.

Leo Rosten (1908–1997)

5

CSP($F^\bullet$) and kernel width

This chapter contains the very last results of this thesis. Unfortunately, time was running out to deepen the kernel width method presented here, however a first step has been made toward the comprehension of mechanisms of kernel width, and this tool could be greatly useful to characterize the complexity of some sub-problems of the CSP problem over unary functions, and by extension thanks to Feder, Madelaine and Stewart's result in [FMS04], to sub-problems of the CSP(S) problems.

Universal algebra often uses the concept of *kernel*. We define this notion with the help of equivalence classes. Let $f\colon D \to D$ be a unary function with range $\operatorname{ran} f$. A *d-equivalence class*, for $d \in \operatorname{ran} f$ and denoted by $[d]_f$, is the set of elements $[d]_f = \{x \in D \mid f(x) = d\}$. The kernel $\ker f$ of a function f is the set of all equivalence classes with cardinality strictly greater than 1 for all $d \in \operatorname{ran} f$, *i.e.* $\ker f = \{[d]_f \mid d \in \operatorname{ran} f \text{ and } |[d]_f| \geq 2\}$.

Example 5.0.1 *Let f, g and h be unary functions on the domain $D = \{0,1,2,3\}$ such that*

x	$f(x)$	$g(x)$	$h(x)$
0	0	0	1
1	1	1	1
2	2	0	1
3	3	1	1

Then $\ker f = \emptyset$, $\ker g = \{\{0,2\};\{1,3\}\}$ *and* $\ker h = \{\{0,1,2,3\}\}$. ■

From now on, in order to lighten notations we will write

$$\ker f = \{(a,b);(c,d,e)\}$$

instead of $\ker f = \{\{a,b\};\{c,d,e\}\}$.

Notice we have only defined the notion of kernel for unary functions. Indeed, this notion is quite natural for unary functions, but becomes immediately complicated to extend this notion to k-ary functions. We have not

developed so far a convenient way to exploit kernels of k-ary functions because of this difficulty, but also because we are not convinced by the benefit that such an extension could give us.

Let us introduce the notion of kernel width. We say that a kernel $\ker f$ is a subset of a kernel $\ker g$, denoted by $\ker f \subseteq \ker g$, if for all $a \in \operatorname{ran} f$ there exists $b \in \operatorname{ran} g$ such that we have $[a]_f \subseteq [b]_g$, that is, every equivalence class in $\ker f$ is a subset of an equivalence class of $\ker g$. For instance, if we take Example 5.0.1, we have the inclusions $\ker f \subseteq \ker g$ and $\ker g \subseteq \ker h$ (and then $\ker f \subseteq \ker h$ since the inclusion of kernels is transitive). Thus, two kernels $\ker f$ and $\ker g$ are *independent* if $\ker f \nsubseteq \ker g$ and $\ker g \nsubseteq \ker f$ hold.

From this notion of inclusion, we derive the notion of *kernel width*, or *width* for short. Let F be a set of unary functions. We denote by $\operatorname{width} F$ the width of this set, such that $\operatorname{width} F$ is the maximal anti-chain in the set of non-empty kernels in $[F]$ ordered by the inclusion. In other words, $\operatorname{width} F$ is the maximal number of non-empty pairwise independent kernels of functions in $[F]$. We think that this notion of kernel width may have a great importance for the characterization of the complexity of CSP, in general or at least for some sub-problems.

In this chapter, we first prove that a linear kernel width, that is a kernel width equals to 1, implies the tractability of CSP($F^\bullet$), with F a set of arbitrary unary functions over a fixed domain D. This result holds for every cardinality of the domain D. Then, the next section contains the full characterization of the complexity of CSP($F^\bullet$), where F is a set of arbitrary unary functions over a ternary domain, with a Dichotomy Theorem using the kernel width method. Finally, the last main result of this chapter deals with the NP-completeness of the problem CSP($F^\bullet$) with F a set of homogeneous co-Boolean functions over a fixed domain D if the inequality $\operatorname{width} F \geq |D|$ holds. Notice that the first and the last results presented above were already known since they are covered respectively by the Bulatov's theorem in [Bul06a] and Theorem 4.3.5 in Chapter 4. Actually, this first result of this chapter (that is, the Dichotomy Theorem) has been already found by Broniek and published [Bro05], but our proof is simpler than Broniek's one, using known results in universal algebra, and leads us to a better comprehension of interactions between kernel width and complexity. Moreover, we would like to stress that those results are respectively the first dichotomy theorem and the first proof of NP-completeness over every cardinality of domain via the kernel width method. Besides, the result of tractability in the second section was not known so far.

5.1 width $F = 1$ implies tractability

5.1.1 Intermediary results

This section presents the following results. Let F be a set of unary functions on a fixed size domain D such that width $F = 1$ holds. We show that for all $f, g \in F$, we have the implication $\ker f \circ g \subseteq \ker f \Rightarrow \ker f \circ g = \ker f$. First, we show the following lemma.

Lemma 5.1.1 *Let F be a set of unary functions over a finite domain D, and let $f, g \in F$. Then the following hold:* $\operatorname{ran}(f \circ g) \subseteq \operatorname{ran} f$ *and* $\ker g \subseteq \ker(f \circ g)$.

Proof: Every unary function $f \in F$ is monotone, *i.e.*, if $A \subseteq B$ then $f(A) \subseteq f(B)$ holds for all subsets A, B of D. Since $\operatorname{ran} g \subseteq D$ and f is monotone, we have $f(\operatorname{ran} g) \subseteq f(D)$. Moreover, $f(\operatorname{ran} g)$ is $\operatorname{ran}(f \circ g)$ and $f(D)$ is $\operatorname{ran} f$. Therefore the inclusion $\operatorname{ran}(f \circ g) \subseteq \operatorname{ran} f$ holds. Let $x, y \in \ker g$. We have $g(x) = g(y)$, therefore $(f \circ g)(x) = (f \circ g)(y)$, *i.e.* $x, y \in \ker(f \circ g)$. □

Proposition 5.1.2 *Let F be a set of unary functions on a domain D. For all $f, g \in F$, the implication* $\ker f \circ g \subseteq \ker f \Rightarrow \ker f \circ g = \ker f$ *holds.*

Proof: Let f and g be two functions in F such that $\ker f \circ g \subseteq \ker f$. Then, for all $x, y \in D$ such that $f \circ g(x) = f \circ g(y)$, we have the equation $f(x) = f(y)$.

First, notice that $f(x) \neq f(y) \Rightarrow f \circ g(x) \neq f \circ g(y) \Rightarrow g(x) \neq g(y)$ hold for all x, y since we know that we have $\ker g \subseteq \ker f \circ g$ by Lemma 5.1.1.

Suppose that we have $\ker f \circ g \subsetneq \ker f$. Thus, let $x_1, y_1 \in D, x_1 \neq y_1$, such that the inequality $(f \circ g)(x_1) \neq (f \circ g)(y_1)$ and the equation $f(x_1) = f(y_1)$ hold.

Let $x_2, y_2 \in D$ be such that we have $g(x_1) = x_2$ and $g(y_1) = y_2$. Necessarily, we have $x_2 \neq y_2$ because of the implication from Lemma 5.1.1 $(f \circ g)(x) \neq (f \circ g)(y) \Rightarrow g(x) \neq g(y)$. Then we have $f(x_2) \neq f(y_2)$, and we deduce from $\ker f \circ g \subseteq \ker f$ that the inequality $f \circ g(x_2) \neq f \circ g(y_2)$ holds. We can then defined x_3 and y_3 such that $g(x_2) = x_3$ and $g(y_2) = y_3$, and so on. *(i)*

Notice that every x_i, y_i are obtained from the function g.

Necessarily, there exist $n_0 \in \mathbb{N}$ and $k \in \mathbb{N}$ such that, for every $n \geq n_0$ we have $g^k(x_n) = x_n$ and $g^k(y_n) = y_n$, with x_n and y_n obtained as previously. We have then $g^{k-1}(x_n) = x_{n+k-1}$ and $g^{k-1}(y_n) = y_{n+k-1}$.

Let j be the smallest number of iteration of g such that $g^j(x_1) = x_{n_0}$ and $g^j(y_1) = y_{n_0}$. Then the equations $g^j(x_1) = g^{k'}(x_{n_0})$ and $g^j(y_1) = g^{k'}(y_{n_0})$

hold, with $k' > j$ a multiple of k. Thus, we have obviously $f \circ g^j(x_1) = f \circ g^{k'}(x_{n_0})$ and $f \circ g^j(y_1) = f \circ g^{k'}(y_{n_0})$. Since we have the inclusion $\ker f \circ g \subseteq \ker f$, we deduce the equations $f \circ g^{j-1}(x_1) = f \circ g^{k'-1}(x_{n_0})$ and $f \circ g^{j-1}(y_1) = f \circ g^{k'-1}(y_{n_0})$. By definition, we have $k' > j$. Then, by iteration the equations $f(x_1) = f \circ g^{k'-j}(x_{n_0})$ and $f(y_1) = f \circ g^{k'-j}(y_{n_0})$ hold. Let $x', y' \in D$ be two values such that $x' = g^{k'-j}(x_{n_0})$ and $y' = g^{k'-j}(y_{n_0})$. Notice that x' and y' are obtained through some iteration of g, and by our notation we have in fact $x' = x_{n_0+((k'-j)[k])}$ and $y' = y_{n_0+((k'-j)[k])}$. By hypothesis, we know that $f(x_1) = f(y_1)$, then we can deduce the equation $f(x') = f(y')$ which is a contradiction with (i), since x' and y' are built through the function g starting from x_1 and y_1. Thus we cannot have the strict inclusion $\ker(f \circ g) \subsetneq \ker f$, so we conclude that $\ker(f \circ g) = \ker f$ holds if $\ker(f \circ g)$ is included in $\ker f$. □

We now introduce a notation specifically used in this chapter.

Notation 5.1.3 *We write $x_1, \ldots, x_k \rhd \ker f$ if $x_1, \ldots, x_k$ are elements contained in the same equivalence class of $\ker f$. On the contrary, we denote by $x_1, \ldots, x_k \not\rhd \ker f$ if there is at least two elements x_i and x_j among $x_1, \ldots, x_k$ which do not belong to the same equivalence class of $\ker f$.*

We recall the Example 5.0.1 given in preliminaries.

Example 5.1.4 *Let f, g and h be functions on the domain $D = \{0, 1, 2, 3\}$ such that*

x	$f(x)$	$g(x)$	$h(x)$
0	0	0	1
1	1	1	1
2	2	0	1
3	3	1	1

Then $\ker f = \emptyset$, $\ker g = \{(0, 2); (1, 3)\}$ and $\ker h = \{(0, 1, 2, 3)\}$. ■

Thus, if we consider this example, we have $0, 2 \rhd \ker g$, $1, 3 \rhd \ker g$ but also $0, 1 \not\rhd \ker g$ for instance.

Lemma 5.1.5 *Let D be a domain of size n and F be a set of unary functions on D such that $\operatorname{width} F = 1$. Let h be a function in $[F]$, then for every $x_1, x_2 \in D$ the following holds: we have $h(x_1) = y_1$ and $h(x_2) = y_2$, and there exists a function $f \in [F]$ such that $x_1, x_2 \rhd \ker f$ if and only if there exists a function $g \in [F]$ such that $y_1, y_2 \rhd \ker g$.*

Proof:
$\boxed{\Leftarrow}$: Let y_1, y_2 be such that $y_1, y_2 \rhd \ker g$, for a function $g \in [F]$. Then $x_1, x_2 \rhd \ker g \circ h$ holds.

$\boxed{\Rightarrow}$: Let x_1, x_2 be such that $x_1, x_2 \rhd \ker f$, for a function $f \in [F]$. Suppose that $y_1, y_2 \not\rhd \ker g$ for every $g \in [F]$. Then, we have the inequality $g(y_1) \neq g(y_2)$, that is equivalent to say $g \circ h(x_1) \neq g \circ h(x_2)$. In particular, we have $f \circ h(x_1) \neq f \circ h(x_2)$. Since width $F = 1$, we have either $\ker f \subseteq \ker f \circ h$ or $\ker f \circ h \subseteq \ker f$. In the first case we are done, because this implies the contradiction $f \circ h(x_1) = f \circ h(x_2)$. In the second case, we deduce from Proposition 5.1.2 that $\ker(f \circ h) = \ker f$ and derive the same contradiction. Then we conclude immediately. □

5.1.2 Main result

Let D be a domain of size n and F be a set of unary functions on D such that width $F = 1$, that is, for every functions f and g in $[F]$, we have either $\ker f \subseteq \ker g$ or $\ker g \subseteq \ker f$. Thus, when we speak about smallest or largest kernels, this is regarding to the linear order by inclusion. Moreover, we call a non-trivial kernel a kernel which is neither the empty kernel nor the kernel containing every elements in D in one equivalence class. More formally, a kernel $\ker f$ is non-trivial if there exists $d_1, \ldots, d_k \in D$ such that $d_1, \ldots, d_k \rhd \ker f$ and $|d_1, \ldots, d_k| < |D|$ hold. In this section, we will see that width $F = 1$ implies the fact that $\text{CSP}(F^\bullet)$ is in P.

Let M be the majority operation defined as follow: consider $M(x, y, z)$, with x, y and z pairwise different. If there exists a function $f \in [F]$ such that $\ker f$ is the smallest non-trivial kernel containing at least two elements from $\{x, y, z\}$ in the same equivalence class (for example, $x, z \rhd \ker f$), then $M(x, y, z)$ returns the first element in the input that belongs to $\ker f$. If this last property does not hold, we are in the situation where there is no non-trivial kernels containing two elements from $\{x, y, z\}$ in the same equivalence class (that is, even the largest non-trivial kernel does not contain two elements belonging to the same equivalence class). Then $M(x, y, z)$ returns the first element in the input; in this case M acts like a projection on the first element of the input.

Example 5.1.6 *Consider the following matrix composed of functions in a*

set $F = \{f, g, h, i\}$*:*

x	$f(x)$	$g(x)$	$h(x)$	$i(x)$
0	0	0	1	3
1	1	0	1	3
2	2	2	1	3
3	3	3	2	3
4	4	4	3	3

First, observe that, in this example, the closure $[F]$ *does not create other kernels than* $\ker f$*,* $\ker g$*,* $\ker h$ *and* $\ker i$*.*

The majority $M(2,1,0)$ *must output* 1 *since* $\ker g$ *is the smallest kernel containing two elements among* 0*,* 1 *and* 2 *in the same equivalence class, i.e.* 0 *and* 1*, and* 1 *is the first element in the input* $(2,1,0)$ *among these two elements.*

Similarly, $M(2,3,0)$ *must output* 2*, with this time* $\ker h$ *the smallest kernel verifying our property.*

Last example, $M(2,3,4)$ *must output* 2*, since there is no non-trivial kernels containing at least two elements among* 2*,* 3 *and* 4 *in the same equivalence class.* ■

Theorem 5.1.7 *Let* F *be a set of unary functions on a domain* D *such that* $\text{width}\, F = 1$*. Then every relation in* $F^\bullet$ *is invariant by the previous majority operation* M*.*

Proof: For each relation R in $F^\bullet$, we apply the majority operation M on three tuples (x_1, y_1), (x_2, y_2) and $(x_3, y_3) \in R$, and we will proof that the tuple $(M(x_1, x_2, x_3), M(y_1, y_2, y_3))$ belongs to R.

We can divide the proof into three cases: First, assume that there exist i and j, $i \neq j$ such that $x_i = x_j$. We know that the relation R is the graph of a function, then for each $x \in D$, there exists only one $y \in D$ such that $(x, y) \in R$. Then, the equation $x_i = x_j$ implies the fact that we must have $y_i = y_j$. Hence we deduce that $(M(x_i, x_i, x_k), M(y_i, y_i, y_k)) = (x_i, y_i)$ since M is a majority operation (note that this is true for every permutations of these tuples).

Assume now that x_1, x_2 and x_3 are pairwise different. We have then two sub-cases: y_1, y_2 and y_3 are pairwise different or not.

Assume that there exist i and j, $i \neq j$ such that $y_i = y_j$. Then, we have $x_i, x_j \rhd \ker h$, with h the function such that $R = h^\bullet$. Since M is a majority operation, we have $M(y_i, y_i, y_k) = y_i$. However, as we have defined M, we must have $M(x_i, x_j, x_k) = x_i$ since M outputs the first element belonging

to a kernel. Note that again, this fact is true for every permutations of (x_i, x_j, x_k) and their corresponding tuple (y_i, y_j, y_k).

Assume finally that y_1, y_2 and y_3 are pairwise different. By Lemma 5.1.5, we know that there exists $f \in [F]$ such that $x_i, x_j \rhd \ker f$ if and only if there exists $g \in [F]$ such that $y_i, y_j \rhd \ker g$, for every $i, j \in \{1, 2, 3\}$.

Hence, if at least two elements between x_i, x_j and x_k belong to an equivalence class of a minimal kernel $\ker f$, say x_j and x_k, we output the first one in the input, *i.e.* x_j. We have then $x_j, x_k \rhd \ker f$. By Lemma 5.1.5, there exists a function $g \in [F]$ such that $y_j, y_k \rhd \ker g$ holds, and we know that x_i cannot belongs to the same equivalence class than x_j and x_k for any non-trivial kernel. Then by the same lemma, y_j must be the first element among y_i, y_j and y_k which belongs to an equivalence class of a kernel, and M must output it. We have then $(M(x_i, x_j, x_k), M(y_i, y_j, y_k)) = (x_j, y_j)$. Now, suppose that there are no two elements among x_i, x_j and x_k which belong to an equivalence class of a kernel, that is, we have $x_i, x_j \not\rhd \ker h$, $x_i, x_k \not\rhd \ker h$ and $x_j, x_k \not\rhd \ker h$, for $\ker h$ a maximal non-trivial kernel. We know that it is also the case for y_i, y_j and y_k by Lemma 5.1.5. Thus, following our definition, M must acts like a projection on the first element. We have then $(M(x_i, x_j, x_k), M(y_i, y_j, y_k)) = (x_i, y_i)$, so the application of M to $F^\bullet$ leads again to a tuple in $F^\bullet$: M is then a polymorphism of $F^\bullet$. □

It is a well-know result (Theorem 4.2.1) that $\text{CSP}(S)$ is in P if the set S of relations is closed under a majority. We conclude that, for a set F of unary relations over a finite domain D, the linear width $\text{width}\, F = 1$ implies the tractability of $\text{CSP}(F^\bullet)$.

5.2 Ternary domains: Dichotomy criterion based on the kernel width

In this section, F will be a set of arbitrary unary functions over a ternary domain. Before introducing our Dichotomy Theorem of the problem $\text{CSP}(F^\bullet)$ through the study of the kernel width, we need to show some intermediary results. In the sequel we will deal with tractable and intractable cases, and the last part of this section presents the Dichotomy Theorem and the meta-problem study. Notice that this Dichotomy Theorem is already cover by Bulatov's result in [Bul06a]. However this is the first Dichotomy Theorem proved via the kernel width method.

We consider in the sequel that $F^\bullet$ is a core, otherwise the problem becomes a problem over a domain of size less than or equals to 2, which is

a well-know problem covered by Schaefer's results in [Sch78]. Thus remark that $F^\bullet$ which is a core over a ternary domain cannot be closed under any constant operations.

5.2.1 Intermediary results

First, we need the notions of *circular* and *swap permutation* on a ternary domain.

Definition 5.2.1 *A* circular permutation c *on a ternary domain* $D = \{0, 1, 2\}$ *is a permutation satisfying the condition* $c(x) = (x + k) \bmod |D|$ *with* $k \in D$, *for all* $x \in D$. *A* swap permutation s *on* $D = \{0, 1, 2\}$ *is a permutation satisfying the conditions* $s(x) = y$, $s(y) = x$ *and* $s(z) = z$, *for distinct* $x, y, z \in D$. *The set* $\{x, y\}$ *is called* swap s.

Remark that since we work on a ternary domain, a non-empty kernel $\ker f$ must have exactly one equivalence class. Thus, to lighten the notation in this section, we will consider kernels as sets of two or three elements. For instance, $\ker f = \{0, 2\}$ and $\ker g = \{0, 1, 2\} = D$. Notice also that kernels of unary functions on a ternary domain are limited to only one equivalence class. This can be easily verified by the pigeonhole principle. Moreover, the size of these unique equivalence classes can only be equal to 0, 2 or 3. Therefore we identify in the sequel the kernel of a unary function over a ternary domain with its singleton equivalence class.

We need in this section the notion of circular permutation, swap permutation and swap set. The reader can find definitions of all other operations used in this chapter, like a majority operation or a 2-semilattice operation for instance, in the Definition section of Chapter 4.

Before introducing our results, we first need the following lemma.

Lemma 5.2.2 *Let* F *be a set of unary functions over a ternary domain* D, *and let* $f, g \in F$. *The following conditions hold:*
(i) if $\operatorname{ran} g \not\subseteq \ker f$ *and* $|\ker f| = 2$ *then* $\operatorname{ran}(f \circ g) = \operatorname{ran} f$;
(ii) if $\operatorname{ran} g \not\subseteq \ker f$ *and* $|\ker g| = 2$ *then* $\ker g = \ker(f \circ g)$.

Proof: Let $\operatorname{ran} g \not\subseteq \ker f$ and $|\ker f| = 2$. First, remark that $|\ker f| = 2$ implies $|\operatorname{ran} f| = 2$. We have to show that $\operatorname{ran} f \subseteq \operatorname{ran}(f \circ g)$. Let $y \in \operatorname{ran} f$. Suppose that for all $x \in \operatorname{ran} g$, the inequality $f(x) \neq y$ holds. Let $z \in \operatorname{ran} f$, with $z \neq y$. So for all $x \in \operatorname{ran} g$, we have $f(x) = z$ since $|\operatorname{ran} f| = 2$, which implies $\operatorname{ran} g \subseteq \ker f$: contradiction with the assumption.

Let $\operatorname{ran} g \not\subseteq \ker f$. We have to show that $\ker(f \circ g) \subseteq \ker g$. Suppose that there exist $x, y \in \ker(f \circ g)$ such that $g(x) \neq g(y)$. Since $|\operatorname{ran} g| = 2$, we have

$\{g(x), g(y)\} = \operatorname{ran} g$. However the equality $(f \circ g)(x) = (f \circ g)(y)$ holds, so we have the inclusion $\operatorname{ran} g \subseteq \ker f$, which is a contradiction. Thus, for all $x, y \in \ker(f \circ g)$, we have $x, y \in \ker g$. □

Proposition 5.2.3 *Let D be a ternary domain and F be a set of unary functions on D. If F does not contain any permutations, then the number of independent kernels in F corresponds to the width of F.*

Proof: Let F be a set of unary functions on a ternary domain D such that F does not contain any permutations. Hence, each function $f \in F$ has a kernel such that $|\ker f| = 2$, or $|\ker f| = 3$ if f is a constant function. Then we show that it is sufficient to compare independent kernels in F to compute the kernel width $\operatorname{width} F$, that is, we do not need to compute $[F]$ to know the width of F.

We know by Lemma 5.1.1 that for all $f, g \in F$, $\ker g \subseteq \ker f \circ g$ holds. Moreover, if $|\ker g| = 3$ then $|\ker f \circ g| = 3$, so $f \circ g$ does not change the width of F. If $|\ker g| = 2$, so $|\ker f \circ g|$ can only be greater or equal than $|\ker g|$. If $|\ker g| = |\ker f \circ g|$, then we have necessary the equality $\ker g = \ker f \circ g$, so $f \circ g$ does not change the width of F. If $|\ker f \circ g|$ is strictly greater than $|\ker g|$, the only solution is $|\ker f \circ g| = 3$ and we can deduce that $f \circ g$ is a constant function, so it can not change the width of F.

We can conclude there is no functional compositions which implies a longer anti-chain of kernels of functions in $[F]$ if F does not contain any permutations. □

Proposition 5.2.4 *Let f be a unary function on a ternary domain D such that $|\ker f| = 2$ and s be a swap permutation on D. Combinations of f and s can only produce functions with the same kernel as f if and only if* $\operatorname{swap} s = \ker f$.

Proof: Let f be a function on D such that $|\ker f| = 2$ and s be a swap permutation on D such that $\operatorname{swap} s = \ker f$. Let $a, b \in D$ such that $\ker f = \{a, b\} = \operatorname{swap} s$. Then we have $f \circ s = f$ since $(f \circ s)(a) = f(b)$ and $(f \circ s)(b) = f(a)$, and for $c \in D$ such that $a \neq c \neq b$, we have $s(c) = c$.

By Lemma 5.1.1, we know that the inclusion $\ker f \subseteq \ker(s \circ f)$ holds. Now aiming for a contradiction, suppose that $\ker(s \circ f) \nsubseteq \ker f$. So there exists $c \in D$ such that $c \in \ker(s \circ f)$ and $c \notin \ker f$. Thus, we have necessary $|\ker(s \circ f)| = 3$ since $|\ker(s \circ f)| > |\ker f|$. Then, $s \circ f$ is a constant function. Since s is a swap permutation, so a permutation, f must be a constant function. Contradiction with $|\ker f| = 2$. Therefore we must have $\ker f = \ker(s \circ f)$.

Now, let f and s be such that $\operatorname{swap} s \neq \ker f$. Without loss of generality, let $a, b \in D$ such that $\operatorname{swap} s = \{a, b\}$ and $b, c \in D$ such that $\ker f = \{b, c\}$, with $a \neq c$. Assume that $f(a) = x$ and $f(b) = f(c) = y$, with $x, y \in D$. So $f(s(a)) = f(b) = x$, $f(s(b)) = f(a) = y$ and $f(s(c)) = f(c) = x$. Then, we have $\ker(f \circ s) = \{a, c\}$.

We conclude that we can only produce by combinations of f and s functions with the same kernel as f if and only if $\operatorname{swap} s = \ker f$. □

Proposition 5.2.5 *Let f and g be two unary functions on a ternary domain D with incomparable kernels and s a swap permutation such that* $\ker f \cap \ker g \notin \operatorname{swap} s$. *Then* $\ker(f \circ s) = \ker g$ *and* $\ker(g \circ s) = \ker f$.

Proof: Let f, g and s be such functions. Assume that $\ker f \cap \ker g = a$, for $a \in D$. Then, we have $\operatorname{swap} s = \{b, c\}$, with $b, c \in D$ and a, b, c pairwise different. Moreover, we must have $\ker f = \{a, b\}$ and $\ker g = \{a, c\}$ since we have $\ker f \cap \ker g = a$ and kernels $\ker f$ and $\ker g$ are incomparable.

Hence, the equations $f(a) = f(b)$ and $g(a) = g(c)$ hold. Then we can deduce the equations $(f \circ s)(a) = (f \circ s)(c)$ and $(g \circ s)(a) = (g \circ s)(b)$ since $s(a) = a$, $s(b) = c$ and $s(c) = b$. We conclude that $\ker f \circ s = \ker g$ and $\ker g \circ s = \ker f$. □

Corollary 5.2.6 *Let f and g be two unary functions on ternary domain D with incomparable kernels and s a swap permutation such that* $\ker f \cap \ker g \notin \operatorname{swap} s$. *Let F be the set of functions containing f, g and s only. Then* $\operatorname{width} F = 2$.

Proof: Immediate by Proposition 5.2.5 and Lemmata 5.1.1 and 5.2.2. □

Corollary 5.2.7 *Let f be a unary function on a ternary domain D such that* $|\ker f| = 2$ *and s be a swap permutation on D such that* $\ker f \neq \operatorname{swap} s$. *Let F be the set of functions containing f and s only. Then* $\operatorname{width} F = 2$.

Proof: Immediate by Proposition 5.2.4 and by Corollary 5.2.6. □

Proposition 5.2.8 *Let f and g be two unary functions on a ternary domain D with incomparable kernels and s a swap permutation such that* $\ker f \cap \ker g \in \operatorname{swap} s$. *Then we can produce a function h such that f, g and h have pairwise incomparable kernels.*

Proof: Let f, g and s be such functions. Assume that $\ker f \cap \ker g = a$, for $a \in D$. By the pigeonhole principle, we have necessarily $\ker f = \operatorname{swap} s$

or $\ker g = \operatorname{swap} s$. Assume without loss of generality that we have $\ker f = \operatorname{swap} s = \{a, b\}$, with $b \in D$ such that $a \neq b$. Hence, we must have $\ker g = \{a, c\}$, with a, b, c pairwise different.

The equation $g(a) = g(c)$ holds. Hence, we have $(g \circ s)(b) = (g \circ s)(c)$ since $s(a) = b$, $s(b) = a$ and $s(c) = c$. Then we have a new function $h = g \circ s$ with an incomparable kernel compared to $\ker f$ and $\ker g$. □

Proposition 5.2.9 *Let f be a unary function on a ternary domain D such that $|\ker f| = 2$ and p a permutation on D that is neither a swap permutation nor the identity. Then we can produce by composition two functions g and h such that kernel of f, g and h are pairwise incomparable.*

Proof: Let f be a function on D such that $|\ker f| = 2$ and p a permutation on D that is not a swap permutation nor the identity, that is, p is a circular permutation. Let $a, b \in D$ be such that $\ker f = \{a, b\}$. Assume, without loss of generality, that p is the permutation (abc). Then $\ker(f \circ p) = \{a, c\}$ and $\ker(f \circ p \circ p) = \{b, c\}$. □

The following lemma is true for a set F of co-Boolean functions. However, observe that a set of unary functions over a ternary domain which does not contain any permutation is also a set of co-Boolean functions. Thus this lemma is available in this chapter if there is no permutations in F.

Lemma 5.2.10 *Let F be a set of co-Boolean functions. The relation F^H_{rhs} is **not** closed under a majority operation (respectively a minority operation) if and only if we have a sub-matrix of the matrix representing F^H_{rhs} presenting the shape*

f_1	f_2	f_3
b_1	a_2	a_3
a_1	b_2	a_3
a_1	a_2	b_3

and if there is no row r such that $r[f_1]r[f_2]r[f_3] = a_1a_2a_3$ (respectively $r[f_1]r[f_2]r[f_3] = b_1b_2b_3$) in the matrix representing F^H_{rhs}.

Proof: The "if" direction is obvious; consider the other one. Take:

f_1	f_2	f_3
a_1	a_2	a_3
b_1	b_2	b_3
c_1	c_2	c_3

with $|\{a_i, b_i, c_i\}| = 2$ for $1 \leq i \leq 3$. Assume that this sub-matrix is not closed under a majority operation. Without loss of generality we can assume $a_1 = b_1$, then we can rewrite our sub-matrix like this:

f_1	f_2	f_3
a_1	a_2	a_3
a_1	b_2	b_3
c_1	c_2	c_3

Remark that we must not have $a_2 = b_2$ or $a_3 = b_3$ otherwise it is clear that the sub-matrix is closed under a majority operation. Then, we must have ($a_2 = c_2$ and $b_3 = c_3$) or ($b_2 = c_2$ and $a_3 = c_3$). Thus, we obtain:

f_1	f_2	f_3
a_1	a_2	a_3
a_1	b_2	b_3
c_1	a_2	b_3

or

f_1	f_2	f_3
a_1	a_2	a_3
a_1	b_2	b_3
c_1	b_2	a_3

that is, the expected shape. Moreover, since F^H_{rhs} is not closed under a majority, we cannot have a row r in F^H such that we have $r[f_1]r[f_2]r[f_3] = a_1a_2b_3$ or $r[f_1]r[f_2]r[f_3] = a_1b_2a_3$ (depending of the sub-matrix we have). Proof for the minority operation follows exactly the same pattern. □

Let's call the shape of the matrix in Lemma 5.2.10 an A-shape, that is, we will say that F^H contains an A-shape if we have a sub-matrix of the matrix representing F^H_{rhs} of the form

f_1	f_2	f_3
b_1	a_2	a_3
a_1	b_2	a_3
a_1	a_2	b_3

and if there is no row r such that $r[f_1]r[f_2]r[f_3] = a_1a_2a_3$ (respectively $r[f_1]r[f_2]r[f_3] = b_1b_2b_3$) in the matrix representing F^H_{rhs}.

5.2.2 Tractable case

Let us recall some known facts. Let F be a set of unary functions and $f, g \in F$ be two unary functions. We say that $\ker f$ and $\ker g$ are incomparable kernels if $\ker f \not\subseteq \ker g$ and $\ker g \not\subseteq \ker f$ hold. Recall also that $\operatorname{width} F$ is the maximal number of incomparable non-empty kernels in $[F]$.

Proposition 5.2.11 *Given a set F of unary functions over a ternary domain D with $\operatorname{width} F \leq 2$, we must have $F^\bullet$ closed under a majority operation.*

Proof: First, observe that if we have $\operatorname{width} F = 1$, we can conclude by Theorem 5.1.7.

Thus, assume we have $\operatorname{width} F = 2$. We prove that $F^\bullet$ is closed under the following conservative majority operation: $M(x, y, z) = t$, with x, y and z pairwise different and t such that $t = \ker f_1 \cap \ker f_2$ holds with $f_1, f_2 \in [F]$ two functions with incomparable kernels.

Let $f \in [F]$ such that $f^\bullet = \{(a_1, a_2); (b_1, b_2); (c_1, c_2)\}$. Assume without loss of generality that $\ker f_1 = \{0, 1\}$ and $\ker f_2 = \{1, 2\}$ with $f_1, f_2 \in [F]$ being two functions with incomparable kernels. Hence we must have $M(\pi(0), \pi(1), \pi(2)) = 1$ for every permutation π.

Thus, we have the equation $M(a_1, b_1, c_1) = 1$. We split the proof into two sub-cases. Fist, assume that a_2, b_2 and c_2 are pairwise different. Then we must have $M(a_2, b_2, c_2) = 1$. This is the case if and only if f is the identity operation or a swap permutation such that $\operatorname{swap} f = \{0, 2\}$. The identity operation is always included in the closure of every set of functions, and since we have $\operatorname{width} F = 2$ with $\ker f_1$ and $\ker f_2$ incomparable, by Corollary 5.2.6, we know that such a swap permutation will not increase the width of F. Then the identity operation and such a swap permutation are allowed.

Second, assume that at least two elements among a_2, b_2 and c_2 are equal. Without loss of generality, assume that $a_2 = b_2$. This implies $f(a_1) = f(b_1)$ and $M(a_2, a_2, c_2) = a_2$. Since 1 belongs to the kernel of any function $f \in [F]$, we can be sure that the tuple $(1, a_2)$ belongs to $f^\bullet$.

We conclude that M is a polymorphism of $F^\bullet$. □

5.2.3 Intractable case

This section presents the result of NP-completeness of $\textsc{csp}(F^\bullet)$ where we have $\operatorname{width} F = 3$ on a 3-element domain. Recall that, thanks to Lemma 4.2.6, one can assume that $F^\bullet$ is a core.

We will show that, for every set F of unary functions on a ternary domain such that width $F = 3$, $F^\bullet$ cannot be closed under any constant operations, any majority operations, any affine operations, any binary idempotent operations that are not projections and any semi-projections. Then we deduce by Rosenberg's list [Ros86] that the complexity of CSP($F^\bullet$) is NP-complete.

Proposition 5.2.12 *Let F be a set of unary functions on a ternary domain D such that* width $F = 3$. *Then $F^\bullet$ cannot be closed under any constant operation.*

Proof: We recall we consider $F^\bullet$ to be a core over a ternary domain, and by Lemma 4.2.6, we can consider that $F^\bullet$ contains the graph of every constant functions. It is clear that a set of relations containing at least the graph of two different constant functions cannot be closed under any constant operation. □

Proposition 5.2.13 *Let F be a set of unary functions on a ternary domain D such that* width $F = 3$. *Then $F^\bullet$ cannot be closed under any majority operation.*

Proof: Let $f, g, h \in F$ be three functions such that their kernel are pairwise incomparable. Without loss of generality, we define f, g and h as follows:

x	$f(x)$	$g(x)$	$h(x)$
0	a_f	b_g	b_h
1	b_f	a_g	b_h
2	b_f	b_g	a_h

with $a_f, b_f, a_g, b_g, a_h, b_h \in D$.

Remark that F^H_{rhs} is an A-shape like presented in Lemma 5.2.10. Thus, this lemma we conclude that $F^\bullet$ cannot be closed under any majority operation. □

Proposition 5.2.14 *Let F be a set of unary functions on a ternary domain D such that* width $F = 2$. *Then $F^\bullet$ cannot be closed under any minority operation.*

Proof: Let $f, g \in F$ be two functions such that their kernel are incomparable. Without loss of generality, we define f and g as follow:

x	$f(x)$	$h(x)$
0	a_f	b_h
1	b_f	b_h
2	b_f	a_h

with $a_f, b_f, a_h, b_h \in D$.

Let m be a minority operation. Then, we must have the equation $m(a_f, b_f, b_f) = a_f$, and that implies the equation $m(0,1,2) = 0$. Thus, we must have the equation $m(b_h, b_h, a_h) = b_h$, but this is a contradiction with the fact that m is a minority operation. □

Proposition 5.2.15 *Let F be a set of unary functions on a ternary domain D such that* $\operatorname{width} F = 3$. *Then $F^\bullet$ cannot be closed under any semi-projection.*

Proof: The proof is the same used for Proposition 4.4.14. Let $f, g, h \in F$ be three functions such that their kernel are pairwise incomparable. Without loss of generality, we define f, g and h as follow:

x	$f(x)$	$g(x)$	$h(x)$
0	a_f	b_g	b_h
1	b_f	a_g	b_h
2	b_f	b_g	a_h

with $a_f, b_f, a_g, b_g, a_h, b_h \in D$.

Let s be a semi-projection that is a polymorphism of $F^\bullet$. Let $x, y, z \in D$ be values such that $|\{x, y, z\}| < 3$ holds. Without loss of generality, we consider that $s(x, y, z)$ will project on the last element (here z). Since s is a polymorphism of $F^\bullet$, we have $s(b_f, b_f, a_f) = a_f$, $s(b_g, b_g, a_g) = a_g$ and $s(b_h, b_h, a_h) = a_h$. Then we must have $s(1,2,0) = s(2,1,0) = 0$, $s(2,0,1) = s(0,2,1) = 1$ and $s(1,0,2) = s(0,1,2) = 2$. This implies that s is a projection, which is a contradiction since by definition, a semi-projection is not a projection. □

Proposition 5.2.16 *Let F be a set of unary functions on a ternary domain D such that* $\operatorname{width} F = 3$. *Then $F^\bullet$ cannot be closed under any binary idempotent operation which is not a projection.*

Proof: Let $f, g, h \in F$ be three functions such that their kernel are pairwise incomparable. Without loss of generality, we define f, g and h as follow:

x	$f(x)$	$g(x)$	$h(x)$
0	a_f	b_g	b_h
1	b_f	a_g	b_h
2	b_f	b_g	a_h

with $a_f, b_f, a_g, b_g, a_h, b_h \in D$.

Let q be a binary idempotent operation which is not a projection. Aiming a contradiction, assume that q is a polymorphism of $F^\bullet$. Since q is idempotent, we must have $q(b_f, b_f) = b_f$, $q(b_g, b_g) = b_g$ and $q(b_h, b_h) = b_h$. We conclude that we must have the inclusions $q(1,2) \subseteq \{1,2\}$, $q(0,2) \subseteq \{0,2\}$ and $q(0,1) \subseteq \{0,1\}$. In other words, q must be conservative.

Moreover, q must not be commutative on tuples $(a,b) \in D^2$, that is to say, for every tuple $(a,b) \in D^2$, we have $q(a,b) = c$ and $q(b,a) = \overline{c}$ such that $c = a$ if and only if $\overline{c} = b$ and $c = b$ if and only if $\overline{c} = a$. In order to prove it by contradiction, suppose there exists a tuple $(a_g, b_g) \in D^2$ such that $q(a_g, b_g) = q(b_g, a_g) = d$ with $d \in D$, then we must have the following result:

	0	a_f	2	b_f	0	b_g	1	a_g	1	b_h	2	a_h
	1	b_f	0	a_f	1	a_g	2	b_g	2	a_h	0	b_h
q	$\downarrow$	$\downarrow$	$\downarrow$	$\downarrow$	$\downarrow$	$\downarrow$	$\downarrow$	$\downarrow$	$\downarrow$	$\downarrow$	$\downarrow$	$\downarrow$
	k	c	k'	$\overline{c}$	k	d	k''	d	k''	$\overline{e}$	k'	e

Suppose that the equation $k = 0$ holds. Then, we must have $c = a_f$, $\overline{c} = b_f$, $k' = 2$, $d = b_g$, $k'' = 2$, $\overline{e} = a_h$ and $e = b_h$. However, we have $k' = 2$ and $e = b_h$ but $(2, b_h) \notin h^\bullet$.

Suppose now that the equation $k = 1$ holds. Then, we must have $c = b_f$, $\overline{c} = a_f$, $k' = 0$, $d = a_g$, $k'' = 1$, $\overline{e} = b_h$ and $e = a_h$. However, we have $k' = 0$ and $e = a_h$ but $(0, a_h) \notin h^\bullet$.

Remark also that q must not be commutative on a tuple $(a,b) \in D^2$ such that the set $\{a,b\}$ is different to the range of f, g and h. Indeed, suppose we have $a_f = b_g = 0$, $b_f = a_g = a_h = 1$ and $b_h = 2$, thus $\operatorname{ran} f = \operatorname{ran} g = \{0,1\}$ and $\operatorname{ran} h = \{1,2\}$, and assume we have $\{a,b\} = \{0,2\}$, thus $q(0,2) = q(2,0)$ holds. Since q is a polymorphism of $F^\bullet$, we must have $q(a_f, b_f) = q(b_f, a_f)$, that is to say, $q(0,1) = q(1,0)$. But then, we will find the same contradiction as above when we assumed that $q(a_g, b_g) = q(b_g, a_g)$ holds.

We see that q must be a binary idempotent conservative operation and non-commutative on all tuples $(a,b) \in D^2$. Hence we must have the following result:

	1	b_f	2	b_f	1	a_g	1	a_g	1	b_h	2	a_h
	0	a_f	0	a_f	0	b_g	2	b_g	2	a_h	0	b_h
q	$\downarrow$	$\downarrow$	$\downarrow$	$\downarrow$	$\downarrow$	$\downarrow$	$\downarrow$	$\downarrow$	$\downarrow$	$\downarrow$	$\downarrow$	$\downarrow$
	k	c	k'	c	k	d	k''	d	k''	$\overline{e}$	k'	e

Suppose that the equation $k = 0$ holds. Then, we must have $c = a_f$, $k' = 0$, $d = b_g$, $k'' = 2$, $\overline{e} = a_h$ and $e = b_h$. Since for every tuple $(a,b) \in D^2$,

we have $q(a,b) = c$ and $q(b,a) = \overline{c}$ such that $c = a$ iff $\overline{c} = b$ and $c = b$ iff $\overline{c} = a$, we see here that q is a projection on the second element.

Suppose now that the equation $k = 1$ holds. Then, we must have $c = b_f$, $k' = 2$, $d = a_g$, $k'' = 1$, $\overline{e} = b_h$ and $e = a_h$. Since for every tuple $(a,b) \in D^2$, we have $q(a,b) = c$ and $q(b,a) = \overline{c}$ such that $c = a$ iff $\overline{c} = b$ and $c = b$ iff $\overline{c} = a$, we see here that q is a projection on the first element.

We conclude that if q is a binary idempotent operation which is a polymorphism of $F^\bullet$, then q must be a conservative non commutative operation, and moreover a projection. Then $F^\bullet$ cannot be closed under any binary idempotent operation which is not a projection. □

Proposition 5.2.17 *Let D be a ternary domain, and F be a set of unary functions on D such that* $\operatorname{width} F = 3$. *Then* $\textsc{csp}(F^\bullet)$ *is* NP*-complete.*

Proof: We know by [JCG97] that, for every set of relations S, if the set of polymorphisms $\operatorname{Pol} S$ does not contain a constant operation, a majority operation, a binary idempotent operation, a minority operation or a semi-projection, then $\textsc{csp}(S)$ is NP-complete.

Let F be such a set of functions. By Propositions 5.2.12, 5.2.13, 5.2.14, 5.2.15 and 5.2.16, we know that $\operatorname{Pol} F^\bullet$ does not contain such operations. Indeed, notice that we know by Proposition 5.2.14 that $\operatorname{Pol} F^\bullet$ does not contain any minority operations if we have $\operatorname{width} F = 2$, so this obviously holds also for $\operatorname{width} F = 3$. We conclude that if $\operatorname{width} F = 3$ holds then we have $\textsc{csp}(F^\bullet)$ NP-complete. □

5.2.4 Dichotomy Theorem and meta-problem

We can present now the main result of this section.

Theorem 5.2.18 *Let D be a ternary domain, and F be a set of unary functions on D. The problem* $\textsc{csp}(F^\bullet)$ *is in* P *if* $\operatorname{width} F \leq 2$ *holds. Otherwise, it is* NP*-complete.*

Proof: The proof is direct thanks to Propositions 5.2.11 and 5.2.17. □

Let's focus now on the meta-problem, that is, the decision if the dichotomy criteria is verified or not. Remark we can go further than Proposition 5.2.3: we do not need to compute the clone $[F]$ to know the width of F, whatever the functions contained in F. Thus, the next proposition allows us to determine when the Dichotomy Criterion of Theorem 5.2.18 is verified by just analyzing the kind of functions composing the set F.

Proposition 5.2.19 *Let D be a ternary domain. Let F be a set of unary functions on D. Then we have* $\operatorname{width} F = 3$ *if at least one of these conditions holds:*

(i) F contains at least three functions with incomparable kernels,

(ii) F contains two functions f and g with incomparable kernels and a swap permutation s such that $\ker f \cap \ker g \in \operatorname{swap} s$,

(iii) for every $f, g \in F$, we have either $\ker f \subseteq \ker g$ *or* $\ker g \subseteq \ker f$ *and F contains at least one function f such that* $|\ker f| = 2$ *and a permutation p that is neither a swap permutation nor the identity, or two different swap permutations s_1 and s_2.*

If none of these conditions holds, then we have $\operatorname{width} F \leq 2$.

Proof:
(i) is obvious.
(ii) is immediate by Proposition 5.2.8.

For (iii), if p is a circular permutation without being the identity, then we immediately conclude by Proposition 5.2.9. If F contains two different swap permutations s_1 and s_2, then $\operatorname{swap} s_1 \neq \operatorname{swap} s_2$ holds, and the combination $s_1 \circ s_2$ leads to a circular permutation. Thus, we also conclude bu Proposition 5.2.9.

If none of these three conditions holds, then one of the following case is verified:

(1) $\operatorname{width} F = 2$ and F does not contain any permutation,

(2) $\operatorname{width} F = 2$ and F contains a swap permutation s such that $\ker F \cap \ker g \notin \operatorname{swap} s$, for any $\ker f$ and $\ker g$ incomparable kernels,

(3) $\operatorname{width} F = 2$ and there exists $f \in F$ such that $|\ker f| = 2$, and F contains exactly one swap permutation s such that $\operatorname{swap} s \neq \ker f$ holds, but does not contain any circular permutations.

(4) $\operatorname{width} F = 1$ and there no functions $f \in F$ such that $|\ker f| = 2$,

(5) $\operatorname{width} F = 1$ and there exists $f \in F$ such that $|\ker f| = 2$, but F does not contain any permutation other than the identity,

(6) $\operatorname{width} F = 1$ and there exists $f \in F$ such that $|\ker f| = 2$, and F contains exactly one swap permutation s such that $\operatorname{swap} s = \ker f$ holds, but does not contain any circular permutations.

For (1), it is immediate by Proposition 5.2.3.

For (2), we conclude thanks to Corollary 5.2.6.

For (3), we can conclude using Corollary 5.2.7. Indeed, since $\operatorname{width} F = 1$ then for every functions $f_1, f_2 \in F$ such that $|\ker f_i| = 2$, we must have $\ker f_1 = \ker f_2$.

For (4), remark that every kernel must be a trivial kernel, that is either empty a kernel or kernel with one equivalence class containing all elements in D. Thus, F contains permutations and constant functions only. It is clear that we cannot increase the width by combining such functions.

For (5), this is again direct by Proposition 5.2.3.

For (6), since $\operatorname{swap} s = \ker f$ holds, we conclude thanks to Proposition 5.2.4. □

5.3 Homogeneous co-Boolean csp, NP-completeness and kernel width

In this section, a set of functions F is a set of homogeneous co-Boolean functions such that for every $f \in F$, we have $f(0) = 0$ and $f(1) = 1$, except for two functions f_0 and f_1 which are respectively the constant functions outputting 0 and 1 and which will always be included in F. Thus, for every function f in F we have $\operatorname{ran} f \subseteq \{0, 1\}$.

Notice that in this special configuration, $F^\bullet$ must be a core. It is easy to see that $[F] = F$ because for every $f_i, f_j \in F$ with $f_i \notin \{f_0, f_1\}$, the equation $f_i \circ f_j = f_j$ holds (thus one can see that if n is the biggest number of incomparable kernels in F, then $\operatorname{width} F = n$). Remark also that in this special configuration, the matrix F^H_{rhs} cannot contain any duplicated rows. Notice also that $F^\bullet$ cannot be closed under any constant operations since F contains constant functions f_0 and f_1.

The principal result of this section is the following one. Let F be a set of functions as presented above, and let D be a fixed domain. If $\operatorname{width} F \geq |D|$, then $\textsc{csp}(F^\bullet)$ is NP-complete. Notice that this result does not bring us a new knowledge on the complexity of such CSP because we already know the dichotomy on graphs of co-Boolean functions (see Theorem 4.3.5). However, it is the first known result about width and NP-completeness over every cardinality of finite domain, and we can hope to extend this to every CSP over graph of unary functions.

By Lemma 4.3.1, we know that if F^H_{rhs} is neither closed under majority, nor minority, nor max, nor min, then $\textsc{csp}(F^\bullet)$ is NP-complete. In the following two subsections, we will demonstrate that if $\operatorname{width} F \geq |D|$ then F^H_{rhs}

is not closed under majority, nor minority, nor max and and min, allowing us to conclude immediately about the NP-completeness of CSP($F^\bullet$).

We would like to thank a lot to Vincent Jost and David Savourey for the proofs of Propositions 5.3.1 and 5.3.3.

5.3.1 Non-stability under Max and Min

We prove in this subsection that if width $F \geq |D|$, then F^H_{rhs} is not closed under max and min operations.

Proposition 5.3.1 *Let F be a set of homogeneous co-Boolean functions such that for every $f \in F$, equations $f(0) = 0$ and $f(1) = 1$ hold, except for the two constant functions $f_0, f_1 \in F$ outputting 0 and 1. If* width $F \geq |D|$, *then F^H_{rhs} is not closed under max operation.*

Proof: We prove this proposition for width $F = |D| = n$ such that the extension of the proof for width $F \geq |D|$ will be trivial. Thus, assume that we have width $F = |D| = n$.

First, observe that F^H_{rhs} is a n^2-Boolean matrix such that the first row is composed only with 0s, the second one only with 1s, all row are pairwise different (otherwise, $F^\bullet$ is not a core) and all columns are different (because all functions are different). Actually, F^H_{rhs} can have more than n columns, but this does not influence the proof.

To prove this proposition, we will show that it is impossible to built such a matrix that is closed under a max operation (*i.e.*, rows are not closed under a max operation).

Let F^H_{rhs} a Boolean matrix with n column such that F^H_{rhs} is closed under max, and such that we does not have the first row (*i.e.* the 0-vector) but we have the second one (the 1-vector). Consider the row r with the largest number of 1's, excluding the 1-vector. Let k the number of 1's in r. We show that k should be equal to $n-1$.

Assume that $k \leq n-2$, then there exist at least two 0 in r. Let c_1 and c_2 two columns were there is a 0 in r. We know that c_1 and c_2 should be different, then there must exists a row r_0 that makes the difference between these two columns putting, for instance, a 0 for c_1 and a 1 for c_2. Hence, $max(r_0, r)$ gives us a row in the matrix with $k+1$ values 1 inside, that is a contradiction with r the row with the largest number of 1.

Thus we must have $k = n-1$, that is to say a row r with $n-1$ values 1 and only one 0 in a column c. It is clear that if we remove r and c from the matrix, we obtain an $(n-1)^2$-matrix verifying the properties of the

initial matrix, *i.e.* the stability under max, the pairwise difference between columns, and between rows, where the row with the biggest number of value 1 contains this time $n-2$ values 1, and so on.

Hence, it is clear that we need at least n rows to built such a matrix, but without the 0-vector row. Then we cannot built such a matrix, including the 0-vector, in only n rows.

It is clear that having $m > n$ columns implies the fact that we need $m+1$ rows. Then, the contradiction still holds. □

Proposition 5.3.2 *Let F be a set of homogeneous co-Boolean functions such that for every $f \in F$, equations $f(0) = 0$ and $f(1) = 1$ hold, except for the two constant functions $f_0, f_1 \in F$ outputting 0 and 1. If* $\operatorname{width} F \geq |D|$, *then F^H_{rhs} is not closed under min operation.*

Proof: The proof is the same as the proof of Proposition 5.3.1, with the difference that we do not take the 1-vector to build a matrix closed under min, and then we see that we need at least $n+1$ rows, including the 1-vector, to have such a matrix. □

5.3.2 Non-stability under Majority and Minority

We prove in this subsection that if $\operatorname{width} F \geq |D|$, then F^H_{rhs} is not closed under majority and minority operations. Again, we deal with the case $\operatorname{width} F = |D| = n$, and the extension of the proof for $\operatorname{width} F \geq |D|$ is trivial.

Recall that the xor operation $\oplus$ is the binary operation on the Boolean domain such that $x \oplus y$ is true (or outputs 1 if you prefer) if and only if x and y are evaluate to different values, that is, one is evaluate to 1 and the other to 0.

First, remark that is a matrix is closed under a majority operation, then, composed with the 0-vector, it is equivalent to say that the matrix is closed under min operation, and composed with the 1-vector, to the max operation. Then the non-stability of F^H_{rhs}, such that $\operatorname{width} F = |D| = n$, under a majority operation is done by Propositions 5.3.1 and 5.3.2.

If a matrix is closed under a minority operation, then, composed with the 0-vector, it is equivalent to say that the matrix is closed under the xor operation, and composed with the 1-vector, to the negation of xor. We will prove in the following proposition that F^H_{rhs} cannot be closed under xor and the negation of xor if we have $\operatorname{width} F = |D| = n$.

Proposition 5.3.3 *Let F be a set of homogeneous co-Boolean functions such that for every $f \in F$, equations $f(0) = 0$ and $f(1) = 1$ hold, except for the two constant functions $f_0, f_1 \in F$ outputting 0 and 1. If* $\operatorname{width} F \geq |D|$, *then F^H_{rhs} is not closed under the xor operation.*

Proof: To prove this proposition, we will show that it is impossible to built such a matrix that is closed under the xor operation.

Let F^H_{rhs} be a Boolean matrix with n columns such that F^H_{rhs} is closed under xor, and such that we does not have the 1-vector row, but we have the 0-vector row. We remark that the set of rows and the xor operation form a group G since xor is associative, the 0-vector is the identity element, and each element is its own inverse element, *i.e.* for every row r, $xor(r, r)$ gives us the 0-vector.

Consider the set $S = (r_1, \ldots, r_k)$ the minimal basis of G, that is, every element in G can be obtained through a linear combination of elements in S, and none of elements in S can be obtain through such a linear combination. Remark that S does not contain the 0-vector.

Recall that rows in F^H_{rhs} are pairwise different, otherwise $F^\bullet$ is not a core. Then, we can produce a row in G with the help of the following linear combination

$$r = \lambda_1 r_1 \oplus \lambda_2 r_2 \oplus \ldots \oplus \lambda_k r_k$$

with $\oplus$ the xor operation. Then, it is clear that we have exactly 2^k elements in G since each rows are pairwise different.

Remark that if there exist two columns c_1 and c_2 in the base S which are the same, then one can only produce rows in G such that c_1 and c_2 remains the same. However, this would be a contradiction with the fact that columns in G are pairwise different. Then, columns in S must also be pairwise different.

It is also clear that we need at least $log(n)$ rows to make the difference between n columns. Then, the basis S contains at least $log(n)$ rows, and G, that is to say the matrix F^H_{rhs} itself, must at least contains $2^{log(n)}$ rows, *i.e.* n rows. Then, if we include the 1-vector, we cannot have such a matrix closed under xor with n columns and only n rows pairwise different. □

Proposition 5.3.4 *Let F be a set of homogeneous co-Boolean functions such that for every $f \in F$, equations $f(0) = 0$ and $f(1) = 1$ hold, except for the two constant functions $f_0, f_1 \in F$ outputting 0 and 1. If* $\operatorname{width} F \geq |D|$, *then F^H_{rhs} is not closed under the negation of xor operation.*

Proof: The proof is the same as the proof of Proposition 5.3.3, with the difference that we do not take the 0-vector to build a matrix closed under the negation of xor, and then we see that we need at least $n+1$ rows (actually, even more), including the 0-vector, to have such a matrix. □

Now, we can introduce the theorem of this section.

Theorem 5.3.5 *Let F be a set of homogeneous co-Boolean functions over a finite domain D such that for every $f \in F$, equations $f(0) = 0$ and $f(1) = 1$ hold, except for the two constant functions $f_0, f_1 \in F$ outputting 0 and 1. If the inequality* $\operatorname{width} F \geq |D|$ *holds, then* $\text{CSP}(F^\bullet)$ *is* NP*-complete.*

Proof: Since F is a set of homogeneous co-Boolean functions, by Schaefer's theorem we know that $\text{CSP}(F^H_{rhs})$ is NP-complete if F^H_{rhs} is neither closed under constant operations which output 0 or 1, nor majority, minority, maximum and minimum operations. Lemma 4.2.6 allows us to directly work with cores. Thus, assuming F is a core, F^H_{rhs} cannot be closed under any constant operation. Propositions 5.3.1 and 5.3.2 show that, since we have $\operatorname{width} F \geq |D|$, F^H_{rhs} cannot be closed under a maximum or minimum operation. Finally, Propositions 5.3.3 and 5.3.4 show that, under the condition $\operatorname{width} F \geq |D|$, F^H_{rhs} cannot be closed under a majority or minority operation. We conclude that $\text{CSP}(F^H_{rhs})$ is NP-complete, and then, by Lemma 4.2.4 and Proposition 4.2.3, that $\text{CSP}(F^\bullet)$ is NP-complete. □

5.3.3 Conjecture

Conjecture 5.3.6 *Let F be any set of unary functions. If* $\operatorname{width} F \geq |D|$*, then* $\text{CSP}(F^\bullet)$ *is* NP*-complete.*

5.4 Conclusion

We have seen in this chapter the very first results on complexity of CSP over unary functions obtained through the study of kernel width. This chapter contained three main results: we proved that, for the $\text{CSP}(F^\bullet)$ problem over unary functions on every finite domain, the linear kernel width implies the tractability of $\text{CSP}(F^\bullet)$. Then, we shown a Dichotomy Theorem of the complexity of the $\text{CSP}(F^\bullet)$ problem over unary functions on a ternary domain via a criterion based on the kernel width. Finally, we proved the NP-completeness of homogeneous co-Boolean $\text{CSP}(F^\bullet)$ problem where every unary function $f \in F$ is such that $f(0) = 0$ and $f(1) = 1$ hold, except

for the two constant functions $f_0, f_1 \in F$ outputting respectively 0 and 1. This result holds for every finite domain D if the kernel width is higher than or equals to the cardinality of D. It leads us to the conjecture that, is this criterion is verified, the CSP($F^\bullet$) problem over unary functions is intractable.

By a lack of time, we have not yet reproved our Dichotomy Theorem of CSP($F^\bullet$) with non-homogeneous co-Boolean functions, as well as we have not yet study the complexity of other problems on different family of unary functions, but we place great hopes in this powerful method to develop new results on the tractability and intractability of some CSP problems on unary functions.

Bibliography

[Bod08] M. Bodirsky. Constraint Satisfaction Problems with Infinite Templates. *Complexity of Constraints, LNCS 5250*, 196–228, 2008.

[BG08] M. Bodirsky and M. Grohe. Non-Dichotomies in Constraint Satisfaction Complexity. *Proceedings 35th International Colloquium on Automata, Languages and Programming (ICALP 2008), Part II, Reykjavik (Iceland)*, 184–196, 2008.

[BHR09] M. Bodirsky, M. Hermann and F. Richoux. Complexity of Existential Positive First-Order Logic. *Proceedings 5th Conference on Computability in Europe (CiE 2009), Heidelberg (Germany)*, 2009.

[BKKR69] V. Bodnarchuk, L. Kaluznin, V. Kotov and B. Romov. Galois theory for Post algebras. I and II. *Cybernetics*, 5:243–253, 531–539, 2009.

[BCRV03] E. Böhler, N. Creignou, S. Reith and H. Vollmer. Playing with Boolean blocks, part I: Post's lattice with applications to complexity theory. *SIGACT News, Complexity Theory Column 42*, 34(4):38–52, 2003.

[BCRV04] E. Böhler, N. Creignou, S. Reith and H. Vollmer. Playing with Boolean blocks, part II: Constraint satisfaction problems. *SIGACT News, Complexity Theory Column 43*, 35(1):22–35, 2004.

[Bul06a] A. Bulatov. A Dichotomy Theorem for Constraint Satisfaction Problems on a 3-element Set. *Journal of the Association for Computing Machinery*, 53(1):66–120, 2006.

[Bul06b] A. Bulatov. Combinatorial Problems Raised from 2-semilattices. *Journal of Algebra*, 298(2):321–339, 2006.

[BG05] A. Bulatov and M. Grohe. The Complexity of Partition Functions. *Theoretical Computer Science*, 348(2-3):148–186, 2005.

[BKJ00] A. Bulatov, A. Krokhin and P. Jeavons. Constraint Satisfaction Problems and Finite Algebras. *Proceedings 27th International Colloquium on Automata, Languages and Programming (ICALP 2000), Geneva (Switzerland)*, 272–282, 2000.

[BJK05] A. Bulatov, P. Jeavons and A. Krokhin. Classifying the Complexity of Constraints using Finite Algebras. *SIAM Journal on Computing*, 34(3):720–742, 2005.

[BW87] S. Burris and R. Willard. Finitely Many Primitive Positive Clones. *Proceedings of the American Mathematical Society*, 101(3):427–430, 1987

[Bro05] P. Broniek. Solving Equations over Small Unary Algebras. *Discrete Mathematics and Theoretical Computer Science*, 49–60, 2006.

[CCHS08] V. Chepoi, N. Creignou, M. Hermann, and G. Salzer. Deciding the Satisfiability of Propositional Formulas in Finitely-Valued Signed Logics. *Proceedings 38th IEEE International Symposium on Multiple-Valued Logic (ISMVL 2008), Dallas (USA)*, 100–105, 2008.

[CJJK00] D. Cohen, P. Jeavons, P. Jonsson, and M. Koubarakis. Building Tractable Disjunctive Constraints. *Journal of the Association for Computing Machinery*, 47(5):826–853, 2000.

[CHKS08] N. Creignou, M. Hermann, A. Krokhin and G. Salzer. Complexity of Clausal Constraints Over Chains. *Theory of Computing Systems*, 42(2):239–255, 2008.

[CKV08] N. Creignou, Ph. Kolaitis and H. Vollmer. Complexity of Constraints - An Overview of Current Research Themes. *LNCS 5250, Springer*, 2008.

[CV08] N. Creignou and H. Vollmer. Boolean Constraint Satisfaction Problems: When does Post's Lattice Help? *Complexity of Constraints*, 3–37, 2008.

[DF99] R. Downey and M. Fellows. Parametrized Complexity. *Springer-Verlag*, 1999.

[FMS04] T. Feder, F. Madelaine and I. Stewart. Dichotomies for Classes of Homomorphism Problems Involving Unary Functions. *Theoretical Computer Science*, 314(1-2):1–43, 2004.

[FV98] T. Feder and M. Vardi. The Computational Structure of Monotone Monadic SNP and Constraint Satisfaction: a Study Through Datalog and Group Theory. *SIAM Journal on Computing*, 28(1):57–104, 1998.

[GJ79] M. Garey and D. Johnson. Computers and Intractability: A Guide to the Theory of NP-Completeness. *W.H. Freeman and Co*, 1979.

[Gei68] D. Geiger. Closed systems of functions and predicates. *Pacific Journal of Mathematics*, 27(2):228–250, 1968.

[Her08] M. Hermann. On Boolean Primitive Positive Clones. *Discrete Mathematics*, 308(15):3151–3162, 2008.

[HR09] M. Hermann and F. Richoux. On the Computational Complexity of Monotone Constraint Satisfaction Problems. *Proceedings 3rd Annual Workshop on Algorithms and Computation (WALCOM 2009), Kolkata (India)*, 286–297, 2009.

[Jea98] P. Jeavons. On the algebraic structure of combinatorial problems. *Theoretical Computer Science*, 200(1-2):185–204, 1998.

[JC95] P. Jeavons and D. Cohen. An Algebraic Characterization of Tractable Constraints. *Proceedings of the First Annual International Conference on Computing and Combinatorics (COCOON 1995), Xi'an (China)*, 959:633–642, 1995.

[JCG97] P. Jeavons, D. Cohen, and M. Gyssens. Closure Properties of Constraints. *Journal of the Association for Computing Machinery*, 44(4):527–548, 1997.

[JCG99] P. Jeavons, D. Cohen, and M. Gyssens. How to Determine the Expressive Power of Constraints. *Constraints*, 4(2):113–131, 1999.

[JCP98] P. Jeavons, D. Cohen, and J. Pearson. Constraints and Universal Algebra. *Annals of Mathematics and Artificial Intelligence*, 24(1-2):51–67, 1998.

[Kra38] M. Krasner. Une généralisation de la notion de corps. *Journal de Mathématiques pures et appliquées*, 17:367–385, 1938.

[Lau06] D. Lau. Function algebras on finite sets: Basic course on many-valued logic and clone theory. *Springer*, 2006.

[LLS75] R. Ladner, N. Lynch and A. Selman. A Comparison of Polynomial-Time Reducibilities. *Theoretical Computer Science*, 1(2), 103–123, 1975.

[Mar06] B. Martin. Dichotomies and Duality in First-order Model Checking Problems. *CoRR abs/cs/0609022*, 2006.

[Mar08] B. Martin. First-Order Model Checking Problems Parameterized by the Model. *Proceedings 4th Conference on Computability in Europe (CiE 2008), Athens (Greece)*, 417–427, 2008.

[Mon74] U. Montanari. Networks of Constraints: Fundamental Properties and Applications to Picture Processing. *Information Sciences*, 7:95–132, 1974.

[Pap94] Ch. Papadimitriou. Computational Complexity. *Addison Wesley Longman*, 1994.

[Pös80] R. Pöschel. A General Galois Theory for Operations and Relations and Concrete Characterization of Related Algebraic Structures. *Report R01/80*, Zentralinstitut für Mathematik und Mechanik, Akademie der Wissenschaften der DDR, Berlin, 1980c.

[Pös04] R. Pöschel. Galois Connections for Operations and Relations. *Galois Connections and Applications*, 231–258, 2004.

[Pos41] E. Post. The two-valued iterative systems of mathematical logic. *Annals of Mathematical Studies*, 5:1–122, 1941.

[Ros86] I. Rosenberg. Minimal clones I: the five types. *Lectures in Universal Algebra*, 405–427, 1986.

[Sal03] A. Salomaa. Composition Sequences for Functions over a Finite Domain. *Theoretical Computer Science*, 292(1):263–281, 2003.

[Sch78] T. Schaefer. The Complexity of Satisfiability Problems. In *Proceedings 10th Symposium on Theory of Computing (STOC'78), San Diego (California, USA)*, 216–226, 1978.

[Sip97] M. Sipser. Introduction to the Theory of Computation. *PWS Publishing Company*, 1997.

[YM59] Yu. Yanov and A. Muchnik. On the Existence of k-valued Closed Classes that have no Bases. *Doklady Akademii Nauk SSSR*, 127:44–46, 1959. In Russian.