Practical resolution of satisfiability testing for modal logics

Valentin Montmirail

Practical resolution of satisfiability testing for modal logics

Doctoral thesis

presented and publicly defended on 17^{th} September 2018

to obtain the title of

Doctor of Science from Artois University

(Artificial Intelligence)

by

Valentin Montmirail

Composition of the jury

President:	Laurent Simon	LaBRI, Bordeaux University, France
Reporters:	Martina Seidl	Johannes Kepler University, Austria
	Andreas Herzig	CNRS & Paul Sabatier University, France
Supervisors:	Jean-Marie Lagniez	Artois University, France
	Tiago de Lima	Artois University, France
Advisor:	Daniel Le Berre	Artois University, France

Centre de Recherche en Informatique de Lens – CNRS UMR 8188
Artois University, rue Jean Souvraz, S.P. 18 F-62307, Lens Cedex France
Secretariat : Tél.: +33 (0)3 21 79 17 23 – Fax : +33 (0)3 21 79 17 70
http://www.cril.univ-artois.fr

Layout by classes thesul v0.14 (D. Roegel, LORIA) and memcril v0.20 (B. Mazure, CRIL)
The design of the boxes is made by xeboiboites (A. Flesch)

"We can only see a short distance ahead, but we can see plenty there that needs to be done."

Alan Mathison Turing (1950)

Contents

General Introduction

> "A whole is what has a beginning and middle and end"
>
> *Aristotle (335 BCE)*

Motivation

Usually, the main problem for a logic $\mathcal{L}$ is the problem to determine if, given a formula ϕ in $\mathcal{L}$, it exists an assignment of the variables in ϕ that satisfies it. This decision problem is called the 'satisfiability problem in $\mathcal{L}$'. As an example in the propositional logic (or Boolean logic Boole (1854)), the problem consists in finding, if it exists, an assignment TRUE/FALSE of all the variables in a formula, in such way that each variable can be consistently replaced by the values TRUE/FALSE to have at the end, the formula evaluated to TRUE.

In the cases of propositional modal logics that we will consider in this thesis, the complexity of the satisfiability problems will vary from NP-complete to PSPACE-complete, see Ladner (1977) and Halpern and Moses (1992) for more details.

Theoretically, if the problem is NP-hard or higher, then it is intractable and it may seems hopeless to try to solve it in practice. However, it is worth remembering that we usually talk about the "worst-case" complexity, it may be achievable to solve some instances in practice if they do not represent the "worst" possible case.

Many teams of researchers are creating solvers able to decide the satisfiability in their logic $\mathcal{L}$ and most of the time compete against each other to see which team created the fastest solver and how. By this mean, we can have a pretty clear view about the state-of-the-art approach for solving the satisfiability problem in logic $\mathcal{L}$. A non-exhaustive list of such communities is given in Table 1.

Community	**Start**	**Competition / Evaluation**	**Complexity**
SAT Community	Since 1992	SAT Buro and Büning (1993)	NP
ATP Community	Since 1996	CASC Sutcliffe and Suttner (1997)	Undecidable
QBF Community	Since 2003	QBFEVAL Le Berre et al. (2003)	PSPACE
CP Community	Since 2005	XCSP Roussel and Lecoutre (2009)	NP
MaxSAT Community	Since 2006	MaxSAT Argelich et al. (2008)	NP
PB Community	Since 2005	PB Manquinho and Roussel (2006)	NP
ASP Community	Since 2007	ASPCOMP Gebser et al. (2007)	Σ_2^P

Table 1: Some competitions of solvers of different complexities

These competitions have also an interesting side-effect: they force developers to make their solvers robust, efficient and re-usable from one year to another. This side-effect may be considered as the first step of why SAT solvers are nowadays used in so many applications as for example, in Bounded Model Checking Biere (2009), in Planning Rintanen (2009), in Software dependency Le Berre and Rapicault (2018), in Software Verification Kroening (2009), the list could go on and on. The SAT community manages to make SAT solvers usable for solving industrial problems by forcing developers to make their source-code available from one year to another, by verifying, with an independent tool, when the solver answers a model that the formula is indeed satisfied by it or by verifying the proof of unsatisfiability provided by the solver otherwise Gelder (2002). SAT solvers are such cornerstone tools in Artificial Intelligence nowadays that Edmund Clarke (Turing award 2007) claimed that "Clearly, efficient SAT solving is a key technology for the 21^{st} century computer science" Biere et al. (2009).

Modal logics solvers had also a competition, it was called TANCS (TABLEAUX Non-Classical (Modal) System Comparisons) and it was a competition of non-classical systems held in conjunction with the TABLEAUX[1] conference in 1998 and 2000. These two competitions and their results have been analysed, Balsiger and Heuerding (1998) for the competition in 1998 and Massacci and Donini (2000) for the one in 2000. Unfortunately, to the best of our knowledge since 18 years, no competition of propositional modal logic solvers was organised and here is our diagnostic for that:

1. The benchmarks available do not have the same structure as the one that 'real-life' applications could have;

2. There are no standard input and output formats for solvers of modal logics, the competitions were organised using translators to make the benchmarks readable for the solvers.

As a consequence, there are not enough efficient modal logic solvers to compete with. These are the different points that we try to address in this thesis. Let us stress now how we want to create efficient modal logic solvers. We make a claim here, that we will try to demonstrate during this thesis: to solve efficiently a problem, one of the best **current**[2] technique is to use SAT solvers, in one way or another.

In a nutshell, our approach in this thesis will be to construct a model by encoding its expected shape into a SAT encoding and giving it to the SAT solver. For NP-complete modal logics, we can encode in one shot an equisatisfiable problem in propositional logic and ask the solver, for the PSPACE-complete modal logics, we will bound the size of the Kripke models and allow more and more worlds as it is usually done in the Planning community.

For the modal logic whose satisfiability problems are NP-complete, the way that we propose here to solve them is by applying an efficient translation from modal logic to propositional logic and then call an off the shelves SAT solver on it. Thus, the difficulty of the problem is no longer in the solving phase, but it is in the encoding phase.

The disadvantage of that approach is that a SAT-based approach tends to produce unnecessary large solutions which could be a problem in the case of 'real-life' applications (where solutions have a meaning and may need to be presented to the user). We then want to find the optimal solution (to obtain the most concise solution possible). The problem is that, an optimisation problem related to an NP-complete decision problem is harder than solving the

[1] http://www.tableaux-ar.org/

[2] It could be the case that this claim is no longer true in the future...

satisfiability problem in propositional logic Cook (1971). The technique that we propose is still using a SAT solver but this time, in an incremental way. In a nutshell, we will ask the SAT solver to answer if there exists a solution of size n (while having no solution of size $n-1$). If it is the case, then we solved the problem, if it is not the case, we will ask the SAT solver for a reason of "why there is no solution of size n?", in order to find the optimal size with as few number of calls as possible.

Finally, for the case of modal logic whose satisfiability problems are PSPACE-complete, we know that the propositional logic formula will be of exponential size on the size of the input, so just translating the problem into propositional logic will likely be impractical. However, there is existing literature about how to solve PSPACE-complete problems with a SAT solver, especially recently in the QBF community Janota et al. (2016). Basically, the technique is to transform the original problem into an easier problem (called the abstraction), in such way that if the abstraction has a solution, then so does the original problem. And if the abstraction does not have a solution, the technique is to refine the abstraction, to make it "closer" to the original problem and to solve again the abstraction. They perform such loop until one abstraction is satisfiable or until the abstraction is in fact equivalent with the original problem. Such technique is called CEGAR (Counter-Example Guided Abstraction Refinement) Clarke et al. (2003) and it is efficient for some PSPACE-complete problems like QBF Janota et al. (2016), Planning Seipp and Helmert (2013).

We tried a CEGAR approach for PSPACE-complete modal logics but the results were not as good as we thought they would be. When the problem has a solution, the technique was efficient, but when the problem was unsatisfiable, either we made too many steps of refinement or we run out of the memory. To solve this problem, we propose a totally new abstraction-refinement procedure that we call RECAR. The idea behind it is to use two levels of abstraction and not just only one. One abstraction has less solutions than the original problem, but if we find a solution for it, then we solved the original problem. The other abstraction has more solutions than the original problem, but if we find no solution for it, then we know that it does not exist a solution for the original problem. The key part of the proposed approach is that these two abstractions are used interleavedly. Thus when we solve the first abstraction and the procedure returns unsatisfiable, we use a reason for "why it is unsatisfiable" in order to compute the second abstraction. And when the second abstraction is solved and the procedure returns satisfiable, we use some information from the solution to refine the first abstraction more precisely.

We instantiated our RECAR framework for the modal logic K satisfiability problem with the first abstraction being a translation into a SAT problem, in order to be competitive with the state-of-the-art approaches. We also generalize the RECAR framework in order to deal with composition of abstractions in order to outperform the different existing approaches, not only in K, but in all the PSPACE modal logics we tried.

Synopsis

Let us now explain briefly, how this thesis is organised. First we will explain the existing requirements, needed to understand our contributions:

- Part I, Chapter 1: Complexity Theory. We will present the important concepts of the complexity theory. We will explain what it means for a problem to be NP-complete,

PSPACE-complete, undecidable, *etc.* We will also explain the concept of polynomial reduction. How we can translate one problem into another without loosing informations and without a blow-up in the size of the input.

- Part I, Chapter 2: Logics. Because we will talk about different logics, like propositional logic and modal logics, it is important to define their syntax and semantics properly. These logics can also represent their formula into different forms such as CNF or NNF that we will describe.
- Part I, Chapter 3: Decision Algorithms and Benchmarks. Then, we will describe the different algorithms used to decide the satisfiability in the different logics. We will describe, how modern SAT solvers work, but also the different state-of-the-art approaches to decide the satisfiability in the different modal logics. Because the structure of the benchmark may have a huge impact on the performance of an approach, we will also describe the structural differences between benchmarks that are randomly generated, crafted or resulting from a polynomial reduction, and representing an industrial problem.

Then, we will describe our actual contributions, and how we solved the different modal logics satisfiability problems with a SAT solver as follows:

- Part II, Chapter 1: The NP Modal Logics Satisfiability Problems. In this chapter, we will present the specificity of the NP-complete modal logics. We will extract and prove correct, from this specificity, a theoretical upper-bound on the size of the Kripke model. Thanks to this upper-bound, we will propose an equisatisfiable SAT translation solvable by a SAT solver that outperforms the state-of-the-art approaches on the benchmarks considered.
- Part II, Chapter 2: The Minimal K$\star$5 Satisfiability Problem. Because the SAT-based approach provides unnecessarily large models, we will present in this chapter different technique to obtain the smallest model possible with respect to the number of possible worlds. We will compare a linear/binary search for the model with a PBO/MaxSAT solver approach and with an incremental SAT-based approach.
- Part II, Chapter 3: RECAR: An Abstraction Refinement Procedure. Then, we wanted to solve the PSPACE-complete modal logics especially the modal logic K. What we first tried was a basic CEGAR approach which was not very efficient. But thanks to this failure, we developed a new abstraction-refinement framework using two levels of abstraction that we will present in this chapter.
- Part II, Chapter 4: The Modal Logic K$\star$ Satisfiability Problem. Now that the RECAR framework is presented, we can explain in detail how we instantiated it for the modal logic K satisfiability problem. But also, how we managed to push the performance of our solver MoSaiC forward and how we made it able to deal with several different propositional modal logics.

From this point, all the preliminaries and the contributions will be presented. It will be time to conclude and to sum-up all the different contributions that we proposed in this thesis. But it will be also the time to propose the different perspectives that this thesis opens.

Part I
Preliminaries

Chapter 1

Complexity Theory

Contents

"If P=NP, then the world would be a profoundly different place than we usually assume it to be. There would be no special value in "creative leaps", no fundamental gap between solving a problem and recognizing the solution once it is found. Everyone who could appreciate a symphony would be Mozart; everyone who could follow a step-by-step argument would be Gauß."

Scott Aaronson

This first chapter will present the different theoretical concepts required to fully understand the contributions presented in this thesis. We will first describe the different notions of complexity, what does it mean for a problem to be NP-complete, PSPACE-complete, *etc.* But we will also go through the notion of Turing machine, which can be deterministic, non-deterministic or can possess an oracle. Then, from this notion, we will be able to describe the notion of completeness and explain the difference between being NP-hard and NP-complete for example. And finally, we will present one of the most important result in complexity: the polynomial hierarchy where a complexity-class is recursively defined from sub-classes and Turing machine with oracles. This chapter is definitely not a complete explanation of the complexity theory and we refer the reader to Papadimitriou (1994), Turing (1938), Soare (2016) for more details.

1.1 Computational Complexity

This section aims to describe the basic notions required to understand the following sections. In a nutshell, the complexity of an algorithm is a measure of the resources needed for its execution. Usually we measure the time and memory (space).

The complexity theory studies the asymptotic behaviour of the problems which are computable with a fixed quantity of resources, not if the problem is or not computable. Given a procedure P, the complexity theory studies the time and memory needed to compute $\mathrm{P}(n)$ with respect to the size of the parameter n.

1.1.1 Best, Worst and Average Case

We want to evaluate the complexity of an algorithm independently from the execution speed of a machine and from the quality of the code produced by a compiler. It is thus necessary to consider the time spent by a program as a number of elementary operations $\mathrm{T_P}(n)$ needed to execute a program with an input of size n.

Example 1

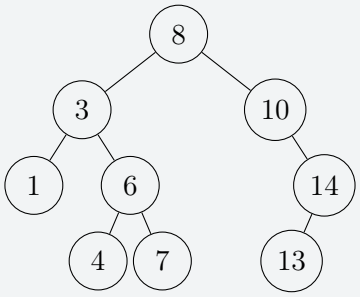


Figure 1.1: Example of a binary search tree of size 9 and depth 3.

Let us consider the following problem: we want to find an element in an ordered binary tree. To perform this search, many techniques can exist. Let us consider the easiest one that we will denote P, it will go through all the elements, by always starting from the top node, until it finds the one we are looking for.

- The best case is that the element we are looking for is the root of the tree. In that case $\mathrm{T_P}(n) = 1$;
- The worst case is when the tree is not well balanced and that we have to go through all the elements. In that case $\mathrm{T_P}(n) = n$;
- The average case depends on how well balanced is the tree. But in the case of a perfectly balanced tree, the average number of operations is $\mathrm{T_P}(n) = \log(n)$;

In practice and basically when no precision is given, it will be the case that when we talk about the complexity of an algorithm, we are discussing its worst case complexity. Indeed, the

number of elementary operations in the best case does not give any information on the real behaviour of the algorithm. For the average case, even if it gives some information, it is difficult to compute and it depends strongly on some hypothesis (in Example 1.1, how well balanced the tree is).

Definition 1 (Computational Complexity)

Let two functions $f : \mathbb{N} \to \mathbb{N}$ and $g : \mathbb{N} \to \mathbb{N}$. We say that $f(n)$ is in $\mathcal{O}(g(n))$ if and only if there exist some $n_0 \in \mathbb{N}$ and some $c \in \mathbb{N}$ such that:

$$f(n) \leq c \times g(n), \forall n \geq n_0$$

1.1.2 Different Types of Algorithm

Obviously, the complexity of an algorithm will have a strong impact on how fast it can finish. Let us assume that we need 10^{-9} seconds to execute an elementary operation (which is more or less the case on nowadays computers). An idea of the time needed to execute an algorithm according to its complexity is given in Table 1.1.

Complexity	**Type**	**n=10**	**n=20**	**n=50**	**n=250**	**n=1000**
$\mathcal{O}(1)$	Constant	1 ns	1 ns	1 ns	1 ns	1 ns
$\mathcal{O}(\log(n))$	Logarithmic	1 ns	1.3 ns	1.7 ns	2.3 ns	3.0 ns
$\mathcal{O}(n)$	Linear	10 ns	20 ns	50 ns	0.25 μs	1.0 μs
$\mathcal{O}(n \times \log(n))$	Linearithmic	10 ns	26 ns	85 ns	0.6 μs	3 μs
$\mathcal{O}(n^2)$	Quadratic	0.1 μs	0.4 μs	2.5 μs	62.5 μs	1 ms
$\mathcal{O}(n^3)$	Cubic	1 μs	8 μs	0.12 ms	15.6 ms	1 s
$\mathcal{O}(2^n)$	Exponential	1 μs	1 ms	13 days	10^{55} years	10^{283} years
$\mathcal{O}(n!)$	Factorial	3.6 ms	77 years	10^{47} years	10^{475} years	10^{2550} years

Table 1.1: Time needed to execute an algorithm according to its complexity

As we can observe on the Table 1.1, the difference in execution time between the complexity measures can reach multiple orders of magnitude. We can split the complexities in two categories: the ones bounded by a polynomial expression and the ones which are not. $\mathcal{O}(\log(n))$ is bounded by $\mathcal{O}(n)$, $\mathcal{O}(n \times \log(n))$ is bounded by $\mathcal{O}(n^2)$, and $\mathcal{O}(n^2)$ is bounded by $\mathcal{O}(n^3)$. The cases $\mathcal{O}(2^n)$ and $\mathcal{O}(n!)$ cannot be bounded by a polynomial expression. From this separation, we can thus split the different algorithms in two categories: those which have a polynomial-time execution and those which have not.

Definition 2 (Polynomial-Time Algorithm)

An algorithm P is said to be polynomial if there is an integer i such that P is of complexity in $\mathcal{O}(n^i)$.

As explained before, the complexity of an algorithm is studied through an asymptotic behaviour. If an algorithm P with an input of size n performs exactly $(n^3 + (3 \times n^2) + (10 \times n) + 9)$ elementary operations, we will consider that the algorithm is of complexity $\mathcal{O}(n^3)$.

Definition 3 (Exponential-Time Algorithm)

An algorithm P is said to be exponential if it is of complexity in $\mathcal{O}(C^{n^i})$ for some constants i and C with $C > 1$.

We exclude the case of $\mathcal{O}(n!)$ here, we will consider only polynomial and exponential-time algorithms. From now on, we know that there exists problems that can be solved with a polynomial-time algorithm and other problems that can be solved with an exponential-time algorithm. But one of the fundamental questions in complexity theory is whether there is at least one problem, that requires in the best case, an exponential-time algorithm, or said otherwise, that can not be solved in polynomial time.

1.2 Computational Problems

Algorithms are created for a given problem but complexity theorists try to classify problems that can or cannot be solved with appropriately restricted resources. In their quest for classifying problems, they created sets of problems that we will define. A computational problem can be viewed as an infinite collection of tuples constructed with an instance together with a solution. The input string for a computational problem is referred to as a problem instance, and should not be confused with the problem itself.

Example 2 (The Primality Test)

Consider the problem of primality testing. The instance is a number (*eg.* 15) and the solution is "yes" if the number is prime and "no" otherwise (in this case "no").
Stated another way, the instance is a particular input to the problem, and the solution is the output corresponding to the given input.
The computational problem of primality testing is the infinite collection of all the possible instances and the corresponding answers.

1.2.1 Decision Problem

From the notion of computational problem, we can then define some specific type of problems according to the expected output. One particular class of computational problems is the class of decision problems. They are one of the central objects of study in complexity theory. A decision problem is a special type of computational problem whose answer is either "Yes" or "No", or alternatively either 1 or 0. A decision problem can be viewed as a formal language, where the members of the language are instances whose output is yes, and the non-members are those instances whose output is no. This kind of problem is formally defined as follows:

Definition 4 (Decision Problem)

A problem Σ is a decision problem if and only if the possible outputs are "Yes" and "No". It means that the set of problems D_Σ of the possibles instances of Σ can be split into two disjoints sets:

- Y_Σ: the set of instances for which there is an algorithm solving Σ and answering "Yes".
- N_Σ: the set of instances for which there is an algorithm solving Σ and answering "No".

The Example 2 is an example of a decision problem, the expected outputs are "Yes" or "No", but there are many more decision problems.

1.2.2 Function Problem

There exists other categories of computational problems than the set of decision problems. Another famous category is the set of function problems. A *function problem* is a computational problem where a single output is expected for every input, but the output is more complex than that of a decision problem. For function problems, the output is not simply "Yes" or "No". Formally we have:

Definition 5 (Function Problem Papadimitriou (1994))

Let us consider L a language and a polynomial-time decidable relation R_L such that for all strings x there is a string y with $\mathrm{R}_L(x, y)$ if and only if $x \in L$. The function problem P associated with L is the following computational problem: Given x, find a string y such that $\mathrm{R}_L(x, y)$ if such string exists, if no such string exists, return "No".

Example 3 (The Prime Decomposition Problem)

One famous example of a function problem is the Prime decomposition problem. Given an integer N, find an integer d with $1 < d < N$ that divides N (or returns $d = N$ to conclude that N is prime). As we can see, the output expected is not just "Yes" or "No", but it is the integer d.

1.2.3 Optimisation Problem

Another set of computational problems that will be addressed in this thesis is the set of optimisation problems. An optimization problem is the problem of finding the *best* solution among all solutions. It can be viewed as an extension of the function problem, when not any y are expected solutions, but only those maximizing (or minimizing) a particular objective function f.

One famous example to illustrate how to define an optimisation problem can be the Knapsack problem, defined precisely by Tobias Dantzig (1930), it refers to the commonplace problem of packing the most valuable or useful items without overloading the luggage.

Example 4 (The Knapsack Problem)

Given a set of items $X = \{x_1, x_2, \ldots, x_n\}$, each with a weight w_i and a value v_i, determine the number of each item to include in a collection so that the total weight is less than or equal to a given limit W and the total value is as large as possible. It is defined formally as follows:

$$\text{maximize} \qquad \sum_{i=1}^{n} v_i \times x_i$$

$$\text{subject to} \qquad \sum_{i=1}^{n} (w_i \times y_i) \leq W \text{ and } y_i \in \{0, 1\}.$$

We could define many more sets of computational problems but what have been presented is sufficient for this thesis. Furthermore, except in some cases, we will talk exclusively about decisions problems. The reason for that is that function problems and optimisation problems can be casted into a decision problem. For Optimisation Problem, instead of minimizing a value X, we just have to ask the question "Is N the optimal value?". If the answer is "Yes", so the problem is decided, if it is not the case, we just need to try another value N' and try again. To illustrate how to cast a function problem into a decision problem, let us go back to Example 3, we can define it as a decision problem as follows:

Example 5 (The Prime Decomposition Decision Problem)

Given an integer N and an integer M with $1 < M < N$, does N have a factor d with $1 < d \leq M$? In that case, the expected output is no longer the factor d, but it is just "Yes" or "No" as in any Decision Problem.
To ask the question if N is not prime, the problem needs to be instantiated with $\sqrt{N} \leq M < N$.

1.2.4 Complement of a Decision Problem

One final definition of kind of problem is the complement of a decision problem. The complement of a decision problem is the decision problem resulting from reversing the yes and no answers. It is defined formally as follows:

Definition 6 (Complement of a Decision Problem)

For a decision problem Σ, its complement is the decision problem $co\Sigma$ such that:

- $Y_{co\Sigma} = N_{\Sigma}$;
- $N_{co\Sigma} = Y_{\Sigma}$;

The complement of a decision problem is the decision problem resulting from reversing the "Yes" and "No" answers. We saw different set of problems and that there exists different kinds of algorithms that can require polynomial-time or exponential-time. One question that can arise is: "do some problems need an exponential time algorithm??" or more generally, do some problems require a certain amount of resources (time, memory, ...). To answer this question, complexity theorists created classes of problems that require a similar amount of a given resource.

1.3 Complexity classes

In this section, we assume that the reader is familiar with the notion of Turing machine (TM) Turing (1938), which can be deterministic, non-deterministic or with an oracle. To formalize the question of knowing if there exists at least one problem, which requires in the best case, an exponential-time algorithm, complexity theorists created sets of problems of related resource-based complexity. This kind of set of problems is called a complexity class and is defined as follows:

Definition 7 (Complexity class)

Let f be a function such that $f : \mathbb{N} \to \mathbb{N}$. A set of problems creates a complexity class C if they can all be solved by an abstract machine (such as a Turing machine) using $\mathcal{O}(f(n))$ of resource R, where n is the size of the input.

From this point, we can define four complexity classes: TIME, NTIME, SPACE and NSPACE, where the resource R will be the time or the space and the abstract machine will be a deterministic Turing-machine or a non-deterministic Turing machine.

- TIME($f(n)$) is the complexity class of all the problems which can be solved with a deterministic Turing machine in time bounded by $\mathcal{O}(f(n))$.
- NTIME($f(n)$) is the complexity class of all the problems which can be solved with a non-deterministic Turing machine in time bounded by $\mathcal{O}(f(n))$.
- SPACE($f(n)$) is the complexity class of all the problems which can be solved with a deterministic Turing machine in space bounded by $\mathcal{O}(f(n))$.
- NSPACE($f(n)$) is the complexity class of all the problems which can be solved with a non-deterministic Turing machine in space bounded by $\mathcal{O}(f(n))$.

We can now describe the important complexity classes which are P, NP, EXPTIME, NEXP, PSPACE and EXPSPACE that will be used in this thesis.

Model of computation	**Time constraint** $f(n)$	**Time constraint** $poly(n)$	**Time constraint** $2^{poly(n)}$
Deterministic TM	TIME(f(n))	P	EXPTIME
Non-Deterministic TM	NTIME(f(n))	NP	NEXP

Table 1.2: Time-complexity classes of problems

Model of computation	**Space constraint** $f(n)$	**Space constraint** $poly(n)$	**Space constraint** $2^{poly(n)}$
Deterministic TM	SPACE(f(n))	PSPACE	EXPSPACE
Non-Deterministic TM	NSPACE(f(n))	PSPACE	EXPSPACE

Table 1.3: Space-complexity classes of problems

For sake of simplicity, when we will talk about a complexity class, we will deal only with the classes of *decision problems*, even if there exists complexity classes for counting problem (*eg.* #P), function problems (*eg.* FP), *etc.* (see Papadimitriou (1994) for more details).

1.3.1 Complement of a Complexity Class

Now that we know what is a complexity class and the complement of a decision problem, it is now straightforward to define the complement of a complexity class. We will say that the class coNP is the complexity class which contains all the complements of the problems inside NP. Same for all the other complexity classes. More formally we have:

Definition 8 (Complement of a Complexity Class)

A set of complement of problems creates a complement of a complexity class if they can all be solved by an abstract machine (such as a Turing machine) using $\mathcal{O}(f(n))$ of resource R, where n is the size of the input.
For a complexity class named P$*$, its complement will be named coP$*$.

Example 6

Let us illustrate some examples of problems which belong to NP and their complements which belong to coNP.

Problems in NP	**Problems in coNP**
Let ϕ be a Boolean formula, is it satisfiable?	Let ϕ be a Boolean formula, is it unsatisfiable?
Let G be a graph, is there an Hamiltonian path?	Let G be a graph, is it true that there is no Hamiltonian path?
Let G be a graph and k an integer, is there a clique of size k in G?	Let G be a graph and k an integer, is it true that G does not have a clique of size k?

If one sees NP as the class where we can verify a model in polynomial time, coNP should be seen as the class where we can verify a counter-example in polynomial time.

1.3.2 Relations Among Complexity Classes

These classes looks all separated but they have relations among each other. We can also define $\mathsf{NP} = \bigcup_{k\in\mathbb{N}} \mathsf{NTIME}(n^k)$ and $\mathsf{PSPACE} = \bigcup_{k\in\mathbb{N}} \mathsf{SPACE}(n^k)$. To represent more easily these complexity classes and how they are linked with each other, let us illustrate their relations with each other in Figure 1.2 and let us give a non-exhaustive list of the known results about these complexity classes and their relations among each other.

- $\mathsf{P} \subseteq \mathsf{NP} \subseteq \mathsf{PSPACE}$;
- $\mathsf{PSPACE} \subseteq \mathsf{EXPTIME}$;
- $\mathsf{EXPTIME} \subseteq \mathsf{EXPSPACE}$;
- $\mathsf{PSPACE} \subsetneq \mathsf{EXPSPACE}$;
- $\mathsf{P} \subsetneq \mathsf{EXPTIME}$;
- $\mathsf{P} =? \mathsf{NP}$.

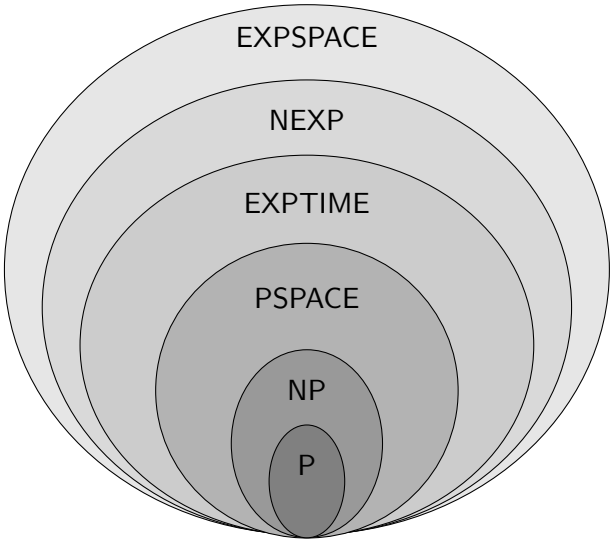


Figure 1.2: A representation of the relations among complexity classes

In the next part, we present the notion of completeness. This notion allows us to define the notion of being the hardest problems in a complexity class.

1.3.3 Polynomial Reductions and Completeness

The notion of polynomial reduction allows us to cast a decision problem into another one that is at least as hard as the first. Let Σ and Σ' be two problems, and P an algorithm that takes as input instances of Σ and returns as output an instance of Σ'. The notion of polynomial reduction can be defined as follows:

Definition 9 (Polynomial Reduction)

P is called a polynomial reduction if and only if:

- P is a polynomial-time algorithm;
- α is a positive instance of Σ if and only if $\mathrm{P}(\alpha)$ is a positive instance of Σ'.

But still, it is not because a problem is in C$*$ that it is C$*$-Hard. All the problems in NP are also in PSPACE, but this does not mean that they are PSPACE-Hard. In order to really separate these problems according to their inherent complexity, complexity theorists created the notion of hardness and completeness. In a nutshell, we can establish that a problem is "as hard as" another one if it can be polynomially reduced to this other one. This notion of hardness is defined as follows:

Definition 10 (C∗-Hard Problem)

A problem Σ is C∗-Hard when for every problem Σ' in $C*$, there exists a polynomial reduction P from Σ' to Σ.

That is, assuming a solution for Σ takes 1 unit time, we can use Σ's solution to solve Σ' in polynomial time. As a consequence, finding a polynomial algorithm to solve any $C*$-hard problem would give polynomial algorithms for all the problems in $C*$.

Remark 1.1. *Because we know from the Figure 1.2, the relation between the complexity classes, we can make the following remarks:*

- *Any problem* PSPACE*-Hard is* NP*-Hard;*
- *Any problem* NP*-Hard is* P*-Hard.*

From this point, we can now define the notion of completeness. If all the problems of one complexity class C∗ can be polynomially reduced to one problem Σ of that complexity class, we will say that Σ is C∗-complete.

Definition 11 (Completeness)

A problem Σ is said to be C∗-complete if and only if Σ is C∗-hard and Σ is in C∗.

Now that we have defined the notion of complexity class, of its complement and its completeness, we can now refine the Figure 1.2. We will just illustrate inside PSPACE to simplify the Figure.

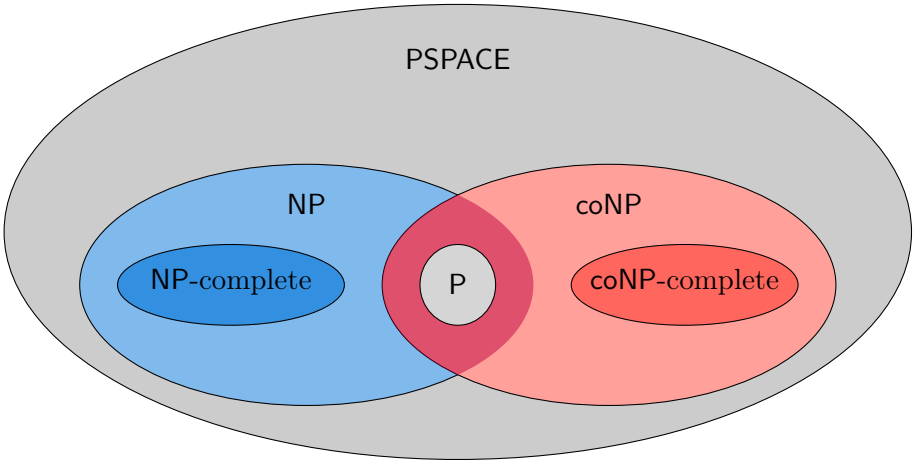


Figure 1.3: Illustration of P, NP, coNP, NP-complete, coNP-complete and PSPACE

The Figure 1.3 is drawn under the assumptions still unproved, presented by Papadimitriou (1994) and Arora and Barak (2009). The assumptions are as follows:

- $\mathsf{P} \neq \mathsf{NP}$;
- $\mathsf{coNP} \neq \mathsf{NP}$;
- $\mathsf{NP} \cap \mathsf{coNP} \neq \mathsf{P}$;
- $\mathsf{NP} \neq \mathsf{PSPACE}$;

We will reason under these assumptions through the rest of the thesis. Complexity theorists introduced a new notion of polynomial hierarchy to go gradually from NP to PSPACE.

1.3.4 Polynomial Hierarchy

We suppose the reader familiar with Turing Machine with Oracle.

Definition 12 (Polynomial hierarchy Stockmeyer (1976))

For the oracle definition of the polynomial hierarchy, we define $\Delta_0^{\mathsf{P}} = \Sigma_0^{\mathsf{P}} = \Pi_0^{\mathsf{P}} = \mathsf{P}$ where P is the set of decision problems solvable in polynomial time. We note by $\mathsf{P}^{\Sigma_i^{\mathsf{P}}}$ a polynomial Turing machine with an Σ_i^{P} oracle. Then for $i \geq 0$ define:

- $\Delta_{i+1}^{\mathsf{P}} = \mathsf{P}^{\Sigma_i^{\mathsf{P}}}$
- $\Sigma_{i+1}^{\mathsf{P}} = \mathsf{NP}^{\Sigma_i^{\mathsf{P}}}$
- $\Pi_{i+1}^{\mathsf{P}} = \mathsf{coNP}^{\Sigma_i^{\mathsf{P}}}$

Such definition of the polynomial hierarchy implies the following relations:

- $\Sigma_i^{\mathsf{P}} \subseteq \Delta_{i+1}^{\mathsf{P}} \subseteq \Sigma_{i+1}^{\mathsf{P}}$
- $\Pi_i^{\mathsf{P}} \subseteq \Delta_{i+1}^{\mathsf{P}} \subseteq \Pi_{i+1}^{\mathsf{P}}$
- $\Sigma_i^{\mathsf{P}} = co\Pi_i^{\mathsf{P}}$

It is an open question whether any of these inclusions are proper, though it is widely believed that they all are and we will assume that they are in this thesis. The union of all classes in the polynomial hierarchy is the complexity class PH. We can define $\mathsf{PH} = \bigcup_{i \geq 0} \Sigma_i^{\mathsf{P}}$. In the Figure 1.4, we can have a graphical representation of PH. The arrows indicate the inclusion of one complexity class into an other.

We can thus refine our known inclusion, we have: $\mathsf{P} \subseteq \mathsf{NP} \subseteq \mathsf{PH} \subseteq \mathsf{PSPACE}$. One famous open question is whether the polynomial hierarchy collapses.

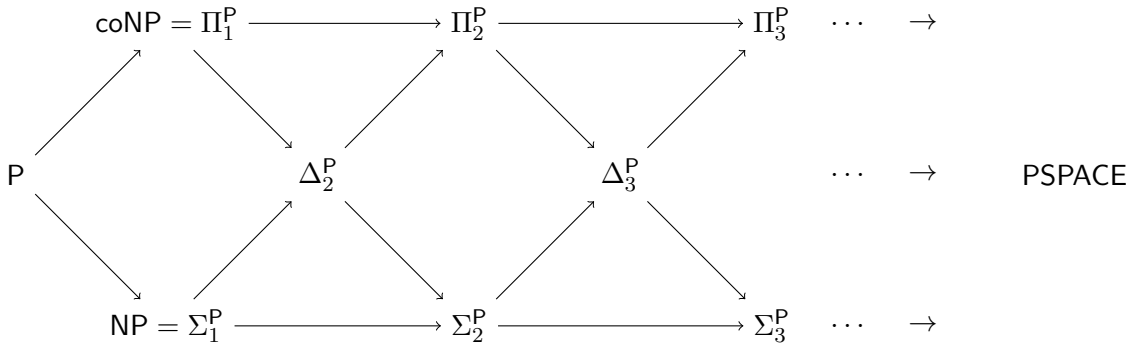


Figure 1.4: Syntactic inclusions in polynomial hierarchy.

Definition 13 (The Collapse of The Polynomial Hierarchy)

We will say that the PH collapses at the level i^{th} if for some $i \geq 1$, $\Sigma_i^P = \Pi_i^P$. This would imply that $\forall j > i\ \Sigma_j^P = \Pi_j^P = \Delta_j^P = \Sigma_i^P$.

The equality between PH and PSPACE is not known and it is still an open question. But this question could be answered by the following property of PH. PH is assumed to have no complete problems, because we assume in this thesis that PH does not collapse.

Theorem 1 (Stockmeyer (1976))

If PH has at least one PH-complete problem then the polynomial hierarchy collapses.

Sketch of Proof. If there exists a problem ϕ that is PH-complete, then ϕ belongs to some Σ_i^P. By the definition of completeness, all other problems in PH can be reduced to ϕ. This means that there cannot be any problems in PH that is harder than ϕ. q.e.d □

Corollary 1

If PH = PSPACE then the polynomial hierarchy collapses.

The intuition behind Corollary 1 is that, because there exists PSPACE-complete problems (*eg.* Modal Logic K Satisfiability Problem Ladner (1977), defined in the next chapter), if PH = PSPACE, then by Theorem 1, the polynomial hierarchy collapses.

1.4 Conclusion

We presented in this chapter many complexity classes and how they are related to each other. In this thesis we will focus mainly on NP, PH and PSPACE but it was important to explain the others. We made a lot of theoretical assumptions that we recall here.

Assumptions: From now on, we assume the following to be true:

1. $\mathsf{P} \neq \mathsf{NP}$;
2. $\mathsf{coNP} \neq \mathsf{NP}$;
3. $\mathsf{NP} \cap \mathsf{coNP} \neq \mathsf{P}$;
4. $\mathsf{NP} \neq \mathsf{PSPACE}$;
5. PH does not collapse;
6. $\mathsf{PH} \neq \mathsf{PSPACE}$;

From this point, we know complexity classes which are sets of problems requiring the same amount of resources but we did not present any actual problem that belongs to these classes and this is what we will do in the next chapter. We will present different logics, different normal forms to represent a formula and we will explain to which complexity class belongs the satisfiability problem in these logics.

Chapter 2

Logics

Contents

> "Logic is invincible, because in order to combat logic it is necessary to use logic."
>
> *Pierre Boutroux*

Reasoning using only logic was one of the dreams of the Greeks philosophers such as Aristotle or Chrysippe de Sole for example. In a nutshell, the Ancient Greeks thought that we could represent every knowledge and every theorem using only logical propositions linked with logical operators. The idea behind it is to put the good axiomiatisation to represent any facts and it is the root of the field of *Knowledge Representation*. According to the axiomiatisation and the logical operators that we allow, we will have a more or less expressive logic which will be more or less difficult to reason about. In this chapter, we will present two different logical-based frameworks.

- Propositional Logic, allowing us to reason about propositions, which was proposed by Boole (1854);
- Modal Logics, allowing to reason about modalities such as necessity, possibility, knowledge, belief and it was proposed by Lewis and Langford (1932).

We will see that Propositional Logic can be seen as a special case of Modal Logics (the case where there is no modality) and that Modal Logics are a decidable fragment of the First Order Logic (Blackburn et al. 2006, Theo.18 p46).

2.1 Propositional Logic

Classical Propositional Logic (also called Boolean Logic Boole (1854)) is called in this thesis CPL. It is the logic used to reason about propositions. It is based on the notion of propositional variables enunciated by Chrysippe de Sole (Stoïcian philosopher) in the III^{rd} century BCE. Somehow it can be seen as the Zero Order Logic, where there is no quantifier nor modality and all the predicates have arity 0. But first, let us define formally its syntax and its semantics.

2.1.1 Syntax

The syntax of the propositional logic can be formally defined as follows:

Definition 14 (Language of Classical Propositional Logic)

Let $\mathbb{P}$ be an infinite countable set of propositional variables. The language of classical propositional logic is the set of formulas containing $\mathbb{P}$, closed under the set of propositional connectives $\{\neg, \wedge\}$. It can also be defined by the following grammar in BNF, where p ranges over $\mathbb{P}$:

$\langle Formula \rangle ::= \langle Variable \rangle$
$\quad | \quad \langle Constant \rangle$
$\quad | \quad (\langle Formula \rangle \wedge \langle Formula \rangle)$
$\quad | \quad \neg \langle Formula \rangle$

$\langle Variable \rangle ::= p$

$\langle Constant \rangle ::= \top \mid \bot$

We denote by $\top$ the propositional constant TRUE (the tautology) and $\bot$ the propositional constant FALSE (the contradiction). $\wedge$ is an operator with arity 2 and $\neg$ is an operator with arity 1. We define some operators as follows:

- $\phi \vee \psi \stackrel{\text{def}}{=} \neg(\neg\phi \wedge \neg\psi)$
- $\phi \rightarrow \psi \stackrel{\text{def}}{=} \neg\phi \vee \psi$
- $\phi \leftrightarrow \psi \stackrel{\text{def}}{=} (\phi \rightarrow \psi) \wedge (\psi \rightarrow \phi)$

The arity of the operators is important to denote if a formula is **well formed** or not. An illustration of this definition is given in the following example:

Example 7

Let $\mathbb{P} = \{a, b, c, d\}$ be a set of propositional variables. The formula $\phi_1 = (a \vee b) \wedge (c \neg d)$ does not belong to CPL because no rule allows to build $(c \neg d)$ whereas $\phi_2 = (a \vee b) \wedge (c \vee \neg d)$ does. We will say in that case that ϕ_1 is **not well formed** whereas ϕ_2 is **well formed**.

Now that we have defined the language (the syntactical aspects) of CPL, it is interesting to have a look at its semantics.

2.1.2 Semantics

Regarding the semantical aspects of the classical propositional logic, the notion of *interpretation* is important. Each variable can be assigned either to TRUE or FALSE. It is defined as follows:

Definition 15 (Interpretation)

Let $\mathbb{P}$ be an infinite countable set of propositional variables An interpretation is an assignment of all propositional variables to TRUE or FALSE.
Said differently, it is a function $\mathbb{P} \to \{\text{TRUE}, \text{FALSE}\}$.

We say that if a formula ϕ is evaluated to true for a given interpretation (also said satisfied by that interpretation), then this interpretation is a **model** of ϕ.

Definition 16 (Satisfaction Relation in CPL)

The relation $\models$ between interpretation $\mathcal{M}$ and formulas ϕ in CPL is recursively defined as follows:

$\mathcal{M} \models p$	iff	$\mathcal{M}(p) = \text{TRUE}$
$\mathcal{M} \models \neg\phi$	iff	$\mathcal{M} \not\models \phi$
$\mathcal{M} \models \phi_1 \wedge \phi_2$	iff	$\mathcal{M} \models \phi_1$ and $\mathcal{M} \models \phi_2$
$\mathcal{M} \models \phi_1 \vee \phi_2$	iff	$\mathcal{M} \models \phi_1$ or $\mathcal{M} \models \phi_2$

Here is now a terminology list that we use in this thesis. The satisfaction relation depends on the logic considered, but, without loss of generality, the notations used are always the same.

- If a formula ϕ has **at least** one model, we will say that this formula is **satisfiable**. We say that the model $\mathcal{M}$ satisfies the formula ϕ and we denote it by $\mathcal{M} \models \phi$.

- If a formula is satisfied by all interpretations, we will say that this formula is **valid**. We denote it $\models \phi$.
- If a formula is not satisfied by any interpretation, we will say that this formula is **unsatisfiable**.
- Let F be a formula and T a theory, F is a theorem of T if and only if for all interpretation $\mathcal{I}$, if $\mathcal{I} \models T$ then $\mathcal{I} \models F$.

We will then denote by $\vdash \phi$ the syntactical entailment (where $\phi \vdash \psi$ means that it exists a formal proof of ψ from the formula ϕ) and by $\models \phi$ the semantical entailment. For the language of propositional logic to be considered as a *logic*, it is still missing some mechanical aspects to be able to reason about formulas. These mechanical aspects are called inference rules, and one famous rule in propositional logic is the **Modus Ponens**, which is a short form for *modus ponendo ponens* which is the Latin for "the way that affirms by affirming".

Definition 17 (Modus Ponens (MP))

The modus ponens can be summarized as "(1) ϕ implies ψ and (2) ϕ is asserted to be true, and therefore ψ must be true". It is noted: $\dfrac{\phi \rightarrow \psi \quad \phi}{\psi}$ or using the sequent notation $\phi \rightarrow \psi, \phi \vdash \psi$.

We just saw the different syntactical and semantical aspects of classical propositional logic. We can in fact list the good properties that a logic system should have, as follows:

Definition 18 (Desirable Properties of a Logic)

- **Soundness**: All theorems of the logic are valid (if $\vdash \phi$ then $\models \phi$);
- **Completeness**: Any valid formula is a theorem (if $\models \phi$ then $\vdash \phi$);
- **Decidability**: For any formula, there is a program that decides if it is a theorem or not.

Theorem 2 (Bernays (1926))

Propositional Logic is sound, complete and decidable.

The propositional logic has these desirable properties as demonstrated by Paul Bernays in 1926. The complexity of the satisfiability problem has been shown to be NP-complete by Cook (1971). This implies that the validity problem is coNP-complete.

2.1.3 Normal Forms

We saw the language of classical propositional logic and we saw that the satisfiability problem in propositional logic is decidable. But to discuss how we can decide efficiently the satisfiability of a formula in propositional logic, we need to define some normal forms that have been created to represent formulas. In a nutshell, it is a way to represent formulas without changing the satisfiability.

Conjunctive Normal Form (CNF)

One famous normal form nowadays used to represent a propositional logic formula and to decide its satisfiability is the Conjunctive Normal Form and it is defined as follows:

Definition 19 (Conjunctive Normal Form (CNF))

In CPL, a formula ϕ is in conjunctive normal form (or clausal normal form) if:

- It is a conjunction of one or more clauses;
- Each clause is a disjunction of literals;
- A literal is a propositional variable or its negation.

Example 8

All of the following formulas on the variables p_1, p_2, p_3, p_4, and p_5 are in CNF:

- $\neg p_1 \wedge (p_2 \vee p_3)$
- $(p_1 \vee p_2) \wedge (\neg p_2 \vee p_3 \vee \neg p_4) \wedge (p_4 \vee \neg p_5)$
- $(p_1 \vee p_2) \wedge p_3$
- $(p_1 \wedge p_2)$

The following formulas are not in CNF:

1. $\neg(p_2 \vee p_3)$
2. $(p_1 \wedge p_2) \vee p_3$
3. $p_1 \wedge (p_2 \vee (p_4 \wedge p_5))$

Every propositional formula can be translated into a logically equivalent formula that is in CNF. This transformation is based on rules about logical equivalences: the double negative law, De Morgan's laws, absorption laws and the distributive law which are as follows:

- $\neg(\neg\phi) ::= \phi$;
- $\neg(\phi \vee \psi) ::= (\neg\phi) \wedge (\neg\psi)$;
- $\neg(\phi \wedge \psi) ::= (\neg\phi) \vee (\neg\psi)$;
- $\phi \vee (\phi \wedge \psi) ::= \phi$;
- $\phi \wedge (\phi \vee \psi) ::= \phi$;
- $\phi \vee (\psi \wedge \chi) ::= (\phi \vee \psi) \wedge (\phi \vee \chi)$;
- $\phi \wedge (\psi \vee \chi) ::= (\phi \wedge \psi) \vee (\phi \wedge \chi)$;

Example 9

Let us take back the three non-examples just mentioned, they are respectively equivalent to the following formulas, in CNF:

1. $\neg p_2 \wedge \neg p_3$
2. $(p_1 \vee p_3) \wedge (p_2 \vee p_3)$
3. $p_1 \wedge (p_2 \vee p_4) \wedge (p_2 \vee p_5)$

One disadvantage of the conversion to CNF is the size of the generated output. In some cases this conversion to CNF can lead to an exponential explosion of the formula. An illustration of such explosion is given in the following Example:

Example 10

Translating the following non-CNF formula ϕ into CNF produces a formula with 2^n clauses:

$$\phi = (X_1 \wedge Y_1) \vee (X_2 \wedge Y_2) \vee \cdots \vee (X_n \wedge Y_n)$$

In particular, the generated formula is:

$$\begin{array}{c}(X_1 \vee X_2 \vee \cdots \vee X_n) \wedge \\ (Y_1 \vee X_2 \vee \cdots \vee X_n) \wedge \\ (X_1 \vee Y_2 \vee \cdots \vee X_n) \wedge \\ (Y_1 \vee Y_2 \vee \cdots \vee X_n) \wedge \\ \cdots \\ (Y_1 \vee Y_2 \vee \cdots \vee Y_n)\end{array}$$

This formula contains 2^n clauses; each clause contains either X_i or Y_i for each i.

There exist transformations into CNF that avoid an exponential increase in size by preserving satisfiability rather than equivalence. One famous translation to do that is the Tseitin Translation Tseitin (1983). These transformations are guaranteed to only linearly increase the size of the formula, but introduce new variables. This means that the original formula and the result of the translation are equisatisfiable but not equivalent. For some specific cases, Plaisted and Greenbaum (1986) is even better because it does not encode equivalences between new variables and sub-formulas but only implications, thus the CNF generated is smaller.

If we go back to ϕ from the Example 10, but this time we use the Tseitin translation, here is what we obtain:

Example 11

Translating the following non-CNF formula ϕ into CNF with the Tseitin translation gives us the following:

$$\phi = (X_1 \wedge Y_1) \vee (X_2 \wedge Y_2) \vee \cdots \vee (X_n \wedge Y_n)$$

The generated formula is:

$$\bigwedge_{i=0}^{n} (Z_i \leftrightarrow (X_i \wedge Y_i)) \wedge \bigwedge_{i=0}^{n} (Z_i)$$

This formula contains n additional variables, but only $4n$ clauses.

Negation Normal Form (NNF)

Another famous normal form is the Negation Normal Form (NNF), which is defined as follows:

Definition 20 (Negation Normal Form (NNF))

We will say that a formula ϕ is in negation normal form for CPL if the negation operator ($\neg$) is only applied to variables and the only other allowed operators are conjunctions ($\wedge$) and disjunctions ($\vee$).

Every formula can be converted into NNF using the following rewrite rules (Robinson and Voronkov 2001, page 204):

- $(\phi \rightarrow \psi) ::= (\neg\phi \vee \psi)$
- $\neg(\phi \vee \psi) ::= (\neg\phi \wedge \neg\psi)$
- $\neg(\phi \wedge \psi) ::= (\neg\phi \vee \neg\psi)$

- $\neg\neg\phi ::= \phi$

One could see the Conjunction Normal Form as a specific case of Negation Normal Form. We will see that these two normal forms can be kept even in other logics than the Propositional Logic. Indeed, we will consider other logics than Propositional Logic for the following limitations of the propositional reasoning:

- Propositional Variables have a "static aspect" which is sometimes not convenient to reason about some objects/properties;
- It is impossible to deal easily with common features from different objects;
- It is impossible to manipulate abstract propositions, without describing them explicitly in the formula.

Obviously, there exists many Normal Forms and we do not plan to discuss all of them, just the ones important in this thesis. We redirect the reader to Robinson and Voronkov (2001), which discuss the different Normal Forms for many logics.

So now that we saw that the propositional logic is decidable but not expressive, it is worth wondering if it exists a logic which is decidable and more expressive than propositional logic. Such a logic exists, in fact there exist many of them according to the axiomiatisation we give, they are called Modal Logics and we will describe them formally in the next section.

2.2 Modal Logics

The term modal logics comes from the fact that these logics are using modality to express a sentence. A modality expresses the semantics of a verb, an adjective, an adverb, ..., on a formula.

Example 12

Here is a non-exhaustive list of some modalities that we can use in modal logics:

- I know that
- I believe that
- It is necessary that
- It is possible that
- It is mandatory that
- It is permissible that

Thanks to these modalities, we can express things in a more concise way than in propositional logic. We can express that something is true and I believe it to be false, without being contradictory.

2.2.1 Syntax

One can consider modal logics in two ways.

- It can be seen as the logic obtained from propositional logic by adding a modal connective $\Box$, *ie.*, if ϕ is a formula, then $\Box\phi$ is also a formula. Intuitively, $\Box\phi$ asserts that ϕ is necessarily true. Dually, $\neg\Box\neg\phi$, abbreviated as $\Diamond\phi$, asserts that ϕ is possibly true.

- It can also be seen as a decidable fragment of First Order Logic, where the modalities replaces the quantifier, where the predicates have a bounded arity and where the models have the finite-model property.

We focus on the definition where modal logics is an extended version of CPL. We will use the Kripke semantics Kripke (1959). For more details on modal logic, especially on the notion of possible worlds, the reader may consult for example Chellas (1980).

Definition 21 (Language of Modal Logic)

Let $\mathbb{P}$ be an infinite countable set of propositional variables and $\Box$ a unary modal operator. The language of modal logic (noted $\mathcal{L}$) is the set of formulas containing $\mathbb{P}$, closed under the set of propositional connectives $\{\neg, \wedge\}$ and $\Box$. The language can be defined by the following grammar in BNF, where p ranges over $\mathbb{P}$:

$$
\begin{aligned}
&\langle \mathit{Formula} \rangle ::= \langle \mathit{Variable} \rangle \\
&\quad | \quad \langle \mathit{Constant} \rangle \\
&\quad | \quad (\langle \mathit{Formula} \rangle \wedge \langle \mathit{Formula} \rangle) \\
&\quad | \quad \neg \langle \mathit{Formula} \rangle \\
&\quad | \quad \Box \langle \mathit{Formula} \rangle
\end{aligned}
$$

$$\langle \mathit{Variable} \rangle ::= p$$

$$\langle \mathit{Constant} \rangle ::= \top \mid \bot$$

We denote by $\top$ the propositional constant TRUE and $\bot$ the propositional constant FALSE. We construct the usual operator in the same way as in CPL plus the modality operator as follows:

- $\Diamond\phi \stackrel{\text{def}}{=} \neg\Box\neg\psi$

The particularity of $\mathcal{L}$ is that there exists many different modal logics according to which axioms are considered. In this section, we will only talk about propositional modal logics respecting axiom (K), but there exists many more modal logics and we invite the reader to consult Blackburn et al. (2006) for more information on this topic.

2.2.2 Axiomatic Theory

Indeed, the axiomiatisation is an important part of the modal logic theory. In this thesis, we will talk about the Normal Modal Logics. A normal modal logic is a logic which extends propositional logic, with a modality $\Box$ and which has a least the following rules:

$$\frac{\phi \to \psi \quad \phi}{\psi}$$ The Modus Ponens (MP)

$$\frac{\phi}{\Box\phi}$$ The Necessitation Rule (N)

$\Box(\phi \to \psi) \to (\Box\phi \to \Box\psi)$ The Kripke Axiom (K)

The Modal Logic K was one of the first modal logic studied, it has the property to be decidable and the complexity of its satisfiability problem is PSPACE-complete (see Ladner (1977), Halpern and Moses (1992) for more details).

There exists many axioms for modal logic, but we will consider here the most common ones which are:

Definition 22 (Most Common Axioms)

$\phi \to \Box\phi$	(Triv)
$\Box(\phi \to \psi) \to (\Box\phi \to \Box\psi)$	Axiom (K)
$\Box\phi \to \Diamond\phi$	Axiom (D)
$\Box\phi \to \phi$	Axiom (T)
$\phi \to \Box\Diamond\phi$	Axiom (B)
$\Box\phi \to \Box\Box\phi$	Axiom (4)
$\Diamond\phi \to \Box\Diamond\phi$	Axiom (5)

The modal logics are named after the axioms they respect. Thus:

- Modal Logic K is the modal logic containing axiom (K) plus the necessitation rule ($\Box\top$).
- Modal Logic KT is the modal logic K containing axiom (T);
- Modal Logic KT5 is the modal logic KT containing axiom (5);
- ....

But even if we can name the logics according to their axioms, some have "historical name" because they have been considered before Kripke (1959). For example:

- S5 = modal logic KT5;
- weak-S5 = modal logic KD45;
- S4 = modal logic KT4;
- B = modal logic KTB;

Obviously, if it exists S5 and S4, there exists also other modal logics S_i, but we will not talk about them in this thesis and we refer the reader to Lewis and Langford (1932) for more details. The axioms have also different property between them, like for example ((T) $\rightarrow$ (D)) or ((T) $\wedge$ (5) $\rightarrow$ (B)).

In fact, we can graphically represent the link between the different logics, with the Modal Logic Cube displayed in Figure 2.1. Interestingly enough, this cube has been formally verified by a High-Order Automated Reasoner. The results of this verification may be found in Benzmüller (2010).

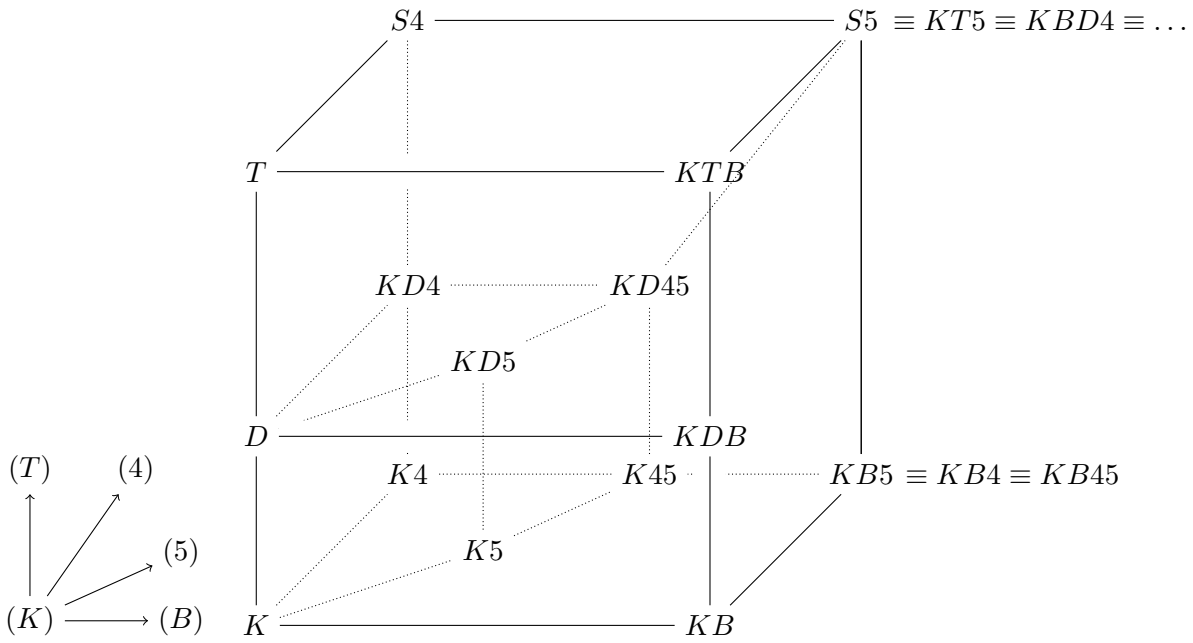


Figure 2.1: The Modal Logic Cube

It is also interesting to notice that on the Figure 2.1, there are only 15 different modal logics whereas if we have 5 axioms (Def. 22), we should have 32 (2^5) different logics. This comes from the fact that some logic may have different axiomatic-name, such as S5 which is KT5 = KBD4 = KBD5 = KBT4 = KBT5 = KDT5 = KT45 = KBD45 = KBT45 = KDT45 = KBDT4 = KBDT5 = KBDT45. Now that we defined modal logics with their syntax and their axiomatic theories, it is time to see what is their semantics.

2.2.3 Semantics

The semantics used here to interpret normal modal logics is the Kripke Semantic Kripke (1959).

Example 13

Let us consider the following example where $\mathbb{P} = \{a, b\}$ and the formula ϕ is $\Diamond(a \wedge \Box b)$. This example will be used through this part of the preliminaries.

Definition 23 (Kripke Structure)

Let $\mathbb{P}$ be a finite non-empty set of propositional variables. A Kripke Structure is a triplet $\mathcal{K} = \langle W, R, V \rangle$, where:

- W is a non-empty set of possible worlds,
- $R \subseteq W \times W$ is a binary accessibility relation
- $V : \mathbb{P} \to 2^W$ is a valuation function which associates, to each $p \in \mathbb{P}$, the set of possible worlds from W where p is true.

A Pointed Kripke Structure is a pair $\langle \mathcal{K}, w \rangle$, where $\mathcal{K}$ is a Kripke Structure and w is a possible world in W. Thereafter, whenever we use the term 'Kripke Structure' we refer to 'Pointed Kripke Structure'.

Definition 24 (Satisfaction Relation)

The relation $\models$ between Kripke Models and formulae in $\mathcal{L}$ is recursively defined as follows:

$\langle \mathcal{K}, w \rangle \models p$	iff	$w \in V(p)$
$\langle \mathcal{K}, w \rangle \models \neg\phi$	iff	$\langle \mathcal{K}, w \rangle \not\models \phi$
$\langle \mathcal{K}, w \rangle \models \phi_1 \wedge \phi_2$	iff	$\langle \mathcal{K}, w \rangle \models \phi_1$ and $\langle \mathcal{K}, w \rangle \models \phi_2$
$\langle \mathcal{K}, w \rangle \models \Box\phi$	iff	$\forall w' s.t.\ (w, w') \in R$ implies $\langle \mathcal{K}, w' \rangle \models \phi$

Definition 25 (Validity)

As usual, a formula $\phi \in \mathcal{L}$ is valid (noted $\models \phi$) if and only if it is satisfied by all Kripke Models $\langle \mathcal{K}, w \rangle$. A formula $\phi \in \mathcal{L}$ is satisfiable if and only if $\not\models \neg\phi$.

Example 14

Here a Kripke model $\langle \mathcal{K}, w \rangle$ satisfying the formula ϕ from Example 22. For the sake of compactness, we represent R as sets.

- $W = \{w_0, w_1, w_2, w_3\}$,
- $R = \{\langle w_0, \{w_3\}\rangle, \langle w_1, \{w_1, w_2, w_3\}\rangle, \langle w_2, \{w_1, w_2, w_3\}\rangle, \langle w_3, \{w_1, w_2, w_3\}\rangle\}$,
- $V = \{\langle a, \{w_3\}\rangle, \langle b, \{w_1, w_2, w_3\}\rangle\}$.

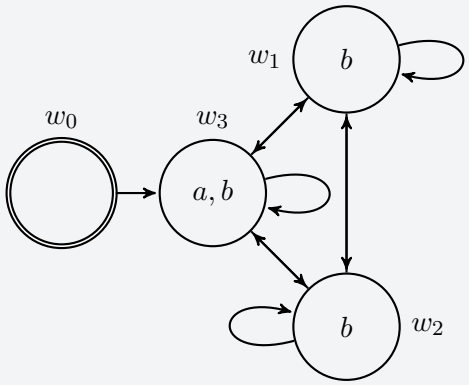


Figure 2.2: A textual and graphical representation of $M \models \Diamond(a \wedge \Box b)$

All Kripke Models satisfy (K) as explained in Chellas (1980), and this is why we will call them here K-models. A K-model may satisfy other schemas, a list of such schemas and the properties corresponding to them is given on Table 2.1. Theses schemas have the particularity that they add constraints on the Kripke models, as an example we can see that a KT-model will be a reflexive K-model.

Name	Condition on $\mathcal{K}$	First Order constraint	Schema
axiom (K)	None		$\Box(\phi \to \psi) \to (\Box\phi \to \Box\psi)$
axiom (T)	Reflexivity	$\forall w.R(w, w)$	$\Box\phi \to \phi$
axiom (B)	Symmetry	$\forall w_1.\forall w_2.(R(w_1, w_2) \to R(w_2, w_1))$	$\phi \to \Box\Diamond\phi$
axiom (D)	Seriality	$\forall w_1.\exists w_2.R(w_1, w_2)$	$\Box\phi \to \Diamond\phi$
axiom (4)	Transitivity	$\forall w_1.\forall w_2.\forall w_3.((R(w_1, w_2) \wedge R(w_2, w_3)) \to R(w_1, w_3))$	$\Box\phi \to \Box\Box\phi$
axiom (5)	Euclideanity	$\forall w_1.\forall w_2.\forall w_3.((R(w_1, w_2) \wedge R(w_1, w_3)) \to R(w_2, w_3))$	$\Diamond\phi \to \Box\Diamond\phi$

Table 2.1: axioms schemata and corresponding structural properties

These correspondences were proposed by Kripke (1959), and it is important to understand them correctly. Every reflexive K-model satisfies T, but there are non-reflexive K-models that satisfy T as well. However, every finite K-model satisfying T has an "equivalent" reflexive K-model. Analogously for the other properties.

Two K-models are equivalent if and only if they are bisimilar. The notion of bisimulation is intended to capture worlds equivalences and relations equivalences. It is formally defined as follows:

Definition 26 (Bisimulation Blackburn et al. (2006))

We denote by $M, s \leftrightarrow N, t$ that there exists, Z, a bisimulation that connects s and t. A bisimulation Z between models $M = \langle W, R, V \rangle$ and $N = \langle W', R', V' \rangle$ is a relation on $W \times W'$ such that if sZt then the following hold:

Invariance $V(s) = V'(t)$ (the two worlds have the same valuation),

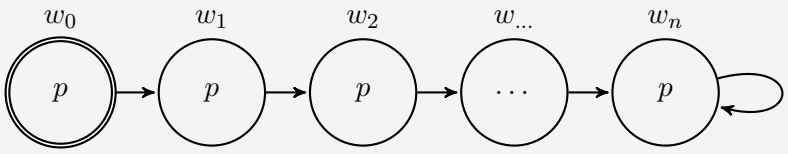
Zig if for some $s' \in W$, $(s, s') \in R$, then there is a $t' \in W'$ with $(t, t') \in R'$ and $s'Zt'$.

Zag same requirement in the other direction.

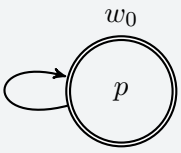
If one wants to find a finite KT-model, it is safe to search only among reflexive K-models. Let M and M' be two models, M' be in bisimulation with M, M' will be call a bisimular of M. Analogously for the other properties.

Example 15

Let us illustrate this bisimulation on an example. Let us consider the following formula $\phi = \Box p$ that we want to determine the satisfiability in modal logic S4. We know from axiom (4) that for any formula ϕ, we have $\Box\phi \to \Box\Box\phi$ and from axiom (T) that $\Box\phi \to \phi$. So one could consider a *line-shaped* Kripke model as follows, to satisfy the formula:



This model satisfies the formula ϕ while respecting axioms (T) and (4) in a way that it respects for any sub-formula the two axioms (T) and (4), but, it is not transitive nor reflexive. However, this model possesses necessary a bisimilar which is transitive and reflexive, thanks to the correspondences presented by Kripke (1959). Such a bisimilar could be for example:



Following Table 2.1, we will call KT-model a reflexive K-model; a KD-model is a serial K-model; a K5-model is an euclidean K-model. In fact, all the combinations of these properties give rise to 15 different kinds of models (presented in Table 2.2).

The satisfiability problems for those different logics have different complexities. But, if we restrict our attention to one kind of model, we can define one modal logic where validities are the formulas that are satisfied only by those models. Formally we have:

Structure Kind	Structural Properties	Structure Kind	Structural Properties
K	$\emptyset$	(S4) KT4 = KDT4	Reflexive and Transitive
KB	Symmetric	KD4	Serial and Transitive
KT = KDT	Reflexive	KD	Serial
K4	Transitive	KDB	Serial and Symmetric
KBT = KBDT	Symmetric and Reflexive	K45	Transitive and Euclidean
K5	Euclidean	KD5	Serial and Euclidean
KB4 = KB5 = KB45	Symmetric and Transitive	KD45	Serial, Transitive and Euclidean
(S5) KT5 = KBD4 = KBD5 = KBT4 = KBT5 = KDT5 = KT45 = KBD45 = KBT45 = KDT45 = KBDT4 = KBDT5 = KBDT45			Equivalence

Table 2.2: The Different Kinds of Kripke Models

Definition 27 ($\star$-Validity)

Let $\star$ range over the model kinds in one of the 15 different kinds in Table 2.2. A formula $\phi \in \mathcal{L}$ is $\star$-valid (noted $\models_\star \phi$) if and only if it is satisfied by all $\star$-models $\langle \mathcal{K}, w \rangle$. A formula $\phi \in \mathcal{L}$ is $\star$-satisfiable if and only if $\not\models_\star \neg\phi$. A $\star$-model that satisfies a formula ϕ will be called a '$\star$-model for ϕ'.

From this definition, it is now interesting to notice that some satisfiability entails some others. For instance, one can see that if a formula ϕ is KT5-satisfiable, then ϕ is also K-satisfiable, KT-satisfiable, S4-satisfiable, *etc.* Whereas, if a formula ϕ is K-unsatisfiable, then it is also KT-unsatisfiable, K4-unsatisfiable, S4-unsatisfiable, *etc.*

If we consider only the axioms we just presented, we can obtain the following theorem:

Theorem 3

Any normal modal logic based on axiom K + $\{(D), (T), (B), (4), (5)\}$ has the good properties presented in Def. 18 Kripke (1959), Ladner (1977).

It was already shown by Ladner that Modal Logic K is PSPACE-complete and that modal logic S5 is NP-complete. The article Halpern and Rêgo (2007) is capital to understand why there is such an NP-PSPACE gap between the satisfiability problems. In a nutshell, their conclusion is that the gap is caused only by the negative introspection axiom also called "axiom (5)". They showed that any modal logic which contains the axiom (5) will have a satisfiability problem NP-complete. All the others which do not contain (5) are PSPACE-complete.

From this moment, it is worth noticing on Figure 2.3 that there is one logic that "contains" the others. We have K5 which is the smallest NP-complete modal logic with respect to the number of axioms. Moreover, it is also known that any normal modal logic has the *Finite Model*

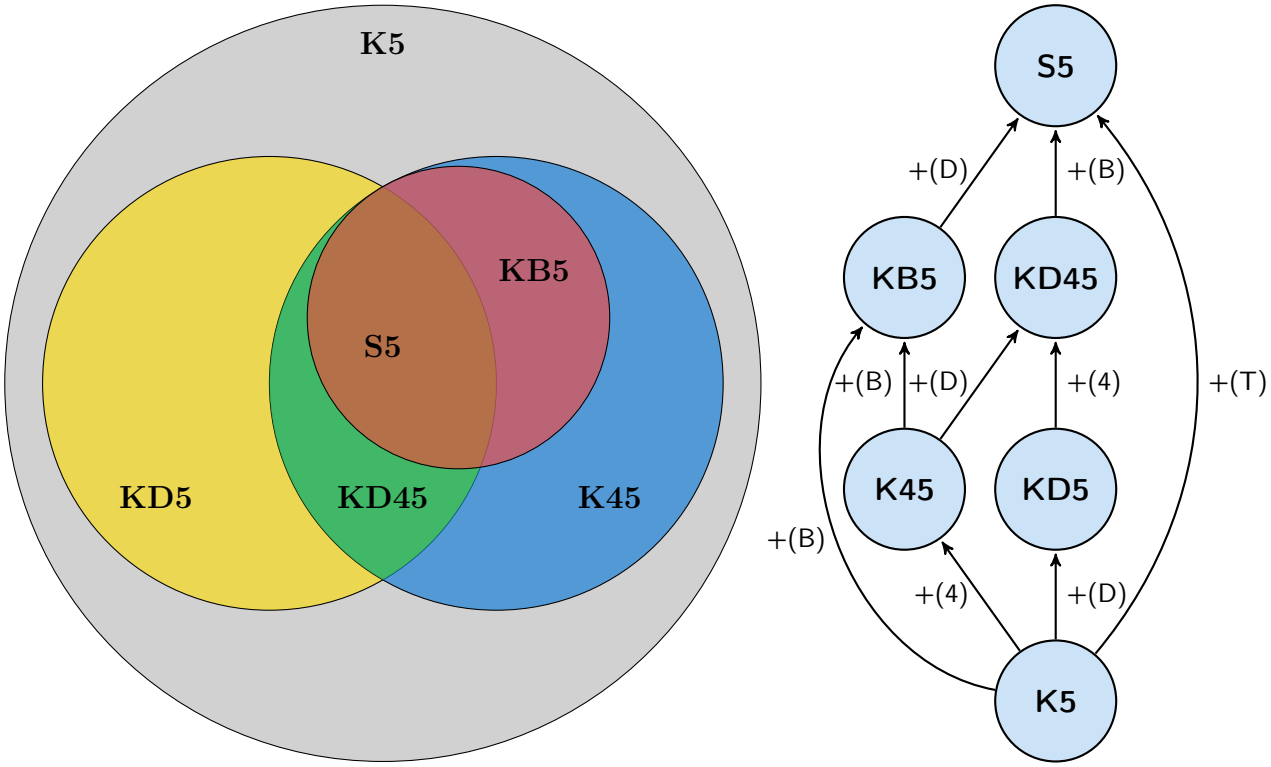


Figure 2.3: Relation between NP modal logics

Property Urquhart (1981). In a nutshell, it means that we know that it exists an upper-bound on the size of a $\star$-model (or at least, an upper-bound on one of its bisimilar). We will then have a series of theorems which state which upper-bound is there for which logic.

It it important to denote also that, as in propositional logic, a modal logic formula can be transformed in Negation Normal Form (NNF) with the following rewrite rules:

- $(\phi \rightarrow \psi) ::= (\neg\phi \vee \psi)$
- $\neg(\phi \vee \psi) ::= (\neg\phi \wedge \neg\psi)$
- $\neg(\phi \wedge \psi) ::= (\neg\phi \vee \neg\psi)$
- $\neg\neg\phi ::= \phi$
- $\neg\Box\phi ::= \Diamond\neg\phi$

From now on, we will consider that any modal logic formula will be in NNF. To define the upper-bound in modal logic K, we need a final definition about the modal logic formulas which is: their modal depth.

Definition 28 (Modal Depth)

The depth of a formula ϕ in $\mathcal{L}$, denoted depth(ϕ), is the highest number of nested modalities as it is defined as follows (where $\oplus \in \{\wedge, \vee, \rightarrow, \leftrightarrow\}$):

$$\text{depth}(p) = \text{depth}(\top) = \text{depth}(\bot) = 0$$

$$\text{depth}(\neg\phi) = \text{depth}(\phi)$$

$$\text{depth}(\phi \oplus \psi) = \max(\text{depth}(\phi), \text{depth}(\psi))$$

$$\text{depth}(\Box\phi) = \text{depth}(\Diamond\phi) = 1 + \text{depth}(\phi)$$

We also need the formal definition of the number of atoms, which is as follows:

Algorithm 2.1: *countAtoms*(ϕ)

Data: ϕ a modal logic formula in NNF, n the numbe of propositional variables in ϕ
Result: The number of distinct occurences of atoms in ϕ

begin
  $nbVarsPos \leftarrow [0, 0, \ldots, 0]$
  $nbVarsNeg \leftarrow [0, 0, \ldots, 0]$
  Function *countNbVars(ϕ)*
    switch ϕ **do**
      case $\phi = (\psi \wedge \chi)$ **do**
        countNbVars(ψ)
        countNbVars(χ)
      end
      case $\phi = (\psi \vee \chi)$ **do**
        countNbVars(ψ)
        countNbVars(χ)
      end
      case $\phi = \Box\psi$ **do** countNbVars(ψ)
      case $\phi = \Diamond\psi$ **do** countNbVars(ψ)
      case $\phi = p$ **do** nbVarsPos[p] = 1
      case $\phi = \neg p$ **do** nbVarsNeg[p] = 1
    end
  end
  countNbVars(ϕ)
  return $\sum_{i=1}^{n}(nbVarsPos[i]) + \sum_{i=1}^{n}(nbVarsNeg[i])$
end

Definition 29 ($|Atom(\phi)|$)

The number of atoms in ϕ in $\mathcal{L}$ is the output of the Algorithm 2.1 on ϕ.

Theorem 4 (Sebastiani and McAllester (1997))

Given a modal logic formula ϕ, if there is no Kripke model of size lower than $Atom(\phi)^{\text{depth}(\phi)}$, then ϕ is unsatisfiable in K.

Theorem 5 (Nguyen (1999))

Given a modal logic formula ϕ, if there is no Kripke model of size lower than $|\phi|^{2\times|\phi|+\text{depth}(\phi)}$, then ϕ is unsatisfiable in $\{K, K4, KD4, KD, KT, KB, KDB, KB, S4\}$ $|\phi|$ denotes the size of the formula (where each atom occurence and operators occurence is counted as one).

Now we also need a bound for the NP-complete modal logics, according to complexity theory, such an upper-bound should be linear in the size of the input and not exponential. We present such an upper-bound is presented in Theorem 6.

Theorem 6 (Halpern and Rêgo (2007))

Given a modal logic formula ϕ, if there is no Kripke model of size lower than $|\phi|$, then ϕ is unsatisfiable in $\{K5, K45, KD5, KB5, KD45, S5\}$. With $|\phi|$ being the size of the formula as denoted in Halpern and Rêgo (2007).

One upper-bound is more precise but only for the modal logic S5 and it is $\text{nm}(\phi)$, where nm is the number of modalities in the formula Ladner (1977). In the rest of the thesis we will denote by $UB(\phi)$, the upper-bound on the size of the Kripke model that could satisfy ϕ. We will make clear all the time, which logic we are talking about.

Now that we described the different modal logics, it is worth to consider how we can decide the satisfiability of a formula in that logic and with which kind of procedure, and this is what we will see in the following chapter. We will first describe the decision procedures for propositional logic and then the decision procedures for modal logics.

Chapter 3

Decision Algorithms and Benchmarks

Contents

> "Proving that I am right would be recognizing that I could be wrong"
>
> *Pierre-Augustin Caron de Beaumarchais*, The Marriage of Figaro, 1778

In this chapter, we will discuss the different procedures that exist to decide the satisfiability of a formula ϕ in propositional logic, especially the famous modern SAT solvers Marques-Silva and Sakallah (1999), Moskewicz et al. (2001) (for a complete view of what is actually a SAT solver see Biere et al. (2009)). We will also discuss satisfiability in modal logics, *ie.* the different state-of-the-art approaches that we will use to compare our approaches during this thesis. Then we will discuss the fact that one technique cannot be the "best technique ever" and is highly dependent on the structure of the benchmarks.

3.1 Modern SAT Solvers

We will discuss the modern SAT solvers in this section because they will be the cornerstone of our contributions in this thesis. Indeed, as stated in the introduction, we want to push, as far as possible, the performance of SAT-based modal logic solvers. But in order to fully understand how the contributions work, one needs to understand the main components of a modern SAT solver and this is what will be presented here.

There exists many techniques to decide the satisfiability of a propositional logic formula ϕ. Usually the formula are considered in CNF (Def. 19). Historically, the first approaches were "incomplete", t@hey did not go through the whole search tree. They were designed to find quickly if the formula is satisfiable, *ie.* it admits a model, but were not efficient (and sometimes even not able) to prove the unsatisfiability. We can cite as such examples of incomplete approaches:

- *Genetic Algorithms* Hao and Dorne (1994);
- *Survey Propagation* Braunstein et al. (2005);
- *Variable Neighbourhood Decomposition* Hansen et al. (2001);
- *The Stochastic Local Search* Selman et al. (1992), Kautz and Selman (1996), Hoos and Stützle (1999).

Worth noticing that early local search solvers were designed to solve optimisation problems and not decision problems. Another specificity of these solvers is that they are much more efficient on random generated benchmarks than their complete counterpart.

Modern SAT solvers are complete, *ie.* they can decide both satisfiability and unsatisfiability, it will go in the worst case, through all the possibility in the search-tree. When we will talk about SAT solvers, we will talk especially about the CDCL (*Conflict Driven Clause Learning*) approaches Marques-Silva and Sakallah (1999), Eén and Sörensson (2003), Biere et al. (2009). We will not go through the historical progression of SAT solver with the Davis and Putnam (DP) Algorithm Davis and Putnam (1960) nor its extension call DPLL Algorithm Davis et al. (1962) and we redirect the reader to Biere et al. (2009) for such historical survey.

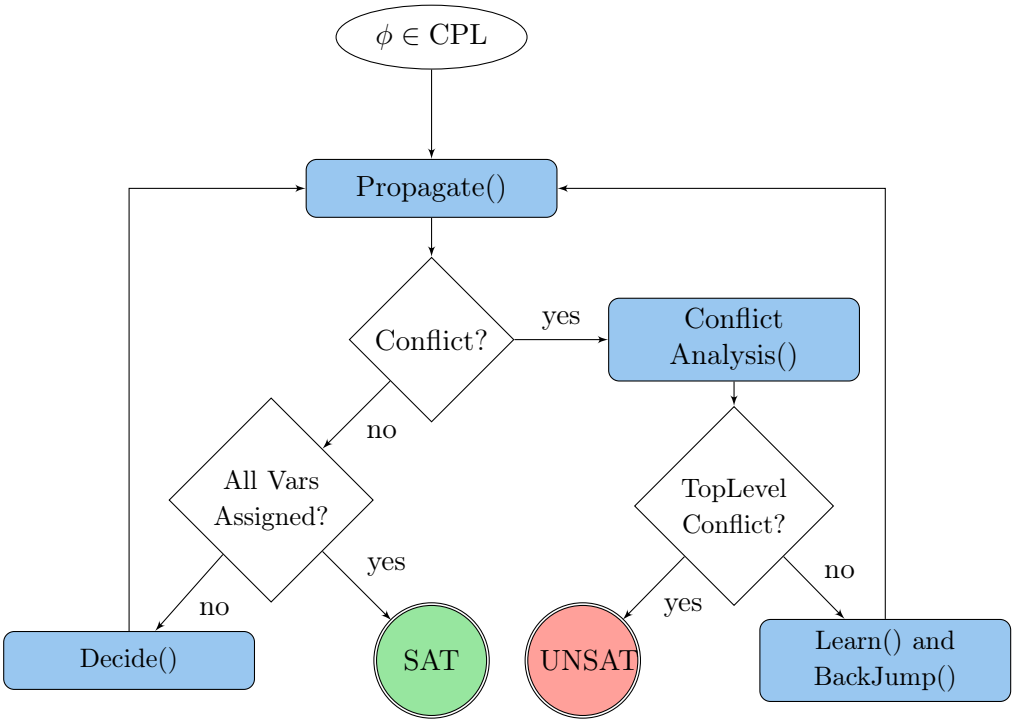


Figure 3.1: Schema of the CDCL Framework

So, as we can see in Figure 3.1, a modern SAT solver has few components that we will start to describe in order to understand globally how such a solver is able to decide the satisfiability of a formula in CPL.

3.1.1 Resolution and Unit Propagation

Two important principles in SAT solvers are the resolution rule and unit propagation. Let us define also that two literals are said to be complements if one is the negation of the other (in the following, $\neg c$ is taken to be the complement to c). The resolution rule is a generalisation of the Modus Ponens. The resolution rule Robinson (1965) is defined as follows:

Definition 30 (Resolution Rule Robinson (1965))

The resolution rule in propositional logic is a single valid inference rule that produces a new clause (called resolvent) implied by two clauses containing complementary literals. Exactly two complementary literals must be used. Formally, we have:

$$\frac{x \vee \alpha \quad \neg x \vee \beta}{\alpha \vee \beta}$$

The resolution rule is sound but incomplete (at least for clausal consequence, it is complete for deciding unsatisfiability). Said otherwise, it is not guaranteed to infer all the clauses implied by a CNF formula. However, the resolution is refutationally complete, *ie.*, it ensures that if a formula is unsatisfiable, then it will infer the empty clause ($\bot$). It is common to represent the sequence of resolutions performed by an algorithm until the empty clause with a refutation-tree.

To obtain a complete proof system, one needs also the factorisation rule which is defined as follows:

Definition 31 (Factorisation rule)

The factorisation rule in propositional logic is the following inference rule:

$$\frac{\alpha \vee \alpha \vee \beta}{\alpha \vee \beta}$$

From this point, we can now define **Unit Resolution**.

Definition 32 (Unit Resolution (UR) Davis and Putnam (1960))

The unit resolution of a CNF formula ϕ is the application of all the resolutions possible which have in common a variable v which appears in a unit clause (a clause which contains only v). Such algorithm will be noted $UR(\phi, v)$.

It safe then to remove all the clauses where v appears, if we have all the resolutions possibles in the formula.

Example 16

Let $\phi = ((\neg a \vee \neg b) \wedge (a \vee c) \wedge (b \vee c) \wedge (\neg c \vee d) \wedge a)$. If we apply the unit resolution on the variable a (which appears in a unit clause), we add the clause $(\neg b)$ in the formula ϕ. Then by removing all the clauses where a appears, we can have $\phi = (\neg b \wedge (b \vee c) \wedge (\neg c \vee d))$.

To apply the unit resolution on a literal l and then delete all the clauses where l appears is equivalent to assign l to the truth value $\top$. More precisely, it consists into replacing all the occurrences of l by TRUE and all the occurrences of $\neg l$ by FALSE and then simplify.

This property came from the fact that all the interpretations that could satisfy the formula ϕ must satisfy all the literals which belong to the unit clauses. The Unit Propagation is to apply the unit resolution until we reach a fixed point.

Definition 33 (Unit Propagation)

Let ϕ be a CNF formula, we denote as ϕ^* the closure by unit propagation of ϕ. ϕ^* is defined by the Algorithm 3.1. Worth to notice that ϕ and ϕ^* are equisatisfiable.

Let us illustrate now, how the UnitPropagation is working in practice. We will give two examples: one to see how UP can decide the satisfiability of a formula, and also how we can retrieve the model and another example to see how UP can reach a fixed point.

Algorithm 3.1: UnitPropagation (UP)

Input: a CNF formula ϕ
Output: $\top$ if ϕ is satisfiable, $\bot$ if ϕ is unsatisfiable, ϕ^* if we reach a fixed point

```
1 φ* = φ;
2 if (∃c, a unit clause containing v in φ*) then
3     φ* = UR(φ*, v); // deletion of all clauses containing v
4     if (φ* = ⊤) then return ⊤ ;
5     if (φ* is contradictory) then return ⊥ ;
6     return UnitPropagation(φ*);
7 else
      // A fixed point is reached
8     return φ*
```

Example 17

Let $\phi = (a \wedge (\neg a \vee \neg b) \wedge (a \vee c) \wedge (b \vee c) \wedge (\neg c \vee d))$. We have:

$$\phi^* = (\underline{a} \wedge (\underline{\neg a} \vee \neg b) \wedge \cancel{(a \vee c)} \wedge (b \vee c) \wedge (\neg c \vee d))$$
$$\phi^* = (\underline{\neg b} \wedge (\underline{b} \vee c) \wedge (\neg c \vee d))$$
$$\phi^* = (\underline{c} \wedge (\underline{\neg c} \vee d))$$
$$\phi^* = (\underline{d})$$
$$\phi^* = \emptyset$$

Thus, ϕ is satisfiable. We can even obtain a model by seeing which variables have been used for the resolution steps. Here the model found is $\mathcal{I} = \{a, \neg b, c, d\}$.

Example 18

Let $\phi = (c \wedge (a \vee b \vee c) \wedge (a \vee \neg b \vee e) \wedge (\neg c \vee \neg e))$. We have:

$$\phi^* = (\underline{c} \wedge \cancel{(a \vee b \vee c)} \wedge (a \vee \neg b \vee e) \wedge (\underline{\neg c} \vee \neg e))$$
$$\phi^* = (\underline{\neg e} \wedge (a \vee \neg b \vee \underline{e}))$$
$$\phi^* = (a \vee \neg b)$$

We no longer have unit clauses, then we reached a fixed point.

For now, it seems straightforward to decide the satisfiability of a CNF formula in CPL. However, the complexity is telling us that it is intractable, the problem is NP-complete Cook (1971), so there must have a tricky-part somewhere.

First, the number of resolvents that can produce a CNF formula can blow-up. It is the case for the PigeonHole problems Haken (1985) where any resolution proof must have an exponential size.

Second, we will often reach a fixed point. Thus we then have to make a choice, and then if we reach the empty clause, it does not mean that the problem is really unsatisfiable, just that we reached a conflict and some choices we did were maybe wrong. Let us then, in the next part explain how we can analyse such a conflict and learn from it to avoid making the same mistake again.

3.1.2 Conflict Analysis and Clause Learning

The goal of the clause learning phase is to find a clause which allows to avoid reproducing twice the same work in the search-tree. To deduce such clauses that we call *Learnt Clauses*, we need to find and analyse the set of literals which is responsible of a conflict Bayardo Jr. and Schrag (1997), Marques-Silva and Sakallah (1999). But to do so, we will need a set of definitions.

Definition 34 (Decision/Propagation Sequence)

Let ϕ a CNF formula in propositional logic.
Let $\mathsf{S}_d = \langle d_1, d_2, \ldots, d_\rangle$ with d_i decided literals be called a **decision sequence** of ϕ.
Let $\mathsf{S}_p = \langle d, \{p_1, p_2, \ldots, p_m\}\rangle$ called a **propagation sequence**, obtained from ϕ, such that:

- d is a decision, or $\emptyset$ to specify that no decision has been taken;
- p_j are propagated literals (unit clauses) thanks to the UnitPropagation.

A **decision/propagation sequence** may be seen as an interpretation $\mathcal{I}$ representing multiple decision sequences and propagation sequences noted:
$\mathcal{I} = \langle\emptyset, \langle x_{0,1}, x_{0,2}, \ldots, x_{0,i}\rangle\rangle, \langle d_1, \langle x_{1,1}, x_{1,2}, \ldots, x_{1,j}\rangle\rangle, \ldots, \langle d_n, \langle x_{n,1}, x_{n,2}, \ldots, x_{n,k}\rangle\rangle$.
In such case n will be called the **decision level** of d_n.
The resulting formula will be noted: $\phi_{|d_1,\ldots,d_n}$.

So now we know how to represent a sequence of decisions and their associated propagation, let us illustrate how we reach a conflict and we can avoid it:

Example 19

Let ϕ be a CNF formula composed of the following six clauses using nine variables:

$$c_1 = (x_1 \vee x_2) \qquad c_2 = (x_1 \vee x_3 \vee x_7) \qquad c_3 = (\neg x_2 \vee \neg x_3 \vee x_4)$$
$$c_4 = (\neg x_4 \vee x_5 \vee x_8) \qquad c_5 = (\neg x_4 \vee x_6 \vee x_9) \qquad c_6 = (\neg x_5 \vee \neg x_6)$$

If we consider the following decision sequence $\mathsf{S}_d = \langle \neg x_7, \neg x_8, \neg x_9, \neg x_1 \rangle$ we have:

$\phi = (x_1 \vee x_2) \wedge (x_1 \vee x_3 \vee \cancel{x_7}) \wedge (\neg x_2 \vee \neg x_3 \vee x_4) \wedge (\neg x_4 \vee x_5 \vee x_8) \wedge (\neg x_4 \vee x_6 \vee x_9) \wedge (\neg x_5 \vee \neg x_6)$
$\phi_{|\neg x_7} = (x_1 \vee x_2) \wedge (x_1 \vee x_3) \wedge (\neg x_2 \vee \neg x_3 \vee x_4) \wedge (\neg x_4 \vee x_5 \vee \cancel{x_8}) \wedge (\neg x_4 \vee x_6 \vee x_9) \wedge (\neg x_5 \vee \neg x_6)$
$\phi_{|\neg x_7, \neg x_8} = (x_1 \vee x_2) \wedge (x_1 \vee x_3) \wedge (\neg x_2 \vee \neg x_3 \vee x_4) \wedge (\neg x_4 \vee x_5) \wedge (\neg x_4 \vee x_6 \vee \cancel{x_9}) \wedge (\neg x_5 \vee \neg x_6)$
$\phi_{|\neg x_7, \neg x_8, \neg x_9} = (\cancel{x_1} \vee x_2) \wedge (\cancel{x_1} \vee x_3) \wedge (\neg x_2 \vee \neg x_3 \vee x_4) \wedge (\neg x_4 \vee x_5) \wedge (\neg x_4 \vee x_6) \wedge (\neg x_5 \vee \neg x_6)$
$\phi_{|\neg x_7, \neg x_8, \neg x_9, \neg x_1} = \cancel{x_2} \wedge \cancel{x_3} \wedge (\neg \cancel{x_2} \vee \neg \cancel{x_3} \vee x_4) \wedge (\neg x_4 \vee x_5) \wedge (\neg x_4 \vee x_6) \wedge (\neg x_5 \vee \neg x_6)$
$\phi_{|\neg x_7, \neg x_8, \neg x_9, \neg x_1, x_2, x_3} = \cancel{x_4} \wedge (\neg \cancel{x_4} \vee x_5) \wedge (\neg \cancel{x_4} \vee x_6) \wedge (\neg x_5 \vee \neg x_6)$
$\phi_{|\neg x_7, \neg x_8, \neg x_9, \neg x_1, x_2, x_3, x_4} = \cancel{x_5} \wedge x_6 \wedge (\neg \cancel{x_5} \vee \neg x_6)$
$\phi_{|\neg x_7, \neg x_8, \neg x_9, \neg x_1, x_2, x_3, x_4, x_5} = x_6 \wedge \neg x_6$
$\phi_{|\neg x_7, \neg x_8, \neg x_9, \neg x_1, x_2, x_3, x_4, x_5, x_6} = \bot$

The conflict is obtained from the following decision/propagation sequence: $\mathcal{I} = \langle \emptyset, \langle \emptyset \rangle \rangle, \langle \neg x_7, \langle \emptyset \rangle \rangle, \langle \neg x_8, \langle \emptyset \rangle \rangle, \langle \neg x_9, \langle \emptyset \rangle \rangle, \langle \neg x_1, \langle x_2, x_3, x_4, x_5, x_6, \neg x_6 \rangle \rangle \rangle$. Let us now assume that we would perform this sequence without obtaining the conflict, a simple idea would be to add the following clause $(x_7 \vee x_8 \vee x_9 \vee x_1)$ to ϕ. In that way, when the first three decisions will be performed, x_1 will be propagated and then we will avoid the conflict.

Even though Example 19 is showing a way to generate *learnt clauses*, they will have a huge size and each of them will represent only one conflict. Let us remark that some works use this technique to generate *learnt clauses*, when these clauses are "small enough". The main problem however is that the information contained in this clause is useless in case of a backtrack and except if a BackJump() is performed, we can safely remove these clauses.

We will not go into the details of the approaches used nowadays in the solvers to generate Learnt clauses, but in a nutshell, they are using a *Direct Acyclic Graph* (DAG) to represents the dependencies between the clauses and the assignment of the truth values obtained by the UnitPropagation. This way to analyse the conflict and to obtain *Learnt clauses* is also giving the approach some information to know at which decision level should be performed a back jump. If this decision level is equal to 0, then we know that the problem is unsatisfiable, the conflict has been made without any decision.

In practice, most modern SAT solvers learn clauses associated to the notion of First *Unique Implication Point* (*UIP*), which is any node at the current decision level such that any path from the decision variable to the conflict node must pass through it. There exists many works based on conflict analysis, for example Sörensson and Biere (2009) which propose an approach consisting of performing resolutions while they do not increase the size of the learnt clause. Other work try to discover subsumed clauses during the conflict analysis Han and Somenzi (2009), Hamadi et al.

(2010). Another approach proposed by Audemard et al. (2008) tries to extend the implication graph in order to consider some clauses satisfied by the formula.

So now, we know all the blocks from the Figure 3.1 except one: how to decide which variable to pick and which polarity. This is what we will explore in the next section.

3.1.3 Choice of variable and choice of polarity

The choice of the next variable that is selected is one of the most important criteria of a SAT solver. Indeed, if we could select only the good literals then we could construct the model. Unfortunately it is as hard to pick a variable to construct a model as to solve the satisfiability problem Liberatore (2000). Finally, the choice of variables in the search has a huge impact of the number of steps that need to do a CDCL approach, thus also on its execution time Li and Anbulagan (1997).

There exists many different heuristics to pick a variable and we redirect the reader to Biere et al. (2009) for more information about them. In this thesis, we will present only the most used one named VSIDS (Variable State Independent Decaying Sum), presented in Zhang et al. (2001). In a nutshell, VSIDS maintains a score for each literal, and the literals with the highest scores are stored in an array used to do the next decision. After learning a clause, the scores of the literals inside it are incremented. More precisely, the score of all the literals which appear in the resolution steps during the Conflict Analysis. VSIDS can in fact be defined as follows:

1. Each literal has a score, initialized to 0
2. When a clause is learnt, the score associated with each literal in the clause is incremented. The score keeps track of how often the literal is used.
3. The unassigned literal with highest score is chosen at each decision point.
4. Ties are broken by random choice.
5. All scores are divided by a constant, periodically.

VSIDS is quite effective because the scores of variable phases is independent of the current variable assignment, so backtracking is much easier. In an nutshell, the goal of VSIDS is to solve the hard part of a formula first.

Obviously, this heuristic has been improved/changed. One can cite as such the MiniSAT's version of it Eén and Sörensson (2003), which is defined as follows:

1. Each variable has a score
2. When a clause is learnt, the score of all clauses encountered during the conflict analysis is incremented.
3. Instead of a decay factor, variable scores are "bumped" with larger and larger values in a floating point representation.
4. When very large values are encountered, all scores are scaled down.
5. 2% of the time, a random decision is made instead. This factor is set at run-time.

In addition, MiniSAT implementation of VSIDS always keeps the variables in order by placing them in an array-based priority queue. Unfortunately, such modifications are making it harder to understand this heuristic. Some authors even published on the matter on how to understand VSIDS branching heuristic in CDCL solvers and we redirect the reader to Liang et al. (2015) for more understanding of this "additive bumping" and this "multiplicative decay".

Now that we saw how a SAT solver is working, we need to present its different features, because obviously, nowadays SAT are not just deciding the problems, they are giving more informations, usable in more complex procedures.

3.1.4 Latest features of modern SAT solvers

Recent SAT solvers are incremental, *ie.*, they are able to check the satisfiability of a formula "under assumptions" Eén and Sörensson (2003) and are able to output a core (a "reason" for the unsatisfiability of the formula). The use of unsatisfiable cores is the corner-stone of many applications, such as MaxSAT Li and Manyà (2009), MCS (Minimal Correction Set) Grégoire et al. (2014), MUS (Minimal Unsatisfiable Set) Belov et al. (2012). The unsatisfiable core is defined as follows:

Definition 35 (Unsatisfiable Core under Assumptions)

Let ϕ be a satisfiable formula in CNF built using Boolean variables from $\mathbb{P}$. Let A be a consistent set of literals built using Boolean variables from $\mathbb{P}$ such that $(\phi \wedge \bigwedge_{a \in A} a)$ is unsatisfiable. $C \subseteq A$ is an unsatisfiable core (UNSAT core) of ϕ under assumptions A if and only if $(\phi \wedge \bigwedge_{c \in C} c)$ is unsatisfiable.

With such unsatisfiable core, we can now see a SAT solver as a piece of software able to output a model if the problem is satisfiable or able to output an unsatisfiable core under some assumptions if the problem is unsatisfiable. From now on, when we will talk about SAT solver, we will talk about such solver defined as follows:

Definition 36 (SAT Solver under Assumptions)

Let ϕ be a formula in CNF. A SAT solver for ϕ, given assumptions A, is a procedure which provides a pair $\langle r, s \rangle$ with $r \in \{\mathsf{SAT}, \mathrm{UNSAT}\}$ such that if $r = \mathsf{SAT}$ then s is a model of ϕ, else if $r = \mathrm{UNSAT}$ then s is an UNSAT core of ϕ under assumptions A.

Now that we know how to solve efficiently the satisfiability problem in propositional logic, which we recall, is a NP-complete problem Cook (1971). Let us see now how we can solve the satisfiability problem in modal logics which can also be NP-complete but also PSPACE-complete with respect to the logic considered Ladner (1977), Halpern and Moses (1992).

3.2 Algorithms for Modal Logics

It seems fair to say that nowadays, the best way to solve the satisfiability problem in propositional logic is to use a CDCL SAT solver. For the problems in modal logics, the choice is not so clear. There exists many different techniques with many solvers available in the community. One of the "basic technique" yet efficient, is to use a Tableau method to decide the satisfiability of the problem.

3.2.1 Tableaux methods

We suppose the reader familiar with the Tableau method in propositional logic proposed by Smullyan (1966). The Tableau method is a refutation procedure; if all the branches of the tableau close, then the initial formula is unsatisfiable. Basically it works as follows: if there is an analytical rule (*ie.*, they produce strict subformulas of the original formula) for every logical connective then the procedure will eventually produce a set which consists only of atomic formulae and their negations, which cannot be broken down any further. Such a set is easily recognizable as satisfiable or unsatisfiable with respect to the semantics of the logic in question. To define the Tableaux methods for modal logics, we will use the notation proposed by Fabio Massacci (2000), which is as follows:

- Prefix: σ a finite non-empty sequence of integers
  - σ is interpreted as a state of a model
  - the concatenation is denoted: $\sigma.\sigma'$
- Prefixed formula: a pair $\sigma : \phi$ where σ is a prefix and ϕ is a modal logic formula
- Tableau: a tree where the nodes are labelled with prefixed formulas
- Branch: standard notion in a tree, defined as a path from the root to a leaf.

From such a definition, we can now have the following fundamental properties of the Tableau calculus:

Definition 37 (Fundamental Properties of the Tableau Calculus)

Here is a list of fundamental properties needed to construct a Tableau to decide the satisfiability of a formula ϕ:

- The root of the Tableau is labelled by $1 : \phi$;
- σ is on a branch if and only if a prefixed formula $\sigma : \phi$ labelled a node on the branch;
- The inference rules are analytic thus if $\sigma.n$ is a new prefix on the branch, then necessary σ was already on the branch;
- The set of Prefix on a branch creates a tree;
- The difference of length between two prefixes of the same rule is at most one ("Single Step Tableaux");

From this point, it is now easy to see that we need to define rules for every possible operators in the language considered, to create a Tableau method to decide the satisfiability in this logic. Let us see the language of modal logic (Def. 21). We know that it has five operators which needs a rule $\{\neg, \wedge, \vee, \Box, \Diamond\}$.

$$\frac{\sigma : (\phi \vee \psi)}{\sigma : \phi \mid \sigma : \psi} \qquad \frac{\sigma : (\phi \wedge \psi)}{\begin{array}{c}\sigma : \psi \\ \sigma : \phi\end{array}} \qquad \frac{\sigma : \neg\neg\phi}{\sigma : \phi} \qquad \frac{\sigma : \Box\phi}{\sigma.n : \phi} \qquad \frac{\sigma : \Diamond\phi}{\sigma.n : \phi \ (n \text{ fresh})}$$

When the label is indicated as fresh, it means that it will be created. When a label is not indicated as fresh, it just means that the rule should match all the labels that are already existing.

And with such Tableaux rules, it is now possible to deal with modal logic K. To deal with all the other modal logics based on K, we need to specify a rule to deal with all the axioms.

$$(K)\ \frac{\sigma : \Box\phi}{\sigma.n : \phi} \qquad (T)\ \frac{\sigma : \Box\phi}{\sigma : \phi} \qquad (4)\ \frac{\sigma : \Box\phi}{\sigma.n : \Box\phi}$$

$$(B)\ \frac{\sigma.n : \Box\phi}{\sigma : \phi} \qquad (D)\ \frac{\sigma : \Box\phi}{\sigma : \Diamond\phi} \qquad (5)\ \frac{\sigma : \Box\phi}{\sigma.n : \Diamond\phi \ (n \text{ fresh})}$$

This tableau-rule presentation of the axioms are there to show the relationship with the properties they force on the Kripke structure. For example, when we take a look at the rule (B), we can easily see the Symmetry property appearing, however it is not clear how a Tableau method having such a rule can always terminates, be sound and complete. It is not the goal of this section, we redirect the reader to the Chapter of Rajeev Goré in the Handbook of Tableau Methods Goré (1999) to know how a Tableau can be constructed for any logic based on K (or in (Massacci 2000, Section 10) to use exactly the same Tableau rules). From this point, we will need few definitions to talk about Tableau methods more precisely. The first definition is the notion of a branch open/close which is defined as follows:

Definition 38 (Closure)

A branch $\mathcal{B}$ is closed if there is a σ such that, for some propositional variable p, both $\sigma : p$ and $\sigma : \neg p$ are present in $\mathcal{B}$. A Tableau is *closed* if every branch is closed.

Definition 39 (Reduced)

A prefixed formula $\sigma : \phi$ is reduced for rule (r) in $\mathcal{B}$

- if (r) has the form $\sigma : \phi \rightarrow \sigma' : \phi'$ and $\sigma' : \phi'$ is in $\mathcal{B}$;
- if (r) has the form $\sigma : \sigma_1 : \phi_1 \mid \sigma_2 : \phi_2$ and at least one of $\sigma_1 : \phi_1$ and $\sigma_2 : \phi_2$ is in $\mathcal{B}$.

The symbol $\rightarrow$ does not denote the implication but the horizontal bar of the tableau rules.

We will say now that:

- A prefixed formula $\sigma : \phi$ is fully-reduced in $\mathcal{B}$ if it is reduced for all applicable rules;
- A prefix σ is (fully) reduced if all prefixed formula $\sigma : \phi$ are (fully) reduced;
- A branch $\mathcal{B}$ is completed if all prefixes in $\mathcal{B}$ are fully reduced for $\mathcal{B}$;
- A branch $\mathcal{B}$ is open if it is completed and not closed;

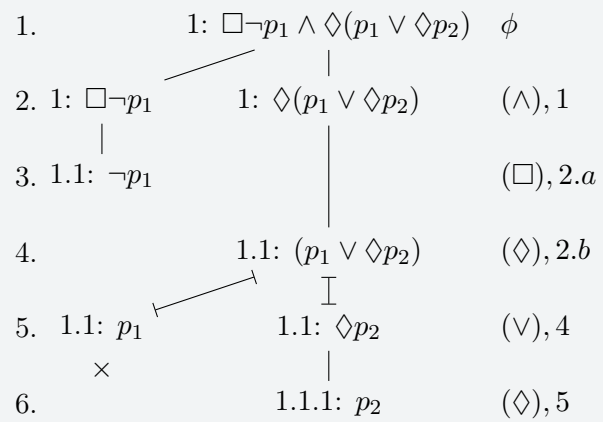
Theorem 7

A formula ϕ is $\star$-satisfiable if and only if there is an open $\star$-Tableau starting with $1 : \phi$. Where $\star$ ranges over (possibly non-empty) subsets of { D, T, 4, B, 5 }.

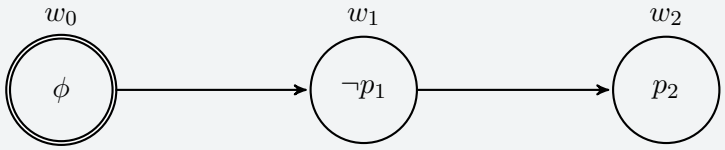
Let us show with an example how a Tableau method is working and how we can show the K-satisfiability of a formula.

Example 20

Let us demonstrate that the formula $\phi = (\Box\neg p_1 \wedge \Diamond(p_1 \vee \Diamond p_2))$ is satisfiable in modal logic K. Or said otherwise, there exist a K-Tableau open.



One can notice that the different notation for an and-branching and for an or-branching. Indeed, in an and-branching, the Tableau must satisfy both branches whereas in the or-branching, the Tableau must satisfy at least one of the branches.
Because there is still at least, one branch which is open, we can conclude that the formula ϕ is satisfied. In fact, thanks to the prefixes, we can retrieve a Kripke model satisfying it.



The Kripke model constructed by the Tableau is depicted above.

There exists many solvers in the literature which use the Tableaux methods to solve the satisfiability problems in modal logics. We can cite as such Spartacus Götzmann et al. (2010) which is to the best of our knowledge, the most efficient Tableau solver for modal logics. There is also FaCT++ Tsarkov and Horrocks (2006), LoTREC Gasquet et al. (2005), MetTeL2 Schmidt and Tishkovsky (2008), KRIS Baader and Hollunder (1991), LWB Balsiger et al. (1998), LCKS5 Abate et al. (2007), *etc.*. There exists many other Tableaux solvers for modal logics, we just cited a few of them. The final Tableau solver we want to talk about is InKreSAT Kaminski and Tebbi (2013) because it makes the link between the two previous sections and the next one. It is an innovative SAT-based system where the SAT solver drives the development of a Tableau.

There exists other approaches to tackle the modal logics satisfiability problems, and one of them is to translate the modal logics satisfiability problem into a $\mathcal{L}$-satisfiability problem where $\mathcal{L}$ could be propositional logic, first order logic, *etc.*

3.2.2 Translation-based methods

Indeed as we just said, the Tableau method is not the only possible approach to solve a modal logic satisfiability problem. Another famous technique is to use another logic $\mathcal{L}$ and to find an efficient way to translate the modal satisfiability problem into an $\mathcal{L}$-satisfiability problem. The propositional logic is widely used in this domain, as explained in Sebastiani and Tacchella (2009). One solver extremely good at that is K$_m$2SAT Sebastiani and Vescovi (2009). To explain how this solver is working, let us make the presentation uniform and let us use the traditional representation of K$_m$ formulas (introduced by Fitting (1983)).

α	α_1	α_2	β	β_1	β_2	π^r	π^r_0	υ^r	υ^r_0
$(\phi \wedge \psi)$	ϕ	ψ	$(\phi \vee \psi)$	ϕ	ψ	$\Diamond_r \phi$	ϕ	$\Box_r \phi$	ϕ
$\neg(\phi \vee \psi)$	$\neg\phi$	$\neg\psi$	$\neg(\phi \wedge \psi)$	$\neg\phi$	$\neg\psi$	$\neg\Box_r \phi$	$\neg\phi$	$\neg\Diamond_r \phi$	$\neg\phi$

Table 3.1: Traditional representation of K$_m$ formulas

Non-literal K$_m$-formulas are grouped into four categories: α (conjunctive), β (disjunctive), π (existential), υ (universal). K$_m$2SAT uses the propositional logic and their encoding is defined recursively as follows:

Definition 40 (Km2SAT Encoding)

Let $A_{\langle,\rangle}$ be an injective function which maps a prefixed formula $\langle\sigma, \psi\rangle$, such that ψ is not in the form $\neg\phi$, into a Boolean variable $A_{\langle\sigma,\psi\rangle}$. They assumed that $A_{\langle\sigma,\top\rangle}$ is $\top$ and $A_{\langle\sigma,\bot\rangle}$ is $\bot$. Let $L_{\langle\sigma,\phi\rangle}$ denote $\neg A_{\langle\sigma,\phi\rangle}$ if ψ is in the form $\neg\phi$, $A_{\langle\sigma,\phi\rangle}$ otherwise.

$$
\begin{aligned}
K_m2SAT(\phi) &\stackrel{\text{def}}{=} A_{\langle\sigma,\phi\rangle} \wedge Def(1,\phi)\\
Def(\sigma,\top) &\stackrel{\text{def}}{=} \top\\
Def(\sigma,\bot) &\stackrel{\text{def}}{=} \bot\\
Def(\sigma,A_i) &\stackrel{\text{def}}{=} \top\\
Def(\sigma,\neg A_i) &\stackrel{\text{def}}{=} \top\\
Def(\sigma,\alpha) &\stackrel{\text{def}}{=} (L_{\langle\sigma,\alpha\rangle} \rightarrow (L_{\langle\sigma,\alpha_1\rangle} \wedge L_{\langle\sigma,\alpha_2\rangle})) \wedge Def(\sigma,\alpha_1) \wedge Def(\sigma,\alpha_2)\\
Def(\sigma,\beta) &\stackrel{\text{def}}{=} (L_{\langle\sigma,\beta\rangle} \rightarrow (L_{\langle\sigma,\beta_1\rangle} \vee L_{\langle\sigma,\beta_2\rangle})) \wedge Def(\sigma,\beta_1) \wedge Def(\sigma,\beta_2)\\
Def(\sigma,\pi^{r,j}) &\stackrel{\text{def}}{=} (L_{\langle\sigma,\pi^{r,j}\rangle} \rightarrow L_{\langle\sigma.j^r,\pi^{r,j}_0\rangle}) \wedge Def(\sigma.j^r,\pi^{r,j}_0)\\
Def(\sigma,\upsilon^r) &\stackrel{\text{def}}{=} \bigwedge_{\text{for every } \langle\sigma,\pi^{r,i}\rangle} (((L_{\langle\sigma,\upsilon^r\rangle} \wedge L_{\langle\sigma,\pi^{r,i}\rangle}) \rightarrow L_{\langle\sigma.i^r,\upsilon^r_0\rangle}) \wedge Def(\sigma.i^r,\upsilon^r_0))
\end{aligned}
$$

Here $\pi^{r,j}$ means that $\pi^{r,j}$ is the j-th distinct π^r formula labelled by σ. $\langle\sigma,\pi^{r,i}\rangle$ ranges over all the labels r, i, accessible from the label r.

We can see that the number of propositional variables and the number of clauses that generates $K_m2SAT(\phi)$ will grow exponentially with the modal depth of ϕ. This has to be the case, as stated by Halpern and Moses (1992), because K-satisfiability is a PSPACE-complete problem Ladner (1977). Moreover, if we take a look at how the formula is encoded in propositional logic, what we see is that K_m2SAT will force a "tree-shaped" Kripke-model, it has no guarantee on the size of the Kripke model outputted. It could exist a much smaller model but K_m2SAT does not search for it. There exists also other solvers that use the help of a SAT solver to decide the satisfiability in modal logics. We already talked about InKreSAT Kaminski and Tebbi (2013), but there is also ⋆SAT Giunchiglia et al. (2002) which interleaves SAT reasoning and domain reasoning. Somehow, ⋆SAT can be seen as an early attempt of SMT reasoning Barrett et al. (2009).

Modal Logic satisfiability problems can also be translated into other logic and/or use another oracle than a SAT solver to be decided. It is the case for example for the solver Moloss[3]. It is a solver based on the SMT encoding presented in Areces et al. (2015) and it uses an SMT solver to decide the modal logic satisfiability problem. There is also KCSP Brand et al. (2003) which uses a Constraint Satisfaction Problem Rossi et al. (2006), Mackworth (1977), Stallman and Sussman (1977) encoding to solve the modal logic K satisfiability problem, unfortunately the solver is not longer accessible on the Internet to the best of our knowledge. Obviously, there is also an approach translating the modal logic K satisfiability problem into another PSPACE-complete problem which is QBF (Quantified Boolean Formula). This approach is designed within the solver QMRES Pan and Vardi (2004). One last logic that can be used to solve modal logic satisfiability problems is the first-order logic (FOL). That is the technique that uses (M)SPASS Hustadt et al. (1999), Weidenbach et al. (2009), but it is also possible to translate directly the modal logic satisfiability problem into a first-order logic satisfiability problem with an external tool such as Optimized Functional Translation (OFT) Horrocks et al. (2006) and to call one of the state-of-the-art FOL solvers such as Vampire Kovács and Voronkov (2013).

Finally, now that we saw the Tableaux approaches and the Translation-based approaches, there are still some solvers which can not be classified into these two categories, they use an "other method" that we will now describe.

3.2.3 Other methods

Indeed, some solvers can not be classified as a Tableau-approach nor as a Translation-based approach, it is the case for example for the solver KЯ Voronkov (1999). The idea behind KЯ is to compare a top-down (Tableau) and a bottom-up (inverse) decision methods. The inverse method can be seen as bottom-up version of the sequent calculus. To simplify greatly the idea behind KЯ, one could say that it is a modalised version of the propositional resolution.

Another approach, where it is a resolution-based solver named K_SP Nalon et al. (2016). They presented a clausal calculus for modal logic K_m which is sound, complete and terminates Nalon et al. (2015). Clauses are labelled by the modal level at which they occur. In order to refer explicitly to modal levels, the modal language is extended with labels. They denote by $ml : \phi$ the fact that ϕ is true at the modal layer ml in a Kripke model, where $ml \in \mathbb{N} \cup \{*\}$. By $* : \phi$ they indicate that ϕ is true at all modal layers in a Kripke model. The motivation for the use of this labelled clausal normal form is that inference rules can then be guided by the semantic

[3]There is no paper talking about Moloss yet but the source-code is accessible at https://github.com/Meleagant/MOLOSS

information given by the labels and applied to smaller sets of clauses, reducing the number of unnecessary inferences, and therefore improving the efficiency of the proof procedure.

Finally, the last solver that we want to talk about is BDDTab Goré et al. (2014). Somehow we could have classified this solver as a Tableau solver, but instead of computing a tableau for the formula in a classical way, they used a Binary Decision Diagram (BDD) as an effective base data structure for computing tableaux.

We just talked about a lot of different solvers, but we did not talk about which modal logics they can deal with, so let us recall all this information in Table 3.2 and list few additional information just after to clarify some points.

Table 3.2: List of solvers and the logics they can deal with

	Solvers		Oracle used	K	KT	S4	K5	KD45	S5
	Spartacus	Götzmann et al. (2010)	–	✓	✓	✓			
	FaCT++	Tsarkov and Horrocks (2006)	–	✓	✓	✓			
	LoTREC	Gasquet et al. (2005)	–	✓	✓	✓	✓	✓	✓
	MetTeL2	Schmidt and Tishkovsky (2008)	–	✓	✓	✓	✓	✓	✓
Tableaux	KRIS	Baader and Hollunder (1991)	–	✓					
	LWB	Balsiger et al. (1998)	–	✓	✓	✓			
	LCKS5	Abate et al. (2007)	–						✓
	InKreSAT	Kaminski and Tebbi (2013)	Minisat Eén and Sörensson (2003)	✓	✓	✓			
	Km2SAT	Sebastiani and Vescovi (2009)	SAT*	✓					
	⋆SAT	Giunchiglia et al. (2002)	SATO Zhang (1997)	✓					
Translation	Moloss	Areces et al. (2015)	Z3 de Moura and Bjørner (2008)	✓	✓	✓	✓	✓	✓
	KCSP	Brand et al. (2003)	CP*	✓	✓	✓			
	QMRES	Pan and Vardi (2004)	QBF*	✓					
	MSPASS	Hustadt et al. (1999)	–	✓	✓	✓	✓		✓
	Vampire	Kovács and Voronkov (2013)	–	✓	✓	✓	✓		✓
	$K\mathfrak{Y}$	Voronkov (1999)	–	✓					
Other	$K_S P$	Nalon et al. (2016)	–	✓	✓	✓			
	BDDTab	Goré et al. (2014)	–	✓		✓			

- KCSP does not deal "officially" with KT and S4, but Stevenson et al. (2008) extends the algorithm to deal with these two logics.
- SAT* in the line of Km2SAT means that this solver can be used in combination with any SAT solver from the literature, *eg.* Glucose Audemard and Simon (2009)
- CP* in the line of KCSP means that this solver can be used in combination with any CP solver from the literature, *eg.* AbsCon Merchez et al. (2001)

- QBF* in the line of QMRES means that this solver can be used in combination with any QBF solver from the literature, *eg.* RaReQS Janota et al. (2016)

- K_SP deals with KT and S4, but in 2018, it is still preliminary and not fully optimized to be competitive.

Now that we know what are the state-of-the-art approaches to solve modal logic $\star$ satisfiability problems, we need to know what are the standard benchmarks to evaluate them. A small amount of benchmarks have been produced for testing the effectiveness of the different technique Giunchiglia et al. (1996), Balsiger et al. (2000), Patel-Schneider and Sebastiani (2003), Massacci (1999), Massacci and Donini (2000). But before describing these benchmarks and explaining how they have been created, let us remind what is known in the SAT community for decades: the structure of the benchmark will influence the experimental results. What A. Einstein said about a fish also applies to solving techniques "If You Judge a Fish by Its Ability to Climb a Tree, It Will Live Its Whole Life Believing that It is Stupid".

3.3 Structural Impact Of The Benchmarks

Indeed, as we just said, the structure of a benchmark impacts the performance of any solver. This is why the ability to visualise the structure of a benchmark has been studied in the SAT community Sinz (2004), Selman (2004). Some graphical tool such as `DPvis` developed by Sinz and Dieringer (2005) were made available to see the structure of the benchmarks in CNF. In a nutshell, what is interesting for us here is depicted in Figure 3.2.

The Figure 3.2 on the left represents the *primal graphs* of some peculiar SAT instances. The primal graph is an undirected graph $G = (V, E)$, where the vertex set V is the set of variables of S and $\{x, y\} \in E$ if and only if there is a clause $c \in S$ that contains both variables x and y.

What we can see in Figure 3.2, is a perfect decomposition of why the SAT instances are classified in three categories: *Application*, *Crafted*, *Random*. We will say in thesis, as in the SAT community, that:

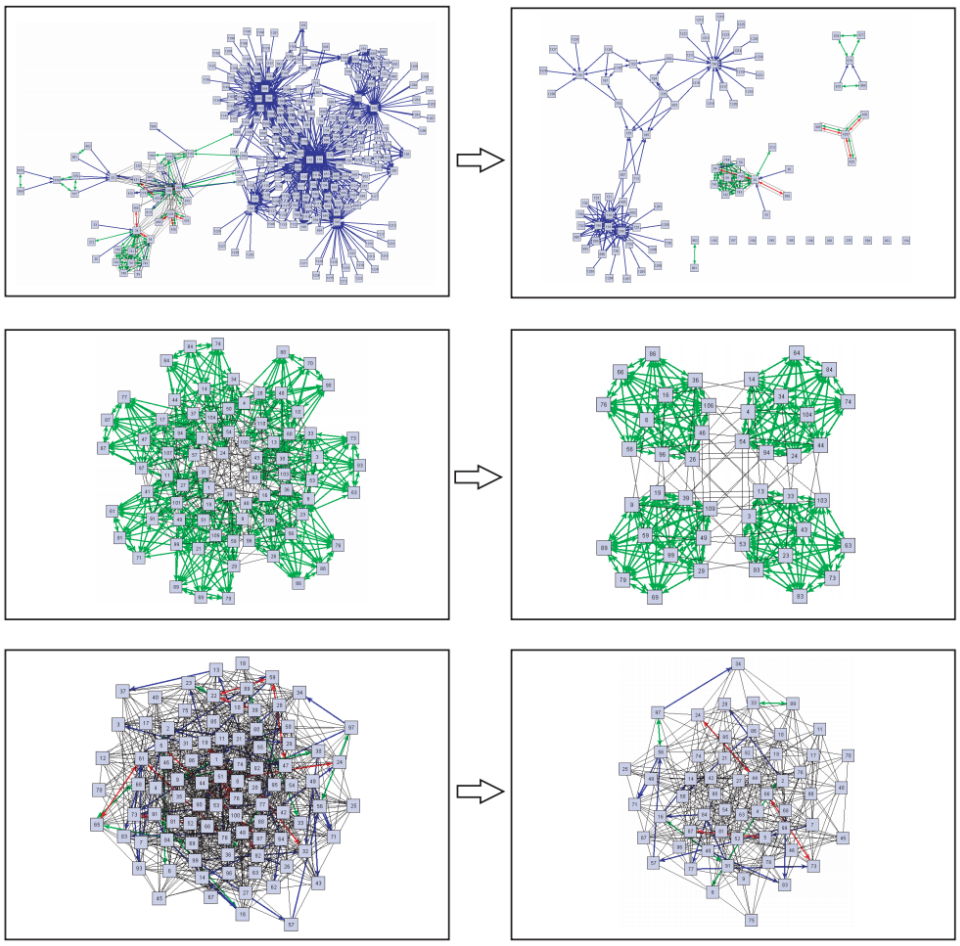


Figure 3.2: "*Interaction graphs for different SAT instances before (left) and after (right) three steps of a DP algorithm run (and subsequent simplification by unit propagation). On the top, an instance from automotive product configuration is shown, in the middle a pigeon hole formula, and on the bottom a random 3-SAT formula with a clause-variable-ratio near the phase-transition point*" Sinz (2004)

Definition 41 (Different categories of benchmark)

- An instance is an *Application* instance if it comes from a modelisation of a real problem (such as the Automotive Product Configuration in the Figure 3.2). The motivation behind this category is to highlight the kind of applications SAT solvers may be useful for.
- An instance is a *Crafted* instance if it is designed to give a hard time to the solvers, or represent otherwise problems that are challenging to typical solvers (including, for example, instances arising from puzzle games), (such as the Pigeon Hole Problem Haken (1985) in the Figure 3.2). These benchmarks are typically small. The motivation behind this category is to highlight current challenging problem domains that reveal the limits of current technology.
- An instance is a *Random* instance if it is has been randomly generated. This category is motivated by the fact that the instances can be fully characterized and by its connection especially to statistical physics.

This separation may sound like a sugar coat on the SAT instances but it is not the case. The performance of solvers have very high variations according to which category of benchmarks we are using to evaluate them. In fact, as explained in Järvisalo et al. (2012), "In 2011, the smallest crafted instance not solved by any solver within the time-out contained only 141 variables, 292 clauses, and 876 literals in total. In contrast, the biggest application instance solved by at least one solver contained 10 million variables, 32 million clauses, and a total of 76 million literals".

Now that we know what are the categories, let us see in which of them are all the already-existing benchmarks from the literature.

3.3.1 Random Benchmarks

The first category of benchmark that we want to talk about are the $3CNF_K$ Giunchiglia et al. (1996).

3CNF$_K$ Benchmarks

An atom in the generator presented in Algorithm 3.2 is either a propositional variable or its negation. The function *flip_ coin(p)* represents a function checking if a generated random number between 0 and 1 is higher or not than p. The function *rand_ propositional_ atom(N)* will generate a random propositional variable with an index being between 1 and N. The algorithm will always generate clauses of size 3 (hence the name 3CNF) and the modalities are randomly added.

The idea behind this generator (presented in Algorithm 3.2) is to have an algorithm which can provide a statistical control of some important features *eg.* hardness and satisfiability rate, of the formulas generated. As the name suggests, it is a generator for Modal Logic K Satisfiability Problem. Because we will not talk in this thesis about multiple agents formulas Blackburn et al. (2006), we will re-write the generator algorithm with only one modality ($\Box$).

Finally, the parameters L and N allow for a "tuning" of the probability of satisfiability and of the hardness of random 3CNF formulas. Indeed, varying the L/N ratio, the plot of satisfiability percentages draws a transition from 100% satisfiability to 100% unsatisfiability Mitchell et al. (1992). The 50% crossover point always located around the fixed value $L/N \approx 4.30$. A precise experimentation of different approach on these benchmarks may be found in Giunchiglia et al. (2000). The conclusion is as follows: the mean and the median CPU times plots reveal an easy-hard-easy pattern always centred in the value $L/N \approx 4.30$. One can thus easily control the satisfiability-rate and the difficulty of an instance thanks to this generator.

CNF-KSP Benchmarks

We just presented one family of benchmarks which look like a propositional logic CNF formula in the construction, with modalities embedded in the clauses. This set was judged "not completely satisfactory" mainly because the Algorithm either generates a too easy formula for the current heavily-optimised solvers or because they generate a high rate of trivial or insignificant instances. Thus a new version of $3CNF_K$ benchmarks has been created Patel-Schneider and Sebastiani (2003).

The name we give in this thesis to this set is not really satisfactory but we decided to keep it for consistency with the articles we published on the subject. We discovered this set of

Algorithm 3.2: $3CNF_K$ Generator

Data: L the number of clauses, N the number of propositional variables, d the modal depth, p a probably with which any random atom is propositional
Result: A $3CNF_K$ formula randomly generated

```
1  begin
2      Function 3CNF(L, N, d, p)
3          for i from 1 to L do C_i := rand_clause(N, d, p)
4          return ⋀_{i=1}^{L} C_i
5      end
6
7      Function rand_clause(N, d, p)
8          for j from 1 to 3 do l_j := rand_lit(N, d, p)
9          return ⋁_{j=1}^{3} l_j
10     end
11
12     Function rand_lit(N, d, p)
13         φ := rand_atom(N, d, p)
14         if (flip_coin(0.5)) then return φ
15         else return ¬φ
16     end
17
18     Function rand_atom(N, d, p)
19         if ((d=0) or flip_coin(p)) then
20             return rand_propositional_atom(N)
21         end
22         else
23             C := rand_clause(N,d-1,p)
24             return □C
25         end
26     end
27 end
```

benchmarks thanks to the article presenting the solver K_SP, so we named it, at this moment, CNF-KSP to make the difference with the original 3CNF. But the real authors of this set are, as mentioned by the paper cited, Peter Patel-Schneider and Roberto Sebastiani.

The new generator is really similar to the first one and is presented in Algorithm 3.3. Unlike 3CNF_K, it allows to control the generated formula thanks to a probability distributions. Here is an example of the kind of formula that it can generate: $(\neg p2 \vee \Box(p4 \vee \Box(\neg p3 \vee p4))) \wedge (p1 \vee \neg(\Box(\neg p3 \vee \Box(p1 \vee \neg p2 \vee p3))))$. As we can see, despite the name being CNF-KSP, the formulas generated are far from being in Conjunctive Normal Form. The generator has two parameters to control the shape of formulas.

- The first parameter, C, is a list of list (*eg.* $[[0, 0, 1]]$) telling it how many disjuncts to put in each disjunction at each modal level. Each internal list represents a finite discrete

Algorithm 3.3: *CNF-KSP* Generator

Data: L the number of clauses, N the number of propositional variables, d the modal depth, C, is a list of list telling it how many disjuncts to put in each disjunction at each modal level, p, is a list of list of lists that controls the propositional/modal rate.

Result: A *CNF-KSP* formula randomly generated

```
1  begin
2      Function CNF-KSP(L, N, d, C, p)
3          for i from 1 to L do
4              repeat C_i := rand_clause(N, d, p) until is_new(C_i)
5          end
6          return ⋀_{i=1}^{L} C_i
7      end
8
9      Function rand_clause(N, d, C, p)
10         K := rand_length(d,C)
11         P := rand_propNum(d,p,K)
12         repeat
13             for j from 1 to P do l_j := rand_sign().rand_atom(N,0,C,p)
14             for j from P + 1 to K do l_j := rand_sign().rand_atom(N,d,C,p)
15             Cl := ⋁_{j=1}^{K} l_j
16         until no_repeated_atoms_in(Cl)
17         return Sort(Cl)
18     end
19
20     Function rand_atom(N, d, C, p)
21         if (d=0) then return rand_propositional_atom(N)
22         else
23             C := rand_clause(N,d-1,C,p)
24             return □C
25         end
26     end
27 end
```

probability distribution. For instance "$[0, 0, 1]$" says "0/1 of the disjunction have 1 disjunct, 0/1 have 2 disjuncts, and 1/1 have 3 disjunctions (fixed length 3)".

- The second parameter, p, is a list of list of lists (*eg.* $[[[], [], [0, 3, 3, 0]]]$) that controls the propositional/modal rate. The top-level elements are for each modal depth (here all the same), the second-level elements are for disjunctions with 1,2,3,... disjunctions (here only the third matters to simulate $3CNF_K$ as all disjunctions have three disjuncts in such case). In the old version ($3CNF_K$), with $p = 0.5$ is now represented by $[[[], [], [0, 3, 3, 0]]]$.

Because the generator eliminates duplicated atoms in a clause, it takes care to not disturb

the probabilities by first determining the "shape" of a clause and only then instantiating it with propositional variables. If a clause has repeated atoms, either propositional or modal, the instantiation is rejected and another instantiation of the shape is performed.

The MQBF Random QBF Benchmarks

The basic benchmark MQBF was first proposed for TANCS in Massacci (1999) and Massacci and Donini (2000). The intuition behind is to encode validity of Quantified Boolean Formula (QBF) into a satisfiability problem in modal logic K. In practice it works as follows: they generate a QBF formula with c clauses, an alternation depth equal to d with at most v variables per quantifier scope. As an example, if we set d=3 and v=2, we can generate a QBF formula looking like:

$$\forall v_{32} v_{31}, \exists v_{22} v_{21}, \forall v_{12} v_{11}, \exists v_{02} v_{01} cnf_{c-clauses}(v_{01} \dots v_{32}) \tag{3.1}$$

For each clause, they randomly generate k different variables (default 4) and each is negated with a probability of 50%. The first and third variable (if it exists) are existentially quantified, whereas the second and fourth variable are universally quantified. This aims at eliminating trivially unsatisfiable formulas Cadoli et al. (2002). The depth of each literal is randomly chosen from 1 to d. The QBF formula can then be translated into modal logic with different encodings:

- An optimization of Ladner's original translation Ladner (1977) that does not introduce new variables;
- A further optimized translation in which the formulas corresponding to the alternation depth are somewhat "compiled away";
- An optimized translation which is close to Schmidt-Schauß and Smolka's reduction of QBF validity into $\mathcal{ALC}$ satisfiability Schmidt-Schauß and Smolka (1991).

From those different translation functions and the different parameters settings, the MQBF family is in fact a collection of five classes, called **qbf**, **qbfL**, **qbfS**, **qbML** and **qbfMS**[4], for a total of 1016 formulas, of which 617 are known to be satisfiable and 399 are known to be unsatisfiable (due to at least one solver being able to solve the formula). Except **qbf** which is just a set of 56 satisfiable benchmark with 1 variable, the different values of the parameters for these categories of benchmarks are as follows: $m \in \{10, 20, 30, 40, 50\}$ denotes the number of clauses; $n \in \{4, 8, 16\}$ denotes the number of variables; $a \in \{4, 6\}$ denotes the alternation depth. For each triplet (a, n, m) we have 8 problems, so a family is composed of 240 problems.

3.3.2 Crafted Benchmarks

MQBF was the last family of randomly generated instances from the literature. The other families are now more in the *Crafted* category in a way that they were designed in sub-family in order to give an hard-time to the different approaches possible.

[4]the 'L' stands for Ladner's translator, the 'S' for the one of Schmidt-Schauß and Smolka

The Logic WorkBench (LWB) Benchmarks

The LWB family is subdivided into 18 classes Balsiger et al. (2000) (1008 formulas, half are satisfiable and half are unsatisfiable by construction of the benchmark classes). Basically the 18 classes should be considered as 9 classes which can be either satisfiable or unsatisfiable by construction. The Logic WorkBench were created to test modal logic solvers in modal logic K, KT and S4. We will not describe all the benchmark generators because they are all described in Balsiger et al. (2000), but let us for some classes of formula (the ones that we found interesting), list the following informations: "why is the formula satisfiable ?", "How is the formula hidden in order to make the problem harder to solve?" and finally a small definition of the formula.

1. **k_branch_p**. The branching formula as defined in Halpern and Moses (1992), plus a negation symbol in front and the additional sub-formulas $\neg\Box^n p_{n \ div \ 3+1}$ in order to make the formula satisfiable. We assume $n < 100$.

$$k\_branch\_p(n) := \neg(p_{100} \wedge \neg p_{101} \bigwedge_{i=0}^{n} (\Box^i(bdepth(n) \wedge det(n) \wedge branching(n)) \vee \neg\Box^n p_{n \ div \ 3+1}))$$

$$bdepth(n) := \bigwedge_{i=1}^{n+1} (p_{100+i} \rightarrow p_{99+i})$$

$$det(n) := \bigwedge_{i=0}^{n} (p_{100+i} \rightarrow (p_i \rightarrow \Box(p_{100+i} \rightarrow p_i)) \wedge (\neg p_i \rightarrow \Box(p_{100+i} \rightarrow \neg p_i)))$$

$$branching(n) := \bigwedge_{i=0}^{n-1} (p_{100+i} \wedge \neg p_{101+i} \rightarrow \Diamond(p_{101+i} \wedge \neg p_{102+i} \wedge p_{i+1}) \wedge \Diamond(p_{101+i} \wedge \neg p_{102+i} \wedge \neg p_{i+1}))$$

2. **k_branch_n**. The branching formula as defined in Halpern and Moses (1992).

$$k\_branch\_n(n) := \neg(p_{100} \wedge \neg p_{101} \bigwedge_{i=0}^{n} (\Box^i(bdepth(n) \wedge det(n) \wedge branching(n))))$$

$bdepth(n) :=$ as in **k_branch_p**
$det(n) :=$ as in **k_branch_p**
$branching(n) :=$ as in **k_branch_p**

3. **k_ph_p**. The pigeonhole formulas as defined in Haken (1985) but modalized. We assume $n < 100$. Some $\Box$ and $\Diamond$ are added to make it harder to find a solution.

$$k\_ph\_p(n) := \Diamond left(n) \rightarrow \Diamond right(n)$$

$$left(n) := \bigwedge_{i=1}^{n+1} (\bigvee_{j=1}^{n} (l(i,j)))$$

$$right(n) := \bigvee_{j=1}^{n} i_1 = 1, \ldots \mathrm{n{+}1}, i_2 = i_1 + 1, \ldots, \mathrm{n{+}1}(l(i_1,j) \wedge l(i_2,j))$$

$$l(i,j) := \begin{cases} \Box p_{100i+j} & \text{if } i < j \\ p_{100i+j} & \text{otherwise} \end{cases}$$

4. **k_ph_n**. The pigeonhole formulas with one missing conjunct on the right-hand side. We assume $n < 100$. Some $\Box$ and $\Diamond$ are added to make it harder to find a solution.

$$\begin{aligned} k\_ph\_p(n) &:= \Diamond left(n) \rightarrow \Diamond right(n) \\ left(n) &:= \text{as in } \mathbf{k\_ph\_p} \\ right(n) &:= \bigvee_{j=1..n, i_1=1..n+1, i_2=i_1+1..n+1} (l2(n,i_1,j) \wedge l2(n,i_2,j)) \\ l2(n,i,j) &:= \begin{cases} \neg l(i,j) & \text{if } i = j \text{ or } i = (2n)\text{div } 3{+}1 \\ l(i,j) & \text{otherwise} \end{cases} \end{aligned}$$

5. **k_poly_p**. The formula $(p_1 \leftrightarrow p_2) \vee (p_2 \leftrightarrow p_3) \vee \ldots \vee (p_{n-1} \leftrightarrow p_n) \vee (p_n \leftrightarrow p_1)$. If we have a polygon with n vertices, and all the vertices are either black or white, then two adjacent vertices have the same color. If n is odd, then this formula is satisfiable in CPL. Many $\Box$ and $\Diamond$ and superfluous subformulas are added to hide the problem and make it harder for the solver to find the solution.

$$\begin{aligned} k\_poly\_p(n) &:= \begin{cases} poly(3n+1) & \text{if n mod } 2 = 0 \\ poly(3n) & \text{otherwise} \end{cases} \\ poly(n) &:= \Box^{n+1} \bigwedge_{i=1}^{n+1} (p_i) \vee f(n,n) \vee \Box^{n+1} \bigwedge_{i=1}^{n+1} (\neg p_{2i}) \\ f(i,n) &:= \begin{cases} \bot & \text{if } i = 0 \\ \Diamond(f(i-1,n) \vee \Diamond^i (p_n \leftrightarrow p_1)) \vee \Box p_{i+2} & \text{i = n} \\ \Diamond(f(i-1,n) \vee \Diamond^i (p_n \leftrightarrow p_{i+1})) \vee \Box p_{i+2} & \text{otherwise} \end{cases} \end{aligned}$$

6. **k_poly_n**. As for **k_poly_p**, but for an even number of vertices. Many $\Box$ and $\Diamond$ and superfluous subformulas are added to hide the problem and make it harder for the solver to detect its unsatisfiability.

$$\begin{aligned} k\_poly\_p(n) &:= \begin{cases} poly(3n) & \text{if n mod } 2 = 0 \\ poly(3n+1) & \text{otherwise} \end{cases} \\ poly(n) &:= \text{as in } \mathbf{k\_poly\_p} \\ f(i,n) &:= \text{as in } \mathbf{k\_poly\_p} \end{aligned}$$

We could continue to enumerate all the different classes but it would be unnecessary, the *Craftedness* has been demonstrated with these 6/18 examples. The formulas for KT and S4 are, except few examples, the same 'kind' of formulas as in K. For example the class **kt_ph_p** is defined as $left(n) \rightarrow \Diamond right(n)$ with the same $left$, $right$ and l functions as in **k_ph_p**.

Now that we showed what is propositional logic, what are the modal logics, what are the state-of-the-art approaches to solve modal logics satisfiability problems and what are the standard benchmarks to evaluate the approach, it is time to present our contributions in this domain. To avoid too many repetitions, We made all the time run all the solvers on the benchmarks represented in InToHyLo format Hoffmann (2010) and we check the Kripke models returned by our solvers (and the other solvers when possible) with the checker `mdk-verifier` Lagniez et al. (2016b) that we developed for the occasion and which is accessible at the following address : `http://www.cril.univ-artois.fr/~montmirail/mdk-verifier/`.

Part II
Contributions

Chapter 1

The NP Modal Logics Satisfiability Problems

Contents

> "Modal Logics can be at the heart of Artificial Intelligence"
>
> *Nicolas Szczepanski (2012)*

SAT solvers have been used successfully to solve a wide range of problems. We report in our contributions our own positive experience in deciding the satisfiability of modal logic formulas. We first focus on the modal logics $\mathcal{L}$ which have the particularity to be NP-complete. Indeed, if one logic is NP-complete, then for sure it exists a polynomial reduction from $\mathcal{L}$ to CPL where the problem can be decided using a SAT solver.

1.1 How To Deal With Modal Logic Formulas in K⋆5

The Kripke structures that can be models of formulas have a size (with respect to the number of worlds in the structure) which is linear in the size of the input due to the polysize model property Bezhanishvili and Marx (2003). The modal logic K⋆5 stands for all modal logics containing the axiom 5. The intuition behind the translation that we propose is as follows: we know that the maximum number of worlds to consider to decide the satisfiability of a formula in modal logic K⋆5 is linear in the size in the formula (proof in Halpern and Rêgo (2007)). However, even if it is linear, such bound (which is $|\phi|$) can hardly be used in practice if one wants to translate a K⋆5 formula into propositional logic: it may lead to CNF too large to be handled by a SAT solver on a standard personal computer. Thus, the first thing we want to know is: can we have a smaller upper-bound on the size of the Kripke models? Because the lower will be the bound, the smaller will be the generated CNF and thus the more efficient will be a SAT-based approach.

1.1.1 A New Upper-Bound For the Size Of The Kripke Models

The idea of our new upper-bound is as follows: if a formula has no diamond, then the upper-bound must be one. Said otherwise, the upper-bound should be linear in the number of diamonds in the case of NP-complete modal logics. So this is the kind of function we want to create. We propose the following upper-bound that we call the diamond-degree.

Definition 42 (Diamond-Degree)

The diamond degree of $\phi \in \mathcal{L}$, noted $\mathtt{dd}(\phi)$, is defined recursively, as follows:

$$\begin{aligned}
\mathtt{dd}(\phi) &= \mathtt{dd}'(\mathrm{nnf}(\phi)) \\
\mathtt{dd}'(\top) &= \mathtt{dd}'(\neg\top) = 0 \\
\mathtt{dd}'(p) &= \mathtt{dd}'(\neg p) = 0 \\
\mathtt{dd}'(\phi \wedge \psi) &= \mathtt{dd}'(\phi) + \mathtt{dd}'(\psi) \\
\mathtt{dd}'(\phi \vee \psi) &= \max(\mathtt{dd}'(\phi), \mathtt{dd}'(\psi)) \\
\mathtt{dd}'(\Box\phi) &= \mathtt{dd}'(\phi) \\
\mathtt{dd}'(\Diamond\phi) &= 1 + \mathtt{dd}'(\phi)
\end{aligned}$$

The theorem that we want to prove to have a valid upper-bound is thus:

Theorem 8

If for a formula ϕ, there is no Kripke model of a size bounded by $\mathtt{dd}(\phi) + 1$, then ϕ is unsatisfiable in $\{K5, K45, KB5, S5\}$. And if there is no Kripke model of size bounded by $\mathtt{dd}(\phi) + 2$ then ϕ is unsatisfiable in $\{KD5, KD45\}$.

We recall here the proof that has been proposed in Caridroit et al. (2017a) for modal logic S5. Let ϕ be a formula in NNF and let $\mathrm{sub}(\phi)$ be the set of all sub-formulas of ϕ. A tableau for ϕ is the smallest non-empty set $T_{S5} = \{s_0, s_1, \ldots, s_n\}$ such that each $s_i \in T_{S5}$ is a subset of $\mathrm{sub}(\phi)$ and $\phi \in s_0$. In addition, each set $s_i \in T_{S5}$ satisfies the following conditions:

1. $\neg\top \notin s$.
2. if $p \in s$ then $\neg p \notin s$.
3. if $\neg p \in s$ then $p \notin s$.
4. if $\psi_1 \wedge \psi_2 \in s$ then $\psi_1 \in s$ and $\psi_2 \in s$.
5. if $\psi_1 \vee \psi_2 \in s$ then $\psi_1 \in s$ or $\psi_2 \in s$.
6. if $\Box\psi_1 \in s$ then $\forall s' \in T_{S5}$ we have $\psi_1 \in s'$.

7. if $\Diamond\psi_1 \in s$ then $\exists s' \in T_{S5}$ s.t. $\psi_1 \in s$.

Lemma 1

Let ϕ be in NNF. The number of elements of the set T_{S5} created by constructing its tableau is bounded by $\mathtt{dd}'(\phi) + 1$.

Proof. Let $\psi \in \text{sub}(\phi)$. Let $g(\psi)$ be the number of sets s added to T_{S5} because of ψ. That is, $g(\psi)$ is the number of times the condition involving operator $\Diamond$ is triggered for sub-formulas of ψ. We show that, for all $\psi \in \text{sub}(\phi)$ we have $g(\psi) \leq \mathtt{dd}'(\psi)$. We do so by induction on the structure of ψ.

Induction base. We consider four cases: (1) $\psi = \top$, (2) $\psi = \neg\top$, (3) $\psi = p$ and (4) $\psi = \neg p$. In all cases, the condition involving $\Diamond$ will never be triggered for formulas in $\text{sub}(\psi)$. Then $g(\psi) = 0 \leq \mathtt{dd}'(\psi)$.

Induction step. We consider four cases:

1. $\psi = \psi_1 \wedge \psi_2$. Assume $\psi \in s$, for some $s \in T_{S5}$. In this case, the algorithm adds ψ_1 and ψ_2 to s. Therefore, $g(\psi)$ is bounded by $g(\psi_1) + g(\psi_2)$. The latter is bounded by $\mathtt{dd}'(\psi_1) + \mathtt{dd}'(\psi_2)$ (by the induction hypothesis). Then $g(\psi) \leq \mathtt{dd}'(\psi)$.

2. $\psi = \psi_1 \vee \psi_2$. Assume $\psi \in s$, for some $s \in T_{S5}$. In this case, the algorithm adds either ψ_1 or ψ_2 to s. Therefore, $g(\phi)$ is bounded by $\max(g(\psi_1), g(\psi_2))$. The latter is bounded by $\max(\mathtt{dd}'(\psi_1), \mathtt{dd}'(\psi_2))$ (by the induction hypothesis). Then $g(\psi) \leq \mathtt{dd}'(\psi)$.

3. $\psi = \Box\psi_1$. Assume $\psi \in s$, for some $s \in T_{S5}$. In this case, the algorithm adds ψ_1 to all $s' \in T_{S5}$. Therefore, $g(\psi)$ is bounded by $g(\psi_1)$. The latter is bounded by $\mathtt{dd}'(\psi_1)$ (by the induction hypothesis). Then $g(\psi) \leq \mathtt{dd}'(\psi)$.

4. $\psi = \Diamond\psi_1$. Assume $\psi \in s$, for some $s \in T_{S5}$. In this case, if there is no s' containing ψ_1 then the algorithm adds a new s'' to T_{S5} and adds ψ_1 to s''. Therefore, $g(\psi)$ is bounded by $1 + g(\psi_1)$. The latter is bounded by $1 + \mathtt{dd}'(\psi_1)$ (by the induction hypothesis). Then $g(\psi) \leq \mathtt{dd}'(\psi)$.

Therefore, we have $|T_{S5}| = 1 + g(\phi) \leq 1 + \mathtt{dd}'(\phi)$. q.e.d □

Thus, for any $\phi \in \mathcal{L}$, each s_i of the tableau T_{S5} corresponds to a $w_i \in W$ in the S5-model, $|T_{S5}| \leq \mathtt{dd}(\phi) + 1$ means that the number of worlds in the S5-model is bounded by $\mathtt{dd}(\phi) + 1$.

For all the other NP modal logics, we will perform also a Tableau method, but this time, the Tableau will be for K5 and we will show that by adding the different axioms, if they can add worlds in the Tableau. Let ϕ be a formula in NNF and let $\text{sub}(\phi)$ be the set of all sub-formulas of ϕ. Basically after defining a K5-Tableau, it will have to respect the following Lemma:

Lemma 2

Let ϕ be in NNF. The number of elements of the set T created by constructing its tableau is bounded by $\mathtt{dd}'(\phi) + 1$.

But before we proof the Lemma 2, we need to define the K5-Tableau and explain the intuition.

Definition 43 (A K5-Tableau)

A K5-Tableau for ϕ is a non-empty set $T = \{s_0, s_1, \ldots, s_n\}$ such that each $s_i \in T$ is a subset of sub(ϕ) and $\phi \in s_0$. We also need to keep the relations (R) between s_0 and the other worlds, we know that $\forall i \neq 0, \forall j \neq 0\ (s_i, s_j) \in R$ due to the euclidean axiom (5). In addition, each set $s_i \in T$ satisfies the following conditions:

1. $\neg\top \notin s_i$.
2. if $p \in s_i$ then $\neg p \notin s_i$.
3. if $\neg p \in s_i$ then $p \notin s_i$.
4. if $\psi_1 \wedge \psi_2 \in s_i$ then $\psi_1 \in s_i$ and $\psi_2 \in s_i$.
5. if $\psi_1 \vee \psi_2 \in s_i$ then $\psi_1 \in s_i$ or $\psi_2 \in s_i$.
6. if $\Box\psi_1 \in s_i$ and $i \neq j \neq 0$ then $\forall s_j \in T$ we have $\psi_1 \in s_j$ and $(s_i, s_j) \in R$.
7. if $\Box\psi_1 \in s_0$ then $\forall s_i \in T$ s.t. $(s_0, s_i) \in R$ we have $\psi_1 \in s_i$.
8. if $\Diamond\psi_1 \in s_i$ and $i \neq 0$ then $\exists s_j \in T$ s.t. $\psi_1 \in s_j$ and $(s_i, s_j) \in R$.
9. if $\Diamond\psi_1 \in s_0$ then $\exists s_i \in T$ s.t. $(s_0, s_i) \in R$ and $\psi_1 \in s_i$.

We need to know also that this K5-Tableau is sound and complete. For the intuition about why this Tableau is sound and complete for modal logic K5, one needs to have a look at the shape of a K5-Structure. An example of such a structure is given in Figure 1.1.

As we can see, the structure can be split into two sub-figures: on one side there is w_0 alone and on the other side there is somehow a sub-KT5-structure. Except w_0 that can access (or not) any other worlds, all the other relations are true. The sub-structure obtained when we remove w_0 is necessarily an equivalence relation due to the euclideanity from axiom (5).

One can use a tool such as Alloy Jackson (2006) to display examples of K5-Structure: an Alloy model is given in appendix of this thesis. We can see on Figure 1.2 that when asked with 3 worlds, there is only three possible kind of K5-structures.

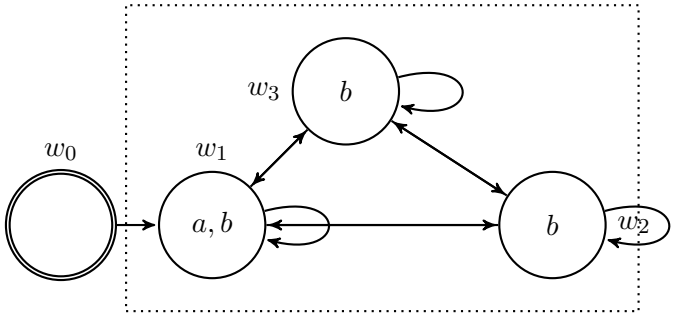


Figure 1.1: Example of a K5-Structure

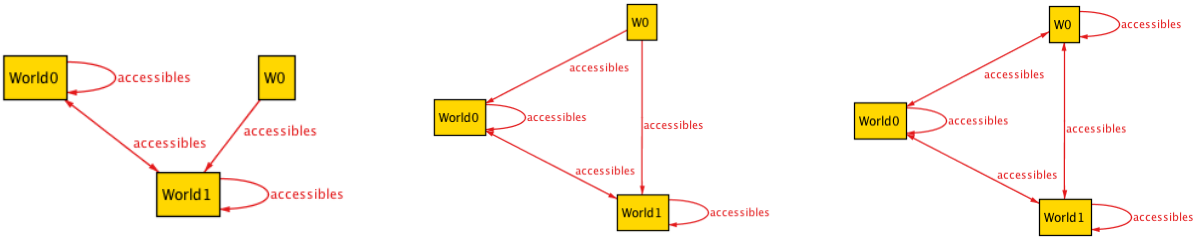


Figure 1.2: Examples from Alloy when asked K5-structures with 3 worlds

Theorem 9

The K5-Tableau proposed in Def.43 is sound and complete.

Proof. The Tableau proposed in Def. 43 is very similar to the well known K-Tableau showed sound and complete by R.Goré (1999). We just split the rules for the modalities $\{\Diamond, \Box\}$ into 2 parts due to axiom (5). Indeed, because we know due to the shape of K5-structures, that for all $s_i, s_j \neq s_0$ $(s_i, s_j) \in R$. q.e.d $\Box$

We need now the proof of the Lemma 2.

Proof. Let $\psi \in \text{sub}(\phi)$. Let $g(\psi)$ be the number of sets s added to T because of ψ. That is, $g(\psi)$ is the number of times the conditions involving operator $\Diamond$ are triggered for sub-formulas of ψ. We show that, for all $\psi \in \text{sub}(\phi)$ we have $g(\psi) \leq \texttt{dd}'(\psi)$. We do so by induction on the structure of ψ.

Induction base. We consider four cases: (1) $\psi = \top$, (2) $\psi = \neg\top$, (3) $\psi = p$ and (4) $\psi = \neg p$. In all cases, the conditions involving $\Diamond$ are not triggered for formulas in sub(ψ). Then $g(\psi) = 0 \leq \texttt{dd}'(\psi)$.

Induction step. We consider six cases:

1. $\psi = \psi_1 \wedge \psi_2$. Assume $\psi \in s$, for some $s \in T$. In this case, the algorithm adds ψ_1 and ψ_2 to s. Therefore, $g(\psi)$ is bounded by $g(\psi_1) + g(\psi_2)$. The latter is bounded by $\texttt{dd}'(\psi_1) + \texttt{dd}'(\psi_2)$ (by the induction hypothesis). Then $g(\psi) \leq \texttt{dd}'(\psi)$.

2. $\psi = \psi_1 \vee \psi_2$. Assume $\psi \in s$, for some $s \in T$. In this case, the algorithm adds either ψ_1 or ψ_2 to s. Therefore, $g(\phi)$ is bounded by $\max(g(\psi_1), g(\psi_2))$. The latter is bounded by $\max(\mathtt{dd}'(\psi_1), \mathtt{dd}'(\psi_2))$ (by the induction hypothesis). Then $g(\psi) \leq \mathtt{dd}'(\psi)$.

3. $\psi = \Box\psi_1$. Assume $\psi \in s_i$, for some $s_i \in T$ s.t $i \neq 0$. In this case, the algorithm adds ψ_1 to all $s_j \in T$. Therefore, $g(\psi)$ is bounded by $g(\psi_1)$. The latter is bounded by $\mathtt{dd}'(\psi_1)$ (by the induction hypothesis). Then $g(\psi) \leq \mathtt{dd}'(\psi)$.

4. $\psi = \Box\psi_1$. Assume $\psi \in s_0$. In this case, the algorithm adds ψ_1 to all $s_i \in T$ s.t $(s_0, s_i) \in R$. Therefore, $g(\psi)$ is bounded by $g(\psi_1)$. The latter is bounded by $\mathtt{dd}'(\psi_1)$ (by the induction hypothesis). Then $g(\psi) \leq \mathtt{dd}'(\psi)$.

5. $\psi = \Diamond\psi_1$. Assume $\psi \in s_i$, for some $s_i \in T$ s.t. $i \neq 0$. In this case, if there is no s_j containing ψ_1 then the algorithm adds a new s_k to T and adds ψ_1 to s_k. Therefore, $g(\psi)$ is bounded by $1 + g(\psi_1)$. The latter is bounded by $1 + \mathtt{dd}'(\psi_1)$ (by the induction hypothesis). Then $g(\psi) \leq \mathtt{dd}'(\psi)$.

6. $\psi = \Diamond\psi_1$. Assume $\psi \in s_0$. In this case, if there is no s_j containing ψ_1 then the algorithm adds a new s_k to T, adds (s_0, s_k) to R and adds ψ_1 to s_k. Therefore, $g(\psi)$ is bounded by $1 + g(\psi_1)$. The latter is bounded by $1 + \mathtt{dd}'(\psi_1)$ (by the induction hypothesis). Then $g(\psi) \leq \mathtt{dd}'(\psi)$.

Therefore, we have $|T| = 1 + g(\phi) \leq 1 + \mathtt{dd}'(\phi)$. q.e.d □

Thus, for any $\phi \in \mathcal{L}$, each s_i of the tableau T corresponds to a $w_i \in W$ in the K5-model, $|T| \leq \mathtt{dd}(\phi) + 1$ means that the number of worlds in the K5-model is bounded by $\mathtt{dd}(\phi) + 1$.

To show that this proof can be extended to any other NP-complete modal logics, we need to show that the different following rules can be added to a K5 Tableau and they will not create an additional world in the model.

$$(T)\ \frac{\Box\phi \in s_i}{\phi \in s_i} \qquad (4)\ \frac{(s_i, s_j) \in R \text{ and } \Box\phi \in s_i}{\Box\phi \in s_j}$$

$$(B)\ \frac{\Box\phi \in s_i \text{ and } (s_i, s_j) \in R}{(s_j, s_i) \in R} \qquad (D)\ \frac{\Box\phi \in s_i}{\Diamond\phi \in s_i}$$

Lemma 3

For any logic K⋆5 in $\{K5, K45, KB5, KT5\}$, the number of worlds in the K⋆5-model for a formula ϕ satisfiable is bounded by $\mathtt{dd}(\phi) + 1$

Proof. We will iterate with all the rules that we can add to our K5 Tableau and we will show that none of them is creating additional worlds.

- (T). If we add (T) then we obtain a KT5 Tableau, and then it is already proved that $\mathtt{dd}(\phi) + 1$ is an upper-bound in the proof of Lemma 1.

- (4). If we have $\Box\phi \in s_i$ and $(s_i, s_j) \in R$ then we have $\Box\phi \in s_j$.
- (B). If we have $\Box\phi \in s_i$ and $(s_i, s_j) \in R$ then we need to add the symmetrical relation $(s_j, s_i) \in R$ and ϕ must be true in s_j because of that.

q.e.d □

Lemma 4

For any logic K⋆5 in $\{KD5, KD45\}$, the number of worlds in the K⋆5-model for a formula ϕ satisfiable is bounded by $\mathtt{dd}(\phi) + 2$

Proof. We will show that if we add to our K5 Tableau the axiom (D), then it can create only one additional world.

- (D). If there exists s_j such that $(s_i, s_j) \in R$ and that ϕ is in s_j then $\Diamond\phi$ is satisfied in s_i. But if it does not exist such a s_j, Tableau needs to create a new s_j, adds $(s_i, s_j) \in R$ and adds ϕ in s_j.

q.e.d □

Now that we have an upper-bound for the size of the Kripke models, the next thing we want to do is to simplify, as much as possible, the input formulas. That way, when we compute the diamond-degree of the simplified formula, it is smaller than the one of the original formula.

1.1.2 A Set Of Simplifications For K⋆5

It is well known that, in modal logic KT5, there exists only two distinct modalities □ and ◇. Said otherwise, it is known that:

Theorem 10 (Garson (2016))

In modal logic KT5, for $\circ \in \{\Diamond, \Box\}$, we have the following equivalences:

$$\circ \circ \circ \circ \circ \circ \Box\phi \leftrightarrow \Box\phi$$
$$\circ \circ \circ \circ \circ \circ \Diamond\phi \leftrightarrow \Diamond\phi$$

Here, we want to generalize this result, by stating that every modal logic formula in NNF containing the axiom (5) has at most 6 different modalities, *viz.* $\Diamond, \Box, \Box\Box, \Diamond\Diamond, \Box\Diamond$ and $\Diamond\Box$.

Theorem 11 (Drop the Middle Theorem 4.23 in Chellas (1980))

In modal logic K⋆5, for $\circ \in \{\Diamond, \Box\}$, we have the following equivalences:

$$\Box \circ \circ \circ \Box\phi \leftrightarrow \Box\Box\phi$$
$$\Box \circ \circ \circ \Diamond\phi \leftrightarrow \Box\Diamond\phi$$
$$\Diamond \circ \circ \circ \Box\phi \leftrightarrow \Diamond\Box\phi$$
$$\Diamond \circ \circ \circ \Diamond\phi \leftrightarrow \Diamond\Diamond\phi$$

We now have the simplifications for any logic containing axiom (5) and we know that if we add the axiom (T), then we obtain the modal logic KT5 which can be even further simplified. The question that may arise now is: what simplification are applicable according to which axioms have been added. For that, we propose the following theorems:

Theorem 12

In modal logic KD5, we have the following equivalences:

(1) $\Box\Box\phi \leftrightarrow \Diamond\Box\phi$ (2) $\Diamond\Diamond\phi \leftrightarrow \Box\Diamond\phi$

Proof. Let us prove (1) and (2) by using the axioms (D), (5) and the Theorem 11.

(1) $\rightarrow$ We have $\Box\Box\phi$. By axiom (D), we know that $\Box\Box\phi \rightarrow \Diamond\Box\phi$.

(1) $\leftarrow$ We have $\Diamond\Box\phi$. By axiom (5), we have $\Box\Diamond\Box\phi$. By Theorem 11, we have $\Diamond\Box\phi \rightarrow \Box\Box\phi$.

(2) $\rightarrow$ We have $\Diamond\Diamond\phi$. By axiom (5), we have $\Box\Diamond\Diamond\phi$. By Theorem 11, we have $\Diamond\Diamond\phi \rightarrow \Box\Diamond\phi$.

(2) $\leftarrow$ We have $\Box\Diamond\phi$. By axiom (D), we have $\Diamond\Diamond\phi$.

q.e.d □

Because the modal logic KD5 contains the axiom (5), we can also apply the simplifications presented in Theorem 11 to further simplify the formulas.

Theorem 13

In modal logic K45, we have the following equivalences:

(1) $\Box\phi \leftrightarrow \Box\Box\phi$ (2) $\Diamond\phi \leftrightarrow \Diamond\Diamond\phi$

Proof. Let us prove (1) and (2) by using the axioms (4), (5) and the Theorem 11.

(1) → We want to prove $\Box\phi \rightarrow \Box\Box\phi$, we already have it from axiom (4) ($\psi \rightarrow \Box\psi$).

(1) ← We have $\Box\Box\phi \rightarrow \Box\Box\phi$ tautologically. Then by adding a diamond, we have $\Diamond\Box\Box\phi \rightarrow \Diamond\Box\Box\phi$. Then from the contraposition of axiom (5) ($\Diamond\Box\phi \rightarrow \Box\phi$), we have $\Box\Box\phi \rightarrow \Diamond\Box\Box\phi$. Then from Theorem 11, we have $\Box\Box\phi \rightarrow \Box\phi$.

(2) → We have $\Diamond\phi$. Then we have $\Diamond\phi \rightarrow \Box\Diamond\phi$ from (axiom 5). Then by adding a diamond, we have $\Diamond\Diamond\phi \rightarrow \Diamond\Box\Diamond\phi$. Then the contrapositive of axiom (4), we have $\Diamond\phi \rightarrow \Diamond\Box\Diamond\phi$. Then by Theorem 11, we have $\Diamond\phi \rightarrow \Diamond\Diamond\phi$.

(2) ← We have $\Diamond\phi$ directly from the contrapositive of axiom (4).

q.e.d □

From this theorem, we know that each K45 formula has at most 4 different modalities, *viz.* $\Diamond, \Box, \Box\Diamond$ and $\Diamond\Box$.

Theorem 14

In modal logic KB5, we have the following equivalences:

(1) $\Box\phi \leftrightarrow \Box\Box\phi$ (2) $\Diamond\phi \leftrightarrow \Diamond\Diamond\phi$

Proof. KB5 is just another name for the modal logic KB45. Thus every simplifications on NP modal logics implied by the axiom (4) can also be applied with the axiom (B) (see Table 2.2). q.e.d □

Obviously, one can combine axioms to simplify even further the formulas. For example, the modal logic KD45 can be easily simplified as follows:

Theorem 15

In modal logic KD45, we have the following equivalences:

(1) $\Box\phi \leftrightarrow \Diamond\Box\phi$ (2) $\Diamond\phi \leftrightarrow \Box\Diamond\phi$

Proof. Let us prove (1) and (2) by using the Theorems 13 and 12. (1) We know that $\Diamond\Box\phi \leftrightarrow \Box\Box\phi$ by Theorem 12. Then we know that $\Diamond\Box\phi \leftrightarrow \Box\phi$ by Theorem 13. (2) We know

that $\Box\Diamond\phi \leftrightarrow \Diamond\Diamond\phi$ by Theorem 12. Then we know that $\Box\Diamond\phi \leftrightarrow \Diamond\phi$ by Theorem 13. q.e.d □

Thus, we know that in modal logic KD45, we have, as in KT5, only two distinct modalities $\Box$ and $\Diamond$. From now on, we know that any modal logic which contains the axiom (5) can be simplified in one way or another in order to reduce the number of modalities in the formula. This is extremely helpful for the diamond-degree which depends only on the number of diamonds in the formula, the fewer there are, the smaller will be the upper-bound for the SAT translation that we propose. There exists a preprocessing, which is validity-preserving in any modal logic containing axiom (K), called the *Box Lifting* and is defined as follows:

Theorem 16 (Box Lifting Sebastiani and Vescovi (2009))

(1) $(\Box\phi \wedge \Box\psi) \leftrightarrow \Box(\phi \wedge \psi)$ (2) $(\Diamond\phi \vee \Diamond\psi) \leftrightarrow \Diamond(\phi \vee \psi)$

It allows to reduce the number of modalities in the formula and it is quite straightforward to understand. For (1), it says that ϕ must be true in all possible worlds, and ψ must be true in all possible worlds. Thus, both must be true in all possible worlds. Similar for (2).

For the NP modal logics, one can go even further in the simplifications. We thus propose the following theorem to split the modalities when the axiom (5) is activated.

Theorem 17 ($\star$5-Lifting)

For any formula ϕ and ψ, in a modal logic which contains the axiom (5), we have the following:

$$(\circ_1\phi \bigoplus \circ_1 \circ_2 \psi) \leftrightarrow \circ_1(\phi \bigoplus \circ_2\psi)$$

for $\circ_1, \circ_2 \in \{\Diamond, \Box\}$ and $\bigoplus \in \{\wedge, \vee\}$.

Proof. This theorem has many cases, let us list them and start proving them one by one.

1. $(\Box\phi \wedge \Box\Box\psi) \leftrightarrow \Box(\phi \wedge \Box\psi)$
2. $(\Box\phi \wedge \Box\Diamond\psi) \leftrightarrow \Box(\phi \wedge \Diamond\psi)$
3. $(\Box\phi \vee \Box\Box\psi) \leftrightarrow \Box(\phi \vee \Box\psi)$
4. $(\Box\phi \vee \Box\Diamond\psi) \leftrightarrow \Box(\phi \vee \Diamond\psi)$
5. $(\Diamond\phi \wedge \Diamond\Box\psi) \leftrightarrow \Diamond(\phi \wedge \Box\psi)$
6. $(\Diamond\phi \wedge \Diamond\Diamond\psi) \leftrightarrow \Diamond(\phi \wedge \Diamond\psi)$
7. $(\Diamond\phi \vee \Diamond\Box\psi) \leftrightarrow \Diamond(\phi \vee \Box\psi)$
8. $(\Diamond\phi \vee \Diamond\Diamond\psi) \leftrightarrow \Diamond(\phi \vee \Diamond\psi)$

The cases (1), (2), (7) and (8) are straightforward from Theorem 16. The cases (3) and (4) can be obtained from (5) and (6) using the duality of box and diamond.
(5) left to right: $(\Diamond\phi\wedge\Diamond\Box\psi) \rightarrow \Diamond(\phi\wedge\Box\psi)$. We know from (Chellas 1980, p.141) that $(\Diamond\chi\wedge\Diamond\delta) \rightarrow \Diamond(\Diamond\chi\wedge\delta)$ is a theorem in modal logic K5. Thus we know that we have $(\Diamond\phi\wedge\Diamond\Box\psi) \rightarrow \Diamond(\Diamond\phi\wedge\Box\psi)$. From this point, we know that if $\exists\mathcal{K} = \langle W, R, V\rangle$, such that $\langle\mathcal{K}, w\rangle \models (\Diamond\phi \wedge \Diamond\Box\psi)$. There is w',

w'' in $R(w)$ s.t. $\langle \mathcal{K}, w' \rangle \models \phi$ and $\langle \mathcal{K}, w'' \rangle \models \Box\psi$. By axiom 5 and thus the euclideanity, we have that w' in $R(w'')$ and also that w'' in $R(w')$. Also by axiom 5, we have that all v'' in $R(w'')$ is also in $R(w')$ and all v' in $R(w')$ is in $R(w'')$. Then we have $\langle \mathcal{K}, w' \rangle \models \Box\psi$. Thus we have $\langle \mathcal{K}, w \rangle \models \Diamond(\phi \wedge \Box\psi)$.

(5) right to left: $\Diamond(\phi \wedge \Box\psi) \rightarrow (\Diamond\phi \wedge \Diamond\Box\psi)$.
If $\exists \mathcal{K} = \langle W, R, V \rangle$, such that $\langle \mathcal{K}, w_0 \rangle \models \Diamond(\phi \wedge \Box\psi)$. So it exists w such that $(w_0, w) \in R$ then $\langle \mathcal{K}, w \rangle \models (\phi \wedge \Box\psi)$. Then we have $\langle \mathcal{K}, w \rangle \models \phi$ and $\langle \mathcal{K}, w \rangle \models \Box\psi$. Then $\forall w'$, s.t. $(w, w') \in R$, we have $\langle \mathcal{K}, w' \rangle \models \psi$. But, because we have axiom (5), we know that $(w_0, w') \in R$, thus we have $\langle \mathcal{K}, w_0 \rangle \models \Diamond\Box\psi$. Finally, we thus know that we have $\langle \mathcal{K}, w_0 \rangle \models (\Diamond\phi \wedge \Diamond\Box\psi)$.

(6) left to right: $(\Diamond\phi \wedge \Diamond\Diamond\psi) \rightarrow \Diamond(\phi \wedge \Diamond\psi)$.
We know from the axiom (5) that we have $\Diamond\Diamond\phi \rightarrow \Box\Diamond\Diamond\phi$. From the Theorem 11, we can thus obtain $\Diamond\Diamond\phi \rightarrow \Box\Diamond\phi$. We then have (i) $\Diamond\Box\phi \rightarrow \Box\Box\phi$. We have $(\Diamond\phi \wedge \Box(\neg\phi \vee \Box\neg\psi)) \rightarrow \Diamond\Box\neg\psi$ by principles of modal logic K. From (i) we obtain $(\Diamond\phi \wedge \Box(\neg\phi \vee \Box\neg\psi)) \rightarrow \Box\Box\neg\psi$. From that we get $(\Diamond\phi \wedge \Diamond\Diamond\psi) \rightarrow \Diamond(\phi \wedge \Diamond\psi)$ by propositional reasoning.

(6) right to left: $\Diamond(\phi \wedge \Diamond\psi) \rightarrow (\Diamond\phi \wedge \Diamond\Diamond\psi)$.
We know from the axiom (5) that $(\Diamond\psi \rightarrow \Box\Diamond\psi)$. By propositional reasoning, we know that $\phi \wedge \Diamond\psi \rightarrow \phi \wedge \Box\Diamond\psi$. Then we know that $\Diamond(\phi \wedge \Diamond\psi) \rightarrow \Diamond(\phi \wedge \Box\Diamond\psi)$ Then by distributing, we have $\Diamond(\phi \wedge \Diamond\psi) \rightarrow (\Diamond\phi \wedge \Diamond\Box\Diamond\psi)$ Then by Theorem 11, we have $\Diamond(\phi \wedge \Diamond\psi) \rightarrow (\Diamond\phi \wedge \Diamond\Diamond\psi)$

q.e.d $\square$

Now we have all the components to propose a polynomial reduction from modal logic K$\star$5 satisfiability problems into a satisfiability problem in CPL (SAT problem).

1.2 A SAT Translation Of The Problems

Indeed, the main contribution here is to demonstrate a SAT encoding of these satisfiability problems and to see the practical efficiency of the proposed approach. To avoid a polynomial blow-up and in order to obtain a CNF as small as possible, all the simplifications presented before are applied (when possible) to ϕ before translating ϕ into propositional logic.

1.2.1 Translation function 'tr'

Definition 44 (Translation function 'tr')

Let ϕ be a formula for which we want to decide the satisfiability in K$\star$5.

$$\mathrm{tr}(\phi, n) = \mathrm{tr}'(\mathrm{nnf}(\phi), 0, n)$$
$$\mathrm{tr}'(\top, i, n) = \top \quad \mathrm{tr}'(\neg\top, i, n) = \neg\top$$
$$\mathrm{tr}'(p, i, n) = p_i \quad \mathrm{tr}'(\neg p, i, n) = \neg p_i$$
$$\mathrm{tr}'((\phi \wedge \delta), i, n) = \mathrm{tr}'(\phi, i, n) \wedge \mathrm{tr}'(\delta, i, n)$$
$$\mathrm{tr}'((\phi \vee \delta), i, n) = \mathrm{tr}'(\phi, i, n) \vee \mathrm{tr}'(\delta, i, n)$$
$$\mathrm{tr}'(\Box\phi, i, n) = \begin{cases} (\neg R_{i,0} \vee \mathrm{tr}'(\phi, 0, n)) \wedge \bigwedge_{j=1}^{n} (\mathrm{tr}'(\phi, j, n)) & (i \neq 0) \\ (\neg R_{0,0} \vee \mathrm{tr}'(\phi, 0, n)) \wedge \bigwedge_{j=1}^{n} (\neg R_{0,j} \vee \mathrm{tr}'(\phi, j, n)) & \textit{otherwise} \end{cases}$$
$$\mathrm{tr}'(\Diamond\phi, i, n) = \begin{cases} (R_{i,0} \wedge \mathrm{tr}'(\phi, 0, n)) \vee \bigvee_{j=1}^{n} (\mathrm{tr}'(\phi, j, n)) & (i \neq 0) \\ (R_{0,0} \wedge \mathrm{tr}'(\phi, 0, n)) \vee \bigvee_{j=1}^{n} (R_{0,j} \wedge \mathrm{tr}'(\phi, j, n)) & \textit{otherwise} \end{cases}$$

The translation adds fresh Boolean variables p_i to the formula, denoting the truth value of p in the world w_i, variables $R_{0,i}$ denoting the truth value of w_i is accessible from w_0 and variables $R_{i,0}$ denoting the truth value of w_0 is accessible from w_i. In such function, the i parameter represents the index of the world. This translation is different from the one proposed in Caridroit et al. (2017a) which deals only with modal logic KT5. Here, thanks to the variables $R_{i,j}$ which starts from (resp. ends at) w_0, we are able to represents the different K$\star$5 Structures.

We need to know how we can add the translation of the axioms $\{(B), (D), (4), (T)\}$ in order to be able to solve all the modal logics K$\star$5. One simple technique to deal with different modal logics is to include the constraints of the different axioms considered to the CNF. This simple extension amount to overload the translation function for the different axioms and to append to the translation the constraints of the axioms considered.

The function is defined over a Negative Normal Form (NNF) formula for sake of simplicity. Note that the result of the translation is not in CNF. As such, a classical translation into CNF such as Tseitin (1983) is needed to use a SAT oracle.

Definition 45 (Translation of axioms)

$$\mathrm{tr}((T),n) = \bigwedge_{j=0}^{n}(R_{0,j} \wedge R_{j,0}) \qquad \mathrm{tr}((D),n) = \bigvee_{j=0}^{n}(R_{0,j})$$
$$\mathrm{tr}((B),n) = \bigwedge_{j=0}^{n}(\neg R_{0,j} \vee R_{j,0}) \qquad \mathrm{tr}((4),n) = \bigwedge_{j=0}^{n}\bigwedge_{k=0}^{n}(\neg R_{0,j} \vee \neg R_{j,k} \vee R_{0,k})$$

Proof. The translation of each axioms came from the relations in First Order Logic, presented by H.Sahlqvist (1975) in 1973 plus the implication of what would be the result if we also add the translation of the Euclideanity. q.e.d □

Theorem 18

- ϕ is satisfiable in modal logic K5 if and only if $tr(\phi, \mathtt{dd}(\phi)+1)$ is satisfiable.
- ϕ is satisfiable in modal logic K45 if and only if $tr(\phi, \mathtt{dd}(\phi)+1) \wedge tr((4), \mathtt{dd}(\phi)+1)$ is satisfiable.
- ϕ is satisfiable in modal logic KB5 if and only if $tr(\phi, \mathtt{dd}(\phi)+1) \wedge tr((B), \mathtt{dd}(\phi)+1)$ is satisfiable.
- ϕ is satisfiable in modal logic KT5 if and only if $tr(\phi, \mathtt{dd}(\phi)+1) \wedge tr((T), \mathtt{dd}(\phi)+1)$ is satisfiable.
- ϕ is satisfiable in modal logic KD5 if and only if $tr(\phi, \mathtt{dd}(\phi)+2) \wedge tr((D), \mathtt{dd}(\phi)+2)$ is satisfiable.
- ϕ is satisfiable in modal logic KD45 if and only if $tr(\phi, \mathtt{dd}(\phi)+2) \wedge tr((D), \mathtt{dd}(\phi)+2) \wedge tr((4), \mathtt{dd}(\phi)+2)$ is satisfiable.

Proof. The translation is simulating the different cases of the Tableau used to prove Lemma 2 and we already proved that dd can be used as a valid upper-bound for modal logic K⋆5.

q.e.d □

Without loss of generality, we will talk about UB(ϕ) to denote the upper-bound which can be $\mathtt{dd}(\phi)+1$ or $\mathtt{dd}(\phi)+2$ according to which NP-modal logics is used. Now that we have the translation from K5 to SAT and that we know that every K⋆5 Structures for ϕ are bounded polynomially by the diamond-degree of ϕ.

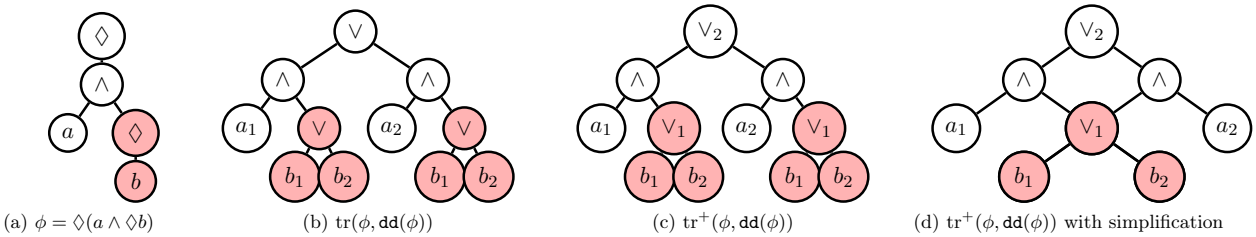
It is interesting to notice in Definition 45 that only the relations related to w_0 are constrained by the axioms. Indeed, the other relations are necessarily true due to the shape of K⋆5 structures.

From this point, if we want to find if a formula ϕ is KD45-satisfiable with a model of size n for example, we will have to solve with a SAT solver the formula: $(\mathrm{tr}(\phi, n) \wedge \mathrm{tr}((D), n) \wedge \mathrm{tr}((4), n)$. During the translation, any $r_{i,j}$ with $i, j > 0$ is directly replaced by $\top$, because it has to be true due the shape of a $K5$-structure. Interesting to notice that if we want to solve the formula ϕ as a modal logic KT5, we will have the propositional logic formula: $(\mathrm{tr}(\phi, \mathtt{dd}(\phi)+1) \wedge \mathrm{tr}((T), \mathtt{dd}(\phi)+1))$, which after simplification of the variables $R_{i,j}$ which are now all equal to true, we will obtain the exact same formula as the one proposed in Caridroit et al. (2017a). From an efficiency perspective, it is worth to consider the logic with the less axioms possible, in order to avoid adding too many clauses which will not change the satisfiability of the formula. Thus for example, if we want to solve the formula ϕ in modal logic KB45, we will just consider axiom (B) which will give the same results with a smaller propositional logic formula.

Moreover, one can see that we can sometimes translate multiple times the same subformula and because it does not depend on the index of the worlds (except for w_0) we will all the time obtain the same set of clauses in propositional logic. One way to avoid such phenomenon is to propose a caching system in order to plug the already-translated result and not translate twice the same sub-formula. We will present how such a caching works in the next subsection.

1.2.2 Caching

Caching is a classical way to avoid redundant work. *SAT performs caching using a "bit matrix" Giunchiglia et al. (2002). Efficient implementation of BDD Bryant (1986) packages also rely on caching, to build an explicit graph. These two examples require additional time and space to search and cache already performed works. Here, our technique is a "simple but efficient" trade-off. It does not memoize the work, so it may not cache all possible formulas, but it only requires a flag to detect redundant work.



(a) $\phi = \Diamond(a \wedge \Diamond b)$ (b) $\mathrm{tr}(\phi, \mathtt{dd}(\phi))$ (c) $\mathrm{tr}^+(\phi, \mathtt{dd}(\phi))$ (d) $\mathrm{tr}^+(\phi, \mathtt{dd}(\phi))$ with simplification

Figure 1.3: Translation of $\Diamond(a \wedge \Diamond b)$ (left), initial translation (middle-left), tagged formula (middle-right), final translation (right)

Example 21

In the example depicted in Figure 1.3.b, sub-formula $(b_1 \wedge b_2)$ appears twice. The translation of the first diamond creates two sub-formulas $a_1 \wedge \Diamond b$ and $a_2 \wedge \Diamond b$, where each $\Diamond b$ needs to be translated. Because in KT5 (all worlds are connected), the translations of $\Diamond b$ on different worlds are equivalent (all the $R_{i,j}$ are true), so we can reuse the same sub-formula. It means that instead of using a tree, we can work with a DAG, which allows a more efficient translation to CNF.

Lemma 5

The result of the translation tr' is independent of the index in the case of a modality in modal logic KT5. More formally we have: $\mathrm{tr}'(\circ\phi, i, n) = \mathrm{tr}'(\circ\phi, j, n)\ \forall i, j$ and $\circ \in \{\Diamond, \Box\}$ in modal logic KT5.

Proof. By definition, there are only 2 cases:

- $(\circ = \Box)$ then, $\mathrm{tr}'(\Box\phi, i, n) = \bigwedge_{k=1}^{n}(\mathrm{tr}'(\phi, k, n))$
- $(\circ = \Diamond)$ then, $\mathrm{tr}'(\Diamond\phi, i, n) = \bigvee_{k=1}^{n}(\mathrm{tr}'(\phi, k, n))$

In each case, the result is independent from i, so choosing j as an index gives the exact same result. q.e.d □

Informally, Lemma 5 shows the fact that no matter how embedded the modal sub-formula is, its translation will always give the same result (independent from the index i). Therefore, we can start by translating the most embedded sub-formula, tag the corresponding node and backtrack. The resulting formula may contain several nodes with the same tag. This means that these sub-formulas are syntactically identical (see Figure 1.3.c). Then, we maintain only one occurrence of the sub-formula, transforming the tree in a DAG (see Figure 1.3.d). Structural caching is thus performed on the fly before translating to CNF. The translation function using this technique is noted tr^{+}. This lazy-caching system leads to huge improvement in KT5 as explained in Caridroit et al. (2017a).

It is trickier to see if such a caching can be extended to any modal logic containing axiom (5). In fact, only a part of the translation can be cached in K⋆5, but not all of it as in KT5. In the more general case, we have:

Lemma 6

If we consider the translation of the modalities as follows:

$$\mathrm{tr}'(\Box\phi, i, n) = \Sigma_i \wedge \Delta \qquad \mathrm{tr}'(\Box\phi, j, n) = \Sigma_j \wedge \Delta$$
$$\mathrm{tr}'(\Diamond\phi, i, n) = \Sigma_i \vee \Delta \qquad \mathrm{tr}'(\Diamond\phi, j, n) = \Sigma_j \vee \Delta$$

$\forall i, j \neq 0$. Then Δ (being the other part of the formula) may be cached.

Proof. By Definition of tr. We know that the second conjunction (resp. disjunction) can be cached if the index is different than 0. q.e.d □

As we can see, the caching system is less efficient than the one that we can obtain if we deal only with KT5. For the sake of understanding, in the experimental part, we did not implement the caching of Caridroit et al. (2017a) in K⋆5SAT when it deals with KT5. That way, one can see the impact of a dedicated approach against a more general one.

Now that we have a set of simplifications, a SAT translation and a lazy-caching system, it is time to see if the whole approach is competitive with the state-of-the-art modal logic K⋆5 solvers. We named our solver K⋆5SAT and it works as follows:

1. It simplifies the formula as much as possible;
2. It translates the formula into a propositional logic formula using caching;
3. It calls Glucose Eén and Sörensson (2003), Audemard et al. (2013) on it to solve it.

1.3 Experimental Evaluation of the SAT-Based Approach

We compared K⋆5SAT against existing solvers considered state-of-the-art in Lagniez et al. (2017a), Nalon et al. (2016) and Caridroit et al. (2017a) for the modal logic K, KT and S4, but able to deal with some/all logics containing axiom (5) namely: Moloss 0.9 Areces et al. (2015), SPASS 3.7 Hustadt et al. (1999), S52SAT Caridroit et al. (2017a), and Vampire 4.0 Kovács and Voronkov (2013) with a combination of the optimized functional translation Horrocks et al. (2006).

One must understand correctly what we are doing here, due to the fact that there does not exist modal logic K⋆5 benchmarks to evaluate the different approach nor dedicated K⋆5 solvers. We took general modal logic solvers (like Moloss and SPASS) and we give them the axioms (5), (D), (4) or (T) according to the logic we consider. Those solvers do not consider that axiom (5) will be by default, thus they expect any kind of Kripke structure to arrive.

For the benchmarks, one thing to understand correctly is which logic entails which other. We already saw that being satisfiable in KT5 entails being satisfiable in K5 and being unsatisfiable in K5 entails being also unsatisfiable in KD45 for example. But this phenomenon is not happening

only in modal logic K⋆5. Indeed, every Kripke structure is at least, a K-Structure, thus the unsatisfiability in modal logic K entails the unsatisfiability in all the modal logics.

Because it would be CPU-consuming to test all the K⋆5 logics and that the CPU ressources of CRIL are mutualised, we restraint ourselves to 3 different K⋆5 logics, namely: K5, KD45 and KT5. K5 being the base of our approach, it is worth to compare against other approach to see if we are competitive without considering any axioms. KT5 is a logic that we already dealt with in Caridroit et al. (2017a), thus it is interesting to compare our general approach against a totally dedicated one. And finally, we test modal logic KD45, because it is a famous modal logic (the modal logic representing the belief Tallon et al. (2004)) and also because it is the logic which adds the most clauses in our SAT approach. We have to translate the axioms (D) and (4), thus it is normally there that we will have the "worst" results.

1.3.1 Results Obtained In Logic WorkBench (LWB)

Solver	**K-SAT**	**K-UNSAT**	**KT-SAT**	**KT-UNSAT**	**S4-SAT**	**S4-UNSAT**	**Total**
# Benchs	(504)	(504)	(504)	(504)	(504)	(504)	(3024)
Moloss	142	100	214	189	308	308	1261
SPASS	295	286	137	126	310	295	1549
Vampire	404	409	317	306	311	288	2035
S52SAT							
K⋆5SAT	**420**	**412**	**403**	**397**	**423**	**398**	**2453**
VBS	420	412	403	397	423	398	2453

Table 1.1: #Instances solved in LWB K,KT,S4 in modal logic $K5$

Solver	**K-SAT**	**K-UNSAT**	**KT-SAT**	**KT-UNSAT**	**S4-SAT**	**S4-UNSAT**	**Total**
# Benchs	(504)	(504)	(504)	(504)	(504)	(504)	(3024)
Moloss	136	119	214	174	305	300	1248
SPASS							
Vampire							
S52SAT							
K⋆5SAT	**380**	**350**	**377**	**321**	**402**	**396**	**2226**
VBS	380	350	377	321	402	396	2226

Table 1.2: #Instances solved in LWB K,KT,S4 in modal logic $KD45$

In Table 1.1, Table 1.2 and Table 1.3, an empty line represents the fact that the solver cannot deal with the logic considered.

As it was the case in Caridroit et al. (2017a), it appears that the SAT-based approaches performs extremely well when the translation is adapted to the problem. Indeed, we can see that S52SAT does better than K⋆5SAT in modal logic KT5. This is obvious because S52SAT

Solver # Benchs	**K-SAT** (504)	**K-UNSAT** (504)	**KT-SAT** (504)	**KT-UNSAT** (504)	**S4-SAT** (504)	**S4-UNSAT** (504)	**Total** (3024)
Moloss	155	142	177	80	288	244	1086
SPASS	287	288	144	130	320	301	1470
Vampire	367	357	398	396	426	399	2343
S52SAT	**480**	**425**	**478**	**409**	**480**	**421**	**2693**
K⋆5SAT	433	420	421	402	453	410	2539
VBS	480	425	478	409	480	421	2693

Table 1.3: #Instances solved in LWB K,KT,S4 in modal logic $KT5$

is dedicated to solve only KT5. For the other logics, Moloss is an SMT-based approach using MiniSAT Eén and Sörensson (2003) as an internal SAT solver and we can see that it does not perform as well as K⋆5SAT. This is mainly due to the fact that Moloss is created to deal with all the modal logics, even the PSPACE-complete one like K, KT or S4 for example. Thus the simplifications are not applied by Moloss. SPASS has good results when we take into account that it is not supposed to deal only with modal logic. SPASS is a resolution-based solver which translates modal logics formulas into first-order logic formulas[5]. And finally Vampire does very good too for an external First-Order Logic solver used after an optimized functional translation Horrocks et al. (2006). Again, Vampire is not really designed to deal with modal logic formulas and it is a complete and efficient tool which can answer the satisfiability as we do here, but it can also do theory reasoning, interpolation, consequence elimination, program analysis and many other.

1.3.2 Results Obtained In TANCS-2000-MQBF

Solver # Benchs	**qbf** (56)	**qbfL** (240)	**qbfS** (240)	**qbfMS** (240)	**qbfML** (240)	**Total** (1016)
Moloss	**56**	0	177	0	0	233
SPASS	50	79	235	108	0	472
Vampire	**56**	**240**	239	**208**	**240**	**983**
S52SAT						
K⋆5SAT	**56**	153	**240**	172	28	649
VBS	56	240	240	240	240	1016

Table 1.4: #Instances solved in MQBF in modal logic $K5$

The instances in MQBF have a very huge number of modalities with a deep modal depth in

[5]We just used the default options in SPASS except the theory for the logic considered (see http://www.cs.man.ac.uk/~schmidt/mspass/manual/script_1.html for more explanation).

Solver	**qbf**	**qbfL**	**qbfS**	**qbfMS**	**qbfML**	**Total**
# Benchs	(56)	(240)	(240)	(240)	(240)	(1016)
Moloss	**47**	0	74	2	0	123
SPASS						
Vampire						
S52SAT						
K⋆5SAT	40	**142**	**240**	**166**	**28**	**450**
VBS	47	142	240	166	28	450

Table 1.5: #Instances solved in MQBF in modal logic $KD45$

Solver	**qbf**	**qbfL**	**qbfS**	**qbfMS**	**qbfML**	**Total**
# Benchs	(56)	(240)	(240)	(240)	(240)	(1016)
Moloss	**56**	0	132	10	0	198
SPASS	**56**	0	108	235	0	399
Vampire	**56**	**240**	**240**	208	**240**	**984**
S52SAT	**56**	178	**240**	**240**	**240**	954
K⋆5SAT	**56**	153	**240**	202	174	825
VBS	56	240	240	240	240	1016

Table 1.6: #Instances solved in MQBF in modal logic $KT5$

general. The minimum modal depth of formulae in this category is 19, the maximum 225, average 69.2 with a standard deviation of 47.5. Thus a solver which would not apply the simplification presented in Theorem 11 has poor chances to solve these instances efficiently. Interestingly, because these formulas have a huge number of modalities and are highly complex (they are representing QBF formulas), Vampire does extremely good results. It is able to infer conclusions out of reach for a SAT solver which allows it to solve efficiently the formulas. Here the weakness of all the SAT-based approach is the translation phase. It is too long to translate the formula (when it does not simply blow-up the memory allocated).

1.3.3 Results Obtained In 3CNF$_{KSP}$

And here the results are extremely logical, the category 3CNF$_{KSP}$ are extremely similar to a SAT problem, so as expected the three techniques based on a SAT solver (Moloss with MiniSAT, S52SAT and K⋆5SAT with Glucose). In fact, when we check the VBS, all the instances are solved in the three logics considered, which was not the case in the other categories. The results of Moloss in the case K5 are surprising compared to the two other logics. But it turns out that the adding of axioms make the results (SAT or UNSAT) easy to obtain for Moloss, without them Moloss runs out of time. SPASS and Vampire, using First-Order Logic (FOL) techniques are not

Solver	**md=1**	**md=2**	**Total**
# Benchs	(240)	(760)	(1000)
Moloss	53	115	168
SPASS	235	287	522
Vampire	**240**	411	651
S52SAT			
K⋆5SAT	221	**760**	**981**
VBS	240	760	1000

Table 1.7: #Instances solved in 3CNF$_{KSP}$ in modal logic $K5$

Solver	**md=1**	**md=2**	**Total**
# Benchs	(240)	(760)	(1000)
Moloss	**240**	**760**	**1000**
SPASS			
Vampire			
S52SAT			
K⋆5SAT	219	**760**	979
VBS	240	760	1000

Table 1.8: #Instances solved in 3CNF$_{KSP}$ in modal logic $KD45$

Solver	**md=1**	**md=2**	**Total**
# Benchs	(240)	(760)	(1000)
Moloss	**240**	**760**	**1000**
SPASS	235	302	537
Vampire	204	359	563
S52SAT	**240**	**760**	**1000**
K⋆5SAT	**240**	**760**	**1000**
VBS	240	760	1000

Table 1.9: #Instances solved in 3CNF$_{KSP}$ in modal logic $KT5$

as efficient as a SAT solver to solve modal logic instances which are closer to a SAT instances than to a highly-structured instance for which FOL solvers are more optimized.

Now that we evaluate our solver K⋆5SAT and we demonstrated that it is now the state-of-the-art approach for solving the K⋆5-SAT problem. We know how many benchmarks from the

community are K$\star$5-Satisfiable, thus it is now interesting to see how far is the diamond-degree from the optimal value, *ie.*, the Kripke model with the smallest number of worlds satisfying the formula. Obviously this problem is a function problem and it is harder to solve than just the satisfiability problem. We will see in the next chapter, how the translation tr can be adapted and how a SAT solver, in incremental mode, can be used to obtain the smallest Kripke model possible.

Chapter 2

The Minimal K⋆5 Satisfiability Problem

Contents

Providing a model, a certificate of satisfiability, is important to check the answer given by the solver. This is true both for the author of the solver or a user of that solver. This is mandatory nowadays in many solver competitions, among them the SAT competition Simon et al. (2005). It has also been shown that those models can help improving NP-oracle based procedures Marques-Silva and Janota (2014): a procedure requiring a polynomial number of calls to a yes/no oracle can be transformed into a procedure requiring only a logarithmic number of calls when the oracle can provide a model.

Another example of the importance for an oracle to provide a model may be found in the model rotation technique Marques-Silva and Lynce (2011), a method for the detection of clauses that are included in all MUSes (Minimal Unsatisfiable Sets) of a given formula via the analysis of models returned by a SAT oracle. Even if the theory does not guarantee a reduction of the number of oracle calls, in practice it provides a huge performance gain (up to a factor of 5) Belov and Marques-Silva (2011).

Finding the smallest model may be even more important in some contexts. The provided model usually has a meaning for the user, like in Hardware Verification Fairtlough and Mendler (1994) where the model is in fact an explanation of the bug found in the design of the hardware. The smaller the model is, the more precise could be the location of the bug. It could also be the case that the model should be inspected by the user or displayed on a screen. Thus, the smaller, the better. There is a huge literature about minimizing models for SAT Iser et al. (2013), Soh and Inoue (2010), Koshimura et al. (2009).

In this chapter, we will see what can be the different techniques to obtain the smallest K⋆5 models with respect to the number of possible worlds.

2.1 The Minimal K⋆5 Satisfiability Problem

In this work, the minimality is achieved on the size of a Kripke structure $\langle M, w\rangle$, noted $|M|$, corresponding to its number of worlds. Let us use a formula as a running example to see how we can obtain the smallest possible:

Example 22

Let $\mathbb{P} = \{a, b\}$. $\phi = ((\Box\neg a \vee \Diamond b) \wedge \Diamond a \wedge \Box b)$ is a modal logic formula. We can easily find a K5 model such as the following one: $W = \{w_0, w_1, w_2\}$, $R = \{(w_0, w_1), (w_0, w_2), (w_1, w_2), (w_1, w_1), (w_2, w_1), (w_2, w_2)\}$, $V = \{\langle a, \{w_1\}\rangle, \langle b, \{w_0, w_1, w_2\}\rangle\}$. The size of $\langle M, w_0\rangle$ equals 3.

Let us remark that obtaining the minimal model for ϕ is not as simple as merging the worlds with the same valuations into only one world in any model of ϕ. The minimality cannot be guaranteed that way. Indeed, ϕ is also satisfied by the following structure containing only one world: $W = \{w_0\}, R = \{(w_0, w_0)\}, V = \{\langle a, \{w_0\}\rangle, \langle b, \{w_0\}\rangle\}$, which is a minimal model. Let us define more formally the problem of Minimal K⋆5 Satisfiability Problem.

Definition 46 (Minimal K⋆5 Satisfiability)

A formula ϕ is min-K⋆5-satisfied by a Kripke structure $\langle M, w\rangle$ (noted $\langle M, w\rangle \models_{\min} \phi$) if and only if $\langle M, w\rangle \models \phi$ and ϕ has no model $\langle M', w'\rangle$ such that $|M'| < |M|$.

Definition 47 (Minimal K⋆5 Satisfiability Problem)

Let a formula ϕ in $\mathcal{L}$ be given. The minimal K⋆5 satisfiability problem (MinK⋆5-SAT) is the problem of finding a structure $\langle M, w\rangle$ such that $\langle M, w\rangle \models_{\min} \phi$.

A very simple way to tackle this problem is to use the solver K⋆5SAT (presented in the previous chapter) with a linear search strategy. Roughly, the procedure starts by trying structures of size $b = 1$. If no model is found, it iterates the process, each time increasing the value of b by 1. It iterates until a model of ϕ is found or the upper bound $UB(\phi)$ is reached. This strategy is called 1toN. It is of course also possible to do it in reverse order: the procedure starts with $b = UB(\phi)$ and decreases the value of b by 1 (this is called Nto1). Yet another possibility is to use a binary search (called Dico). However, these approaches are very naive. If we take, for instance, 1toN when the solution is a model of size m, it will perform m translations from K⋆5 to SAT and then m calls to a SAT solver. The problem here is that such strategy does not take advantage of the previous UNSAT answer of the SAT solver to solve the new formula.

2.2 How To Solve The MinK⋆5 Satisfiability Problem

In this section, we will present different techniques that can be used to solve the MinK⋆5 Satisfiability Problem. The first of them is to use an incremental-SAT solver. Modern SAT solvers are able to take advantage of previous calls when they are used incrementally Eén and Sörensson (2003), Audemard et al. (2013). The usual way to do that is to add selectors (assumptions) to the input formula and to get as output, on the suitable cases, some kind of "reason" for its unsatisfiability, in terms of these selectors. We propose here a way to add such selectors in the translation from K⋆5 to SAT.

2.2.1 An Assumption-Based Translation

In order to take advantage of the ability of modern SAT solvers to return a reason for unsatisfiability, we also add more information to the translated formula. Here, we change the translations of $\Box\psi$ and $\Diamond\psi$ in order to add fresh propositional variables (s_i) that we call selectors, in such a way that the truth value of the selectors determines if the world i is 'active' or 'inactive' in the current Kripke structure. Formally (and with the additional parameters), we have:

Definition 48 (Translation function '$\mathrm{tr_s}$' with selectors)

Let $\phi \in \mathcal{L}$ which contains the axiom (5).

$$\mathrm{tr_s}(\phi, n) = \mathrm{tr_s}'(\mathrm{nnf}(\phi), 0, n)$$
$$\mathrm{tr_s}'(\top, i, n) = \top \quad \mathrm{tr_s}'(\neg\top, i, n) = \neg\top$$
$$\mathrm{tr_s}'(p, i, n) = p_i \quad \mathrm{tr_s}'(\neg p, i, n) = \neg p_i$$
$$\mathrm{tr_s}'((\phi \wedge \cdots \wedge \delta), i, n) = \mathrm{tr_s}'(\phi, i, n) \wedge \cdots \wedge \mathrm{tr_s}'(\delta, i, n)$$
$$\mathrm{tr_s}'((\phi \vee \cdots \vee \delta), i, n) = \mathrm{tr_s}'(\phi, i, n) \vee \cdots \vee \mathrm{tr_s}'(\delta, i, n)$$
$$\mathrm{tr_s}'(\Box\phi, i, n) = \begin{cases} (\neg R_{i,0} \vee \mathrm{tr_s}'(\phi, 0, n)) \wedge \bigwedge_{j=1}^{n} (\neg s_j \vee \mathrm{tr_s}'(\phi, j, n)) & (i \neq 0) \\ (\neg R_{0,0} \vee \mathrm{tr_s}'(\phi, 0, n)) \wedge \bigwedge_{j=1}^{n} (\neg R_{0,j} \vee (\neg s_j \vee \mathrm{tr_s}'(\phi, j, n))) & \textit{otherwise} \end{cases}$$
$$\mathrm{tr_s}'(\Diamond\phi, i, n) = \begin{cases} (R_{i,0} \wedge \mathrm{tr_s}'(\phi, 0, n)) \vee \bigvee_{j=1}^{n} (s_j \wedge \mathrm{tr_s}'(\phi, j, n)) & (i \neq 0) \\ (R_{0,0} \wedge \mathrm{tr_s}'(\phi, 0, n)) \vee \bigvee_{j=1}^{n} (R_{0,j} \wedge (s_j \wedge \mathrm{tr_s}'(\phi, j, n))) & \textit{otherwise} \end{cases}$$

First, it is important to notice that, there is always at least one selector affected at true because in modal logic there is always at least one world, called the actual-world. A K⋆5-model has to have at least one possible world (the actual world). The reader may verify that the selector s_0 does not exist. It is not added to the translation because it is considered to be always set to

true. In the remainder of this Chapter, we denote by $\mathrm{tr_s}(\phi)$ the formula $\mathrm{tr_s}(\phi, UB(\phi))$ and the set of all selectors of $\mathrm{tr_s}(\phi)$ is denoted by $\mathcal{S}(\phi)$ (i.e., $\mathcal{S}(\phi) = \{s_i \mid 1 \leq i \leq \mathtt{dd}(\phi)\}$).

Let us go back to Example 22 with the formula $\phi = \Diamond(a \wedge \Box b)$. The translation $\mathrm{tr_s}(\phi)$ is:

$$(R_{0,0} \wedge a_0 \wedge (\neg R_{0,0} \vee b_0) \wedge (\neg R_{0,1} \vee \neg s_1 \vee b_1)) \vee (R_{0,1} \wedge s_1 \wedge a_1 \wedge (\neg R_{1,0} \vee b_0) \wedge (\neg s_1 \vee b_1))$$

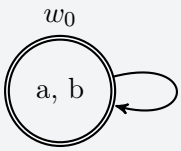
Intuitively, every formula with subscript i is a formula that is true at the possible world i. As we can see, if one selector s_i is false, then the sub-formula connected to it will be considered false as well. We thus respect the idea that if the selector s_i is false, then the world i is not present in the model (and we do not care about the valuation of the propositions there in).

Example 23 (Example of "$\mathrm{tr_s}$")

Below, we have a formula that is equivalent to deactivate s_1 (because s_0 is considered to be for sure activated).

$$(R_{0,0} \wedge a_0 \wedge (\neg R_{0,0} \vee b_0) \wedge (\neg R_{0,1} \vee \neg \bot \vee b_1)) \vee (R_{0,1} \wedge \bot \wedge a_1 \wedge (\neg R_{1,0} \vee b_0) \wedge (\neg \bot \vee b_1))$$

After the application of the simplification, the formula is equivalent to $(R_{0,0} \wedge a_0 \wedge (\neg R_{0,0} \vee b_0))$ And if we take a closer look, this formula is $\mathrm{tr}(\phi, 1)$ (the translation without any selectors and allowing only 1 world). It corresponds to the problem of deciding if ϕ is satisfiable in a model of size 1.



In fact, the model that the SAT solver finds is this $K5$-Structure which is a $K5$-model of ϕ.

Moreover, even though we are using here the incremental-SAT technique, it is still possible to simulate the approaches presented before (1toN, Nto1 and Dicho). The disadvantages is that the size of the formula generated with this translation is bigger due to the fact that we must bound it by $UB(\phi)$ compared to the other approach where the bound is necessary smaller or equal. However, as pointed out in Caridroit et al. (2017a), the size of the formulas generated with the theoretical upper-bound are reasonable and adding the selectors does not increase too much the size. Moreover, because the translation is the bottleneck of the approach and not the SAT solving, translating the formula only once gives us a speed-up compared to the approaches presented before (1toN, Nto1, Dicho).

As we can see, the problem of solving the minimal K⋆5 satisfiability problem is now equivalent to the problem of satisfying $\mathrm{tr_s}(\phi)$ and minimize the number of s_i, for $i > 0$, assigned to true (or, equivalently, maximize the number of s_i assigned to false). This problem is well known in the literature as the Partial MaxSAT problem Li and Manyà (2009), which consist on satisfying all the hard clauses and the maximum number of soft clauses. In our case, the hard clauses are those generated by the translation function, and the soft-clauses are the unit clauses $\{\neg s_i \mid s_i \in$

$\mathcal{S}(\phi)$ and $i > 0\}$, which are added to the problem. Obviously it can also be seen as a pseudo-Boolean optimization problem Roussel and Manquinho (2009), where the optimization function to be minimized is the number of selectors assigned to true.

We can thus use state-of-the-art Partial MaxSAT or PBO solvers. However, it is not the only way, as we show in the following section that, by considering the structure of K⋆5-models, extracting a MSS can also be used to decide the problem.

Without loss of generality, in the following sections, we represent a set of unit soft clauses as the set of selectors composing it (eg.: $\{s_1, s_2, s_3\}$ rather than $\{\neg s_1, \neg s_2, \neg s_3\}$). We will also consider formulas Σ in CNF as a set of clauses.

2.2.2 Cardinality Optimality Equals Subset Optimality

In this section, we will need the notion of MSS and co-MSS, so let us defined it. The problem of computing a Maximal Satisfiable Set of clauses (MSS problem) consists in extracting a maximal set of clauses from a formula in CNF that are consistent together O'Sullivan et al. (2007). The minimal correction subset (MCS or co-MSS) is the complement of its MSS.

Definition 49

Let a unsatisfiable formula Σ in CNF be given. $S \subseteq \Sigma$ is a Maximal Satisfiable Subset (MSS) of Σ if and only if S is satisfiable and $\forall c \in \Sigma \setminus S$, $S \cup \{c\}$ is unsatisfiable.

Definition 50

Let a unsatisfiable formula Σ in CNF be given. $C \subseteq \Sigma$ is a Minimal Correction Subset (MCS or co-MSS) of Σ if and only if $\Sigma \setminus C$ is satisfiable and $\forall c \in C$, $\Sigma \setminus (C \setminus \{c\})$ is unsatisfiable.

It is also possible to define a partial version of the MSS problem, where the objective is to compute a MSS such that some given sub-set of the clauses (the hard clauses) must be in it. This problem is related to the Partial MaxSAT problem. In fact, a solution to a Partial MaxSAT problem is one of the biggest MSS that satisfies the set of hard-clauses. In general, a partial MSS is not a solution to a Partial MaxSAT problem but, in the specific case of MinK⋆5-SAT, a partial MSS is also a solution to its corresponding Partial MaxSAT problem.

Proposition 1

Let ψ be a formula in CNF equi-satisfiable to $\mathrm{tr_s}(\phi)$, and let χ be the formula $\bigwedge_{i=1}^{UB(\phi)} \neg s_i$. With $UB(\phi) = \mathtt{dd}(\phi) + 2$ (or $UB(\phi) = \mathtt{dd}(\phi) + 1$ if the axiom (D) is not considered). A MSS of $(\psi \wedge \chi)$, where ψ is the set of hard clauses, is also a solution to the Partial MaxSAT problem $\langle \psi, \chi \rangle$.

The proof of Proposition. 1 uses the following lemma.

Lemma 7

Let $\psi = \mathrm{tr_s}(\phi)$, and let χ be the formula $\bigwedge_{s_i \in \mathcal{S}'} \neg s_i$, where $\mathcal{S}' \subseteq \mathcal{S}(\phi)$. If $(\psi \wedge \chi)$ is satisfiable then so is the formula $(\psi \wedge \chi')$, where χ' is obtained from χ by replacing the occurrences of one selector $s \in \mathcal{S}'$ by another selector $s' \in \mathcal{S}(\phi) \setminus \mathcal{S}'$.

Proof. [Sketch] The proof is done by an induction on the length of the formula ϕ. In the induction base, $\psi = p$, for some $p \in \mathbb{P}$. We have $\psi = p_1$ and $\mathcal{S}(\phi) = \{\}$, which means that the claim is true (because $\mathcal{S}(\phi) \setminus \mathcal{S}' = \emptyset$).

We have several cases on the induction step. Since their proofs are all similar, we show only one of them here.

Let $\phi = \Box\phi'$. If $i \neq n$ We have $\psi = (\neg R_{i,0} \vee \mathrm{tr_s}'(\phi, 0, n)) \wedge \bigwedge_{j=1}^{n}(\neg s_j \vee \mathrm{tr_s}'(\phi, j, n))$, $\chi = \bigwedge_{s_i \in S'} \neg s_i$, and $\mathcal{S}(\phi) = \{s_1, \ldots, s_n\}$. Now, let χ' be obtained from χ where s_i is replaced by $s_j \in \mathcal{S}(\phi) \setminus S'$. If $(\psi \wedge \chi)$ is satisfied by a model M then we construct a new model M', which equals M except that the truth assignment of all propositional variables with subscript i are the same as those with subscript j. We immediately have that if $M \models \chi$ then $M' \models \chi'$. We also have that if $M \models \neg s_i$ then $M' \models \neg s_j$. Finally, for each $1 \leq i \leq n$, if $M \models \mathrm{tr_s}(\phi', i, n))$ then $M' \models \mathrm{tr_s}(\phi', j, n))$, by the induction hypothesis (since the length of ϕ' is strictly smaller than that of ϕ). Therefore, $M' \models \psi \wedge \chi'$. q.e.d □

Proof. Towards a contradiction, assume that there exists a MSS $\delta_1 = (\psi \wedge \chi_1)$, where $\chi_1 = \bigwedge_{s \in S_1} \neg s$, which is not the biggest one. Thus, there exists another MSS $\delta_2 = (\psi \wedge \chi_2)$, where $\chi_2 = \bigwedge_{s \in S_2} \neg s$ and such that $|S_1| < |S_2|$. Now, let $S_3 = S_2 \setminus \{s\} \cup \{s'\}$, where $s \in S_2$ and $s' \in S_1$. By Lemma 7, the formula $\delta_3 = (\psi \wedge \chi_3)$, where $\chi_3 = \bigwedge_{s \in S_3} \neg s$ is satisfiable, because it is δ_2 with one of the selectors of S_2 in χ_2 replaced by another selector. It is easy to see that one can keep replacing selectors in this set until we have the set S_k, such that $S_1 \subseteq S_k$. The formula $\delta_k = (\psi \wedge \chi_k)$, where $\chi_k = \bigwedge_{s \in S_k} \neg s$, is satisfiable, by applying Lemma 7 $|S_1|$ times. Then S_k a MSS that includes S_1, which contradicts the assumption. This means that every MSS of the initial formula is one of the biggest ones. Therefore, any MSS of $(\psi \wedge \chi)$ is also a solution to the partial MaxSAT problem $\langle \psi, \chi \rangle$. q.e.d □

As a direct consequence of Proposition 1, we can always find a MSS such that the indexes of the selectors inside it are contiguous. This means that we can consider an optimisation that reduces the search space (breaks the symmetry), by adding the following constraint:

$$(\bigwedge_{i=1}^{n-1} \neg s_i \rightarrow \neg s_{i+1}) \tag{2.1}$$

By giving as input $\mathrm{tr_s}(\phi)$ plus $\mathcal{S}(\phi)$, we can solve the MinK⋆5-SAT problem with a MaxSAT solver, or a PBO solver. If we also add Equation 2.1 to the input, we can then use a MSS-extractor. However, we demonstrate in the following section that we can push the envelope further by considering a dedicated approach using an incremental SAT solver with unsatisfiable cores.

2.2.3 Only Unsatisfiable Cores Size Matters

Before defining formally the core-guided approach and its different properties, let us consider the following case: Let ϕ the input formula and let $UB(\phi) = 10$. We translate ϕ using selectors and start by trying to find a model for it. Assume that, after some computation, we conclude that 4 worlds cannot be deactivated altogether, i.e., if the selectors s_i, s_j, s_k and s_l are set to false, we have an inconsistency. We can infer that we will need at least 7 worlds in the K⋆5-model for ϕ. This comes from the fact that the '4 worlds which cannot be deactivated altogether' can be, in fact, any group of 4 worlds. Indeed, in the sequel, we demonstrate that if we have a group of m selectors forming an unsatisfiable core and the upper-bound equals n, then we need at least $(n - m + 1)$ worlds in the K⋆5-model of the input formula.

Proposition 2

Let $\phi \in \mathcal{L}$ such that $UB(\phi) = n$. If C is a UNSAT core of ϕ under assumptions $\mathcal{S}(\phi)$ then $\mathrm{tr}_{\mathrm{s}}(\phi, n')$ is unsatisfiable for all $n' \in \{1, \ldots, (n - |C|)\}$.

Lemma 8

If C is an UNSAT core of ϕ with assumptions $\mathcal{S}(\phi)$ then any set of literals $C' = \{\neg s \mid s \in \mathcal{S}(\phi)\}$ such that $|C'| = |C|$ is a UNSAT core of ϕ.

Proof. Assume that C is an UNSAT core of ϕ with assumptions $\mathcal{S}(\phi)$. We have that $(\phi \wedge \bigwedge_{s \in C} s)$ is unsatisfiable. Now, towards a contradiction, also assume that there exists a set $C' = \{\neg s \mid s \in \mathcal{S}(\phi)\}$ such that $|C'| = |C|$ and $(\phi \wedge \bigwedge_{s \in C} \neg s)$ is satisfiable. By Lemma 7, we can obtain a new set D from C' by replacing the selectors in C' by those in C such that $(\phi \wedge \bigwedge_{s \in D} \neg s)$ is satisfiable. Because $D = C$, we have a contradiction. Therefore, any set of literals C' obtained as such is a UNSAT core of ϕ. q.e.d □

We can now prove the Proposition 2 thanks to the Lemma 8.

Proof. The formula has n worlds. The SAT solver returns a core C of size m. So one of the selectors has to be true. But due to Lemma 8, we have to put at least one selector to true to all the possible unsatisfiable cores of size m. Said otherwise, we must have $(n - m + 1)$ selectors to be true together, or the formula will be necessarily unsatisfiable. This also means that $\forall b' \in [1 \ldots (n - m)]$ $\mathrm{tr}_{\mathrm{s}}(\phi, b')$ is unsatisfiable. q.e.d □

From this property, it is possible to construct an interactive algorithm which is based on incremental SAT. The SAT solver will be able to return an unsatisfiable core, and by interpreting it as explained in Proposition 2, we can refine the bound used in the translation. Such a technique is presented in Algorithm 2.1.

Algorithm 2.1: K⋆5SAT 1toN with selectors

Data: $\phi \in \mathcal{L}$
Result: $\langle M, w\rangle$ such that $\langle M, w\rangle \models_{min} \phi$, else UNSAT

```
1  begin
2      b ← 1 ;
3      ⟨r, s⟩ ← SAT-SOLVER(tr_s(φ, b)) ;
4      n ← UB(φ) ;
5      while (r ≠ SAT ∧ (b ≤ n)) do
6          b ← (n − |s| + 1) ;
7          ⟨r, s⟩ ← SAT-SOLVER(tr_s(φ, b)) ;
8      if (r ≠ SAT) then
9          return UNSAT
10     else
11         ⟨M, w⟩ ← getModel(s) ;
12         return ⟨M, w⟩;
```

The procedure starts by trying structures of size $b = 1$. If no model is found, it iterates the process, each time increasing the value of b by $(UB(\phi) - |s| + 1)$ (where $|s|$ is the size of the core). It iterates until a model of ϕ is found or the upper bound $UB(\phi)$ is reached. The procedure SAT-SOLVER() that appears in the algorithm is the SAT solver "GLUCOSE" presented in Audemard et al. (2013), Eén and Sörensson (2003). The procedure $getModel()$ is a procedure that generates the K⋆5-model from the SAT model given as its argument. We present only the approach $1toN_c$ (1toN with selectors) but the approaches $Nto1_c$ and $Dicho_c$ are similar.

2.3 Experiment For The MinK⋆5 Satisfiability Problem

Now that we evaluate our solver K⋆5SAT and we demonstrated that it is now the state-of-the-art approach for solving the K⋆5-SAT problem. We know how many benchmarks from the community are K⋆5-Satisfiable, thus it is now interesting to see how far the upper-bound proposed ($UB(\phi)$) from the optimal value and how many instances we can solved optimally (out of the many satisfiable instance that we could at least solved). We compare several different approaches to solve the MinK5-SAT problem:

- K⋆5SAT with five different strategies: 1toN$_c$, 1toN, Nto1, Dicho$_c$, Dicho.
- CNF plus MaxSAT solver: maxHS-b Davies and Bacchus (2013), mscg2015b dos Reis Morgado et al. (2014), and MSUnCore Heras et al. (2011).
- Pseudo-Boolean (PB) translation plus *PBO* solver: NaPS Sakai and Nabeshima (2015), SAT4J-PB Le Berre and Parrain (2010), SCIP Maher et al. (2017)
- CNF plus symmetry breaking plus MCS extraction with the LBX solver Mencía et al. (2015).

The symmetry breaking could have been applied to all the direct approaches (MaxSAT, PBO, MCS) but we choose to apply it only on MCS extraction to display the impact of such symmetry breaking.

2.3.1 Results On The Benchmarks From The Literature

We deal only with K5 here because the other logics will not change the interpretation that we do of the results. Moreover this is the logic which has the more satisfiable instances in the benchmarks that we considered. The study for KT5 was published in Lagniez et al. (2018a).

Method	**K**	**KT**	**S4**	**Total**	**md=1**	**md=2**	**Total**	**qbf**	**qbfS**	**Total**
# Benchs	(185)	(279)	(160)	(624)	(62)	(27)	(89)	(56)	(171)	(227)
K⋆5SAT	185	279	160	624	62	27	89	56	171	227
1toN	168	226	100	494	48	0	48	**56**	**171**	**227**
Nto1	124	126	12	262	0	0	0	**56**	112	168
Dicho	140	200	89	429	40	0	40	**56**	**171**	**227**
1toN$_c$	**185**	**279**	**160**	**624**	**62**	**27**	**89**	**56**	**171**	**227**
Dicho$_c$	**185**	245	**160**	590	**62**	15	77	**56**	**171**	**227**
maxHS	125	159	68	352	20	0	20	**56**	0	56
MSCG	137	183	71	391	**62**	10	72	**56**	**171**	**227**
MSUnCore	115	138	60	313	19	0	19	**56**	0	56
NaPS	147	217	78	442	**62**	10	72	**56**	**171**	**227**
SAT4J	62	129	91	282	19	0	19	**56**	0	56
SCIP	124	234	**160**	518	16	0	16	**56**	56	112
LBX	148	235	153	536	**62**	14	76	**56**	**171**	**227**
VBS	185	279	160	624	62	27	89	56	171	227

a: in LWB K,KT,S4 b: in 3CNF$_{KSP}$ c: in $MQBF$

Table 2.1: #Instances solved with one of the smallest model possible

One may wonder why there is such a difference in the results between the approaches using K⋆5SAT and the MaxSAT solvers. This came from the fact that MaxSAT solvers cannot take into account inherent properties of modal logic K⋆5, they just have embedded cardinality constraints used to count the number of satisfied/falsified clauses to return the smallest model not the fact that the s_i with $i \neq 0$ are all interchangeable.

We can also observe that 1toN and Dicho approaches are more successful than Nto1. The explanation is that these benchmarks can usually be solved with few possible worlds, thus starting with the theoretical upper-bound is less efficient.

In the case of MQBF displayed in Table 2.1c, we can see that Dicho, Dicho$_c$, 1toN and 1toN$_c$ approaches are better than the other ones. Moreover, it is interesting to see that the whole **qbf** family is $K5$-satisfiable (even KT5 in fact), even though they are normally used to evaluate modal logic K solvers. It is worth noticing that the performance of a MaxSAT or a PB approach are globally worst than the MSS-extraction approach. However, if we add the symmetry breaking

from Equation 2.1 then the performances become equivalent.

The problem with these benchmarks is that they were originally designed to evaluate K, KT and S4 modal logic solvers. Thus the smallest value in the number of possibles worlds is usually pretty small. To circumvent this problem, we proposed new benchmarks designed for modal logic KT5 and having some kind of structures to allow bigger model.

2.3.2 Results On A Proposed Set Of Benchmarks With Structures

We proposed new benchmarks based on planning with uncertainties in the initial states, to check the performance of the different approaches on structured benchmarks. In such planning problems, some fluent f may be initially true, initially false, or neither. If the fluent f is initially instantiated, we consider it as $\Box f_0$ stating that it is necessary that the fluent f is true at the time $t = 0$, if not, we consider $\Diamond f_0 \wedge \Diamond \neg f_0$ stating that it is possible that f is true at the time $t = 0$ and it is possible that f is false at the time $t = 0$. The rest of the translation is a basic translation from planning to SAT with a bound on the size of the plan except that here, each part of the rest of the translation is necessarily true, thus after a box $\Box$ modality. If the latter case, two different initial situations are possible. As a result, instead of a single initial state s_0, we may have several different initial states, which are consistent with available knowledge about the system (see Eiter et al. (2000) for more details and applications). By construction, all instances considered here have a plan to minimize. Modal logic KT5 formulas are generated with a CEGAR approach Clarke et al. (2003). We increase the value of the bounded-horizon until we reach the smallest value for which there exists a plan as explained in Rintanen (2009).

Method	**block**	**bomb**	**cube**	**omelet**	**ring**	**safe**	**Total**
# Benchs	(20)	(30)	(100)	(30)	(40)	(60)	280
1toN	10	25	**100**	0	**40**	0	175
Nto1	5	10	20	0	20	0	55
Dicho	**20**	20	70	0	**40**	0	150
$1toN_c$	**20**	25	**100**	10	**40**	30	225
$Dicho_c$	**20**	**30**	**100**	**18**	**40**	**34**	**242**
maxHS	17	22	82	0	**40**	5	166
MSCG	**20**	25	72	0	**40**	4	161
MSUnCore	10	22	68	0	38	0	138
NaPS	**20**	**30**	74	2	**40**	8	174
SAT4J	**20**	20	58	0	33	0	131
SCIP	**20**	**30**	**100**	5	**40**	10	200
LBX	12	26	79	0	**40**	0	157
VBS	20	30	100	18	40	34	242

Table 2.2: #instances of Planning solved in modal logic KT5

We performed experimental evaluations on a variety of planning benchmarks. It includes the traditional conformant benchmarks, namely: Bomb-in-the-toilet, Ring, Cube, Omelet and Safe (see Petrick and Bacchus (2002) for more details) modelled here as planning with uncertainties in the initial state. We also performed evaluations on classical benchmarks: Blocksworld, Logistics, and Grid, in which the authors of Hoffmann and Brafman (2006) introduced uncertainty about the initial state. All the benchmarks are available for download[6].

To select the "minimalizable" benchmarks, we set a time-out of 1500 seconds. We managed to solve 28 benchmarks out of the 119 available. We tried other solvers: Spartacus Götzmann et al. (2010) solved 15 instances and SPASS Hustadt et al. (1999) solved 5, with both being a subset of the 28 solved by S52SAT. Our generator has negligible execution times and is available for download[7]. Each of these benchmarks has a plan of size N (where N can be different for each benchmark) which has been verified. We then generated modal logic benchmarks from these instances by fixing the horizon at N, $N+1$, ..., $N+9$ having thus 280 benchmarks, all KT5-satisfiable, to test our minimisation techniques.

As in the MQBF, CNF-KSP and LWB benchmarks before, we can see in Table 2.2 that the use of selectors allows us to solve more benchmarks. But, surprisingly, here the best approach is to use a dichotomic search instead of a linear search from 1 to N. This is mainly due to the size of the smallest model, which is rarely a small number, as it was the case in LWB for example. Moreover, each call to the SAT solver is more time-consuming because the instances are harder to solve in practice. This again reminds us that the benchmarks considered can influence the result obtained.

2.3.3 General Analysis Of The Results Obtained

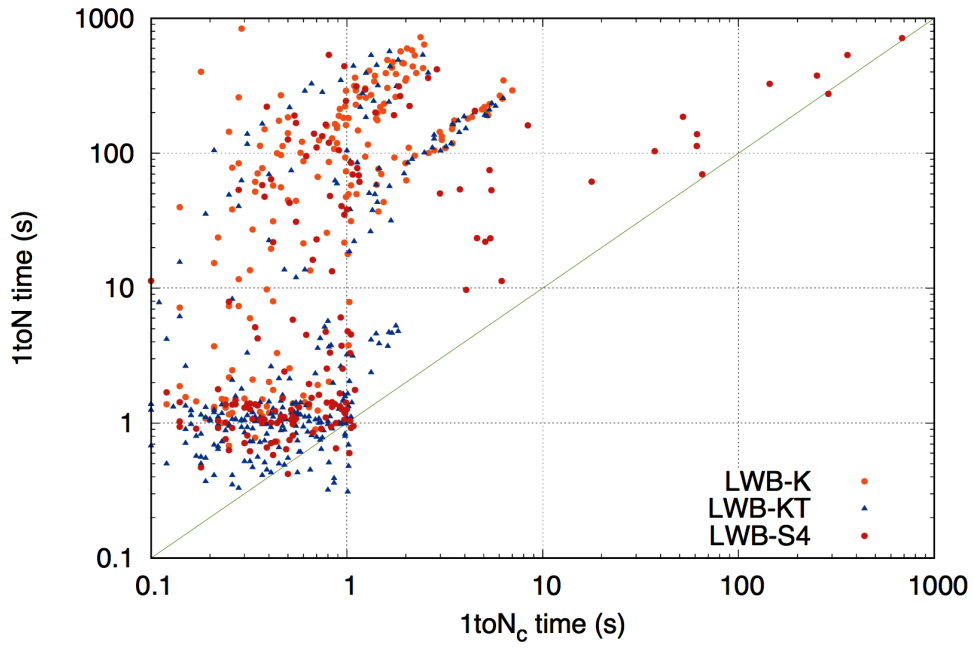


Figure 2.1: Scatter-plot of 1toN vs 1toN$_c$

[6] https://fai.cs.uni-saarland.de/hoffmann/ff/cff-tests.tgz
[7] http://www.cril.fr/~montmirail/planning-to-s5/

For the analysis of the results obtained, we use the one provided in Lagniez et al. (2018a) which are on the same sets of benchmarks but solved in modal logic KT5. First thing we can see is, the use of selectors leads to a speed-up in the runtime of the approach to find the smallest model possible, as depicted in Figure 2.1. Two questions remain though, what is the cost of obtaining the smallest model possible against just one model? and is the search for a small model worth it?

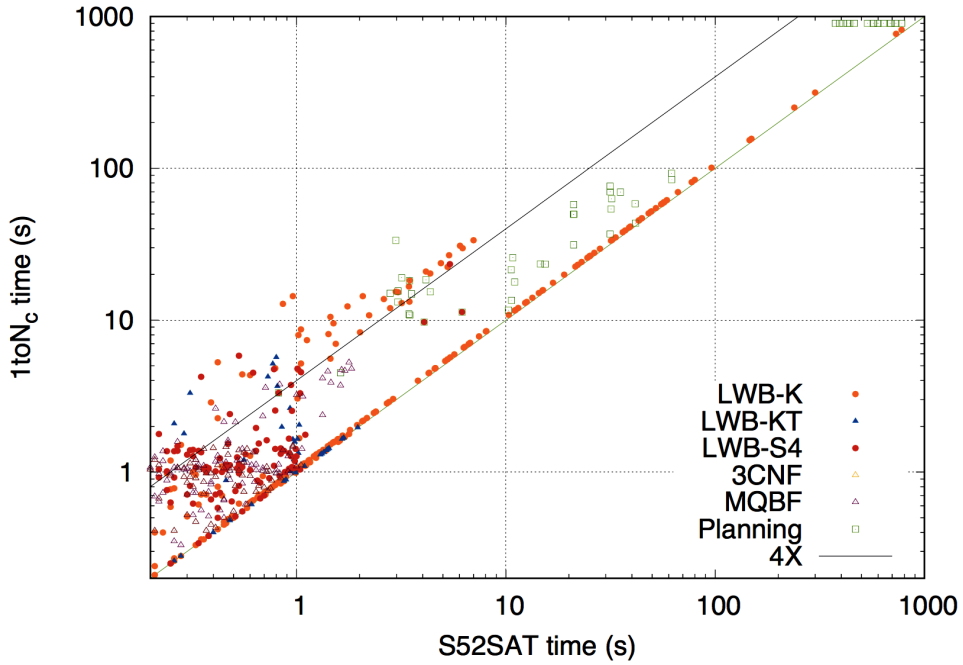


Figure 2.2: S52SAT vs S52SAT-1toN$_c$ (time)

The answer to the first question: is it time-consuming to obtain the smallest model possible? if yes, but for a reasonable cost. The difference in the runtime between the smallest and just one model is depicted in Figure 2.2. As we can see, for approximately four times the runtime, the model can be optimal with respect to the number of possible worlds.

However, it is definitely worth it to search for the smallest model as depicted in Figure 2.3. Problems which have a diamond-degree of hundreds can sometimes be solved with just one world, which is quite a problem if one wants to evaluate solvers. Moreover, one can see on the Figure 2.3 that the Planning benchmarks that we proposed have a better scalability property. The bigger the instance become, the harder it is to solve it and the bigger will be the smallest model possible. Finally, because the translator we proposed is reading a standard format for Planning problems, one can use it to translate any PDDL problems into an InToHyLo formula to evaluate solvers.

Now that we showed how to solve NP modal logic satisfiability problems, it is time to attack the PSPACE modal logic satisfiability problems. But because we assume in this thesis that NP $\neq$ PSPACE, translating in one shot the PSPACE modal logic satisfiability problem into a satisfiability problem in propositional logic will lead to an exponential blow-up that we want to avoid. However, because we still want to rely on a SAT solver, we will explain in the next section the framework that we proposed to solve efficiently PSPACE problems with a SAT solver as an oracle.

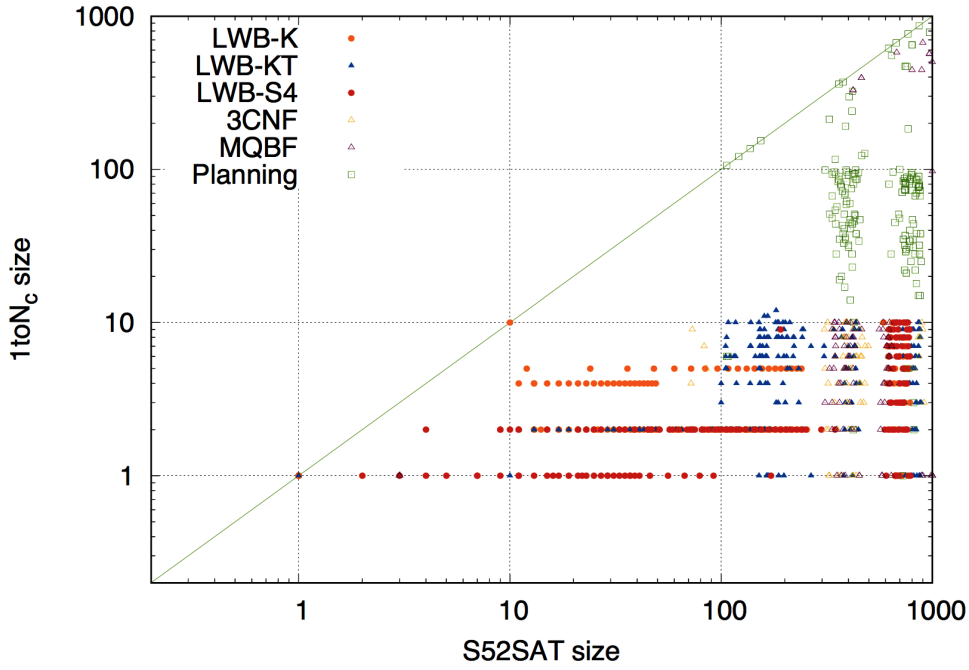


Figure 2.3: S52SAT vs S52SAT-1toN_c (size)

Chapter 3

RECAR: An Abstraction Refinement Procedure

Contents

> "To understand recursion, one must first understand recursion."
>
> *Stephen Hawking*

As we just said in the previous chapter if we want to solve PSPACE problems with a SAT solver, a direct translation may not be the most efficient approach. In the literature, one technique to solve problems which are above NP in the complexity hierarchy is to abstract the problem, *ie.* to solve problems which are easier in practice. Moreover, one could also note that the problems are easier to solve also in theory, because NP oracles are used to approximate PSPACE problems. The drawback being that the abstraction usually does not have the same number of models as the original problem, which must be taken into account.

For this reason, we can classify abstractions into two categories. The *Over Abstractions*, with less models than the original problem (one should read it as a problem *over-constrained*), and the *Under-Abstractions* with more models than the original problem (one should read it as a problem *under-constrained*). The advantage of such approach is that we can decide sometimes the original problem by solving its abstraction.

- If an *over-abstraction* has a model, then the original problem has also a model.
- If an *under-abstraction* has no model, then neither has the original problem.

Another important feature is that the abstraction is refined according to the feedback of the solver.

This makes abstractions procedure extremely successful in various domain such as QBF solving Tentrup (2017), Janota et al. (2016), Software Verification Mordan and Mutilin (2016), Bounded Model Checking Clarke et al. (2003), Bug Detection Wang et al. (2007), Planning Seipp and Helmert (2013), Satisfiability in the Propositional Fragment of First-Order Logic Khasidashvili et al. (2015), Satisfiability Modulo Theory Brummayer and Biere (2009) and many other domains. But even if these domains seems all very different from one another, they all rely on the same principles that we will describe in the next sections.

3.1 Abstraction Functions

The first notion that we will need to present RECAR and its implications is the notion of Abstraction. It is a notion widely studied in the theory of the *Abstract Interpretation* Cousot and Cousot (1977; 1979). This theory studies the abstractions, their soundness and completeness. Even if *Abstract Interpretation* comes from the late 1970s, the mathematical background is much older. It goes back to *Évariste Galois*, a French mathematician, who is at the origins of the Galois connections Ore (1944) which works as follows:

Definition 51 (Galois connection)

Given two lattices $\langle C, \sqsubseteq_C \rangle$ and $\langle A, \sqsubseteq_A \rangle$, a Galois connection noted $C \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} A$ is defined with:

- $\alpha : \langle C, \sqsubseteq_C \rangle \to \langle A, \sqsubseteq_A \rangle$, where α is called an *abstraction* function;
- $\gamma : \langle A, \sqsubseteq_A \rangle \to \langle C, \sqsubseteq_C \rangle$, where γ is called an *concretization* function;

such that $\forall x \in C, \forall y \in A$ we have that $x \sqsubseteq_C \gamma(y) \leftrightarrow \alpha(x) \sqsubseteq_A y$, with $\sqsubseteq$ being the partial orders which compose the lattices.

Note that $\sqsubseteq$ here corresponds in logic to the consequence relation $\models$. From this Galois connection, we can obtain many results:

- $\gamma \circ \alpha$ is extensive (*ie.*, $(\gamma \circ \alpha)(x) \sqsubseteq_C x$) and it represents the information lost by the abstraction;
- α preserves $\sqcup$ and γ preserves $\sqcap$ (with $\sqcup$ and $\sqcap$ being the least and greatest upper-bound in the lattices);
- If $\gamma \circ \alpha$ is the identity, then C and A are isomorphic from the information standpoint;
- Abstraction functions can be composed (*ie.*, an abstraction of an abstraction is still an abstraction).

Galois connection is already used in the SAT/CP Community without being named. For instance, in Constraint Programming, there are used when solving continuous problems. Indeed, as the intervals with real bounds are not representable in a nowadays computer, there are approximated with intervals with floating-point bounds. The reader may find additional work on how to use *Abstract Interpretation* in the CP domain in the book of Marie Pelleau (2015). The transition from one representation to the other forms a Galois connection as shown in the following example.

Example 24

Let $\mathbb{J}$ be the set of intervals with real bounds and $\mathbb{I}$ the set of floating-point bounds intervals. Given two lattices $\langle \mathbb{I}^n, \subset \rangle$ and $\langle \mathbb{J}^n, \subset \rangle$, there exists a Galois connection:

$$\mathbb{J}^n \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} \mathbb{I}^n$$
$$\alpha([x_1, y_1] \times \cdots \times [x_n, y_n]) = [\underline{x_1}, \overline{y_1}] \times \cdots \times [\underline{x_n}, \overline{y_n}]$$
$$\gamma([x_1, y_1] \times \cdots \times [x_n, y_n]) = [x_1, y_1] \times \cdots \times [x_n, y_n]$$

In this example, the abstraction function α transforms a Cartesian product of real bound intervals into a Cartesian product of floating-point bounds intervals. It approximates each real bounds by the closest floating-point number rounded in $\mathbb{F}$ in the corresponding direction. As for the concretization function γ, it is direct since a floating-point number is also a real.

Abstract Interpretation and the *Galois connection* behind it are rich theories, widely used in computer science to formalize reasoning involving the sound and complete abstraction of the semantics of formal systems. We can give as example the semantics of programs describes their possible runtime executions in all possible execution Cousot and Cousot (1992), Formal proofs of program correctness Cousot (2000), Static Analysis Cousot (1981), Model Checking Clarke and Schlingloff (2001) and Counter-Example Guided Abstraction Refinement Clarke et al. (2003) that we will define more precisely in a next section.

In this thesis, we will consider decision problems, more precisely satisfiability problems in different logic. To that aim, as stated at the beginning of this chapter, we will not consider *concretization* functions, but only *abstraction* functions. We will classify *abstraction* functions into two categories the *over-abstraction* and the *under-abstraction*.

3.1.1 Over-Abstraction

The *over-abstraction* functions are functions that takes a problem ϕ in parameter and outputs a problem $\hat{\phi}$ which is more constrained than ϕ. The property being that if $\hat{\phi}$ has a model, then so does ϕ. Formally they can be defined as follows:

Definition 52 (Over-Abstraction Function)

For a problem ϕ defined in language $\mathcal{L}^1$, an over-abstraction function $\hat{\alpha}$ from $\mathcal{L}^1$ to $\mathcal{L}^2$, with $\phi \in \mathcal{L}^1$, is defined as follows:

$$\begin{aligned} \hat{\alpha} \colon \mathcal{L}^1 &\to \mathcal{L}^2 \\ \phi &\mapsto \hat{\phi} \end{aligned}$$

This function preserves that if there is M_2 such that $M_2 \models_2 \hat{\phi}$, then there is M_1 such that $M_1 \models_1 \phi$. $\models_i$ denotes the satisfiability relation in $\mathcal{L}^i$, thus M_1 is a model for $\mathcal{L}^1$ and M_2 is a model for $\mathcal{L}^2$.

This kind of functions are well known in the Planning community, by those who solves Planning problems with a SAT-based planner Rintanen (2009). Indeed, it is known that Planning problems may have plan of exponential size. But many approaches translate the original problem into a SAT problem by bounding the size of the authorized plan. Like this, if the SAT solver finds a model, then it means that it exists a plan of size n, if not it just means that we need to increase the size of the authorized plan.

3.1.2 Under-Abstraction

The *under-abstraction* functions are functions that takes a problem ϕ in parameter and outputs a problem $\check{\phi}$ which is less constrainted than ϕ. The property being that if $\check{\phi}$ has no model, then so does ϕ. Formally they can be defined as follows:

Definition 53 (Under-Abstraction Function)

For a problem ϕ defined in language $\mathcal{L}^1$, an under-abstraction function $\check{\alpha}$ from $\mathcal{L}^1$ to $\mathcal{L}^2$, with $\phi \in \mathcal{L}^1$, is defined as follows:

$$\begin{aligned} \check{\alpha} \colon \mathcal{L}^1 &\to \mathcal{L}^2 \\ \phi &\mapsto \check{\phi} \end{aligned}$$

This function preserves that if there is M_1 such that $M_1 \models_1 \phi$ then there is M_2 such that $M_2 \models_2 \check{\phi}$. $\models_i$ denotes the satisfiability relation in $\mathcal{L}^i$, thus M_1 is a model for $\mathcal{L}^1$ and M_2 is a model for $\mathcal{L}^2$.

This kind of functions are well known in the Constraint community, but in this field, they are called *Relaxation* Davies and Bacchus (2013). Indeed, it is common in this field to forget some constraints. But then, if the problem is unsatisfiable, then for sure it is also unsatisfiable with the missing constraints.

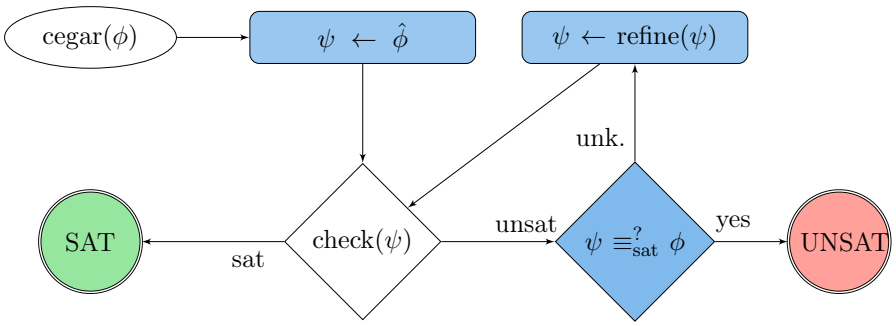


Figure 3.1: The CEGAR framework with over-abstraction

Using the two kind of functions that we just presented is done in many fields in many different ways, but Clarke et al. (2003) tried to framework the use of them in a procedure called CEGAR which stands for *Counter-Example Guided Abstraction Refinement*. It is the procedure which is widely use and on which RECAR is based, so let us it introduce it in the next section.

3.2 Counter-Example Guided Abstraction Refinement

As we just said, CEGAR is a framework to make "user-friendly" the use of over/under abstractions functions to solve problems. It takes into consideration decision problems and there are two kinds, according to which kind of abstraction functions you are using.

3.2.1 CEGAR-over

Let us first describe the CEGAR-over, *ie.* the CEGAR framework instantiated with over-abstraction functions. An example of a CEGAR using over-abstractions is given on Fig. 3.1. It receives a formula ϕ as input and computes an over-abstraction ψ. Then it uses an oracle (check) to check whether ψ is satisfiable. If so it concludes that ϕ is satisfiable by construction of an over-abstraction function. Otherwise, ψ is refined, *ie.* it gets closer to ϕ, until it is satisfiable, or until the refined over-abstraction is detected to be equisatisfiable to ϕ, denoted $\psi \equiv_{\text{sat}} \phi$, (*ie.* $\exists M, M \models_1 \psi$ iff $\exists M', M' \models_2 \phi$)[8], where it concludes that ϕ is unsatisfiable. In the following, $\phi \equiv^{?}_{\text{sat}} \psi$ means an incomplete efficient equi-satisfiability test which returns yes or unknown. An illustration of how the search space of the different over-abstraction are becoming closer and closer from the search space of the original problem is given in Figure 3.2.

$$\boxed{\hat{\phi}} \subseteq \boxed{\text{refine}(\hat{\phi})} \subseteq \boxed{\text{refine}^2(\hat{\phi})} \subseteq \cdots \subseteq \boxed{\text{refine}^n(\hat{\phi})} = \boxed{\phi}$$

Figure 3.2: How the over-abstractions search-space become closer to the original problem's one

This framework is widely used, for example, for solving Planning problems with a SAT solver Rintanen (2009). Indeed, it is known that the worst-case possible is a plan of exponential size. But in practice, the worst-case is rare, thus the technique to abstract the problem into a SAT

[8] $\models_1$ and $\models_2$ denote possibly different consequence relations (for propositional logic and modal logic K for instance).

problem asking the question "can we find a plan of size 0?" if it is the case, then the problem is decided. If not, the size of plan is refine and a new SAT problem is created : "can we find a plan of size 1?", *etc.* Now that we describe the CEGAR-over, let us go to the description of its under-counterpart.

3.2.2 CEGAR-under

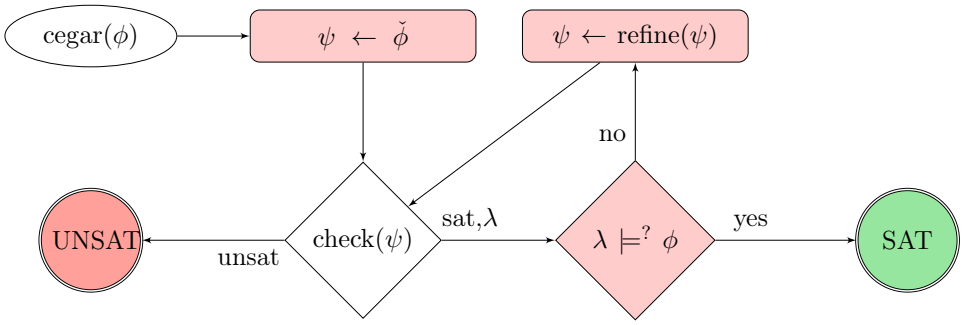


Figure 3.3: The CEGAR framework with under-abstraction

Indeed, the CEGAR framework can also be instantiated with under-abstraction functions. In fact, it is much more common to find it with such abstraction when it is about solving NP problems with a SAT solver. As a matter of fact, CEGAR-under is more used for NP problems whereas CEGAR-over is more used for PSPACE problems even if, from the framework, nothing is forcing it.

An example of a CEGAR using under-abstractions is given on Fig. 3.3. It receives a formula ϕ as input and computes an under-abstraction ψ. Then it uses an oracle (check) to check whether ψ is unsatisfiable. If so it concludes that ϕ is unsatisfiable by construction of an under-abstraction function. Otherwise, ψ is refined, *ie.* it gets closer to ϕ, until it is satisfiable, or until the model λ for the refined under-abstraction is detected to be also a model for ϕ, denoted $\lambda \models^? \phi$[9], where it concludes that ϕ is satisfiable.

An illustration of how the search space of the different under-abstraction are becoming closer and closer from the search space of the original problem is given in Figure 3.4.

$$\boxed{\check{\phi}} \supseteq \boxed{\text{refine}(\check{\phi})} \supseteq \boxed{\text{refine}^2(\check{\phi})} \supseteq \cdots \supseteq \boxed{\text{refine}^n(\check{\phi})} = \boxed{\phi}$$

Figure 3.4: How the under-abstractions search-space become closer to the original problem's one

Let us illustrate it with an article that we wrote for the conference CP 2018 Glorian et al. (2018), which is not related to modal logics, but which is definitely a CEGAR-under approach of the problem.

3.2.3 CEGAR-under For $\mathcal{RCC}8$

The $\mathcal{RCC}8$ language is a widely-studied formalism for describing topological arrangements of spatial regions. Two fundamental reasoning problems that are associated with $\mathcal{RCC}8$ are the

[9] *ie.* $\exists \lambda \models \psi$ then $\exists f$ s.t. $f(\lambda) \models \phi$

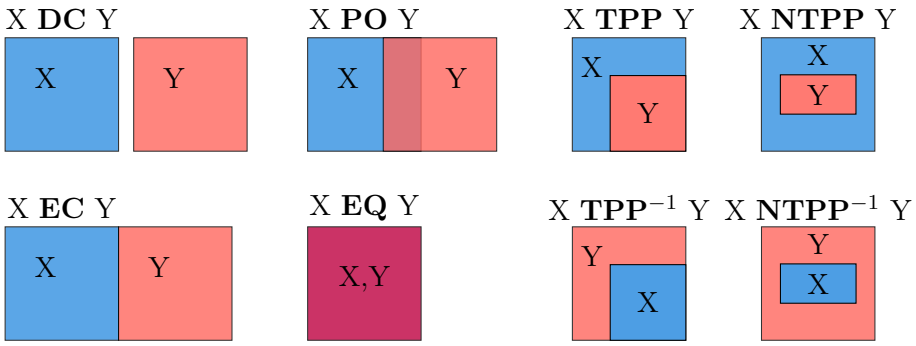


Figure 3.5: Illustration of the base $\mathcal{RCC}8$ relations

problems of *satisfiability* and *realization*. Given a qualitative constraint network (QCN) of $\mathcal{RCC}8$, the satisfiability problem is deciding whether it is possible to assign regions to the spatial variables of the QCN in such a way that all of its constraints are satisfied (*solution*). The realization problem is producing an actual spatial model that can serve as a solution.

$\mathcal{RCC}8$ is based on the following eight relations: equals (**EQ**), partially overlaps (**PO**), externally connected (**EC**), disconnected (**DC**), tangential proper part (**TPP**) and its inverse (**TPP**$^{-1}$), and non-tangential proper part (**NTPP**) and its inverse (**NTPP**$^{-1}$). These spatial relations are illustrated in Fig. 3.5.

Given a qualitative constraint network (QCN) of $\mathcal{RCC}8$, we are particularly interested in its *satisfiability* problem, which is the problem of deciding if there exists a spatial interpretation of the variables of the QCN that satisfies its constraints. The satisfiability problem for $\mathcal{RCC}8$ (and $\mathcal{RCC}5$) is NP-complete Renz and Nebel (1999). Once a QCN of $\mathcal{RCC}8$ is known to be satisfiable, one typically deals with the *realization problem* in order to produce an actual spatial model that can serve as a solution, which is a tractable problem (see Li (2006)).

Research in $\mathcal{RCC}8$ usually focuses either on symbolically checking the satisfiability of a QCN or on presenting a method to realize (valuate) a satisfiable QCN. To the best of our knowledge, combining those two lines of research in an interrelating manner has not been considered in the literature, as the first line deals with native constraint-based methods, and the second one with rich mathematical structures that are difficult to implement.

We bind those two lines of research together in a unified and homogeneous approach by means of an incremental SAT-based technique known as CEGAR. The idea is as follows: instead of creating an equisatisfiable propositional formula as per the state of the art Huang et al. (2013), we generate an *under-approximation* formula (a formula which is under-constrained, also called *relaxation* in other domains). Meaning, if an under-approximation is unsatisfiable, then by construction the original formula is unsatisfiable; otherwise, the SAT solver outputs a model that can then be checked. It could be the case that the approach is lucky and the model of the under-approximation is also a model of the original formula, in which case the problem is decided. In general, the under-approximation is constantly refined, *i.e.*, it comes closer to the original formula and, in the worst-case, it will eventually become equisatisfiable with the original formula after a finite number of refinements. Let us give us a small preliminary about $\mathcal{RCC}8$ before presenting our CEGAR-under encoding.

Region Connection Calculus

The Region Connection Calculus ($\mathcal{RCC}$) Randell et al. (1992) is a first order theory for representing and reasoning about mereotopological information between regions of some topological space. Its relations are based on a connectedness relation **C**. In particular, using **C**, a set of binary relations is defined. From this set, the $\mathcal{RCC}8$ fragment can be extracted: { **DC**, **EC**, **PO**, **EQ**, **TPP**, **NTPP**, $\mathbf{TPP}^{-1}$, $\mathbf{NTPP}^{-1}$ }. These eight ones are jointly exhaustive and pairwise disjoint, meaning that only one of those can hold between any two regions. As noted in the introduction, this fragment (illustrated in Fig.3.5), will be referred to simply as $\mathcal{RCC}8$ for convenience.

We can view regions in $\mathcal{RCC}$ as non-empty regular subsets of some topological space that do not have to be internally connected and do not have a particular dimension, but that are usually required to be *closed* Renz (2002) (*i.e.*, the subsets equal the closure of their respective interiors). Let $R(\mathcal{X})$ denote the set of all regions of some topological space $\mathcal{X}$. Then, we can have the following interpretation for the basic relations of $\mathcal{RCC}8$, where $\mathbf{R}_i$ denotes the interpretation of **R** for two instantiated region variables. Semantically, binary relation **R** contains all the possible instantiations of its pair of region variables.

Definition 54 (Set Notation of $\mathcal{RCC}8$)

Given two regions X and Y in $R(\mathcal{X})$, then:[a]

$\mathbf{EQ}_i(X,Y)$	iff	$X = Y$
$\mathbf{DC}_i(X,Y)$	iff	$X \cap Y = \emptyset$
$\mathbf{EC}_i(X,Y)$	iff	$\mathring{X} \cap \mathring{Y} = \emptyset, X \cap Y \neq \emptyset$
$\mathbf{PO}_i(X,Y)$	iff	$\mathring{X} \cap \mathring{Y} \neq \emptyset, X \not\subseteq Y, Y \not\subseteq X$
$\mathbf{TPP}_i(X,Y)$	iff	$X \subset Y, X \not\subseteq \mathring{Y}$
$\mathbf{TPP}_i^{-1}(X,Y)$	iff	$Y \subset X, Y \not\subseteq \mathring{X}$
$\mathbf{NTPP}_i(X,Y)$	iff	$X \subset \mathring{Y}$
$\mathbf{NTPP}_i^{-1}(X,Y)$	iff	$Y \subset \mathring{X}$

[a] $\mathring{A}$ denotes the interior of A.

Given two basic relations **R** and **S** of $\mathcal{RCC}8$ that involve the pair of variables (i,j) and (j,k) respectively, the *weak composition* of **R** and **S**, denoted by *CT(**R**,**S**)*, yields the strongest relation of $\mathcal{RCC}8$ that contains $\mathbf{R} \circ \mathbf{S}$, *i.e.*, it yields the smallest set of basic relations such that, each of which can be satisfied by the instantiated variables i and k for some possible instantiation of variables i, j, k with respect to relations **R** and **S**. We remind the following definition of the weak composition operation from Renz and Ligozat (2005):

Definition 55 (Weak Composition *CT*)

For two basic relations **R**, **S** of $\mathcal{RCC}8$, their weak composition *CT(**R**,**S**)* is defined to be the smallest subset $\{\mathbf{T}_1, \mathbf{T}_2, \ldots, \mathbf{T}_n\}$ of $2^{\mathcal{RCC}8}$ such that $\mathbf{T}_i \cap (\mathbf{R} \circ \mathbf{S}) \neq \emptyset \ \forall i \in \{1, \ldots, n\}$.

CT	**DC**	**EC**	**PO**	**TPP**	**NTPP**	**TPP^{-1}**	**NTPP^{-1}**	**EQ**
DC	*	DC EC PO TPP NTPP	DC EC PO TPP NTPP	DC EC PO TPP NTPP	DC EC PO TPP NTPP	DC	DC	DC
EC	DC EC PO TPP^{-1} NTPP^{-1}	DC EC PO TPP TPP^{-1} EQ	DC EC PO TPP NTPP	EC PO TPP NTPP	PO TPP NTPP	DC EC	DC	EC
PO	DC EC PO TPP^{-1} NTPP^{-1}	DC EC PO TPP^{-1} NTPP^{-1}	*	PO TPP NTPP	PO TPP NTPP	DC EC PO TPP^{-1} NTPP^{-1}	DC EC PO TPP^{-1} NTPP^{-1}	PO
TPP	DC	DC EC	DC EC PO TPP NTPP	TPP NTPP	NTPP	DC EC PO TPP TPP^{-1} EQ	DC EC PO TPP^{-1} NTPP^{-1}	TPP
NTPP	DC	DC	DC EC PO TPP NTPP	NTPP	NTPP	DC EC PO TPP NTPP	*	NTPP
TPP^{-1}	DC EC PO TPP^{-1} NTPP^{-1}	EC PO TPP^{-1} NTPP^{-1}	PO TPP^{-1} NTPP^{-1}	PO TPP TPP^{-1} EQ	PO TPP NTPP	TPP^{-1} NTPP^{-1}	NTPP^{-1}	TPP^{-1}
NTPP^{-1}	DC EC PO TPP^{-1} NTPP^{-1}	PO TPP^{-1} NTPP^{-1}	PO TPP^{-1} NTPP^{-1}	PO TPP^{-1} NTPP^{-1}	PO TPP NTPP TPP^{-1} NTPP^{-1} EQ	NTPP^{-1}	NTPP^{-1}	NTPP^{-1}
EQ	DC	EC	PO	TPP	NTPP	TPP^{-1}	NTPP^{-1}	EQ

Table 3.1: The $\mathcal{RCC}8$ *CT*, where * specifies the universal relation

The result of the weak composition operation for each possible pair of basic relations of $\mathcal{RCC}8$ is provided by a dedicated table, called the *weak composition table* Li and Ying (2003) ($\mathcal{RCC}8$ *CT* for short), shown in Table 3.1. The weak composition operation for two general $\mathcal{RCC}8$ relations can be computed by unifying the results (sets) of the weak composition operations for all ordered pairs of basic relations that involve a basic relation from the first general relation and a basic relation from the second one. Henceforward, a general $\mathcal{RCC}8$ relation will be represented by the set of its basic relations.

In order to concretely capture the qualitative spatial information that is entailed by a knowl-

edge base of $\mathcal{RCC}8$ relations, we will use the notion of a Qualitative Constraint Network (QCN), defined as follows:

Definition 56 (Qualitative Constraint Networks (**QCN**))

A QCN of $\mathcal{RCC}8$ is a pair $\mathcal{N} = (V, C)$ where V is a non-empty finite set of variables (each one corresponding to a region), and C is a mapping associating a relation $C(v, v') \in 2^{\mathcal{RCC}8}$ with each pair (v, v') of $V \times V$. Further, mapping C is such that $C(v, v) \subseteq \{\mathbf{EQ}\}$ and $C(v, v') = (C(v', v))^{-1}$.

Concerning a QCN $\mathcal{N} = (V, C)$, we have the following definitions: An instantiation of V is a mapping σ defined from V to the domain $R(\mathcal{X})$. A solution (*realization*) σ of $\mathcal{N}$ is an instantiation of V such that for every pair (v, v') of variables in V, $(\sigma(v), \sigma(v'))$ satisfies $C(v, v')$, i.e., there exists a base relation $b \in C(v, v')$ such that $(\sigma(v), \sigma(v')) \in b$. $\mathcal{N}$ is satisfiable if and only if it admits a solution. The constraint graph of a QCN $\mathcal{N}$ is the graph (V, E), denoted by G_N, for which we have that $\{v, v'\} \in E$ if and only if $C(v, v') \neq \mathcal{RCC}8$ (i.e., $C(v, v')$ corresponds to a non-universal relation) and $v \neq v'$.

SAT Encoding of the $\mathcal{RCC}8$ Satisfiability Problem

To obtain a SAT encoding of the $\mathcal{RCC}8$ satisfiability problem, we need to define how to translate the different possible relations. We will represent a region i as a set of four variables $\{x_i^-, y_i^-, x_i^+, y_i^+\}$ as illustrated in Fig. 3.6.

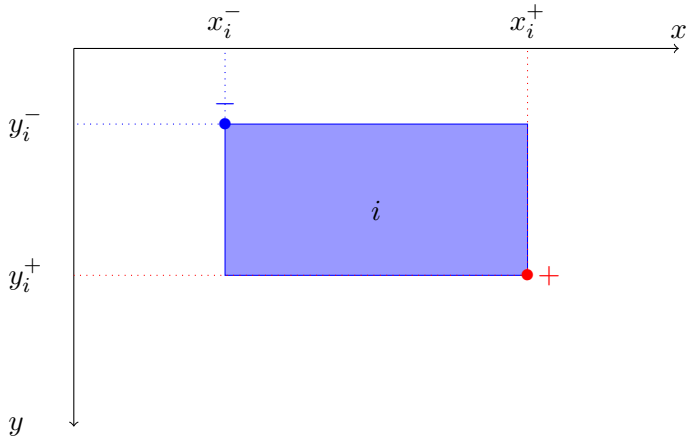


Figure 3.6: Illustration of how a region is represented

All the possibles cases for every relation may be found and proved, along with their link with Point Algebra, in (Long 2017, Table 6.2). From this encoding, we can then propose the following SAT encoding which translates all the edges possible:

Definition 57 (SAT Translation – `translate`)

For all relations R in all the given edges $(\mathtt{i}, \mathtt{j})$ of the input problem $\mathcal{N}$ we have:

$$\mathtt{translate}(\mathcal{N}) := \bigwedge_{\forall(R,i,j)\in\mathcal{N}} \mathtt{translate}(R, i, j)$$

Then from (Long 2017, Table 6.2), if we want to translate for example the relation **EC** between nodes `i` and `j` (the procedure is similar for other $\mathcal{RCC}8$ relations), we will have the following SAT encoding as per Def. 57:

Definition 58 (SAT Translation of **EC** on the edge i-j)

$$\mathtt{translate}(\mathbf{EC}, i, j) := \mathbf{EC}(i, j) \rightarrow (\mathbf{EC}_r(i, j) \vee \mathbf{EC}_l(i, j) \vee \mathbf{EC}_u(i, j) \vee \mathbf{EC}_d(i, j))$$

From this definition, we can see that the relation **EC** for the edge $(\mathtt{i}, \mathtt{j})$ can only be satisfied by 4 different cases, viz., *left*, *right*, *up*, *down*. Each case is defined as follows:

$$\begin{aligned}\mathbf{EC}_r(i,j) \rightarrow &((x_i^- < x_j^-) \wedge (x_i^- < x_j^+)) \quad \wedge\\ &((x_i^- = x_j^+) \wedge (x_i^+ < x_j^+)) \quad \wedge\\ &((y_i^- < y_j^+) \vee (y_i^- < y_j^+)) \quad \wedge\\ &((y_i^+ < y_j^-) \vee (y_i^+ < y_j^-))\end{aligned}$$

$$\begin{aligned}\mathbf{EC}_u(i,j) \rightarrow &((x_i^- < x_j^-) \vee (x_i^- = x_j^-)) \quad \wedge\\ &((x_i^- > x_j^+) \vee (x_i^- = x_j^+)) \quad \wedge\\ &((y_i^- < y_j^-) \wedge (y_i^- < y_i^+)) \quad \wedge\\ &((y_i^+ = y_j^-) \wedge (y_i^+ < y_j^+))\end{aligned}$$

$$\begin{aligned}\mathbf{EC}_l(i,j) \rightarrow &((x_i^- > x_j^-) \wedge (x_i^- = x_j^+)) \quad \wedge\\ &((x_i^- > x_j^+) \wedge (x_i^+ > x_j^+)) \quad \wedge\\ &((y_i^- < y_j^+) \vee (y_i^- < y_j^+)) \quad \wedge\\ &((y_i^+ < y_j^-) \vee (y_i^+ < y_j^-))\end{aligned}$$

$$\begin{aligned}\mathbf{EC}_d(i,j) \rightarrow &((x_i^- < x_j^-) \vee (x_i^- = x_j^-)) \quad \wedge\\ &((x_i^- > x_j^+) \vee (x_i^- = x_j^+)) \quad \wedge\\ &((y_i^- > y_j^-) \wedge (y_i^- = y_i^+)) \quad \wedge\\ &((y_i^+ > y_j^-) \wedge (y_i^+ > y_j^+))\end{aligned}$$

The inverse relations are defined as usual: $\mathbf{TPP}^{-1}(i, j) = \mathbf{TPP}(j, i)$ and $\mathbf{NTPP}^{-1}(i, j) = \mathbf{NTPP}(j, i)$. For every node in the QCN with N nodes that we want to solve, we will add the following constraint assuring that all the point coordinates are in good order:

$$\bigwedge_{i=1}^{N}((x_i^- < x_i^+) \wedge (y_i^- < y_i^+))$$

We want to point out that, if the propositional variable $(A < B)$ is true, then the variables $(A > B)$ and $(A = B)$ are false. To express this, we use the following clauses:

$$\texttt{AMO} := \bigwedge_{a\in\{x,y\}} \bigwedge_{c_1\in\{-,+\}} \bigwedge_{c_2\in\{-,+\}} \bigwedge_{i=1}^{N} \bigwedge_{j=1}^{N} \begin{pmatrix} ((a_i^{c_1} < a_j^{c_2}) \ \vee \ (a_i^{c_1} = a_j^{c_2}) \ \vee \ (a_i^{c_1} > a_j^{c_2})) \wedge \\ (\neg(a_i^{c_1} < a_j^{c_2}) \ \vee \ \neg(a_i^{c_1} = a_j^{c_2})) \wedge \\ (\neg(a_i^{c_1} < a_j^{c_2}) \ \vee \ \neg(a_i^{c_1} > a_j^{c_2})) \wedge \\ (\neg(a_i^{c_1} = a_j^{c_2}) \ \vee \ \neg(a_i^{c_1} > a_j^{c_2})) \wedge \end{pmatrix} \tag{3.1}$$

Thanks to Equation 3.1 (AMO – At Most One), we can thus replace, for example, in **EC** (i,j) (u), $(x_i^- < x_j^+) \vee (x_i^- = x_j^+)$ by $\neg(x_i^- > x_j^+)$. The same applies for all the disjunctions in (Long 2017, Table 6.2). Last but not least, we want to ensure the transitivity of the relations on all the possible coordinates; this will have the biggest impact on the size of the generated CNF. For all the triplets (i, j, k) in a triangulation (chordal completion of the constraint graph of an input QCN), we must add the following rules for every combination (c1, c2) that can be assured by transitivity $\in \{(-,-),(-,+),(+,-),(+,+)\}$ and for both axis $a \in \{x, y\}$:

$$\texttt{transitivity}(i,j,k) := \bigwedge \begin{pmatrix} ((a_i^{c1} = a_j^{c1}) \wedge \ (a_j^{c1} = a_k^{c2})) \rightarrow (a_i^{c1} = a_k^{c2}) \\ ((a_i^{c1} < a_j^{c1}) \wedge \neg(a_j^{c1} > a_k^{c2})) \rightarrow (a_i^{c1} < a_k^{c2}) \\ ((a_i^{c1} > a_j^{c1}) \wedge \neg(a_j^{c1} < a_k^{c2})) \rightarrow (a_i^{c1} > a_k^{c2}) \\ ((a_j^{c1} > a_k^{c2}) \wedge \neg(a_i^{c1} < a_j^{c1})) \rightarrow (a_i^{c1} > a_k^{c2}) \\ ((a_j^{c1} < a_k^{c2}) \wedge \neg(a_i^{c1} > a_j^{c1})) \rightarrow (a_i^{c1} < a_k^{c2}) \end{pmatrix}$$

We will not enter the details of how a graph can be made chordal; it is a standard procedure and we redirect the reader to Savický and Vomlel (2009) for more information about how it can be done. It is worth noting that triangulating a graph can take linear time in the size of the output chordal graph. Before moving to the CEGAR part, we need to prove that the encoding we designed is sound and complete.

Theorem 19

Let $\mathcal{N} = (V, C)$ be a QCN of RCC8, and G a chordal supergraph of the constraint graph of $\mathcal{N}$. If $\texttt{toSAT}(\mathcal{N})$ is defined as follows:

$$\texttt{toSAT}(\mathcal{N}) := \texttt{translate}(\mathcal{N}) \wedge \texttt{AMO} \wedge \bigwedge_{i=1}^{N} ((x_i^- < x_i^+) \wedge (y_i^- < y_i^+)) \wedge \bigwedge_{(i,j,k)\in G} \texttt{transitivity}(i,j,k)$$

then $\texttt{toSAT}(\mathcal{N})$ is equisatisfiable with $\mathcal{N}$.

Proof. We need to show that $\texttt{toSAT}(\mathcal{N})$ is equisatisfiable with $\mathcal{N}$. In order to do so, we split $\texttt{toSAT}(\mathcal{N})$ in two parts. The first one $(\texttt{translate}(\mathcal{N}) \wedge \texttt{AMO} \wedge \bigwedge_{i=1}^{N} ((x_i^- < x_i^+) \wedge (y_i^- < y_i^+)))$ is obviously the input problem; this representation comes from Long (2017) and the relation with Point Algebra (PA). As proven in Sioutis and Koubarakis (2012), it is enough to check the path

consistency with respect to the chordal graph, so the real difficulty of this proof is demonstrating why by adding a finite number of transitivity constraints does the translation become equisatisfiable. The intuition is that, each time we add a transitivity constraint $\texttt{transitivity}(i, j, k)$, we force the SAT solver to find only relations in this triangle which match the weak composition table CT (Table 3.1). For this purpose, we need to enumerate all the cases pertaining to the CT and show that, each time, the transitivity constraints force the solver to find only relations allowed by the CT. Let us consider the following case: (i,j) has the relation **EQ** and (j,k) has the relation **NTPP**. Then by the CT, the transitivity constraints should force to find **NTPP** on (i,k). Because of what we assume true, we have the following propositional variables assigned to true: $(x_i^- = x_j^-), (y_i^- = y_j^-), (x_i^- < x_j^+), (y_i^- < y_j^+), (x_i^+ > x_j^-), (y_i^+ > y_j^-), (x_i^+ = x_j^+), (y_i^+ = y_j^+)$ (which encodes $\mathbf{EQ}(i, j)$) and $(x_j^- > x_k^-), (y_j^- > y_k^-), (x_j^- < x_k^-), (y_j^- < y_k^+), (x_j^- > x_k^+), (y_j^- > y_k^-), (x_j^- < x_k^+), (y_j^- < y_k^+)$ (which encodes $\mathbf{NTPP}(j, k)$). Then, we want to obtain the following:

$$
\begin{array}{ll}
(1.a)\ (x_i^- > x_k^-) & (1.b)\ (y_i^- > y_k^-) \\
(2.a)\ (x_j^- < x_k^-) & (2.b)\ (y_j^- < y_k^+) \\
(3.a)\ (x_j^- > x_k^+) & (3.b)\ (y_j^- > y_k^-) \\
(4.a)\ (x_j^- < x_k^+) & (4.b)\ (y_j^- < y_k^+)
\end{array}
$$

1.a We have $(x_i^- = x_j^-)$ and $(x_j^- > x_k^-)$. Due to the transitivity constraint $\texttt{transitivity}(i, j, k)$ at one point, we add the clause: $((x_i^- = x_j^-) \wedge \neg(x_j^- < x_k^-) \rightarrow (x_i^- > x_k^-))$. Then, because of AMO, we have $((x_j^- > x_k^-) \rightarrow \neg(x_j^- < x_k^-))$. Thus we have $(x_i^- > x_k^-)$.

2.a We have $(x_i^- = x_j^-)$ and $(x_j^- < x_k^+)$. Due to the transitivity constraint $\texttt{transitivity}(i, j, k)$ at one point, we add the clause: $(\neg(x_i^- < x_j^-) \wedge (x_j^- < x_k^+) \rightarrow (x_i^- < x_k^+))$. Then, because of AMO, we have $((x_i^- = x_j^-) \rightarrow \neg(x_i^- < x_j^-))$. Thus we have $(x_i^- < x_k^+)$.

3.a We have $(x_i^+ = x_j^+)$ and $(x_j^+ > x_k^-)$. Due to the transitivity constraint $\texttt{transitivity}(i, j, k)$ at one point, we add the clause: $(\neg(x_i^+ < x_j^+) \wedge (x_j^+ > x_k^-) \rightarrow (x_i^+ > x_k^-))$. Then, because of AMO, we have $((x_i^+ = x_j^+) \rightarrow \neg(x_i^+ < x_j^+))$. Thus we have $(x_i^+ > x_k^-)$.

4.a We have $(x_i^+ = x_j^+)$ and $(x_j^+ < x_k^+)$. Due to the transitivity constraint $\texttt{transitivity}(i, j, k)$ at one point, we add the clause: $(\neg(x_i^+ > x_j^+) \wedge (x_j^+ < x_k^+) \rightarrow (x_i^+ > x_k^+))$. Then, because of AMO, we have $((x_i^+ = x_j^+) \rightarrow \neg(x_i^+ > x_j^+))$. Thus we have $(x_i^+ < x_k^+)$.

The cases on the y-axis (1.b, 2.b, 3.b and 4.b) are similar to the one on the x-axis. From this point forward, we just have to enumerate in that way all the possible cases pertaining to the weak composition table. This would be extremely space-consuming so we leave it as exercise for the reader. q.e.d □

We just saw that if we encode all the transitivity cases for every triangle in the respective chordal graph in accordance with our description, we obtain an equisatisfiable SAT encoding. However, when we take a closer look at what can be time-consuming, function `transitivity` is exactly what we want to avoid at any cost due to the size of its encoding, and that is why we propose a CEGAR approach to circumvent it.

Translating parsimoniously the transitivity constraints

As we explained earlier, the function `transitivity` can be costly and it hence needs to be avoided if we want to have a competitive approach. This is exactly the hypothesis on which our CEGAR approach rests. Let us take the example illustrated in Fig. 3.7, a $\mathcal{RCC}8$ problem with 5 nodes and 5 relations given as input **(the relations are shown in black in Fig. 3.7)**. Thus, when we translate this problem into a propositional logic formula, we obtain the following formula:

$$\begin{aligned}\texttt{under}(\mathcal{N}) = \ & \texttt{translate}(\mathbf{EC}, 0, 1) \wedge \texttt{translate}(\mathbf{EC}, 1, 2) \\ & \wedge \texttt{translate}(\mathbf{EC}, 2, 3) \\ & \wedge \texttt{translate}(\mathbf{EQ}, 3, 4) \wedge \texttt{translate}(\mathbf{DC}, 3, 4) \\ & \wedge \texttt{translate}(\mathbf{EQ}, 4, 0) \wedge \texttt{translate}(\mathbf{DC}, 4, 0) \\ & \wedge EC_{0,1} \wedge EC_{1,2} \wedge EC_{2,3} \wedge (EQ_{3,4} \vee DC_{3,4}) \wedge (EQ_{4,0} \vee DC_{4,0}) \\ & \wedge \texttt{AMO} \wedge \bigwedge_{i=0}^{4} ((x_i^- < x_i^+) \wedge (y_i^- < y_i^+))\end{aligned}$$

Theorem 20

Let $\mathcal{N}$ be a QCN, then `under`$(\mathcal{N})$ is an under-abstraction of $\mathcal{N}$ (*i.e.*, it has at least the same amount of models).

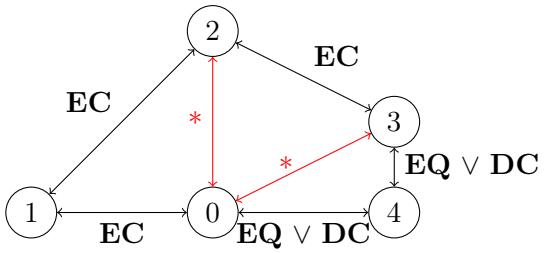


Figure 3.7: An example of the constraint graph of an $\mathcal{RCC}8$ problem $\mathcal{N}$, (the labels denote the corresponding $\mathcal{RCC}8$ relations), in red the edges added to make the graph chordal

Proof. `under`$(\mathcal{N})$ is a subset of clauses of the equisatisfiable encoding (Thm. 19). Thus if under$(\mathcal{N})$ is unsatisfiable, then $\mathcal{N}$ is also unsatisfiable by definition of the logical conjunction. q.e.d □

At this point, the translation is an under-abstraction of the original problem, *i.e.*, if it is unsatisfiable, then for sure the problem is unsatisfiable, but a model of this translation does not imply that it exists a model for the original problem. The CEGAR Assumption (2) is respected

by construction of the translation, to obtain this equisatisfiability we need to have:

$$\begin{aligned}\text{toSAT}(\mathcal{N}) = \ & \texttt{under}(\mathcal{N}) \\ & \wedge \texttt{transitivity}(0,1,2) \\ & \wedge \texttt{transitivity}(0,2,3) \\ & \wedge \texttt{transitivity}(0,3,4)\end{aligned}$$

As the number of triangles in a chordal graph is bounded by a number N (worst case: $N = |V|^3$ when the graph is complete), we can now easily see that after we translate the transitivity of each triangle (an operation that we will call a refinement in what follows), we can refine the problem N times and obtain an equisatisfiable formula.

We need to find a way to check efficiently if a returned model of the under-abstraction is also a model of the original formula. For this purpose, we used the algorithm Directional Path Consistency (DPC) presented in Dechter et al. (1991), Long et al. (2016). The algorithm 3.1 performs the model-checking and returns the triangle which results in the assignment of the empty set to some relation. From this point forward, if the checker returns the triangle (i,j,k) and we consequently add the transitivity constraint $\texttt{transitivity}(i,j,k)$ in the propositional formula, then it is impossible for the checker to return once again the same triangle. As discussed earlier, the maximum set of transitivity constraints that we need to add is of finite size, at which point we will have an equisatisfiable formula.

We now have all the pieces to create two different ways to solve the satisfiability and at the same time the realization problem in $\mathcal{RCC}8$. The first one is by using a direct encoding (with the function $\texttt{toSAT}(\mathcal{N})$). The second one is by using a CEGAR approach for it, like the one presented in Algorithm 3.2, which in the worst-case (the case where all the transitivity rules must be considered) will end-up being just a slightly slower version of the direct encoding; however, this has been experimentally found to never occur in practice as explained later. Moreover, every time we have that the instance is satisfiable, we also obtain an interpretation of the model returned by the SAT solver. In other words, we solve the satisfiability and realization problems together.

Algorithm 3.1: check(λ, N)

Data: N=(V,C) with n variables, λ a partial assignment of N
Result: An inconsistent triangle if false, otherwise null (the model is a realization of N)

```
1  N ← assign(N, λ);
2  G ← (V, E ← E(G_N));
3  for v_k from v_n to v_1 do
4      F_k ← {v_s : {v_s, v_k} ∈ E ∧ (s < k)} ;
5      foreach {v_i, v_j} ∈ F_k with (i < j) do
6          if ({v_i, v_j} ∉ E) then E ← E ∪ {{v_i, v_j}} ;
7          Temp ← R_{i,j} ∩ (R_{i,k} ⋄ R_{k,j});
8          if Temp ⊂ R_{i,j} then
9              R_{i,j} ← Temp;
10             R_{j,i} ← Temp^{-1};
11         else if R_{i,j} = ∅ then return (i,j,k) ;
12 return null;
```

Algorithm 3.2: CEGAR-$\mathcal{RCC}8(\mathcal{N})$

Data: $\mathcal{N}$=(V,C) with n variables
Result: A realization of $\mathcal{N}$ if it is possible to obtain one, UNSAT otherwise

```
1  G ← (V, E ← E(G_N));
2  setOfTriangles ← Chordal(G);
3  transitivity ← ⊤;
4  ψ ← under(N) ;                                   // under-abstraction step
5  while (setOfTriangle ≠ ∅) do
6      λ ← SAT-Solver(ψ ∧ transitivity) ;                          // solve step
7      if (λ = ⊥) then return UNSAT ;
8      res ← check(λ, N) ;                                        // check step
9      if (res = null) then return interpret(λ) ;
10     else
11         setOfTriangle.remove(res);
12         transitivity ← transitivity ∧ transitivity(res) ;   // refinement step
13 λ ← SAT-Solver(ψ ∧ transitivity) ;          // worst case: equisatisfiability
14 if (λ = ⊥) then return UNSAT ;
15 else return interpret(λ) ;
```

Experimental Results

Now that we have a new SAT encoding and a CEGAR approach for solving the satisfiability and realization problems in $\mathcal{RCC}8$, we want to compare against the state-of-the-art. For this purpose, we implemented the approach within the solver Churchill[10] and we used Glucose Audemard et al. (2013), Eén and Sörensson (2003) as an internal SAT solver. We will compare Churchill in direct-encoding and CEGAR mode against the state-of-the-art qualitative spatial reasoners for $\mathcal{RCC}8$, which are GQR Westphal et al. (2009), Renz-Nebel01 Renz and Nebel (2001), RCC8SAT Huang et al. (2013), PPyRCC8 Sioutis and Koubarakis (2012), and Chordal-Phalanx Sioutis and Condotta (2014). Each solver is using default settings, except GQR, which is using the flag "-c horn". By using the flag "-c horn", GQR decomposes an $\mathcal{RCC}8$ relation into horn sub-relations (which is standard behavior for the rest of the solvers), instead of basic relations; this changes the branching factor from 4 to ~ 1.4. Moreover, PPyRCC8 and Chordal-Phalanx are run using PyPy as recommended by their authors to improve the overall performance. We compare these solvers on four categories of benchmarks.

1. The first set consists of random hard instances that have been generated with:

   "`gencsp -i 100 -n 100 -d 10 15 0.5 -r nprels`".[11]

   In particular, it consists of 100 instances of 100 nodes for every average degree from 10.0 to 15.0 with a 0.5 step and using only relations that result to NP-completeness (*nprels*); thus, a total of 1 100 QCNs were generated.

2. The second set is generated exactly like the first one but with 500 nodes instead of 100 and for a range of d between 10.0 and 20.0, consisting of a total of 2100 QCNs.

[10] The name comes from the historical figure who used to also do a lot of CEGAR
[11] The generator comes with the Renz-Nebel01 solver

Figure 3.8: Runtime distribution on the 1st set

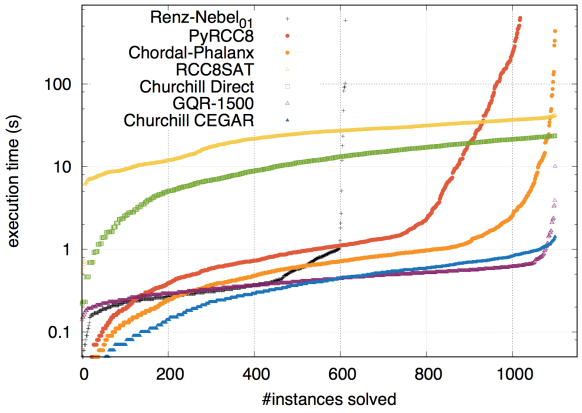
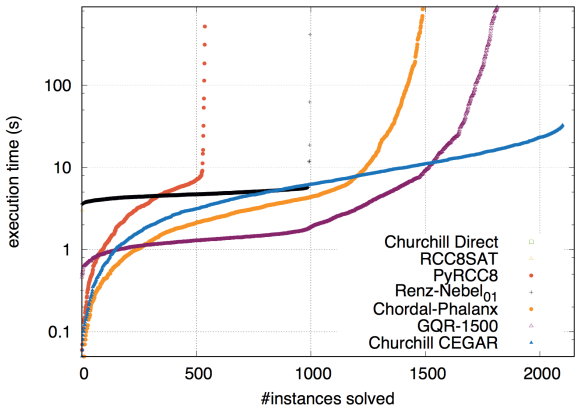


Figure 3.9: Runtime distribution on the 2nd set



3. The third set is the *random-scale-free-like-instances* Sioutis et al. (2016), which consists of 300 instances, 30 instances for every size from 1 000 to 10 000 nodes with a 1 000 step. These instances are normal to hard.

4. The fourth set is the *random-scale-free-like-np8-instances* Sioutis et al. (2016), which consists of 70 instances, 10 for every size from 500 to 3 500 nodes with a 500 step. These instances are hard to very hard as they are defined solely by *nprels*.

Regarding sets 3 and 4, scale-free networks are networks whose degree distribution follows a power law Barabási and Albert (1999); this kind of structured networks have been used extensively in the recent literature Huang et al. (2013), Sioutis and Condotta (2014). The experiments were ran on a cluster of Xeon, 4 cores, 3.3 GHz with CentOS 7.0 with a memory limit of 32GB and a runtime limit of 900 seconds per solver per benchmark. All solvers' answers were checked by verifying if all the solvers gave the same output for each benchmark. No discrepancy was found.

Regarding Fig. 3.8 and Fig. 3.9, which show the runtime distributions of the different solvers for the 1st and 2nd set of instances, we can see that Churchill and GQR are extremely fast to solve the respecting sets. Indeed, it took at most 1.42 seconds for Churchill to solve the hardest instance of the 1st set and 10.10 seconds for GQR, and 32.70 seconds on the 2nd set for Churchill. For Churchill it is mainly due to the fact that the networks are small (100 and 500 nodes), thus the triangulation of the graph and the SAT translation are extremely efficient. Moreover, the speed-up is also because we perform in average a small number of CEGAR loops (avg: 8.90 for 1st set and 11.87 for 2nd set). However, when we take a look at the direct translations (RCC8SAT and Churchill Direct), 500 nodes is already too much and this blows up the allowed memory.

Table 3.2 shows the number of benchmarks solved for sets 3 and 4. The best results of a given row are presented in bold colour and the number of benchmarks which cannot be solved because of lack of memory is provided between parenthesis (if such benchmarks do not exist a dash is displayed). The line VBS represents the Virtual Best Solver (a practical upper-bound on the performance achievable by picking the best solver for each benchmark). On the 3rd set, we can see the scalability of a CEGAR approach against a direct encoding (Churchill Direct) or via a CP representation. The results are clear, when the number of nodes is too big, the SAT approaches require too much memory or too much time for the translation of the problem and,

Table 3.2: Results on sets 3 (left) and 4 (right)

	random-scale-free-like-instances						*random-scale-free-like-np8-instances*						
#Nodes (x1000)	< 6	6	7	8	9	10	0.5	1	1.5	2	2.5	3	3.5
#Instances	150	30	30	30	30	30	10	10	10	10	10	10	10
Renz-Nebel01	0	0	0	0	0	0	0	0	0	0	0	0	0
	-	-	-	-	-	-	-	-	-	-	-	-	-
RCC8SAT	0	0	0	0	0	0	0	0	0	0	0	0	0
	(150)	(30)	(30)	(30)	(30)	(30)	(10)	(10)	(10)	(10)	(10)	(10)	(10)
PPyRCC8	134	19	20	21	23	23	**10**	9	**10**	8	7	3	3
	(14)	(8)	(7)	(7)	(7)	(4)	-	-	-	-	(1)	(5)	(7)
Chordal-Phalanx	**150**	**30**	**30**	**30**	**30**	**30**	**10**	**10**	**10**	**10**	**10**	**10**	**10**
	-	-	-	-	-	-	-	-	-	-	-	-	-
GQR-1500	**150**	**30**	**30**	**30**	**30**	**30**	**10**	**10**	9	6	**10**	8	9
	-	-	-	-	-	-	-	-	-	-	-	-	-
Churchill Direct	0	0	0	0	0	0	0	0	0	0	0	0	0
	(150)	(30)	(30)	(30)	(30)	(30)	(10)	(10)	(10)	(10)	(10)	(10)	(10)
Churchill CEGAR	**150**	**30**	**30**	18	8	6	**10**	**10**	**10**	**10**	**10**	**10**	**10**
	-	-	-	(10)	(20)	(24)	-	-	-	-	-	-	-
VBS	150	30	30	30	30	30	10	10	10	10	10	10	10

hence, become inefficient. The bigger the network, the more time Churchill CEGAR spends model-checking the output of the SAT solver and the more space is required to add transitivity constraints. In some cases, we just reach the space limit and are unable to solve instances.

On the 4th set, we study the scalability on very hard instances that are of reasonable size. We can see here that SAT solvers still have a hard time with the size of the input, and that using a CEGAR approach instead of a direct encoding leads to a huge improvement. Indeed, Churchill CEGAR managed to solve all the instances, but, unfortunately, it took more time than Chordal-Phalanx to do so in most cases (median: 37.97s for Churchill vs 16.78 for Chordal-Phalanx); however, it was faster in the worst-case (max: 163.10s for Churchill vs 714.12s for Chordal-Phalanx). This is mainly due to the fact that model-checking many times, which is typically the case when the network has a size between 2 000 and 3 500 nodes, is time-consuming. In fact, a sum-up of how the runtime of Churchill is distinguished by Triangulation time, Checking time, and Solving time is given in Table 3.3.

When we analyse the results given in Table 3.3, the analysis is clear: when the network is small (1st and 2nd sets) the main percentage of the time is spent in the triangulation of the graph. When the network is big (3rd and 4th sets) the main percentage of the time is spent in the Checking time. But in any case, the SAT solver is not the bottleneck here.

For all the results, it is worth remembering that even if we are a little bit slower on sets 3 and 4, we are solving in the same time the realization problem, *i.e.*, we output a realization for the input problem, not only a decision about the satisfiability of that problem.

Now that we saw what are the CEGAR over and under and an instantiation of the CEGAR-under framework for the $\mathcal{RCC}8$ satisfiability problem, let us make a small sum-up of the situation

Table 3.3: Sum-up of times for the three steps in Churchill

Time (s)	Triangulation			Checking			Solving		
	min	med	max	min	med	max	min	med	max
1st set	0.220	0.390	0.630	0.015	0.030	0.151	0.002	0.003	0.010
2nd set	3.910	12.910	20.153	0.430	1.170	2.057	0.015	0.030	0.370
3rd set	8.708	55.96	128.96	7.900	222.3	668.57	0.015	0.860	16.38
4th set	3.765	21.530	68.230	0.560	24.410	58.950	0.012	0.250	15.73

before presenting RECAR. We can easily sum-up the framework as follows: a classic CEGAR approach with over-abstraction and a SAT short-cut performs well when the input is satisfiable. But generally, it does not perform well in problems which are unsatisfiable. Respectively, a classic CEGAR approach with under-abstraction and an UNSAT short-cut performs well when the input is unsatisfiable. But generally, it does not perform well in problems which are satisfiable. The reason for both is that it may have to keep refining until it reaches equi-satisfiability with the original problem.

One way to address this issue is to mix SAT and UNSAT short-cuts, as in Brummayer and Biere (2009) and Wang et al. (2007). In these approaches, the methods alternate between over and under abstractions. Another way is to theorize a new framework able to use both kind of abstractions. This is what we did and what we will present in the next section.

3.3 Recursive Explore and Check Abstraction Refinement

In this section, we present a new framework that we call RECAR which stands for *Recursive Explore and Check Abstraction Refinement* which is a sound, complete and terminating framework to mix both kinds of abstractions and to be able to interleave them during the search. As in the CEGAR framework, it can be used in two modes, that we will present now.

3.3.1 RECAR-over

The RECAR-over approach, depicted in Fig. 3.10 and Algorithm. 3.3, interleaves both kinds of abstractions: each abstraction is performed with the information retrieved from solving the previous one. The UNSAT short cut is implemented using a recursive call to the main procedure when a strict under-abstraction $\check{\phi}$ can be built. One should also note that the proposed approach permits abstractions on two different levels: one is used to simplify the problem at the domain level (recursive call), while the other one is used to approximate the problem at the oracle level. In order to apply RECAR, the under-abstraction $\check{\phi}$ and the over-abstraction $\hat{\phi}$ must satisfy some properties. In the following, isSAT(ϕ) means that ϕ is satisfiable ($\not\models_1 \neg\phi$) and isUNSAT(ϕ) means ($\models_2 \neg\phi$), but on possibly different consequence relations (therefore the $_1$ and $_2$). $RC(\phi, \check{\phi})$ denotes a Boolean function deciding if a Recursive Call should occur.

Definition 59 (RECAR-over Assumptions)

To be able to perform a RECAR-over framework to decide a problem, one must verify a list of assumptions. We assume that isSAT is the satisfiability in the corresponding logic of $\hat{\phi}$ and that $\equiv^{?}_{\text{sat}}$ is sound.

1. Function 'check' is a sound and complete implementation of 'isSAT' which terminates.
2. $\text{isSAT}(\hat{\phi})$ implies $\text{isSAT}(\text{refine}(\hat{\phi}))$.
3. There exists $n \in \mathbb{N}$ such that $\text{refine}^n(\hat{\phi}) \equiv^{?}_{\text{sat}} \phi$.
4. $\text{isUNSAT}(\check{\phi})$ implies $\text{isUNSAT}(\phi)$.
5. Let $\text{under}(\phi) = \check{\phi}$. $\exists n \in \mathbb{N}$ s.t. $RC(\text{under}^n(\phi), \text{under}^{n+1}(\phi))$ evaluates to false.

Note that we have $\hat{\phi}$ is satisfiable implies ϕ is satisfiable by Assumptions 2 and 3 together. In the following, we show that, under these assumptions, RECAR is sound, complete and terminates. To do so, we present the algorithm recar-over(ϕ) in Algorithm. 3.3.

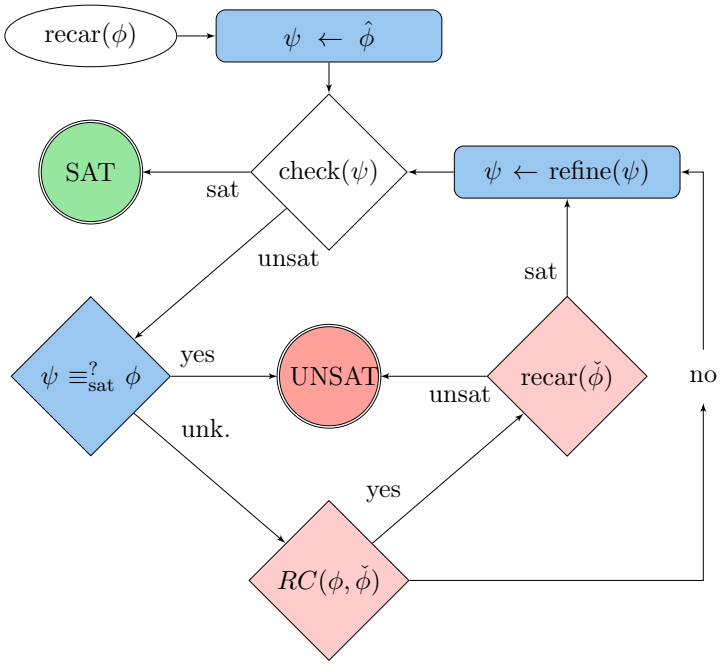


Figure 3.10: The RECAR framework starting with over-abstraction

Algorithm 3.3: recar - over(ϕ)

```
1 ψ ← φ̂ ;
2 while (ψ ≡?sat φ returns "unknown") do
3     if (check(ψ) = SAT) then return SAT ;
4     if (RC(φ, φ̌) returns "yes") then
5         if (recar - over(φ̌) = UNSAT) then return UNSAT ;
6     ψ ← refine(ψ) ;
7 return check(ψ) ;
```

Theorem 21 (Soundness)

If recar - over(ϕ) returns SAT then ϕ is satisfiable.

Proof. Assume that recar - over(ϕ) returns SAT. This happens only if check(ψ) returns SAT, either from line 3 or from line 7. Thus, we know that isSAT(ψ) holds (by Assump. 1). But ψ equals to $\hat{\phi}$ or equals to refine$^n(\phi)$ for some $n \in \mathbb{N}$. Then ϕ is satisfiable (by Assump. 2 and 3).

q.e.d □

The intuition behind the proof of Th. 22 is that there are two ways to conclude that ϕ is UNSAT. In the first case, $\hat{\phi}$ is refined a finite number of times until it is detected equi-satisfiable to ϕ and check returns UNSAT. Then ϕ is unsatisfiable. In the second case, one of the under-abstractions is shown UNSAT, then ϕ is UNSAT (by Assump. 4).

Theorem 22 (Completeness)

If recar - over(ϕ) returns UNSAT then isUNSAT(ϕ).

Proof. By induction on the number k of recursive calls to recar - over (Line 5). Assume recar - over(ϕ) returns UNSAT after k recursive calls.

In the induction base $k = 0$ (no recursive call). Then we must have exited the loop ($\psi \equiv^?_{\text{sat}} \phi$) (by Assump. 3) and check(ψ) returns UNSAT. This means that ψ is unsatisfiable (by Assump. 1) and therefore isUNSAT(ϕ) holds (because of equi-satisfiability).

The induction hypothesis is: for all $k \leq n$, if recar - over(ϕ) returns UNSAT after k recursive calls then isUNSAT(ϕ).

In the induction step $k = n + 1$. Then the conditions of lines 4 and 5 of the algorithm are true. This means that recar - over($\check{\phi}$) returns UNSAT after k recursive calls to recar - over. Then isUNSAT($\check{\phi}$) (by I.H.). Then isUNSAT(ϕ) (by Assump. 4).

q.e.d □

The intuition behind the proof of Th. 23 is that the function performs a finite number of recursive calls (Assump. 5). Moreover, each of these calls will have a finite number of refinements before terminating (Assump. 3).

Theorem 23 (Termination)

recar - over terminates for any input ϕ.

Proof. We have that (1) For all ϕ, there exists $n \in \mathbb{N}$ such that $RC(\text{under}^n(\phi), \text{under}^{n+1}(\phi))$ evaluates to false (by Assump. 5) and (2) For each $i \leq n$ there is $m_i \in \mathbb{N}$ such that $\text{refine}^{m_i}(\hat{\phi}) \equiv^{?}_{\text{sat}} \phi$ (by Assump. 3).

Then, for any input ϕ, the recursive call of line 6 of the algorithm will be executed at most n times before the condition of line 5 becomes false. For each one of these recursive calls, the while-loop of the algorithm will be executed at most m_i times before the condition of line 2 becomes false. Therefore, for any input, recar - over halts after a finite number of recursive calls.

q.e.d □

An instantiation of this framework is given in the next chapter to decide the satisfiability of PSPACE modal logics Lagniez et al. (2017a). The over-abstraction function is a bounded-reduction into propositional logic, the under-abstraction function is a function able to smartly remove conjunction in the formula that are "supposedly" not responsible for the unsatisfiability of the formula. Obviously, as in the CEGAR framework, there exists an under version of RECAR. But to the best of our knowledge, in 2018, this RECAR-under framework has never been instantiated.

3.3.2 RECAR-under

The principle is the same as for RECAR-over, we have a list of assumptions that must be respected, we have a general schema and a proof for the correctness, completeness and termination of the algorithm.

Definition 60 (RECAR-under Assumptions)

We assume that isUNSAT is the unsatisfiability in the corresponding logic of $\check{\phi}$, that isSAT is the satisfiability of the corresponding logic of $\hat{\phi}$.

1. Function 'check' is a sound and complete implementation of 'isUNSAT' which terminates.
2. isUNSAT($\check{\phi}$) implies isUNSAT(refine($\check{\phi}$)).
3. There exists $n \in \mathbb{N}$ and λ such that $\lambda \models^? \text{refine}^n(\check{\phi})$ and $\lambda \models^? \phi$.
4. isSAT($\hat{\phi}$) implies isSAT(ϕ).
5. Let over(ϕ) $= \hat{\phi}$. $\exists n \in \mathbb{N}$ s.t. $RC(\text{over}^n(\phi), \text{over}^{n+1}(\phi))$ evaluates to false.

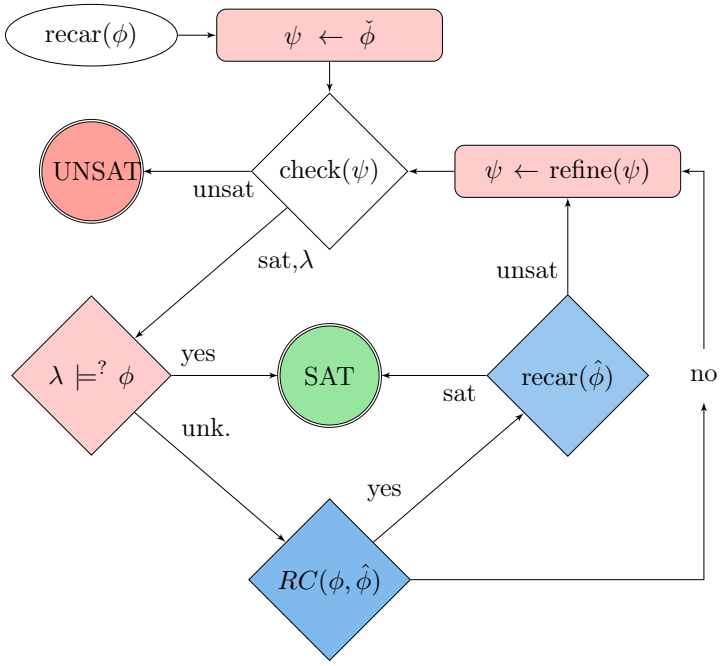


Figure 3.11: The RECAR framework starting with under-abstraction

Theorem 24 (Soundness)

If recar - under(ϕ) returns SAT then ϕ is satisfiable.

Proof. By induction on the number k of recursive calls to recar - under (Line 5). Assume recar - under(ϕ) returns SAT after k recursive calls. In the induction base $k = 0$ (no recursive

Algorithm 3.4: recar - under(ϕ)

```
1  ψ ← φ̌ ;
2  λ ← ⊤;
3  while (λ ⊨? φ returns "unknown") do
4      if (λ ← check(ψ) = UNSAT) then
5          return UNSAT
6      if (RC(φ, φ̂) returns "yes") then
7          if (recar - under(φ̂) = SAT) then
8              return SAT
9      ψ ← refine(ψ) ;
10 return check(ψ) ;
```

call). Then we must have exited the loop ($\lambda \models^? \phi$) and check(ψ) returns SAT. This means that ψ is satisfiable (by Assump. 1) and therefore isSAT(ϕ) holds (becaus $\lambda \models^? \phi$ by Assump. 3). The induction hypothesis is: for all $k \leq n$, if recar - under(ϕ) returns SAT after k recursive calls then isSAT(ϕ). In the induction step $k = n+1$. Then the conditions of lines 4 and 5 of the algorithm are true. This means that recar - under($\hat{\phi}$) returns SAT after k recursive calls to recar - under. Then isSAT($\hat{\phi}$) (by I.H.). Then isSAT(ϕ) (by Assump. 4).

q.e.d □

Theorem 25 (Completeness)

If recar - under(ϕ) returns UNSAT then isUNSAT(ϕ).

Proof. Assume that recar - over(ϕ) returns UNSAT. This happens only if check(ψ) returns UNSAT, either from line 3 or from line 7. Thus, we know that isUNSAT(ψ) holds (by Assump. 1). But ψ equals to $\check{\phi}$ or equals to refine$^n(\phi)$ for some $n \in \mathbb{N}$. Then ϕ is unsatisfiable (by Assump. 2 and 3). q.e.d □

Theorem 26 (Termination)

recar - under(ϕ) terminates for any input ϕ.

Proof. We have that (1) For all ϕ, there exists $n \in \mathbb{N}$ such that $RC(\text{over}^n(\phi), \text{over}^{n+1}(\phi))$ evaluates to false (by Assump. 5) and (2) For each $i \leq n$ there is $m_i \in \mathbb{N}$ such that it exists $\lambda \models^? \text{refine}^{m_i}(\check{\phi})$ and $\lambda \models^? \phi$ (by Assump. 3). Then, for any input ϕ, the recursive call of line 6 of the algorithm will be executed at most n times before the condition of line 5 becomes false.

For each one of these recursive calls, the while-loop of the algorithm will be executed at most m_i times before the condition of line 2 becomes false. Therefore, for any input, recar - over halts after a finite number of recursive calls. q.e.d □

Now that we showed our theoretical contribution which is the RECAR framework and that we proved that it is sound, complete and that it terminates whether it is used with first an over-abstraction (RECAR-over) or with an under-abstraction (RECAR-under). Let us now go through the next chapter where we present a RECAR-over instantiation for the modal logic K Satisfiability Problem first, and then we show how the solver can be extended for all the other modal logics and what are the necessary optimization to make the solver competitive with the state-of-the-art approaches.

3.4 Explanation of How The Abstractions Are Called

We just presented different framework that are able to use abstractions to decide a problem. But they all work differently. In fact, one can represent how the abstractions are refined with the Figure 3.12.

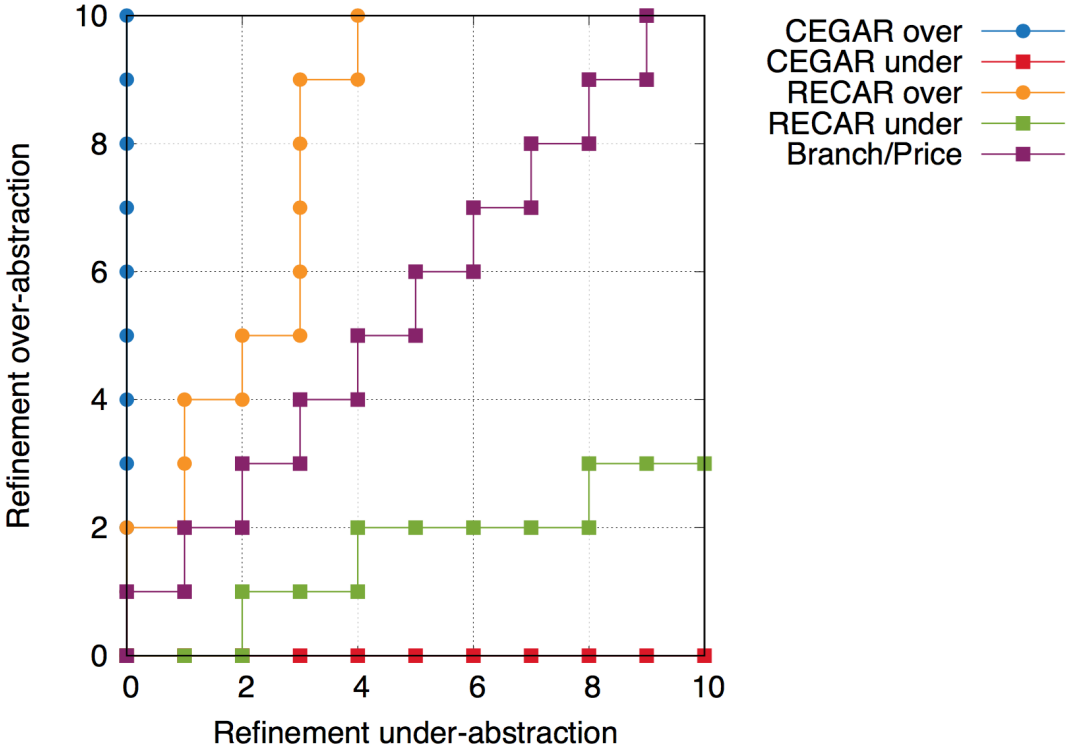


Figure 3.12: Illustration of the different kind of abstractions framework

Indeed, what we can see is that first, the CEGAR-over and CEGAR-under are not moving in two dimensions, they are set with one kind of abstractions, and they just refined it until they reach a fix-point to decide the problem. Indeed, a CEGAR-over cannot perform an under-abstraction and a CEGAR-under cannot perform an over-abstraction. There exists also the Branch and Price algorithm Savelsbergh (1997) (one famous method to solve combinatorial optimisation

problems). It is basically doing an alternation between an over-abstraction and under-abstraction until it is able to decide the satisfiability of the problem. This approach is doing better than the CEGAR framework in the way that, this time, it is able to move in two dimensions, but unfortunately, the *trajectory* is known in advance, it will first do an over-abstraction, then an under-abstraction, then an over-abstraction *etc.* until deciding the problem.

The RECAR framework that we proposed is freer in the way that the trajectory is not known in advance. It could be the case that the function RC, which decide if we should perform a recursive call, is never true. In that case the RECAR-over is equals to the CEGAR-over and it will never perform an under-abstraction. It could also be the case that the function RC is always true, then the RECAR-over and RECAR-under will behave in the same way as a Branch and Price algorithm and it will just alternate the abstractions.

The goal of the RECAR framework is to give the freedom, thanks to the RC function, to adapt the direction to the problem that we want to solve. It could be the case that for some instances of a problem, behaving as a CEGAR-over or as a CEGAR-under framework is the best behaviour. It could also be the case that for some other instances of a problem, behaving as Branch and Price is the best behaviour. The RECAR framework, when it is well instantiated, will be able to behave as both and to adapt its behaviour to the instances.

Chapter 4

The Modal Logic K⋆ Satisfiability Problem

Contents

> À ce propos, rien n'est plus arbitraire qu'une logique modale : "J'ai fini cette logique, je peux en avoir une autre ?" semble-t-on dire de ce côté-là.
>
> *J-Y Girard, Le Point aveugle*

Now that we just proposed a theoretical framework that we showed sound, complete and terminating, it is time to see how such a framework can be instantiated to solve problems. The first instantiation that we proposed is an instantiation using a SAT solver to decide the Modal Logic K Satisfiability Problem within the solver MoSaiC.

4.1 RECAR Approach For The Modal Logic K

As our over-abstraction function, we define a translation from modal logic K to classical propositional logic

4.1.1 Over-Abstraction

Definition 61 (Translation)

$$
\begin{aligned}
\mathrm{tr}(\phi, n) &= \mathrm{tr}'(\mathrm{nnf}(\phi), 0, n) \\
\mathrm{tr}'(\top, i, n) &= \top \\
\mathrm{tr}'(\bot, i, n) &= \bot \\
\mathrm{tr}'(p, i, n) &= p_i \\
\mathrm{tr}'(\neg p, i, n) &= \neg p_i \\
\mathrm{tr}'(\phi \wedge \psi, i, n) &= \mathrm{tr}'(\phi, i, n) \wedge \mathrm{tr}'(\psi, i, n) \\
\mathrm{tr}'(\phi \vee \psi), i, n) &= \mathrm{tr}'(\phi, i, n) \vee \mathrm{tr}'(\psi, i, n) \\
\mathrm{tr}'(\Box\phi, i, n) &= \bigwedge_{j=0}^{n} (r_{i,j} \rightarrow \mathrm{tr}'(\phi, j, n)) \\
\mathrm{tr}'(\Diamond\phi, i, n) &= \bigvee_{j=0}^{n} (r_{i,j} \wedge \mathrm{tr}'(\phi, j, n))
\end{aligned}
$$

The translation adds fresh variables p_i and $r^a_{i,j}$ to the formula: p_i denotes that variable p is true in the world w_i whereas $r^a_{i,j}$ corresponds to w_j being accessible from w_i by the relation a. We have the following as an immediate result (where $UB(\phi)$ is the theoretical upper-bound presented in Theorem 5).

Theorem 27

ϕ is K-satisfiable if and only if $\mathrm{tr}(\phi, Atom(\phi)^{\mathrm{depth}(\phi)})$ is satisfiable in propositional logic.

Therefore, in order to decide the satisfiability of a formula $\phi \in \mathcal{L}$, one can simply feed a SAT solver with $\mathrm{tr}(\phi, Atom(\phi)^{\mathrm{depth}(\phi)})$. The main issue is that the translation may generate an exponentially larger formula that the RECAR framework wants to avoid. Now, in order to apply the RECAR approach, we first need to find an over-abstraction which respects the assumptions presented in Definition 59.

Definition 62 (Over-abstraction)

Let $\phi \in \mathcal{L}$. The over-abstraction of ϕ, denoted $\hat{\phi}$, is the formula $\mathrm{tr}(\phi, 1)$.

Definition 63 (Refinement)

Let $1 \leq n \leq Atom(\phi)^{\text{depth}(\phi)}$. The refinement of $\text{tr}(\phi, n)$, noted $\text{refine}(\phi, n)$ is the formula $\text{tr}(\phi, n+1)$.

It is important to notice that $Atom(\phi)$ counts the number of different atoms in the formula ϕ. If we consider $\phi = p \wedge \neg p$, there are two different atoms in this formula, thus $Atom(\phi) = 2$. The following theorem will demonstrate that the previous over-abstraction can satisfied the RECAR-over Assump. 2 and 3.

Theorem 28

If $\text{isSAT}(\text{tr}(\phi, n))$ then $\text{isSAT}(\text{tr}(\phi, n+1))$, for all $1 < n \leq Atom(\phi)^{\text{depth}(\phi)}$. (RECAR-over Assump. 2)

Proof. [Proof Sketch] The idea is that if ϕ is satisfied by a model $\mathcal{M}$ with n worlds, then we can find a model $\mathcal{M}'$ with $n+1$ worlds satisfying ϕ. The additional world is just not accessible from the ones already in $\mathcal{M}$. q.e.d □

The latter result allows us to use this over-abstraction and refinement in the RECAR-over approach. It is easy to see that RECAR-over Assumptions 2 and 3 are satisfied. The RECAR-over Assumption 1 is satisfied by using a SAT solver which is sound, complete and terminates. The following section will demonstrate how one can satisfy the RECAR-over Assumption 4 and 5.

4.1.2 Under-Abstraction

To understand the intuition behind the under-abstraction we use an example. Let $\phi = (\Diamond p \wedge \Box\neg p \wedge \chi)$ for some $\chi \in \mathcal{L}$, where $\text{depth}(\chi)$ is huge. This is clearly unsatisfiable because $(\Diamond p \wedge \Box\neg p)$ is unsatisfiable. One can see that right away without even knowing what χ looks like. However, a CEGAR approach using the over-abstraction and refinement defined earlier will take a long time before finally conclude it. The reason is that each refinement $\text{tr}(\phi, n+1)$ of the original formula will be shown unsatisfiable and it will not stop until the theoretical upper-bound is reached.

To avoid these pathological cases, the RECAR approach also performs under-abstractions. To see how it works, let us take that formula ϕ again. First, we add to each conjunct in ϕ a fresh variable s_i (a selector) that will be assumed to be true by the SAT solver, as done in Figure 4.1. Then, we make the first over-abstraction $\text{tr}(\phi, 1)$ and give it to a modern SAT solver. The solver will return UNSAT with an unsatisfiable core. From this core, we extract a set of selectors *core*. Let us assume, in our example, that $core = \{s_1, s_2\}$. This means that the formula $\check{\phi} = (\Diamond p \wedge \Box\neg p)$, which is the one labelled by the selectors, is enough to prove the unsatisfiability of ϕ with only 1 possible world. Proving the unsatisfiability of $\check{\phi}$ will imply that

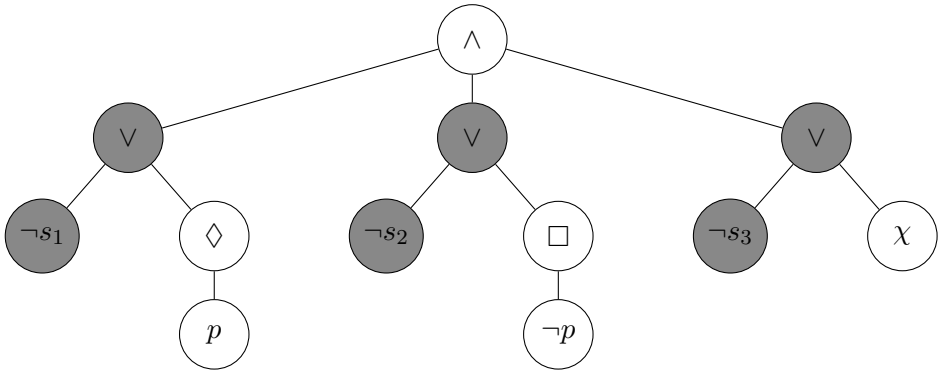


Figure 4.1: How selectors are applied to $\phi = (\Diamond p \wedge \Box \neg p \wedge \chi)$

ϕ is unsatisfiable. Note that, in this specific case, $UB(\check{\phi})$ is much smaller than $UB(\phi)$. Thus the CEGAR approach applied to $\check{\phi}$ will succeed much earlier, while it may have failed for the entire formula ϕ. Formally, we have the following.

Definition 64 (Under-Approximation)

$$
\begin{aligned}
&\text{under}(p, core) = p \\
&\text{under}(\neg p, core) = \neg p \\
&\text{under}(\Box\phi, core) = \Box(\text{under}(\phi, core)) \\
&\text{under}(\Diamond\phi, core) = \Diamond(\text{under}(\phi, core)) \\
&\text{under}((\phi \wedge \psi), core) = \text{under}(\phi, core) \wedge \text{under}(\psi, core) \\
&\text{under}((\psi \vee \chi), core) = \begin{cases} \text{under}(\chi, core) & \text{if } \psi = \neg s_i, s_i \in core \\ \top & \text{if } \psi = \neg s_i, s_i \notin core \\ (\text{under}(\psi, core) & \\ \vee \text{under}(\chi, core)) & \text{otherwise} \end{cases}
\end{aligned}
$$

Theorem 29

under$(\phi, core)$ is unsatisfiable in K and $s_i \in core$ means that $\forall w$ in all Kripke models, $s_i \in V(w)$, implies ϕ is unsatisfiable in K.

The intuition of the proof is that each selector s_i enables an operand in a conjunction of the formula. Each time function 'under' is called with a non-empty *core*, operands not enabled with a selector from the *core* will be removed from the formula.

Proof. Let ϕ be in NNF. We show that under$(\phi, core)$ is unsatisfiable in K implies ϕ is unsatisfiable in K by induction on the structure of ϕ. The induction hypothesis being that for every formula ϕ smaller than n, if $\langle M, w\rangle \models \phi$ then $\langle M, w\rangle \models$ under$(\phi, core)$. We assume because it is necessarily the case, that if $s_i \in core$ then $\forall w$ in all Kripke models, $s_i \in V(w)$ and if $s_i\neg \in core$ then $\forall w$ in all Kripke models, $\neg s_i \in V(w)$. Assume ϕ satisfiable in K. Then $\exists\ M, w$ s.t. $\langle M, w\rangle \models \phi$. There are two cases in the induction base: (1) $\phi = p$ and (2) $\phi = \neg p$. In both of them under$(\phi, core) = \phi$. There are four cases in the induction step:
(1) $\phi = \Diamond(\psi)$. $\exists\ \langle M, w\rangle$ s.t. $\langle M, w\rangle \models \Diamond(\psi)$. Then $\exists\ \langle M, w'\rangle$ s.t. $(w, w') \in R$, $\langle M, w'\rangle \models \psi$. Then $\langle M, w'\rangle \models$ under$(\psi, core)$ by induction hypothesis. Thus $\langle M, w\rangle \models$ under$(\phi, core)$;
(2) $\phi = \Box(\psi)$. This case is analogous to (1).
(3) $\phi = (\psi \wedge \chi)$. $\exists\ \langle M, w\rangle \models (\psi \wedge \chi)$. Then $\langle M, w\rangle \models \psi$ and $\langle M, w\rangle \models \chi$. Then $\langle M, w\rangle \models$ under$(\psi, core)$ and $\langle M, w\rangle \models$ under$(\chi, core)$ by induction hypothesis. Thus $\langle M, w\rangle \models$ under$(\phi, core)$;
(4) $\phi = (\psi \vee \chi)$. We consider the three cases:
(4.a) $\psi = \neg s_i$ and $s_i \in core$. Then $\exists\ \langle M, w\rangle \models (\neg s_i \vee \chi)$ but $s_i \in V(w)$, then $\langle M, w\rangle \models \chi$. Then $\langle M, w\rangle \models$ under$(\chi, core)$ by induction hypothesis. Thus $\langle M, w\rangle \models$ under$(\phi, core)$;
(4.b) $\psi = \neg s_i$ and $s_i \notin core$. $\exists\ \langle M, w\rangle \models (\neg s_i \vee \chi)$. but we always have $\langle M, w\rangle \models \top$. Thus $\langle M, w\rangle \models$ under$(\phi, core)$.
(4.c) This case is analogous to (3). q.e.d $\Box$

Theorem 29 shows that function 'under' satisfies RECAR-over Assump. 4. To see that it also satisfies Assump. 5, note that the length of under$^{n+1}(\phi, core)$ is smaller or equal to that of under$^{n}(\phi, core')$ (even though the sets $core$ and $core'$ usually differ).

4.1.3 MoSaiC: A RECAR-over Approach

The function *addS* will add selectors on the different conjuncts in the formula. Here the function *RC* from the previous chapter is replaced by the simple equality test $\check{\phi} = \phi$. We implemented the RECAR-over approach for modal logic K satisfiability problem within the solver MoSaiC, using the over and under abstractions defined in the previous sections. MoSaiC combines several features found in state-of-the-art solvers. As in Km2SAT Sebastiani and Vescovi (2009), it optimises the input by performing the rules: Box Lifting, Flattening, and Truth Propagation through modal and Boolean operators (see Sebastiani and Vescovi (2009) for more details). MoSaiC also uses the SAT solver GLUCOSE in incremental mode Eén and Sörensson (2003), Audemard et al. (2013) to decide the satisfiability of each ψ.

Note that some implementation details differ a bit from Figure 4.2. For instance, we do not call GLUCOSE on ψ but on an updated ψ' with selectors on conjuncts under the assumption that these selectors are satisfied; we do not need to generate the under abstraction $\check{\phi}$ to test the condition $\check{\phi} = \phi$: we just need to know the number of selectors involved in the unsatisfiability of the formula. We also return a Kripke model in the main procedure, not just SAT/UNSAT. We take advantage of such information to provide a new bound for l. And finally, note that in our case $\max(|M|, l + 1)$ always returns $|M|$ because it is not possible to find a model smaller than M by construction of $\check{\phi}$.

The condition *Jump() ?* visible in Figure 4.2 is an heuristic to improve in practice, the performance obtained by MoSaiC. More explanation will be given in the section talking about the experimental evaluation, but in a nutshell here is what we detect. At one point in the search, we reach a sub-formula after making an under-abstraction, which is so small that we would spend too many times making iteration. Thus, instead of making these iterations, we

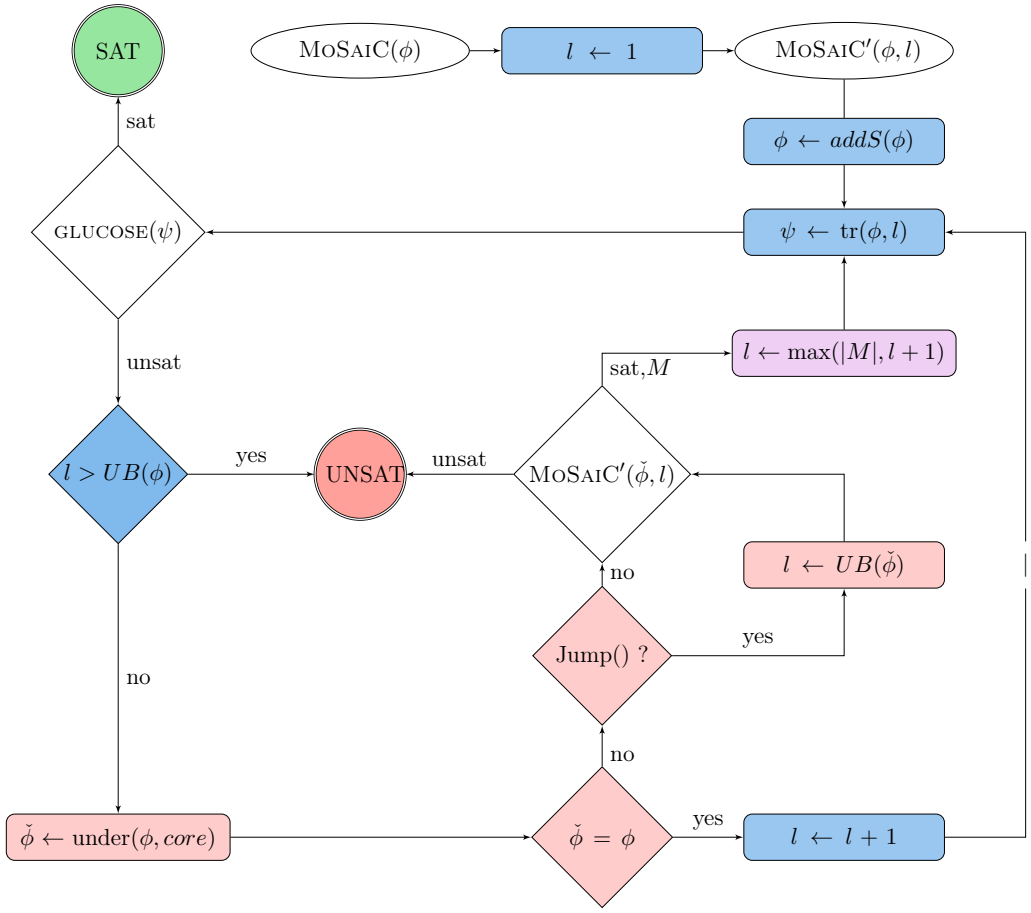


Figure 4.2: MoSaiC: RECAR-over for modal logic K

detect heuristically that if we put the bound l to the theoretical upper-bound of the sub-formula and that it does not generate a memory-out, then we should do it. In that way, we generate a propositional logic formula which is equisatisfiable with the sub-formula and we can decide in one shot thanks to GLUCOSE, the satisfiability of this under-abstraction. Heuristically, the function *Jump()* is as follows: $(|\check{\phi}| \times UB(\check{\phi}) < BIG)$ with BIG set according to the memory-limit that we allow for the solver.

From now on, in this thesis, we saw how to solve NP modal logics, how to solve modal logic K satisfiability problem thanks to a RECAR-over approach. The last part is now related to how to deal with all kind of modal logics. Said otherwise, how we can extend MoSaiC in such way that it can deal with all the logics based on K. This is what we will see in the next section.

4.2 Extensions Of MoSaiC For The Other Modal Logics

As we just said, we want to extend MoSaiC in order to make it able to deal with the different modal logics. As stated in the Preliminaries part of this thesis, we all know that each modal logic is just modal logic K with additional axioms. It seems natural to search how we can make MoSaiC encodes the different axioms of modal logic and this is what we will see know in this section.

4.2.1 How To Encode The Axioms

It is well known that some axioms correspond to constraints on Kripke structures Sahlqvist (1975). For instance, every reflexive K-structure satisfies T. Even if there are non-reflexive K-structures satisfying T as well, it is always possible to find an "equivalent" reflexive K-structure. Two K-structures are equivalent if and only if they are bi-similar. The reader may find the definition of bi-simulation, for instance, in Blackburn et al. (2006). Therefore, if one wants to find a finite KT-model, it is safe to search only among reflexive K-structures. Analogous reasoning may also be used for the other properties. Consequently, following Table 2.1, we call KT-structure a reflexive K-structure and we call S4-structure (or KT4-structure) a reflexive and transitive K-structure.

Therefore, to deal with different modal logics, we append to the translation into CNF, the following constraints corresponding to the different axioms (with m the number of modal operators and n the number of worlds):

Definition 65 (Translation of Axioms)

$$\mathrm{tr}((T), n) = \bigwedge_{i=0}^{n} (r_{i,i}) \quad \mathrm{tr}((D), n) = \bigwedge_{i=0}^{n} \bigvee_{j=0}^{n} (r_{i,j})$$

$$\mathrm{tr}((B), n) = \bigwedge_{i=0}^{n} \bigwedge_{j=0}^{n} (r_{i,j} \rightarrow r_{j,i})$$

$$\mathrm{tr}((4), n) = \bigwedge_{i=0}^{n} \bigwedge_{j=0}^{n} \bigwedge_{k=0}^{n} ((r_{i,j} \wedge r_{j,k}) \rightarrow r_{i,k})$$

$$\mathrm{tr}((5), n) = \bigwedge_{i=0}^{n} \bigwedge_{j=0}^{n} \bigwedge_{k=0}^{n} ((r_{i,j} \wedge r_{i,k}) \rightarrow r_{j,k})$$

The translation of each axiom came from the relations in First Order Logic, presented by Sahlqvist. Thus, when axiom (T) is considered (i.e. modal logic KT), the over-abstraction function is $(\mathrm{tr}(\phi, n) \wedge \mathrm{tr}((T), n))$. When both axioms (T) and (4) (i.e. modal logic S4) are considered, the over-abstraction function is $(\mathrm{tr}(\phi, n) \wedge \mathrm{tr}((T), n) \wedge \mathrm{tr}((4), n))$.

When considering modal logic axioms other than K, the formulas usually need more worlds to be satisfied. Consequently, the SAT translation becomes too big to be handled (some CNF have hundreds of million of clauses). Thus, we provide a new space-aware over-abstraction function and a new axiom-aware under-abstraction function, to be used in the framework.

4.2.2 Axiom-Aware Under-Abstraction

So far, when the formula was unsatisfiable, the only way to prove its unsatisfiability was to cut some branch rooted in AND nodes in order to produce an unsatisfiable sub-formula that can be translated into a CNF. When dealing with axiom (T), we have $\Box\phi \rightarrow \phi$ and then, as

demonstrated in the following property, it is possible to construct an under-abstraction of the formula that also removes some boxes.

Proposition 3

Let us consider an NNF formula $\phi \in \mathcal{L}$ in a modal logic satisfying (T). If we replace a sub-formula $\Box\psi$ by ψ, then the resulting formula ϕ' is an under-abstraction of ϕ.

Proof. To prove that this property holds, it is enough to check that if ϕ is satisfiable then so is ϕ'. W.l.o.g., in the following, we suppose that all the OR and AND nodes are binary and we remind that the Boolean operators are commutative. Let us consider an NNF modal logic formula ϕ, and ϕ' a copy of ϕ which differs on only one sub-formula rooted on a box node $\Box\psi \in \phi$ where $\Box\psi$ has been replaced by ψ in ϕ'. We show that, if there exists a Kripke model $\mathcal{K} = \langle W, R, V\rangle$ and a possible world $w \in W$, s.t. $\langle \mathcal{K}, w\rangle \models \phi$ then $\langle \mathcal{K}, w\rangle \models \phi'$ by induction on the structure of ϕ. Induction base: $\phi = \Box\psi$ and $\phi' = \psi$, where ψ contains no operator $\Box$. If there is $\langle \mathcal{K}, w\rangle \models \Box\psi$ then $\langle \mathcal{K}, w\rangle \models \psi$ due to axiom (T). Let us now prove the different cases on the induction step:
(1): $\phi = (\chi_1 \wedge \chi_2)$ and χ_1 contains $(\Box\psi)$ and $\phi' = (\chi_1' \wedge \chi_2)$ where χ_1' is χ_1 where $(\Box\psi)$ has been replaced by ψ. The case where χ_2 contains $(\Box\psi)$ is analogous due to the commutativity. We have $\langle \mathcal{K}, w\rangle \models (\chi_1 \wedge \chi_2)$ iff $\langle \mathcal{K}, w\rangle \models \chi_1$ and $\langle \mathcal{K}, w\rangle \models \chi_2$. By IH, $\langle \mathcal{K}, w\rangle \models \chi_1'$ and $\langle \mathcal{K}, w\rangle \models \chi_2$, iff $\langle \mathcal{K}, w\rangle \models (\chi_1' \wedge \chi_2)$. Thus $\langle \mathcal{K}, w\rangle \models \phi'$.
(2): $\phi = (\chi_1 \vee \chi_2)$ and χ_1 contains $(\Box\psi)$ and $\phi' = (\chi_1' \vee \chi_2)$ where χ_1' is χ_1 where $(\Box\psi)$ has been replaced by ψ. Anologous to (1).
(3): $\phi = \Box\chi$ and χ contains $(\Box\psi)$ and $\phi' = \Box\chi'$ where χ' is χ where $(\Box\psi)$ has been replaced by ψ. We have $\langle \mathcal{K}, w\rangle \models \Box\chi$ iff $\forall w'$ if $(w, w') \in R_a$ then $\langle \mathcal{K}, w'\rangle \models \chi$. By IH, $\langle \mathcal{K}, w'\rangle \models \chi'$, then $\forall w'$ if $(w, w') \in R_a$ then $\langle \mathcal{K}, w'\rangle \models \Box\chi'$. Thus $\langle \mathcal{K}, w\rangle \models \phi'$.
(4): $\phi = \Diamond\chi$ and χ contains $(\Box\psi)$ and $\phi' = \Diamond\chi'$ where χ' is χ where $(\Box\psi)$ has been replaced by ψ. Anologous to (3). q.e.d $\Box$

Note that such property does not hold for modal logic K: $\Box\bot$ is satisfiable in K but inconsistent in KT.

Definition 66 (Under-abstraction carve function)

$$\text{carve}(p, core) = p \quad \text{carve}(\neg p, core) = \neg p$$
$$\text{carve}(\Box\phi, c) = \Box(\text{carve}(\phi, core))$$
$$\text{carve}(\Diamond\phi, core) = \Diamond(\text{carve}(\phi, core))$$
$$\text{carve}((\neg s_i \vee \Box\chi) \wedge \chi, core) = \begin{cases} \text{carve}(\Box\chi, core) & \text{if } s_i \in core \\ \text{carve}(\chi, core) & \text{if } s_i \notin core \end{cases}$$
$$\text{carve}((\phi \wedge \psi), c) = \text{carve}(\phi, core) \wedge \text{carve}(\psi, core)$$
$$\text{carve}((\psi \vee \chi), core) = \begin{cases} \text{carve}(\chi, core) & \text{if } \psi = \neg s_i, s_i \in core \text{ and } \chi = (\chi_1 \wedge \chi_2) \\ \top & \text{if } \psi = \neg s_i, s_i \notin core \text{ and } \chi = (\chi_1 \wedge \chi_2) \\ \text{carve}(\psi, core) \vee \text{carve}(\chi, core) & \text{otherwise} \end{cases}$$

In order to select the boxes which will be replaced, we propose to improve the under-abstraction presented in the previous section, which combines selectors and unsatisfiable cores to search unsatisfiable sub-formulas. There we proposed to add a selector to each branch rooted in a AND node to be able to activate/deactivate some parts of the formula. When the solver is called to check the satisfiability of the formula, it is called with the set of selectors as assumptions. If the solver returns UNSAT, then the unsatisfiable core returned is used to remove parts of the formula that are not in the reason of its inconsistency. What we propose here consists in also replacing each $\Box\psi$ by $((\neg s_k \vee \Box\psi) \wedge \psi)$. This somehow says 'when we activate the selector s_k, then we translate the whole modality, if not, we just translate ψ in the current world'. We define formally such a function in Definition 66.

One question that arise when we see such a function, guided by the unsatisfiable core returned by the SAT solver is: is it really worth-it to trust the SAT solver? Is it really more efficient than just cutting boxes randomly in the formula. We will demonstrate experimentally the importance to trust the SAT solver in detecting the reason for why a formula is unsatisfiable.

Now, we can again use the core returned by the solver to extract a sub-formula that is unsatisfiable. That under-abstraction will remove some boxes from the formula which are not involved in its inconsistency. The difference that is when we translate a box into SAT it is not useful to translate twice the formula ψ in the current world.

4.2.3 Space-Aware Over-Abstraction

As already pointed out in the introduction, the bottleneck of CEGAR-based approaches using SAT oracle is the size of generated CNF formulas. For the case we are interested in, this bottleneck is reached when the over-abstraction function is called with a large number of worlds. However, it is possible to estimate the size of the CNF formula before computing it and then be aware that the translation will exhaust memory on targeted hardware.

In the following, we propose a space-aware over-abstraction function that is used instead of the original one when the space taken by the CNF reaches a given threshold. This new over-

abstraction is performed by disabling some disjuncts from the original formula. For this function to exist and to respect the RECAR assumptions, it needs to verify that cutting edges rooted in an OR node produces a weaker formula, i.e. every model of the resulting formula is also a model of the initial formula.

Property 1

Let us consider an NNF modal logic formula $\phi \in \mathcal{L}$. If we cut an edge rooted in an OR node, then the resulting formula ϕ' is an over-abstraction of ϕ.

Proof. To prove that this property holds, it is enough to check that every model of ϕ' is also a model of ϕ. W.l.o.g., in the following we suppose that all the OR and AND nodes are binary. Indeed, $(\phi_1 \oplus \phi_2 \oplus \cdots \oplus \phi_n)$ can be rewritten as $(\phi_1 \oplus (\phi_2 \oplus (\cdots \oplus (\phi_{n-1} \oplus \phi_n))))$, where $\oplus \in \{\wedge, \vee\}$. Let us remind also that the boolean operators are commutative, so we only need to prove that property on the left or on the right side. Let ϕ be an NNF formula in $\mathcal{L}$ containing $(\psi_1 \vee \psi_2)$ and ϕ' be equals to ϕ but with $(\psi_1 \vee \psi_2)$ replaced by ψ_1. We show that, if there exists a Kripke model $\mathcal{K} = \langle W, R, V\rangle$ and a possible world $w \in W$, s.t. $\langle \mathcal{K}, w\rangle \models \phi'$ then $\langle \mathcal{K}, w\rangle \models \phi$ by induction on the structure of ϕ.

Induction base: $\phi = (\psi_1 \vee \psi_2)$ and $\phi' = \psi_1$, where ψ_1 contains no operator $\vee$. The claim is clearly true.

Let us now prove the different cases in the induction step:
(1): $\phi = (\chi_1 \wedge \chi_2)$ and χ_1 contains $(\psi_1 \vee \psi_2)$ and $\phi' = (\chi' \wedge \chi_2)$ where χ' is χ where $(\psi_1 \vee \psi_2)$ has been replaced by ψ_1. The case where χ_2 contains $(\psi_1 \vee \psi_2)$ is analogous due to the commutativity. We have $\langle \mathcal{K}, w\rangle \models (\chi'_1 \wedge \chi_2)$ iff $\langle \mathcal{K}, w\rangle \models \chi'_1$ and $\langle \mathcal{K}, w\rangle \models \chi_2$. By IH, since we have $\langle \mathcal{K}, w\rangle \models \chi'_1$, we have $\langle \mathcal{K}, w\rangle \models \chi_1$. And because we also have $\langle \mathcal{K}, w\rangle \models \chi_2$, then we have $\langle \mathcal{K}, w\rangle \models (\chi_1 \wedge \chi_2)$. Thus $\langle \mathcal{K}, w\rangle \models \phi$.
(2): $\phi = (\chi_1 \vee \chi_2)$ and χ_1 contains $(\psi_1 \vee \psi_2)$ and $\phi' = (\chi' \vee \chi_2)$ where χ' is χ where $(\psi_1 \vee \psi_2)$ has been replaced by ψ_1. The case where χ_2 contains $(\psi_1 \vee \psi_2)$ is analogous due to the commutativity. We have $\langle \mathcal{K}, w\rangle \models (\chi'_1 \vee \chi_2)$ iff $\langle \mathcal{K}, w\rangle \models \chi'_1$ or $\langle \mathcal{K}, w\rangle \models \chi_2$. Two cases have to be considered, if we have $\langle \mathcal{K}, w\rangle \models \chi_2$, then we also have $\langle \mathcal{K}, w\rangle \models (\chi_1 \vee \chi_2)$. Otherwise, we have $\langle \mathcal{K}, w\rangle \models \chi'_1$, by IH we have $\langle \mathcal{K}, w\rangle \models (\chi_1$. Then we have $\langle \mathcal{K}, w\rangle \models (\chi_1 \vee \chi_2)$. Thus we have $\langle \mathcal{K}, w\rangle \models \phi$.
(3): $\phi = \Box\chi$ and χ contains $(\psi_1 \vee \psi_2)$ and $\phi' = \Box\chi'$ where χ' is χ where $(\psi_1 \vee \psi_2)$ has been replaced by ψ_1. We have $\langle \mathcal{K}, w\rangle \models \Box\chi'$ iff $\forall w'$ s.t. if $(w, w') \in R_a$ then we have $\langle \mathcal{K}, w'\rangle \models \chi'$. By IH, $\langle \mathcal{K}, w'\rangle \models \chi$, then $\forall w'$ s.t. if $(w, w') \in R_a$ then we have $\langle \mathcal{K}, w\rangle \models \Box\chi$. Thus $\langle \mathcal{K}, w\rangle \models \phi$.
(4): $\phi = \Diamond\chi$ and χ contains $(\psi_1 \vee \psi_2)$ and $\phi' = \Diamond\chi'$ where χ' is χ where $(\psi_1 \vee \psi_2)$ has been replaced by ψ_1. Analogous to (3).

Thus, $\forall\phi$ in NNF, if $\langle \mathcal{K}, w\rangle \models \phi'$ then $\langle \mathcal{K}, w\rangle \models \phi$. q.e.d □

Let us remark that Proposition 1 can be extended to the case where a set of edges rooted in OR nodes are cut. Consequently, it is possible to consider a new kind of over-abstraction that cuts edges rooted in OR nodes. More precisely, we create such over-abstraction sub-formula by cutting heuristically a set of edges rooted in OR nodes which have the bigger impact on the upper-bound and we translate it into a CNF formula. We call this function cut-or(ϕ, b), where the parameter b is the number of OR nodes cut. Obviously, the refinement function is linked to

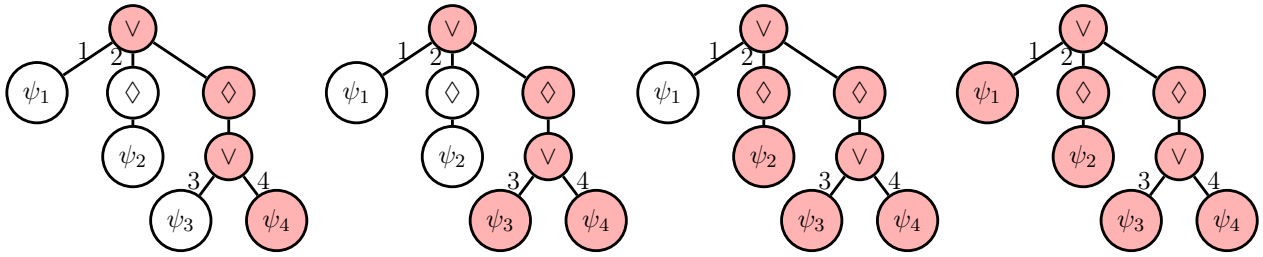


Figure 4.3: Function cut-or$(\phi, 3)$ and its refinements

this over-abstraction (it is not possible to cut an edge and then increase the number of worlds without violating the RECAR Assumptions). Thus, we consider a new refinement function refine$_{or}(\hat{\phi}, b)$ that is inductively defined as follows:

$$\text{refine}_{or}(\hat{\phi}, b) = \begin{cases} \hat{\phi} & \text{if } b = 0 \\ \phi' & \forall b > 0 \text{ and } \text{refine}_{or}(\hat{\phi}, b-1) \neq \phi \\ \phi & \text{otherwise} \end{cases}$$

where ϕ' is defined such that refine$_{or}(\hat{\phi}, b-1) \subsetneq \phi' \subseteq \phi$. Intuitively, refine$_{or}(\hat{\phi}, b)$ restores b OR nodes to $\hat{\phi}$ if possible.

Basically the refine function consists in restoring at least one edge which were cut at each induction step. Now, let us demonstrate that the proposed couple over-abstraction cut-or(ϕ, b) and refinement function refine$_o^n(\psi)$ satisfies the needed conditions recalled in preliminaries. An illustration of the functions "cut-or$''$ and "refine$_o''$ is done in Figure 4.3.

Theorem 30

ψ = cut-or(ϕ, n) is satisfiable implies refine$_{or}(\psi, 1)$ is satisfiable (Assump. 2) and refine$_{or}(\psi, n) \equiv_{\text{sat}} \phi$ (Assump. 3).

Proof. The proof is straightforward. From Proposition 1 we know that adding back an edge rooted in an OR node preserves the satisfiability. Because refine$_{or}(\psi, b)$ can only add edges back, then we have directly that ψ = cut-or(ϕ, n) is satisfiable implies refine$_{or}(\psi, 1)$ is satisfiable. For the Assumption 3, the result comes directly from the definition of refine$_{or}(\psi, n)$ and the fact that we can only add a finite number of edges. q.e.d □

4.2.4 Chain of Modalities Simplifications

Once all above simplifications are performed, the resulting "abstracted" formula may contain chains of modalities. This is a critical information that can be exploited if we want to further improve the performance of a solver.

Simplifications for Modal Logic S4

They are known simplifications of chain of modalities in modal logic S4 which preserves logical equivalency (Van Benthem 2010, Sec. 5.5).

$$\Box\Box\phi \leftrightarrow \Box\phi \tag{4.1}$$
$$\Diamond\Diamond\phi \leftrightarrow \Diamond\phi \tag{4.2}$$
$$\Box\Diamond\Box\Diamond\phi \leftrightarrow \Box\Diamond\phi \tag{4.3}$$

As such, it means that any chain of modal operators (involving the same modality a) can be reduced to a chain of maximum 3 modal operators in modal logic S4, which is tolerable for the translation into CNF done by the over-approximation function.

Because modal logic K and modal logic KT "have infinitely many non-equivalent modalities" Van Benthem (2010), there is no such result to the best of our knowledge for K and KT. To deal with chains of modalities in those logics, we propose the following simplifications which preserve satisfiability.

Simplifications for Modal Logic K

Let us consider a simple case where the modal prefix contains only diamonds. In this case, it is safe to test only the end of the chain without taking into account the modalities. The resulting sub-formula is equisatisfiable to the original formula.

Theorem 31

$\Diamond\Diamond\ldots\Diamond\psi \equiv_{\text{sat}} \psi$ in modal logic K.

Proof. ($\Rightarrow$) If $\langle\mathcal{K}, w\rangle \models \Diamond\ldots\Diamond\psi$ then there is $w' s.t. \langle\mathcal{K}, w'\rangle \models \psi$. ($\Leftarrow$) If $\langle\mathcal{K}, w\rangle \models \psi$ then we can add a world w' to $\mathcal{K}$ s.t. $w \in R(w')$. In this case, $\langle\mathcal{K}, w'\rangle \models \Diamond\psi$. By continuing doing so, we can show that $\Diamond\ldots\Diamond\psi$ is sat. q.e.d □

Theorem 32

$\Diamond\ldots\Diamond\Box\psi$ is sat in modal logic K.

Proof. $\Box\psi$ is satisfiable for all $\psi \in \mathcal{L}$. Therefore, by Theorem 31, $\Diamond\ldots\Diamond\Box\psi$ is satisfiable for all $\psi \in \mathcal{L}$. q.e.d □

Simplifications for Modal Logic KT

Modal logic KT is different. Reflexivity (Axiom (T)) implies that $\Box\bot$ is unsatisfiable, which means we cannot apply the same technique as in K. Moreover, because of the absence of transitivity (Axiom (4)), we cannot simplify $(\Box\Diamond\Box\Diamond\phi)$ into $(\Box\Diamond\phi)$, as in S4. We thus propose new simplifications which preserve satisfiability (but not equivalence) in KT. However it is still possible to retrieve a model for the original formula by finding a model for the simplified formula.

Theorem 33

$\Box \circ \psi \equiv_{\text{sat}} \circ\psi$, for $\circ \in \{\Box, \Diamond\}$.

Proof. Case 1: $\circ = \Diamond$. ($\Rightarrow$) If $\Box\Diamond\psi$ is sat then, by Axiom (T), $\Diamond\psi$ is sat. ($\Leftarrow$) Assume $\langle\mathcal{K}, w\rangle \models \Diamond\psi$. We can create a new model $\langle\mathcal{K}', w\rangle$ which is the unwind of $\langle\mathcal{K}, w\rangle$. This is an infinite tree with root at w which is bisimilar to $\langle\mathcal{K}, w\rangle$. By assumption, there at least one world $w' \in R(w)$ s.t. $\langle\mathcal{K}', w'\rangle \models \psi$. Also note that the reflexivity property of the original structure $\mathcal{K}$ implies $\langle\mathcal{K}', w'\rangle \models \Diamond\psi$. There can also be some worlds $w'' \in R(w)$ s.t. $\langle\mathcal{K}, w''\rangle \not\models \psi$. We can remove all such branches from the root. In the model obtained, all worlds accessible from w satisfy ψ. To make this model a KT-model, we add a reflexive arrow only on the root w. This final model satisfies $\Box\Diamond\psi$. Case 2 is similar. The only difference is that there are no branches w'' to be removed. q.e.d □

Theorem 34

$\Diamond\psi \equiv_{\text{sat}} \psi$

Proof. ($\Rightarrow$) If there is $\langle\mathcal{K}, w\rangle \models \Diamond\psi$, then there is $w' \in R(w)$ s.t. $\langle\mathcal{K}, w'\rangle \models \psi$. ($\Leftarrow$) If $\langle\mathcal{K}, w\rangle \models \psi$, by the contraposition Axiom (T), $\langle\mathcal{K}, w\rangle \models \Diamond\psi$. q.e.d □

The original version of MoSaiC, *ie.* the version without the Axiom-Aware under-abstraction and the Space-aware over-abstraction, will be call for now on MoSaiC 1.0. Whereas the one with these two abstractions plugged and the use of simplifications will be called MoSaiC 2.0. Let us see in the next section, how such a solver, in its version 1.0 and 2.0 is able to compete against state-of-the-art approaches for modal logic K, KT and S4.

4.3 Experimental Evaluation Of MoSaiC

First, let us go through the experimental conditions that we put to evaluate the different modal logic solvers. The experiments ran on a cluster of Xeon, 4 cores, 3.3 GHz with CentOS 6.4 with a memory limit of 32GB and a runtime limit of 900 seconds per solver per benchmark, no matter the logic considered. All solvers answers have been checked since the satisfiability of each benchmark is known by design. No discrepancy was found.

4.3.1 Experimental Evaluation of MoSaiC 1.0

We compared MoSaiC 1.0, against state-of-the-art solvers for the modal logics K, namely:

- K_SP 0.1.2 Nalon et al. (2016)
- Km2SAT 1.0 Sebastiani and Vescovi (2009)
- Vampire 4.0 Kovács and Voronkov (2013) with a combination of the optimized functional translation Horrocks et al. (2006)
- *SAT Giunchiglia et al. (2002)
- BDDTab 1.0 Goré et al. (2014)
- FaCT++ 1.6.4 Tsarkov and Horrocks (2006)
- InKreSAT 1.0 Kaminski and Tebbi (2013)
- Spartacus 1.1.3 Götzmann et al. (2010)
- MoSaiC 1.0 Lagniez et al. (2017a) in two modes RECAR and CEGAR

What we mean by MoSaiC 1.0 in CEGAR mode is that, we use MoSaiC 1.0 to solve the instance, but we do not create any under-abstraction. Thus what MoSaiC 1.0 in CEGAR mode is doing could be represented by the Figure 4.4.

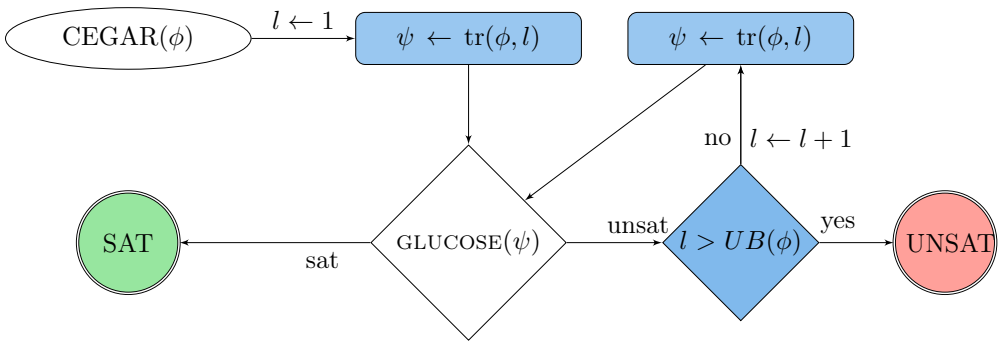


Figure 4.4: MoSaiC 1.0 in CEGAR mode

What we expect from MoSaiC 1.0 in CEGAR mode is to be efficient for satisfiable benchmarks but far from efficient for unsatisfiable benchmarks. Indeed, its only way to decide the unsatisfiability is to reach the theoretical upper-bound of the original formula or by simplifications to obtain $\bot$.

For our first analysis of MoSaiC 1.0, our goal is not yet to analyse its results precisely against the state-of-the-art, we will do that in MoSaiC 2.0 because it is its latest version thus the one usable by others. Our first goal is to see if the RECAR short-cut is efficient and leads to progress against just a simple CEGAR loop. We deal with all the benchmarks for modal logic K in one shot without any distinction yet. So we compare the solvers on the classical LWB benchmarks for modal logics K Balsiger et al. (2000), on the MQBF Massacci and Donini (2000)

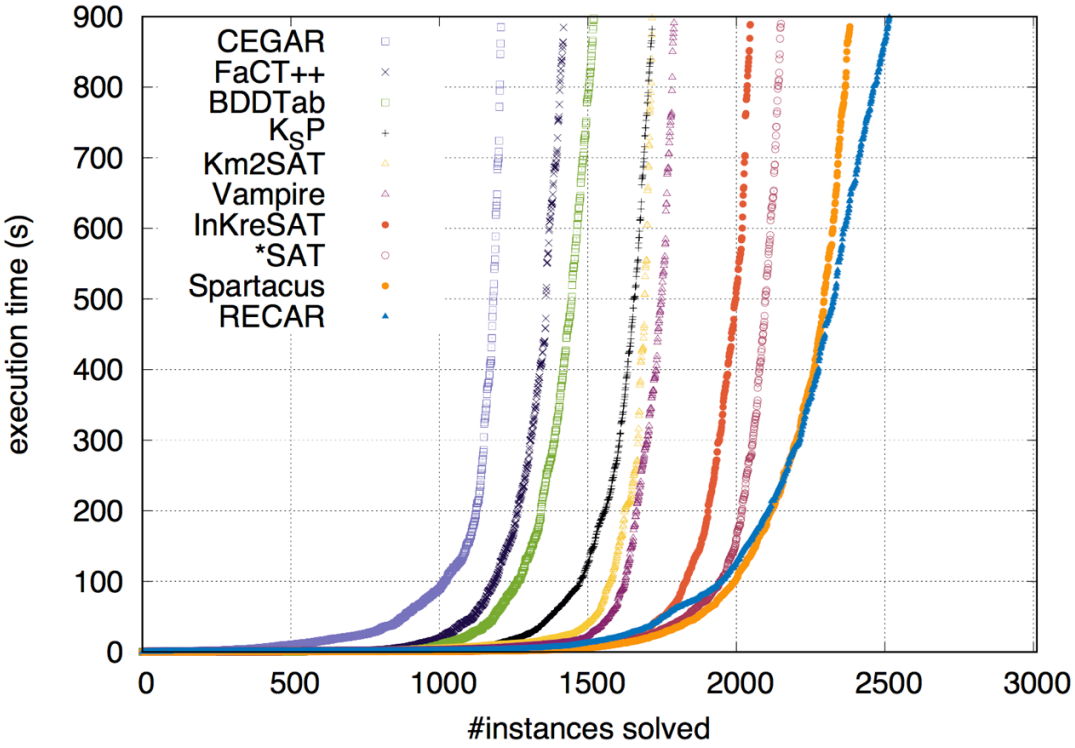


Figure 4.5: Runtime distribution on a different benchmarks for modal logic K

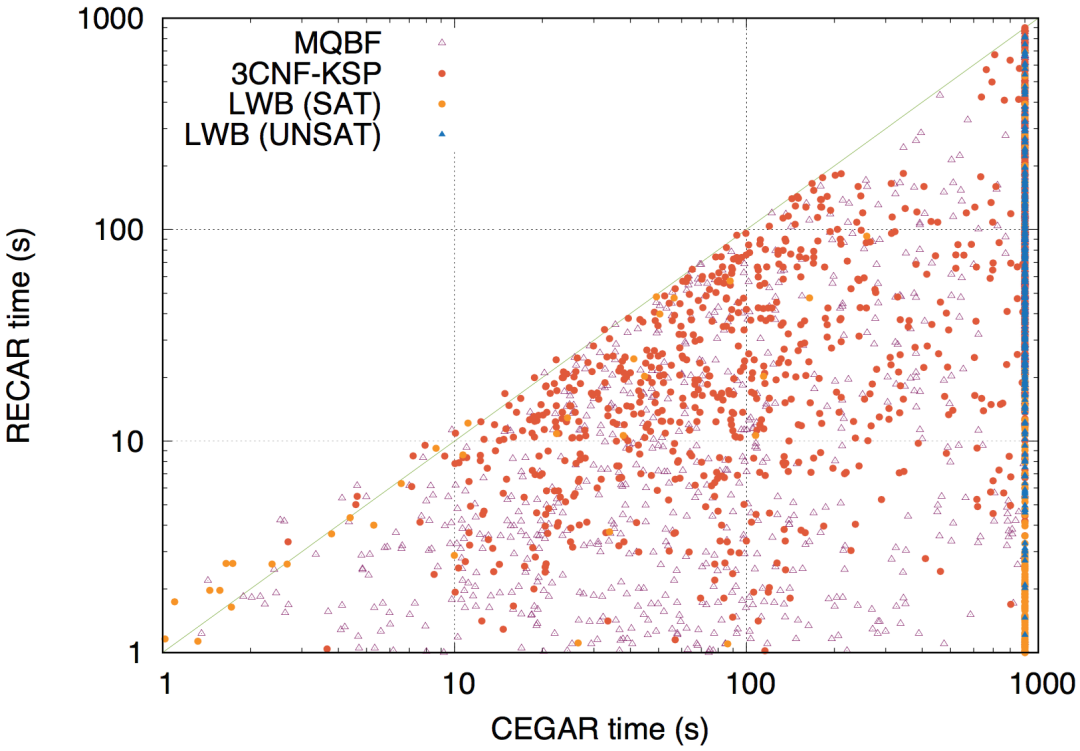


Figure 4.6: Scatter-plot CEGAR vs RECAR

and on the CNF-KSP Patel-Schneider and Sebastiani (2003). This leads to a mega-set of 3024 (1008 + 1016 + 1000) benchmarks respectively. With 1578 (504 + 617 + 457) benchmarks being satisfiable and 1367 (504 + 399 + 464) benchmarks being unsatisfiable.

Figure 4.5 shows the runtime distribution on all the benchmarks for all the solvers that we consider to solve modal logic K Satisfiability Problem. We can see that our over-abstraction CEGAR approach is the worst solver whereas our RECAR approach outperforms the other solvers. This is easy to understand, There is basically 1367 benchmarks which are almost impossible for CEGAR to solve. Km2SAT performs specific reasoning to detect earlier some UNSAT benchmarks without generating the CNF, which explains why it performs much better than our CEGAR approach. *SAT interleaves SAT reasoning and domain reasoning, and can be considered as an under-approximation cegar approach. It shows good results, despite being tied with the old SAT solver SATO[12].

Our best competitor, Spartacus, is based on a tableaux method, not on SAT: SAT based-techniques were not the best way to tackle such problems up to now. Spartacus reaches the time-out on unsolved benchmarks while we exhaust the available memory: the solvers behave quite differently and have different limits. We will analyse more finely such difference in the next section.

We can see in Figure 4.6 that for most benchmarks, the RECAR approach outperforms the CEGAR one. The under approximation often provides a formula with a much smaller upper-bound, which produces a CNF of reasonable size to be handed in to the SAT solver. Note that in this plot, memory out for the CEGAR approach is denoted by a timeout, *ie.* a point at 900 seconds. This is mainly due to improvements in solving unsatisfiable benchmarks (1118/1367) for RECAR vs (155/1367) for CEGAR. For satisfiable benchmarks, the bound update resulting from the recursive call helps to reach faster a satisfiable formula. 1446 for RECAR vs 1053 for CEGAR.

From this analysis, the results were clear: MoSaiC is slower than Spartacus on many benchmarks because we spend too many times translating the formula instead of just deciding it directly. Moreover, Spartacus is able to deal with more logics than just modal logic K, this is mainly why we also decide to do so. Basically MoSaiC 1.0 was a confirmation: a simple CEGAR approach cannot work, the RECAR framework is efficient to decide PSPACE-complete problems, but to push the envelope forward, we will need other abstractions, more adapted to the problems and the benchmarks that we consider to evaluate the solvers.

4.3.2 Experimental Evaluation of MoSaiC 2.0

We chose to compare the solvers on the classical LWB benchmarks for modal logics K, KT and S4 Balsiger et al. (2000). These benchmarks are generated using the script from Nalon et al. (2016) using 56 formulas with 18 parameter settings, for a total of 1008 formulas, 504 satisfiable, 504 unsatisfiable Balsiger et al. (2000) for each logic. We compared MoSaiC 2.0, against state-of-the-art solvers for the modal logics K, KT and S4, namely:

- Moloss 0.9 Areces et al. (2015)
- K_SP 0.1.2 Nalon et al. (2016)
- BDDTab 1.0 Goré et al. (2014)

[12] *SAT is deeply integrated with SATO, which makes very difficult an update to a more recent SAT solver.

Solver	$\mathbf{LWB}_K$ SAT	$\mathbf{LWB}_K$ UNSAT	$\mathbf{Total}_K$	$\mathbf{LWB}_{KT}$ SAT	$\mathbf{LWB}_{KT}$ UNSAT	$\mathbf{Total}_{KT}$	$\mathbf{LWB}_{S4}$ SAT	$\mathbf{LWB}_{S4}$ UNSAT	$\mathbf{Total}_{S4}$
#Instances	504	504	1008	504	504	1008	504	504	1008
Moloss	71 (0)	83 (0)	154 (0)	68 (0)	170 (0)	238 (0)	269 (0)	203 (0)	472 (0)
InKreSAT	192 (24)	247 (0)	439 (24)	155 (9)	193 (0)	348 (9)	248 (0)	304 (0)	552 (0)
BDDTab	248 (5)	277 (4)	525 (9)	–	–	–	211 (0)	270 (0)	481 (0)
FaCT++	264 (10)	284 (19)	548 (29)	184 (30)	226 (59)	410 (89)	298 (42)	338 (25)	636 (67)
MoSaiC 1.0	263 (241)	306 (198)	569 (439)	230 (251)	222 (253)	452 (504)	277 (229)	225 (277)	502 (506)
K_SP	249 (4)	**328** (3)	577 (7)	130 (2)	93 (0)	223 (2)	223 (0)	205 (0)	428 (0)
Spartacus	331 (33)	320 (10)	651 (43)	207 (74)	251 (59)	458 (133)	273 (17)	350 (13)	623 (30)
MoSaiC 2.0	**362** (142)	317 (78)	**679** (220)	**304** (167)	**245** (223)	**549** (390)	**360** (8)	**381** (40)	**741** (48)
VBS	362	342	704	304	245	549	364	381	741

Table 4.1: Number of LWB instances solved in K, KT and S4

- FaCT++ 1.6.4 Tsarkov and Horrocks (2006)
- InKreSAT 1.0 Kaminski and Tebbi (2013)
- Spartacus 1.1.3 Götzmann et al. (2010)
- MoSaiC 1.0 Lagniez et al. (2017a) extended with the translation of the axioms.

In practice, MoSaiC 2.0 is able to change which over-abstraction function it is using, compared to MoSaiC 1.0 which is just increasing the number of worlds as an over-abstraction. We determined experimentally as an heuristic for when stop using the original abstraction and start using the one that we proposed here that the threshold ($|\phi| \times currentBound > 5000$) worked fine for these benchmarks on our computer with memory limit of 32GB. When this threshold is reached, MoSaiC starts to cut some edges in order to make the formula smaller and thus faster to translate.

It is important to notice that K_SP (kindly provided by its authors) is still under development for modal logics KT and S4. Its results should be considered as preliminary. We can see that the number of memory-out between MoSaiC 1.0 and MoSaiC 2.0 reduces drastically in the three logics thanks to the new over-abstraction, which reduces the size of the formula that need to be translated. Thus, the memory used to solve the instance (if it manages to solve it).

When we take a closer look at the runtime, we can see that not only the number of memory-out reduces, but the new abstractions provide a speed-up. When we look at Figure 4.7, we can see that, even if MoSaiC 1.0 was designed to deal with modal logic K, its main problem is that it spends too much time translating the formula instead of solving them.

Small sum up of the difference between MoSaiC 1.0 and 2.0 The only difference in modal logic K between MoSaiC 1.0 and 2.0 are the new over-abstraction and some simplifications on the prenex of modalities. Basically, if the new over-abstraction provides such a gain is an explanation about the fact that, if for a SAT-based approach, we try to translate part by part and not the formula as a whole, we manage to solve many more instances.

We can see the progression of SAT-based approaches between Km2SAT Sebastiani and Vescovi (2009) which translates smartly the formula as a whole, MoSaiC 1.0 Lagniez et al. (2017a) which translates the whole-formula with a bound on the size of the Kripke model and MoSaiC 2.0 which translates smartly chosen sub-formula with a bound on the size of the Kripke model. The

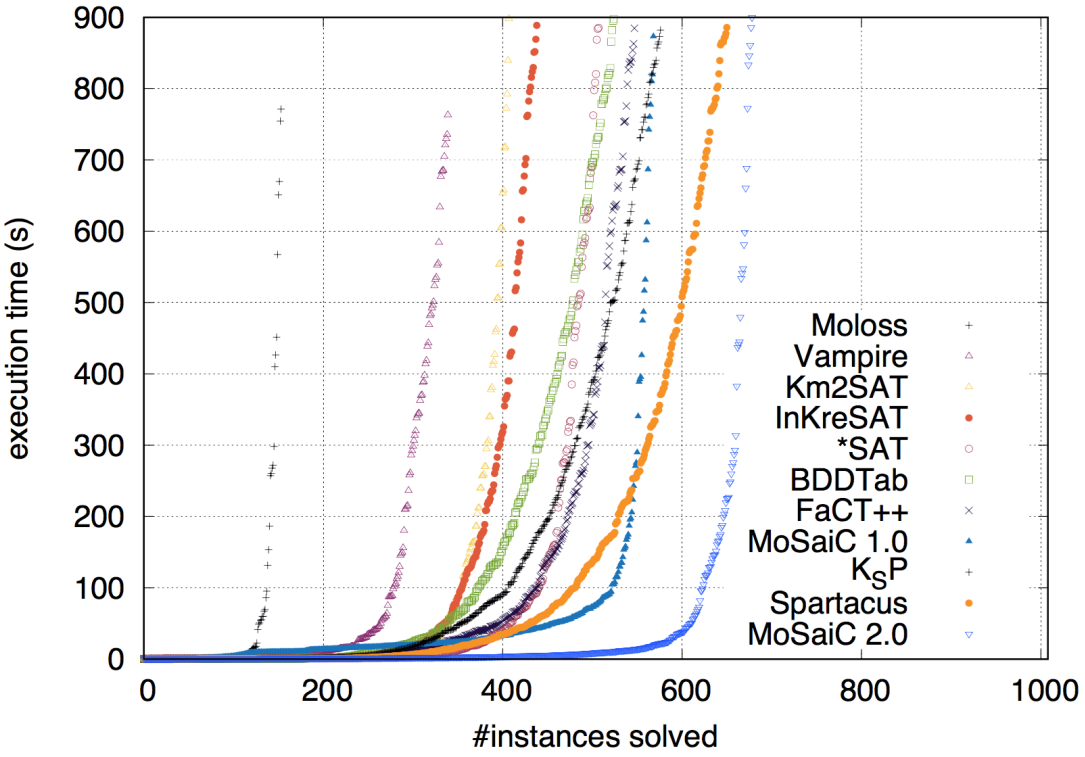


Figure 4.7: Runtime distribution on LWB in modal logic K

results are clear: the smaller is the need-to-be-translate formula, the better are the results of the solver.

When we take a look at the results on modal logic KT, which are depicted in Table 4.1 and Figure 4.8 we were surprised by the results obtained by MoSaiC 1.0 which is not designed for modal logic KT at the beginning. But these results can easily be explained: the translation of the axiom (T) is adding unit clauses in the propositional logic formula. This is boosting the unit propagation of the embedded SAT solver which can thus decide quickly the satisfiability of the formula. However, as we can see on Figure 4.8, the new under-abstraction and the simplifications lead to a huge gain in the performance of MoSaiC.

In modal logic S4, we can see that the gap between MoSaiC 1.0 and MoSaiC 2.0 is bigger than in modal logic K or KT, as depicted in Figure 4.9. However, the speed-up obtained thanks to the translation of axiom (T) which add unit clauses is killed by the translation of axiom (4), which is here, making the formula harder by adding many clauses.

The SAT-based approach for S4 is extremely efficient due to the simplifications of modality chains, which reduce translation time. Indeed, the simplifications can be apply on any sub-formula which is definitely helping the solver by reducing the size of formula. Moreover, because the time of translation is reduced, due to the new over-abstraction and the new under-abstraction, MoSaiC 2.0 is much faster than MoSaiC 1.0 (2.3s vs 31s median time).

In order to understand the difference on the efficiency on solving these logics, we collected information about the size of the computed Kripke models. As depicted in Table 4.2, adding axioms tends to increase the size of the models found in number of worlds. This can be partially explained by the fact that these models must satisfy more constraints on the Kripke model we are

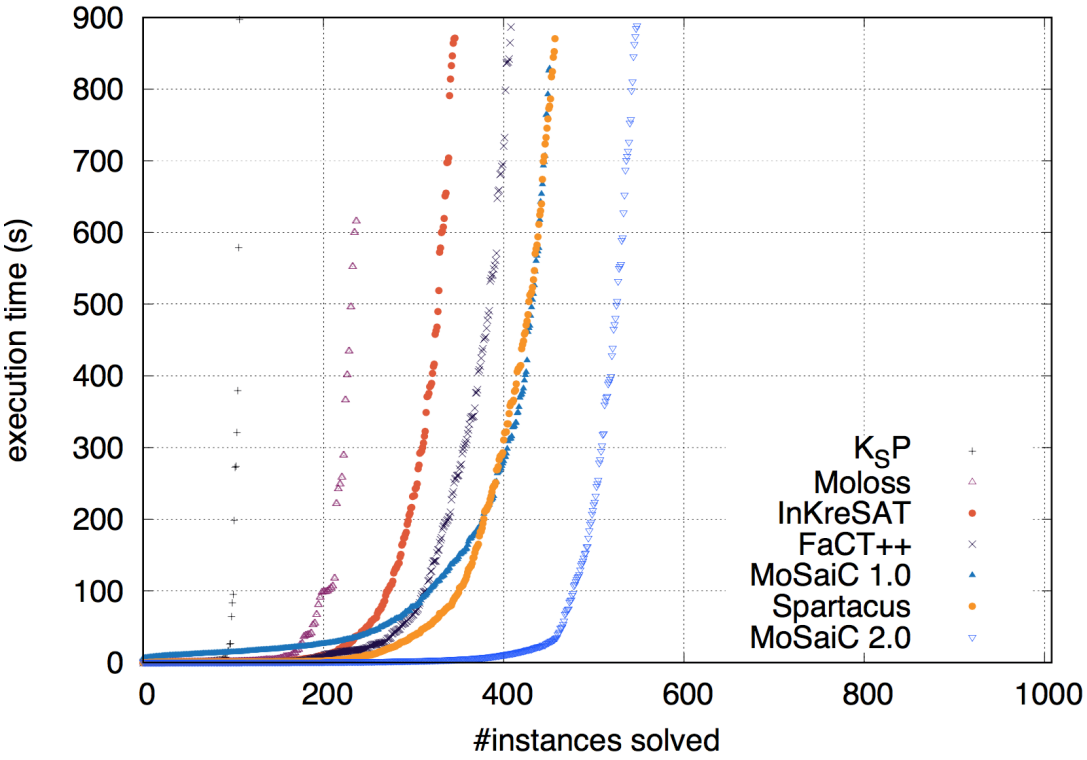


Figure 4.8: Runtime distribution on LWB in modal logic KT

	min	Q_1	med	mean	Q_3	max
K	1	2	22	174	279	903
KT	1	52	207	364	613	1505
S4	1	33	113	256	399	1217

Table 4.2: Sizes of the Kripke structures for satisfiable benchmarks for each modal logic

searching. It happens fewer times in S4 because we can greatly reduce the number of modalities.

4.3.3 General Analysis Of The Results Obtained

As explained in Section 3.4, the behaviour of a RECAR-over solver should be to have trajectories between a CEGAR-over behaviour and a Branch and Price behaviour. Let us see now, MoSaiC 2.0 on some peculiar benchmarks to see exactly how it is behaving and how the abstractions are used to decide problems.

On Figure 4.10, we can see the behaviour of MoSaiC 2.0 on different instances, they were picked especially to display a behaviour. There are unfortunately too many benchmarks and some times too many steps of refinements to display them all. So we just pick eight different instances and display only until 100 steps of refinements.

First thing we can see is that, because MoSaiC is not design to do so, they are never a line below the Branch And Price line. Indeed, MoSaiC can do at most as many over-abstraction refinement steps as under-abstraction refinement steps. Each under-abstraction step is performed by a recursive call, and the first thing it does after this call is to perform an over-abstraction.

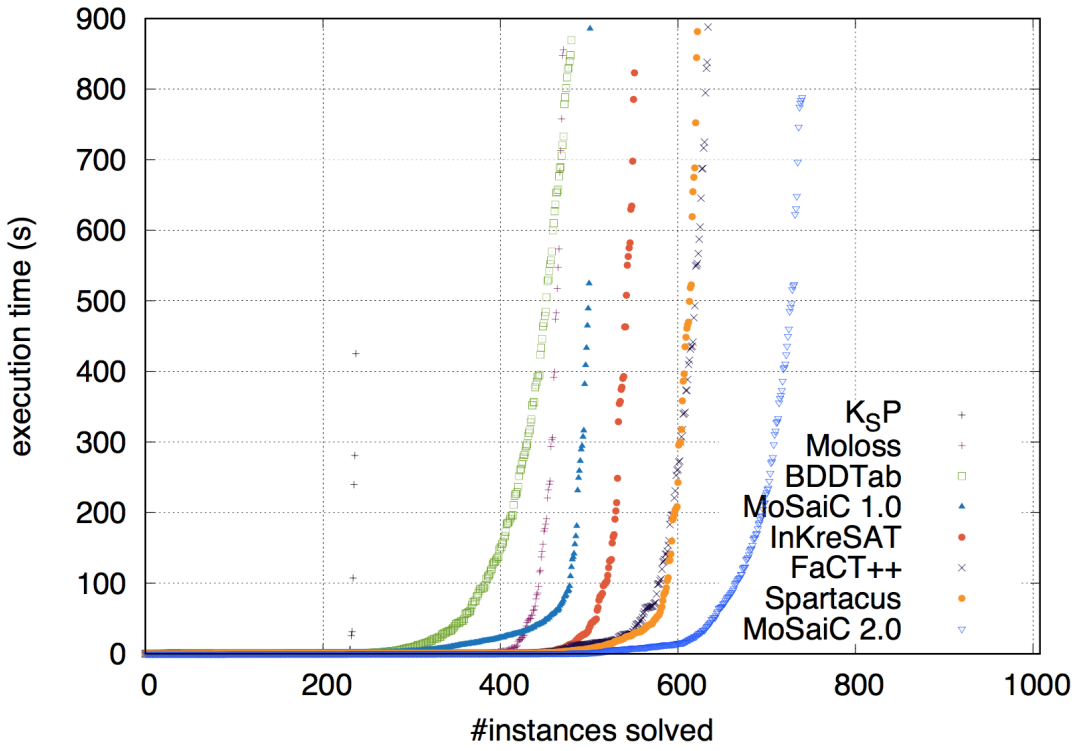


Figure 4.9: Runtime distribution on LWB in modal logic S4

Then, we can also see that the behaviour of MoSaiC when it want to solve pigeon-hole problems is very similar to the behaviour that could have a CEGAR-over approach. Indeed in such case, it is usually the case that all the selectors are part of the unsatisfiable core, so there are few under-abstractions. Moreover, even one there is one, after one refinement we decide that it is satisfiable and we go back dealing with the whole formula.

On the other side of the lines is the branch-formulas as defined in Halpern and Moses (1992) where here we do the opposite and what we do in the case of pigeon-hole problems. We do almost at each-step, an under-abstraction refinement. The problem again in this case is that we do not output much information from it. Those are the worst-case possible for us, and more generally they are the worst problems for SAT-based approach, as explained in Sebastiani and Vescovi (2009). They force us to do many steps of under-abstraction refinement, but we do not take many information from it. So because of that, we are just extremely slow to decide these problems, when we manage to do so.

And between these two lines, they are many different trajectories. *poly*, *lin* and *3CNF* are (and it is also the case for the other benchmarks of these categories) usually few under-abstraction at the beginning, which allow to reduce greatly, the size of the formula. And then just a very fast and high number of over-abstraction refinement steps which allow MoSaiC to decide these problems. The three last lines are part of the MQBF set of benchmarks are here, they are more or less the *normal* behaviour of MoSaiC.

We can even see, on the *modKLadn* benchmark, MoSaiC deciding the problem. What is happening here is that, the under-abstraction managed to reduce the input to a very small sub-formula and with an upper-bound around 90, we manage to decide that the problem was unsatisfiable.

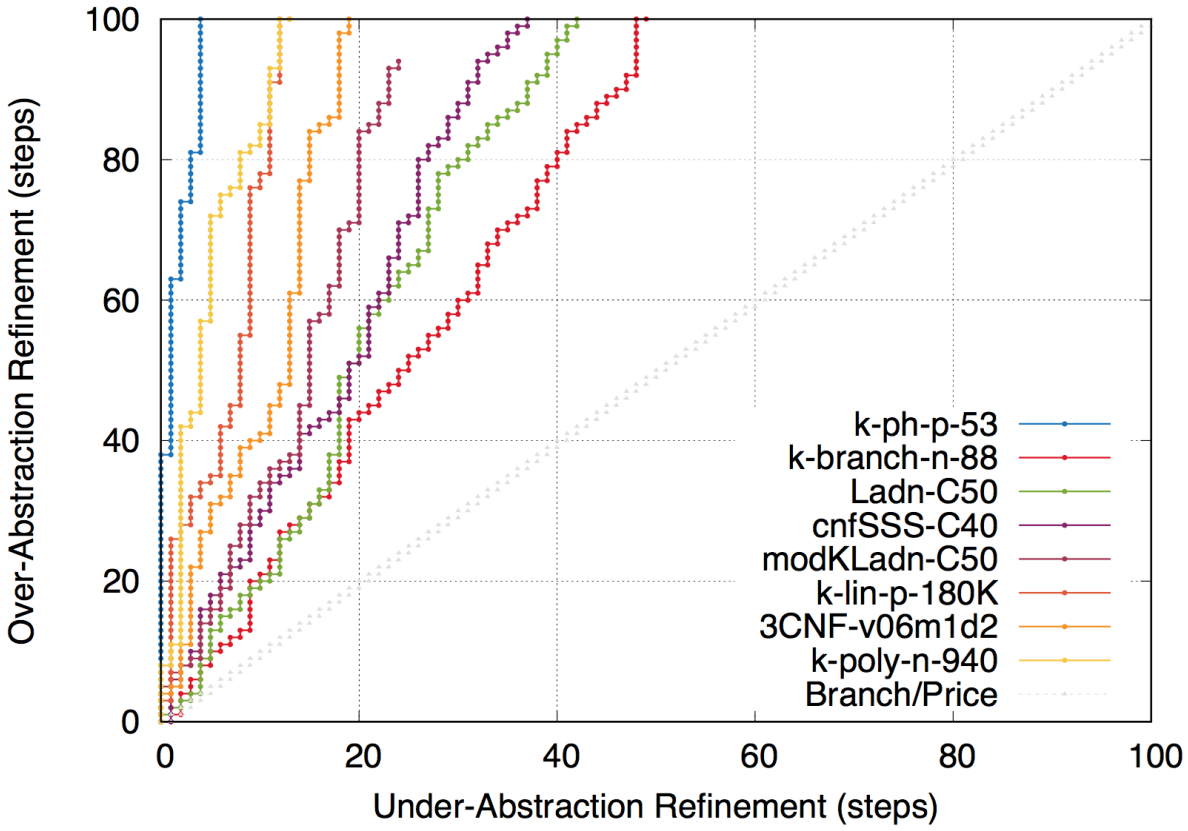


Figure 4.10: Behaviour of MoSaiC 2.0 on some benchmarks in modal logic K

Conclusion And Perspectives

Conclusion

In this work we have explored the idea of exploiting direct and incremental SAT-based techniques for the automated reasoning in all the Modal Logics based on K. Just after introducing, in Part I, the necessary elements to understand this thesis, we proposed in Part II, different approaches to solve the different modal logic satisfiability problems:

- First of all, we explored the idea of solving the NP-complete modal logics with a direct encoding into a propositional logic satisfiability problem which can thus be solved with an off-the-shelves SAT solver. By doing so, we proposed a syntactic-property that we called the *diamond-degree* and we proved that the maximum number of worlds to satisfy a formula in modal logics K$\star$5 is linear in the *diamond-degree* in the formula. Once this upper-bound has been established, we proposed a series of axiom-dependent simplifications in order to reduce the size of the input formula without loosing the equi-satisfiability. Then, we presented a SAT encoding of the problem, which can deal with all the axioms of the normal modal logics. We demonstrated experimentally that such a SAT encoding is extremely efficient in practice to decide the satisfiability of the problem.

- In a second chapter, once we analysed the results obtained by our SAT encoding, we realized that the upper-bound proposed (*diamond-degree*) is usually far too big compared to what is the minimum required to satisfy the formula. Having this idea in mind, we thus proposed a new optimization problem that we call the Minimal K$\star$5 Satisfiability Problem which is to find the smallest Kripke structure which K$\star$5-satisfy a formula. To solve this optimization problem we presented different techniques. The first one being a simple linear search with the direct encoding. The second one being an encoding into a MaxSAT or PBO problems solvable with an off-the-shelves MaxSAT or PBO solvers. Then we proposed an incremental-SAT based approaches, using unsatisfiable cores to speed-up the linear search. We demonstrated experimentally that "giving knowledge" to the SAT solver, by using selectors and analysing them to understand why a formula is unsatisfiable, is the best approach to solve the Minimal K$\star$5 Satisfiability Problem. During this chapter, we also proposed a new series of benchmarks, thanks to a translator from Planning problems with uncertainties in the initial states into modal logic S5 formulas and we show that the results that we can draw from the experiments are benchmarks-dependent. We do not draw the same conclusion if we just use the state-of-the-art benchmarks or our Planning benchmarks.

- Once the NP-complete modal logics were explored, we wanted to deal with PSPACE-complete modal logics. Unfortunately a direct encoding into a SAT problem is first, already proposed by the extremely efficient Km2SAT Sebastiani and Vescovi (2009) solver, and second not necessary the best outcome possible due to the translation being exponential in the size of the input. To thus deal efficiently with PSPACE-complete modal logics, we presented a framework that we call RECAR, based on the CEGAR framework, which is a generic way to incrementally solve PSPACE problems without having to deal, except in the worst-case, with the exponential blow-up. We demonstrated that this framework is

sound, complete and terminates and that it can be instantiated in two ways, as CEGAR. The first way, that we call RECAR-over, is to first perform an over-abstraction and to call recursively the framework on an under-abstraction. The second way, that we call RECAR-under, is to first perform an under-abstraction and to call recursively the framework on an over-abstraction.

- Finally, we wanted to instantiate our framework to see its efficiency on PSPACE-complete problems. To do so, we developed the solver MoSaiC, able to deal with modal logic K in its 1.0 version. The results we obtained were extremely encouraging because, except being slower sometimes than the state-of-the-art approaches, we manage to solve more problems. The 2.0 version was thus already found: we need to solve faster instances and we want to deal with other PSPACE modal logics. To do so, we presented axiom-aware simplifications and new over-abstraction and under-abstraction. MoSaiC 2.0 was then tested against the state-of-the-art solvers for modal logics K, KT and S4 and we demonstrated experimentally that the RECAR framework instantiated with the over and under-abstractions that we proposed managed to solve faster and more instances than the state-of-the-art approaches.

We have at the end of this thesis, if we want to sum-up quickly our contributions: a new solver able to deal with all the NP-complete modal logics, a new set of benchmarks based on Planning with uncertainties problems, a new solver able to deal with all the PSPACE-complete modal logics, and finally, a framework totally generic, usable in two ways, to deal with PSPACE (or even beyond) problems.

Obviously, this thesis raises more questions than it solves and we give in the next sections, few directions that we consider worth exploring.

Future Works

We see several important research lines to explore in order to extend or enhance our results:

1. The first one of them, is a problem that we did not have time to consider in only three years of thesis: a generic upper-bound for PSPACE-complete modal logics. We conjecture that an upper-bound linear in $\mathtt{dd}(\phi)^{\mathrm{depth}(\phi)}$ could be an upper-bound for PSPACE-complete modal logics. It respects the theory saying that the bound must be exponential, it could make the link between NP and PSPACE modal logics and it respects the remark from Halpern and Moses (1992) stating that such an upper-bound has to be exponential in the modal depth of the formula.

2. Another perspective would be to deal with multiple modalities (multi-agent) and to generate "real world" benchmarks capturing such a meaning. What is proposed in MoSaiC can be extended easily for multi-agent modal logics. We did not do it basically because the state-of-the-art benchmarks are only mono-agent, so it was not worth to make even harder the explanation in this thesis.

3. A third perspective would be to finally instantiated the RECAR-under framework, which has not be the case to the best of our knowledge. One idea to do so is to deal with EXPTIME modal logics problems by adding the Common-Knowledge. The extension consists of the introduction of a group G of agents, and of n modal operators $\Box_i$ (with i = 1, ..., n) with the

intended meaning that "agent i knows." Thus $\Box_i\varphi$ (where φ is a formula of the calculus) is read “agent i knows φ”. We can define an operator E_G with the intended meaning of "everyone in group G knows" by defining it with the axiom $E_G\varphi \Leftrightarrow \bigwedge_{i\in G} \Box_i\varphi$ (more details about Common-Knowledge in Costa and Benevides (2005)). In such context, the under-abstraction is clear: if only one agent does not know φ, then it is impossible to have $E_G\varphi$, if it does, then we need to refine with two agents, then three, until, in the equisatisfiable case, where all the agents must know φ. With such an under-abstraction we can obtain a simple modal logic formula which can be decide with MoSaiC.

4. To see if MoSaiC could be using all the recent progress in Parallel-SAT solving, using just a Parallel SAT solver Balyo and Sinz (2018) such as D-Syrup Audemard et al. (2017) or AmPharoS Audemard et al. (2016) is not really satisfying because it does not exploit all the possibilities. One could think about different over-abstraction performed all in a parallel way and procedures about how to mix all the informations returned from all the over-abstraction solved. Or to get inspired by Bonacina (2018) which works is about how to solve First Order Logic formulas in parallel. Modal Logics being just a decidable fragment, it could be interesting to see how it can be adapted.

5. And finally a last perspective which is very broad: to spread the RECAR framework in many other domains where the CEGAR approaches are already efficient. Such as Model Checking Kupferman (2018) where there is, in the Handbook of Model Checking 2018, even a chapter about Abstraction and Abstraction Refinements Dams and Grumberg (2018). Or in QBF solving, in Planning, *etc.*

All these perspectives are for us extremely interesting and exciting future work.

Publications During The Thesis

This thesis has permit to publish a set of national and international publications. They have been used to explain the different contributions in this thesis. But I also worked with other colleagues on work unrelated with modal logics. All the publications have their names in **alphabetical order** and does not reflect the importance of each co-author. We all contributed equally to the different works, even though I am the main developper of all the softwares realized for these articles. In order to find more easily what were the contributions of three years of research, let us list them here:

International Conference Papers with Proceedings

- Lagniez et al. (2018b) Jean-Marie Lagniez, Daniel Le Berre, Tiago de Lima and Valentin Montmirail: Space-Awareness in a SAT-Based Approach For PSPACE Modal Logics. In: Proceedings of the Fifteenth International Conference of Knowledge Representation and Reasoning (KR 2018), October 2018, S. ?–?. – URL ? (to appear).
  - This article is presented in the Part II, Chapter 4, Section 4.2.
- Glorian et al. (2018) Gael Glorian, Jean-Marie Lagniez, Valentin Montmirail and Michael Sioutis: An Incremental SAT-Based Approach to Reason Efficiently On Qualitative Constraint Network. In: Proceedings of the 24th International Conference of Principles and Practice of Constraint Programming (CP 2018), August 2018, S. 160–178. – URL https://doi.org/10.1007/978-3-319-98334-9_11
  - This article is presented in Part II, Chapter 3, Section 3.2.
- Lagniez et al. (2018a) Jean-Marie Lagniez, Daniel Le Berre, Tiago de Lima and Valentin Montmirail: An Assumption-Based Approach for Solving The Minimal S5-Satisfiability Problem. In: Proceedings of the 9th International Joint Conference on Automated Reasoning (IJCAR 2018), July 2018, S. 1–18. – URL https://doi.org/10.1007/978-3-319-94205-6_1
  - This article is generalized in Part II, Chapter 2, Section 2.1.
  - This article presents only how to minimize and search for a S5-model.
- Lagniez et al. (2017a) Jean-Marie Lagniez, Daniel Le Berre, Tiago de Lima and Valentin Montmirail: A Recursive Short-Cut for CEGAR: Application To The Modal Logic K Satisfiability Problem. In: Proceedings of the 26th IJCAI International Joint Conference on Artificial Intelligence (IJCAI 2017), URL https://www.ijcai.org/proceedings/2017/94, August 2017, S. 674–680
  - This article is generalized in Part II, Chapter 3, Section 3.3.
  - The article presents only the RECAR-over framework.
- Caridroit et al. (2017a) Thomas Caridroit, Jean-Marie Lagniez, Daniel Le Berre, Tiago de Lima and Valentin Montmirail: A SAT-based approach for solving the modal logic S5 satisfiability problem. In: Proceedings of the 31st AAAI Conference on Artificial Intelligence (AAAI 2017), AAAI, Feb. 2017, S. 3864–3870. – URL http://aaai.org/ocs/index.php/AAAI/AAAI17/paper/view/14380/14111

  - This article is generalized in Part II, Chapter 1, Section 1.2.
  - This article talks only about S52SAT and a SAT translation from modal logic S5 to propositional logic.

International Workshop Papers with Proceedings

- Lagniez et al. (2016b) Jean-Marie Lagniez, Daniel Le Berre, Tiago de Lima and Valentin Montmirail: On Checking Kripke Models for Modal Logic K. In: 5th Workshop on Practical Aspects of Automated Reasoning (PAAR@IJCAR'16), Springer, June 2016, S. 69–81. – URL http://ceur-ws.org/Vol-1635/#paper-07

National Conference Papers with Proceedings

- Lagniez et al. (2018c) Jean-Marie Lagniez, Daniel Le Berre, Tiago de Lima and Valentin Montmirail: Une approche SAT incrémentale pour le problème de satisfiabilité minimale en logique modale S5. In: Actes des 14es Journées Francophones de Programmation par Contraintes (JFPC 2018), URL https://home.mis.u-picardie.fr/~evenement/JFPC2018/articles/JFPC_2018_papier_1.pdf, June 2018, S. 1-10
- Lagniez et al. (2017b) Jean-Marie Lagniez, Daniel Le Berre, Tiago de Lima and Valentin Montmirail: Un raccourci récursif pour CEGAR : Application au problème de satisfiabilité en logique modale K. In: Actes des 11es Journées d'Intelligence Artificielle Fondamentale (JIAF 2017), URL https://pfia2017.greyc.fr/share/actes/IAF/Lagniez_IAF_2017.pdf, July 2017, S. 169–176
- Caridroit et al. (2017b) Thomas Caridroit, Jean-Marie Lagniez, Daniel Le Berre, Tiago de Lima and Valentin Montmirail: Une approche basée sur SAT pour le problème de satisfiabilité en logique modale S5. In: Actes des 13es Journées Francophones de Program- mation par Contraintes (JFPC 2017), URL http://www.cril.univ-artois.fr/jfpc2017/articles/JFPC_2017_paper_11.pdf, June 2017, S. 45–53
- Lagniez et al. (2016a) Jean-Marie Lagniez, Daniel Le Berre, Tiago de Lima and Valentin Montmirail: A propos de la vérification de modèles en logique modale K. In: Actes des 10es Journées d'Intelligence Artificielle Fondamentale (JIAF 2016), URL https://www.supagro.fr/jfpc_jiaf_2016/Articles.IAF.2016/Lagniez_IAF_2016.pdf, June 2016, S. 149–157

National Journal Papers with Proceedings

- Defourneau et al. (2017) Thibault Defourneau, Florent Dewez and Valentin Montmirail: Le Jeu du Lights Out : une approche visuelle des mathématiques au travers d'un atelier. In: MathemaTICE: Volume 54 (2017), March. – URL http://revue.sesamath.net/spip.php?article950. – (Online article)

Unpublished International Work

- Dewez and Montmirail (2017) Florent Dewez and Valentin Montmirail: The Hill Cipher: A Weakness Studied Through Group Action Theory. November 2017. – URL https://hal.archives-ouvertes.fr/hal-01631232. – (Unpublished yet)

Alloy Script to visualize the shape of the K* Structure

```
module modal

sig World {
    accessibles : set World
}

one sig W0 extends World {}

pred five {
    all u,v,w:World | u->v in accessibles and u->w in accessibles implies v->w in accessibles
}

pred t {
    all u:World | u->u in accessibles
}

pred d {
    all u : World | some u.accessibles
}

pred b {
    all u,v:World | u->v in accessibles implies v->u in accessibles
}

pred four {
    all u,v,w:World | u->v in accessibles and v->w in accessibles implies u->w in accessibles
}

fact w0 {
    all w:World | w = W0 or w in W0.^accessibles
}

pred show[] {
    some World
    five
    four
    d
}

run show for exactly 4 World
```

One just need to change the predicate "show" and the line "run show for exactly n World" to see the shape of the structure with n worlds.

Bibliography

[Abate et al. 2007] Abate, Pietro ; Goré, Rajeev ; Widmann, Florian: Cut-Free Single-Pass Tableaux For The Logic of Common Knowledge. In: *Workshop on Agents and Deduction (WAD@TABLEAUX'07)*, Springer, July 2007, p. 1–20. – URL http://users.cecs.anu.edu.au/~rpg/Submissions/lck.pdf

[Areces et al. 2015] Areces, Carlos ; Fontaine, Pascal ; Merz, Stephan: *Modal Satisfiability via SMT Solving*. p. 30–45. In: De Nicola, Rocco (Editor) ; Hennicker, Rolf (Editor): *Software, Services, and Systems: Essays Dedicated to Martin Wirsing on the Occasion of His Retirement from the Chair of Programming and Software Engineering*, Springer International Publishing, 2015. – URL https://doi.org/10.1007/978-3-319-15545-6_5. – ISBN 978-3-319-15545-6

[Argelich et al. 2008] Argelich, Josep ; Li, Chu M. ; Manyà, Felip ; Planes, Jordi: The First and Second Max-SAT Evaluations. In: *JSAT* 4 (2008), Nr. 2-4, p. 251–278. – URL https://satassociation.org/jsat/index.php/jsat/article/view/53

[Aristotle 335 BCE] Aristotle: *De Poetica*. Forgotten Books, 335 BCE. – URL https://books.google.fr/books?id=EYuJEx4GKncC. – ISBN 9781605063362

[Arora and Barak 2009] Arora, Sanjeev ; Barak, Boaz: *Computational Complexity - A Modern Approach*. Cambridge University Press, 2009. – URL http://www.cambridge.org/catalogue/catalogue.asp?isbn=9780521424264. – ISBN 978-0-521-42426-4

[Audemard et al. 2008] Audemard, Gilles ; Bordeaux, Lucas ; Hamadi, Youssef ; Jabbour, Saïd ; Sais, Lakhdar: A Generalized Framework for Conflict Analysis. In: Büning, Hans K. (Editor) ; Zhao, Xishun (Editor): *Theory and Applications of Satisfiability Testing - SAT 2008, 11th International Conference, SAT 2008, Guangzhou, China, May 12-15, 2008. Proceedings* Volume 4996, Springer, 2008, p. 21–27. – URL https://doi.org/10.1007/978-3-540-79719-7_3. – ISBN 978-3-540-79718-0

[Audemard et al. 2013] Audemard, Gilles ; Lagniez, Jean-Marie ; Simon, Laurent: Improving Glucose for Incremental SAT Solving with Assumptions: Application to MUS Extraction. In: (Järvisalo and Gelder 2013), p. 309–317. – URL https://doi.org/10.1007/978-3-642-39071-5_23. – ISBN 978-3-642-39070-8

[Audemard et al. 2016] Audemard, Gilles ; Lagniez, Jean-Marie ; Szczepanski, Nicolas ; Tabary, Sébastien: An Adaptive Parallel SAT Solver. In: Rueher, Michel (Editor): *Principles and Practice of Constraint Programming - 22nd International Conference, CP 2016, Toulouse, France, September 5-9, 2016, Proceedings* Volume 9892, Springer, 2016, p. 30–48. – URL https://doi.org/10.1007/978-3-319-44953-1_3. – ISBN 978-3-319-44952-4

[Audemard et al. 2017] Audemard, Gilles ; Lagniez, Jean-Marie ; Szczepanski, Nicolas ; Tabary, Sébastien: A Distributed Version of Syrup. In: Gaspers, Serge (Editor) ; Walsh, Toby (Editor): *Theory and Applications of Satisfiability Testing - SAT 2017 - 20th International Conference, Melbourne, VIC, Australia, August 28 - September 1, 2017, Proceedings* Volume 10491, Springer, 2017, p. 215–232. – URL https://doi.org/10.1007/978-3-319-66263-3_14. – ISBN 978-3-319-66262-6

[Audemard and Simon 2009] Audemard, Gilles ; Simon, Laurent: Predicting Learnt Clauses Quality in Modern SAT Solvers. In: Boutilier, Craig (Editor): *IJCAI 2009, Proceedings of the 21st International Joint Conference on Artificial Intelligence, Pasadena, California, USA, July 11-17, 2009*, Interational Joint Conference on Artificial Intelligence, 2009, p. 399–404. – URL http://ijcai.org/Proceedings/09/Papers/074.pdf

[Baader and Hollunder 1991] Baader, Franz ; Hollunder, Bernhard: A Terminological Knowledge Representation System with Complete Inference Algorithms. In: Boley, Harold (Editor) ; Richter, Michael M. (Editor): *Processing Declarative Knowledge, International Workshop PDK'91, Kaiserslautern, Germany, July 1-3, 1991, Proceedings* Volume 567, Springer, 1991, p. 67–86. – URL https://doi.org/10.1007/BFb0013522. – ISBN 3-540-55033-X

[Balsiger and Heuerding 1998] Balsiger, Peter ; Heuerding, Alain: Comparison of Theorem Provers for Modal Logics - Introduction and Summary. In: (de Swart 1998), p. 25–26. – URL https://doi.org/10.1007/3-540-69778-0_4. – ISBN 3-540-64406-7

[Balsiger et al. 1998] Balsiger, Peter ; Heuerding, Alain ; Schwendimann, Stefan: Logics Workbench 1.0. In: (de Swart 1998), p. 35–37. – URL https://doi.org/10.1007/3-540-69778-0_8. – ISBN 3-540-64406-7

[Balsiger et al. 2000] Balsiger, Peter ; Heuerding, Alain ; Schwendimann, Stefan: A Benchmark Method for the Propositional Modal Logics K, KT, S4. In: *J. Autom. Reasoning* 24 (2000), Nr. 3, p. 297–317. – URL https://doi.org/10.1023/A:1006249507577

[Balyo and Sinz 2018] Balyo, Tomás ; Sinz, Carsten: Parallel Satisfiability. In: (Hamadi and Sais 2018), p. 3–29. – URL https://doi.org/10.1007/978-3-319-63516-3_1. – ISBN 978-3-319-63515-6

[Barabási and Albert 1999] Barabási, Albert-László ; Albert, Réka: Emergence of Scaling in Random Networks. In: *science* 286 (1999), Nr. 5439, p. 509–512

[Barrett et al. 2009] Barrett, Clark W. ; Sebastiani, Roberto ; Seshia, Sanjit A. ; Tinelli, Cesare: Satisfiability Modulo Theories. In: (Biere et al. 2009), p. 825–885. – URL https://doi.org/10.3233/978-1-58603-929-5-825. – ISBN 978-1-58603-929-5

[Bayardo Jr. and Schrag 1997] Bayardo Jr., Roberto J. ; Schrag, Robert: Using CSP Look-Back Techniques to Solve Real-World SAT Instances. In: Kuipers, Benjamin (Editor) ; Webber, Bonnie L. (Editor): *Proceedings of the Fourteenth National Conference on Artificial Intelligence and Ninth Innovative Applications of Artificial Intelligence Conference, AAAI 97, IAAI 97, July 27-31, 1997, Providence, Rhode Island.*, AAAI Press / The MIT Press, 1997, p. 203–208. – URL http://www.aaai.org/Library/AAAI/1997/aaai97-032.php. – ISBN 0-262-51095-2

[Belov et al. 2012] Belov, Anton ; Lynce, Inês ; Marques-Silva, João: Towards efficient MUS extraction. In: *AI Commun.* 25 (2012), Nr. 2, p. 97–116. – URL https://doi.org/10.3233/AIC-2012-0523

[Belov and Marques-Silva 2011] Belov, Anton ; Marques-Silva, João: Accelerating MUS extraction with recursive model rotation. In: Bjesse, Per (Editor) ; Slobodová, Anna (Editor): *International Conference on Formal Methods in Computer-Aided Design, FMCAD*

'11, Austin, TX, USA, October 30 - November 02, 2011, FMCAD Inc., 2011, p. 37–40. – URL http://dl.acm.org/citation.cfm?id=2157663. – ISBN 978-0-9835678-1-3

[Benzmüller 2010] Benzmüller, Christoph: Verifying the Modal Logic Cube Is an Easy Task (For Higher-Order Automated Reasoners). In: Siegler, Simon (Editor) ; Wasser, Nathan (Editor): *Verification, Induction, Termination Analysis - Festschrift for Christoph Walther on the Occasion of His 60th Birthday* Volume 6463, Springer, 2010, p. 117–128. – URL https://doi.org/10.1007/978-3-642-17172-7_7. – ISBN 978-3-642-17171-0

[Bernays 1926] Bernays, Paul: *Axiomatische Untersuchung des Aussagen-Kalkuls der "Principia mathematica"*. Springer, 1926. – URL https://books.google.fr/books?id=-hBntAEACAAJ

[Bezhanishvili and Marx 2003] Bezhanishvili, Nick ; Marx, Maarten: All Proper Normal Extensions of S5-square have the Polynomial Size Model Property. In: *Studia Logica* 73 (2003), Nr. 3, p. 367–382. – URL https://doi.org/10.1023/A:1023383112908

[Biere 2009] Biere, Armin: Bounded Model Checking. In: (Biere et al. 2009), p. 457–481. – URL https://doi.org/10.3233/978-1-58603-929-5-457. – ISBN 978-1-58603-929-5

[Biere et al. 2009] Biere, Armin (Editor) ; Heule, Marijn (Editor) ; Maaren, Hans van (Editor) ; Walsh, Toby (Editor): *Frontiers in Artificial Intelligence and Applications*. Volume 185: *Handbook of Satisfiability*. IOS Press, 2009. – URL https://www.iospress.nl/book/handbook-of-satisfiability/. – ISBN 978-1-58603-929-5

[Blackburn et al. 2006] Blackburn, Patrick ; Benthem, Johan van ; Wolter, Frank: *Handbook of Modal Logic*. Volume 3. Elsevier, 2006. – URL https://hal.inria.fr/inria-00120237. – ISBN 978-0444516909

[Bonacina 2018] Bonacina, Maria P.: Parallel Theorem Proving. In: (Hamadi and Sais 2018), p. 179–235. – URL https://doi.org/10.1007/978-3-319-63516-3_6. – ISBN 978-3-319-63515-6

[Boole 1854] Boole, George: *An Investigation of the Laws of Thought: On which are Founded the Mathematical Theories of Logic and Probabilities*. Walton and Maberly, 1854 (George Boole's collected logical works). – URL https://books.google.fr/books?id=SWgLVT0otY8C

[Brand et al. 2003] Brand, Sebastian ; Gennari, Rosella ; Rijke, Maarten de: Constraint Methods for Modal Satisfiability. In: Apt, Krzysztof R. (Editor) ; Fages, François (Editor) ; Rossi, Francesca (Editor) ; Szeredi, Péter (Editor) ; Váncza, József (Editor): *Recent Advances in Constraints, Joint ERCIM/CoLogNET International Workshop on Constraint Solving and Constraint Logic Programming, CSCLP 2003, Budapest, Hungary, June 30 - July 2, 2003, Selected Papers* Volume 3010, Springer, 2003, p. 66–86. – URL https://doi.org/10.1007/978-3-540-24662-6_4. – ISBN 3-540-21834-3

[Braunstein et al. 2005] Braunstein, Alfredo ; Mézard, Marc ; Zecchina, Riccardo: Survey propagation: An algorithm for satisfiability. In: *Random Struct. Algorithms* 27 (2005), Nr. 2, p. 201–226. – URL https://doi.org/10.1002/rsa.20057

[Brummayer and Biere 2009] Brummayer, Robert ; Biere, Armin: Effective Bit-Width and Under-Approximation. In: Moreno-Díaz, Roberto (Editor) ; Pichler, Franz (Editor) ; Quesada-Arencibia, Alexis (Editor): *Computer Aided Systems Theory - EUROCAST 2009,*

12th International Conference, Las Palmas de Gran Canaria, Spain, February 15-20, 2009, Revised Selected Papers Volume 5717, Springer, 2009, p. 304–311. – URL https://doi.org/10.1007/978-3-642-04772-5_40. – ISBN 978-3-642-04771-8

[Bryant 1986] Bryant, Randal E.: Graph-Based Algorithms for Boolean Function Manipulation. In: *IEEE Trans. Computers* 35 (1986), Nr. 8, p. 677–691

[Buro and Büning 1993] Buro, M. ; Büning, H.K.: Report on a SAT competition. In: *Bulletin of the European Association for Theoretical Computer Science* 49 (1993), p. 143–151. – URL http://stamm-wilbrandt.de/en/Report_on_a_SAT_competition.pdf

[Cadoli et al. 2002] Cadoli, Marco ; Schaerf, Marco ; Giovanardi, Andrea ; Giovanardi, Massimo: An Algorithm to Evaluate Quantified Boolean Formulae and Its Experimental Evaluation. In: *J. Autom. Reasoning* 28 (2002), Nr. 2, p. 101–142. – URL https://doi.org/10.1023/A:1015019416843

[Caridroit et al. 2017a] Caridroit, Thomas ; Lagniez, Jean-Marie ; Le Berre, Daniel ; de Lima, Tiago ; Montmirail, Valentin: A SAT-based approach for solving the modal logic S5 satisfiability problem. In: *Proceedings of the 31st AAAI Conference on Artificial Intelligence (AAAI 2017)*, AAAI, Feb. 2017, p. 3864–3870. – URL http://aaai.org/ocs/index.php/AAAI/AAAI17/paper/view/14380/14111

[Caridroit et al. 2017b] Caridroit, Thomas ; Lagniez, Jean-Marie ; Le Berre, Daniel ; de Lima, Tiago ; Montmirail, Valentin: Une approche basée sur SAT pour le problème de satisfiabilité en logique modale S5. In: *Actes des 13es journées Francophones de Programmation par Contraintes (JFPC 2017)*, ., June 2017, p. 45–53. – URL http://www.cril.univ-artois.fr/jfpc2017/articles/JFPC_2017_paper_11.pdf

[Chellas 1980] Chellas, Brian F.: *Modal Logic: An Introduction*. Cambridge University Press, 1980. – URL https://books.google.fr/books?id=v4YIAQAAIAAJ. – ISBN 978-0521295154

[Clarke et al. 2003] Clarke, Edmund M. ; Grumberg, Orna ; Jha, Somesh ; Lu, Yuan ; Veith, Helmut: Counterexample-Guided Abstraction Refinement For Symbolic Model Checking. In: *J. ACM* 50 (2003), Nr. 5, p. 752–794. – URL http://doi.acm.org/10.1145/876638.876643

[Clarke et al. 2018] Clarke, Edmund M. ; Henzinger, Thomas A. ; Veith, Helmut ; Bloem, Roderick: *Handbook of Model Checking*. Volume 1. Springer International Publishing, 2018. – URL https://link.springer.com/book/10.1007/978-3-319-10575-8. – ISBN 978-3-319-10575-8

[Clarke and Schlingloff 2001] Clarke, Edmund M. ; Schlingloff, Bernd-Holger: Model Checking. In: (Robinson and Voronkov 2001), p. 1635–1790. – URL https://www.sciencedirect.com/science/book/9780444508133. – ISBN 0-444-50813-9

[Cook 1971] Cook, Stephen A.: The Complexity of Theorem-Proving Procedures. In: Harrison, Michael A. (Editor) ; Banerji, Ranan B. (Editor) ; Ullman, Jeffrey D. (Editor): *Proc. of ACM'71*, ACM, 1971, p. 151–158. – URL http://doi.acm.org/10.1145/800157.805047

[Costa and Benevides 2005] Costa, Vania ; Benevides, Mario R. F.: Formalizing Concurrent Common Knowledge as Product of Modal Logics. In: *Logic Journal of the IGPL* 13 (2005), Nr. 6, p. 665–684. – URL https://doi.org/10.1093/jigpal/jzi049

[Cousot 1981] Cousot, Patrick: Semantic Foundations of Program Analysis. In: Muchnick, S.S. (Editor) ; Jones, N.D. (Editor): *Program Flow Analysis: Theory and Applications*. Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1981, Chap. 10, p. 303–342

[Cousot 2000] Cousot, Patrick: Partial Completeness of Abstract Fixpoint Checking. In: Choueiry, Berthe Y. (Editor) ; Walsh, Toby (Editor): *Abstraction, Reformulation, and Approximation, 4th International Symposium, SARA 2000, Horseshoe Bay, Texas, USA, July 26-29, 2000, Proceedings* Volume 1864, Springer, 2000, p. 1–25. – URL https://doi.org/10.1007/3-540-44914-0_1. – ISBN 3-540-67839-5

[Cousot and Cousot 1977] Cousot, Patrick ; Cousot, Radhia: Abstract Interpretation: A Unified Lattice Model for Static Analysis of Programs by Construction or Approximation of Fixpoints. In: Graham, Robert M. (Editor) ; Harrison, Michael A. (Editor) ; Sethi, Ravi (Editor): *Conference Record of the Fourth ACM Symposium on Principles of Programming Languages, Los Angeles, California, USA, January 1977*, ACM, 1977, p. 238–252. – URL http://doi.acm.org/10.1145/512950.512973

[Cousot and Cousot 1979] Cousot, Patrick ; Cousot, Radhia: Systematic Design of Program Analysis Frameworks. In: Aho, Alfred V. (Editor) ; Zilles, Stephen N. (Editor) ; Rosen, Barry K. (Editor): *Conference Record of the Sixth Annual ACM Symposium on Principles of Programming Languages, San Antonio, Texas, USA, January 1979*, ACM Press, 1979, p. 269–282. – URL http://doi.acm.org/10.1145/567752.567778

[Cousot and Cousot 1992] Cousot, Patrick ; Cousot, Radhia: Inductive Definitions, Semantics and Abstract Interpretation. In: Sethi, Ravi (Editor): *Conference Record of the Nineteenth Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, Albuquerque, New Mexico, USA, January 19-22, 1992*, ACM Press, 1992, p. 83–94. – URL http://doi.acm.org/10.1145/143165.143184. – ISBN 0-89791-453-8

[Dams and Grumberg 2018] Dams, Dennis ; Grumberg, Orna: *Abstraction and Abstraction Refinement*. Volume 1. p. 385–419. see (Clarke et al. 2018). – URL https://doi.org/10.1007/978-3-319-10575-8_13. – ISBN 978-3-319-10575-8

[Dantzig 1930] Dantzig, Tobias: *Number: The language of Science*. Nature, 1930. – URL https://www.nature.com/articles/147009a0

[Davies and Bacchus 2013] Davies, Jessica ; Bacchus, Fahiem: Exploiting the Power of MIP Solvers in MaxSAT. In: (Järvisalo and Gelder 2013), p. 166–181. – URL https://doi.org/10.1007/978-3-642-39071-5_13. – ISBN 978-3-642-39070-8

[Davis et al. 1962] Davis, Martin ; Logemann, George ; Loveland, Donald W.: A Machine Program For Theorem-Proving. In: *Commun. ACM* 5 (1962), Nr. 7, p. 394–397. – URL http://doi.acm.org/10.1145/368273.368557

[Davis and Putnam 1960] Davis, Martin ; Putnam, Hilary: A Computing Procedure for Quantification Theory. In: *J. ACM* 7 (1960), Nr. 3, p. 201–215. – URL http://doi.acm.org/10.1145/321033.321034

[de Moura and Bjørner 2008] de Moura, Leonardo Mendonça ; Bjørner, Nikolaj: Z3: An Efficient SMT Solver. In: Ramakrishnan, C. R. (Editor) ; Rehof, Jakob (Editor): *Tools and Algorithms for the Construction and Analysis of Systems, 14th International Conference,*

TACAS 2008, Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2008, Budapest, Hungary, March 29-April 6, 2008. Proceedings Volume 4963, Springer, 2008, p. 337–340. – URL https://doi.org/10.1007/978-3-540-78800-3_24. – ISBN 978-3-540-78799-0

[Dechter et al. 1991] Dechter, Rina ; Meiri, Itay ; Pearl, Judea: Temporal Constraint Networks. In: *Artificial Intelligence* 49 (1991), Nr. 1-3, p. 61–95. – URL https://doi.org/10.1016/0004-3702(91)90006-6

[Defourneau et al. 2017] Defourneau, Thibault ; Dewez, Florent ; Montmirail, Valentin: Le Jeu du Lights Out : une approche visuelle des mathématiques au travers d'un atelier. In: *MathemaTICE: Volume 54* (2017), March. – URL http://revue.sesamath.net/spip.php?article950. – (Online article)

[Dewez and Montmirail 2017] Dewez, Florent ; Montmirail, Valentin: *The Hill Cipher: A Weakness Studied Through Group Action Theory*. November 2017. – URL https://hal.archives-ouvertes.fr/hal-01631232. – (Unpublished yet)

[Eén and Sörensson 2003] Eén, Niklas ; Sörensson, Niklas: An Extensible SAT-solver. In: (Giunchiglia and Tacchella 2003), p. 502–518. – URL https://doi.org/10.1007/978-3-540-24605-3_37. – ISBN 3-540-20851-8

[Eiter et al. 2000] Eiter, Thomas ; Faber, Wolfgang ; Leone, Nicola ; Pfeifer, Gerald ; Polleres, Axel: Planning under Incomplete Knowledge. In: Lloyd, John W. (Editor) ; Dahl, Verónica (Editor) ; Furbach, Ulrich (Editor) ; Kerber, Manfred (Editor) ; Lau, Kung-Kiu (Editor) ; Palamidessi, Catuscia (Editor) ; Pereira, Luís Moniz (Editor) ; Sagiv, Yehoshua (Editor) ; Stuckey, Peter J. (Editor): *Computational Logic - CL 2000, First International Conference, London, UK, 24-28 July, 2000, Proceedings* Volume 1861, Springer, 2000, p. 807–821. – URL https://doi.org/10.1007/3-540-44957-4_54. – ISBN 3-540-67797-6

[Fairtlough and Mendler 1994] Fairtlough, Matt ; Mendler, Michael: An Intuitionistic Modal Logic with Applications to the Formal Verification of Hardware. In: Pacholski, Leszek (Editor) ; Tiuryn, Jerzy (Editor): *Computer Science Logic, 8th International Workshop, CSL '94, Kazimierz, Poland, September 25-30, 1994, Selected Papers* Volume 933, Springer, 1994, p. 354–368. – URL https://doi.org/10.1007/BFb0022268. – ISBN 3-540-60017-5

[Fitting 1983] Fitting, Melvin: *Proof Methods for Modal and Intuitionistic Logics*. Springer, 1983. – URL https://www.springer.com/la/book/9789027715739. – ISBN 978-90-277-1573-9

[Garson 2016] Garson, James: Modal Logic. In: Zalta, Edward N. (Editor): *The Stanford Encyclopedia of Philosophy*. Spring 2016. Metaphysics Research Lab, Stanford University, 2016, p. .. – URL https://plato.stanford.edu/archives/spr2016/entries/logic-modal/

[Gasquet et al. 2005] Gasquet, Olivier ; Herzig, Andreas ; Longin, Dominique ; Sahade, Mohamad: LoTREC: Logical Tableaux Research Engineering Companion. In: Beckert, Bernhard (Editor): *Automated Reasoning with Analytic Tableaux and Related Methods, International Conference, TABLEAUX 2005, Koblenz, Germany, September 14-17, 2005, Proceedings* Volume 3702, Springer, 2005, p. 318–322. – URL https://doi.org/10.1007/11554554_25. – ISBN 3-540-28931-3

[Gebser et al. 2007] Gebser, Martin ; Liu, Lengning ; Namasivayam, Gayathri ; Neumann, André ; Schaub, Torsten ; Truszczynski, Miroslaw: The First Answer Set Programming System Competition. In: Baral, Chitta (Editor) ; Brewka, Gerhard (Editor) ; Schlipf, John S. (Editor): *Logic Programming and Nonmonotonic Reasoning, 9th International Conference, LPNMR 2007, Tempe, AZ, USA, May 15-17, 2007, Proceedings* Volume 4483, Springer, 2007, p. 3–17. – URL https://doi.org/10.1007/978-3-540-72200-7_3. – ISBN 978-3-540-72199-4

[Gelder 2002] Gelder, Allen V.: Extracting (Easily) Checkable Proofs from a Satisfiability Solver that Employs both Preorder and Postorder Resolution. In: *International Symposium on Artificial Intelligence and Mathematics, AI&M 2002, Fort Lauderdale, Florida, USA, January 2-4, 2002*, CiteSeerx, 2002, p. 1–10. – URL http://rutcor.rutgers.edu/~amai/aimath02/PAPERS/33.ps

[Giunchiglia et al. 2000] Giunchiglia, Enrico ; Giunchiglia, Fausto ; Sebastiani, Roberto ; Tacchella, Armando: SAT vs. translation based decision procedures for modal logics: a comparative evaluation. In: *Journal of Applied Non-Classical Logics* 10 (2000), Nr. 2, p. 145–172. – URL https://doi.org/10.1080/11663081.2000.10510994

[Giunchiglia and Tacchella 2003] Giunchiglia, Enrico (Editor) ; Tacchella, Armando (Editor): *Theory and Applications of Satisfiability Testing, 6th International Conference, SAT 2003. Santa Margherita Ligure, Italy, May 5-8, 2003 Selected Revised Papers.* Volume *2919*. Springer, 2003. (Lecture Notes in Computer Science). – URL https://doi.org/10.1007/b95238. – ISBN 3-540-20851-8

[Giunchiglia et al. 2002] Giunchiglia, Enrico ; Tacchella, Armando ; Giunchiglia, Fausto: SAT-Based Decision Procedures for Classical Modal Logics. In: *Journal of Automated Reasoning* 28 (2002), Feb, Nr. 2, p. 143–171. – URL https://doi.org/10.1023/A:1015071400913. – ISSN 1573-0670

[Giunchiglia et al. 1996] Giunchiglia, Fausto ; Roveri, Marco ; Sebastiani, Roberto: A New Method for Testing Decision Procedures in Modal and Terminological Logics. In: Padgham, Lin (Editor) ; Franconi, Enrico (Editor) ; Gehrke, Manfred (Editor) ; McGuinness, Deborah L. (Editor) ; Patel-Schneider, Peter F. (Editor): *Proceedings of the 1996 International Workshop on Description Logics, November 2-4, 1996, Cambridge, MA, USA* Volume WS-96-05, AAAI Press, 1996, p. 119–123. – URL https://pdfs.semanticscholar.org/2477/d66cf583e0942f1854e351721f2cd3f4e7c1.pdf. – ISBN 1-57735-014-6

[Glorian et al. 2018] Glorian, Gael ; Lagniez, Jean-Marie ; Montmirail, Valentin ; Sioutis, Michael: An Incremental SAT-Based Approach to Reason Efficiently On Qualitative Constraint Network. In: *Principles and Practice of Constraint Programming - 24th International Conference, CP 2018, Lille, France, August 27-31, 2018, Proceedings*, Springer, August 2018, p. 160–178. – URL https://doi.org/10.1007/978-3-319-98334-9_11

[Goré 1999] Goré, Rajeev: *Tableau Methods for Modal and Temporal Logics.* p. 297–396. In: D'Agostino, Marcello (Editor) ; Gabbay, Dov M. (Editor) ; Hähnle, Reiner (Editor) ; Posegga, Joachim (Editor): *Handbook of Tableau Methods*, Springer, 1999. – URL https://doi.org/10.1007/978-94-017-1754-0_6. – ISBN 978-94-017-1754-0

[Goré et al. 2014] Goré, Rajeev ; Olesen, Kerry ; Thomson, Jimmy: Implementing Tableau Calculi Using BDDs: BDDTab System Description. In: Demri, Stéphane (Editor) ; Kapur,

Deepak (Editor) ; Weidenbach, Christoph (Editor): *Automated Reasoning - 7th International Joint Conference, IJCAR 2014, Held as Part of the Vienna Summer of Logic, VSL 2014, Vienna, Austria, July 19-22, 2014. Proceedings* Volume 8562, Springer, 2014, p. 337–343. – URL https://doi.org/10.1007/978-3-319-08587-6_25. – ISBN 978-3-319-08586-9

[Götzmann et al. 2010] Götzmann, Daniel ; Kaminski, Mark ; Smolka, Gert: Spartacus: A Tableau Prover for Hybrid Logic. In: *Electr. Notes Theor. Comput. Sci.* 262 (2010), p. 127–139. – URL https://doi.org/10.1016/j.entcs.2010.04.010

[Grégoire et al. 2014] Grégoire, Éric ; Lagniez, Jean-Marie ; Mazure, Bertrand: An Experimentally Efficient Method for (MSS, CoMSS) Partitioning. In: Brodley, Carla E. (Editor) ; Stone, Peter (Editor): *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence, July 27 -31, 2014, Québec City, Québec, Canada.*, AAAI Press, 2014, p. 2666–2673. – URL http://www.aaai.org/ocs/index.php/AAAI/AAAI14/paper/view/8324. – ISBN 978-1-57735-661-5

[Haken 1985] Haken, Armin: The Intractability of Resolution. In: *Theor. Comput. Sci.* 39 (1985), p. 297–308. – URL https://doi.org/10.1016/0304-3975(85)90144-6

[Halpern and Moses 1992] Halpern, Joseph Y. ; Moses, Yoram: A guide to completeness and complexity for modal logics of knowledge and belief. In: *Artificial Intelligence* 54 (1992), Nr. 3, p. 319 – 379. – URL http://www.sciencedirect.com/science/article/pii/0004370292900494. – ISSN 0004-3702

[Halpern and Rêgo 2007] Halpern, Joseph Y. ; Rêgo, Leandro C.: Characterizing the NP-PSPACE Gap in the Satisfiability Problem for Modal Logic. In: *J. Log. Comput.* 17 (2007), Nr. 4, p. 795–806. – URL https://doi.org/10.1093/logcom/exm029

[Hamadi et al. 2010] Hamadi, Youssef ; Jabbour, Saïd ; Sais, Lakhdar: Learning for Dynamic Subsumption. In: *International Journal on Artificial Intelligence Tools* 19 (2010), Nr. 4, p. 511–529. – URL https://doi.org/10.1142/S0218213010000303

[Hamadi and Sais 2018] Hamadi, Youssef (Editor) ; Sais, Lakhdar (Editor): *Handbook of Parallel Constraint Reasoning*. Springer, 2018. – URL https://doi.org/10.1007/978-3-319-63516-3. – ISBN 978-3-319-63515-6

[Han and Somenzi 2009] Han, HyoJung ; Somenzi, Fabio: On-the-Fly Clause Improvement. In: (Kullmann 2009), p. 209–222. – URL https://doi.org/10.1007/978-3-642-02777-2_21. – ISBN 978-3-642-02776-5

[Hansen et al. 2001] Hansen, Pierre ; Mladenovic, Nenad ; Pérez-Brito, Dionisio: Variable Neighborhood Decomposition Search. In: *J. Heuristics* 7 (2001), Nr. 4, p. 335–350. – URL https://doi.org/10.1023/A:1011336210885

[Hao and Dorne 1994] Hao, Jin-Kao ; Dorne, Raphaël: An Empirical Comparison of Two Evolutionary Methods for Satisfiability Problems. In: *Proceedings of the First IEEE Conference on Evolutionary Computation, IEEE World Congress on Computational Intelligence, Orlando, Florida, USA, June 27-29, 1994*, IEEE, 1994, p. 451–455. – URL https://doi.org/10.1109/ICEC.1994.349908. – ISBN 0-7803-1899-4

[Heras et al. 2011] Heras, Federico ; Morgado, António ; Marques-Silva, João: Core-Guided Binary Search Algorithms for Maximum Satisfiability. In: Burgard, Wolfram (Editor) ; Roth, Dan (Editor): *Proceedings of the Twenty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2011, San Francisco, California, USA, August 7-11, 2011*, AAAI Press, 2011, p. 1–7. – URL http://www.aaai.org/ocs/index.php/AAAI/AAAI11/paper/view/3713

[Hoffmann 2010] Hoffmann, Guillaume: *Tâches de raisonnement en logiques hybrides*, Henri Poincaré University, Nancy, Ph.D. thesis, 2010. – URL https://tel.archives-ouvertes.fr/tel-01746342v2/document

[Hoffmann and Brafman 2006] Hoffmann, Jörg ; Brafman, Ronen I.: Conformant Planning via Heuristic Forward Search: A New Approach. In: *A.I.* 170 (2006), Nr. 6-7, p. 507–541. – URL https://doi.org/10.1016/j.artint.2006.01.003

[Hoos and Stützle 1999] Hoos, Holger H. ; Stützle, Thomas: Towards a Characterisation of the Behaviour of Stochastic Local Search Algorithms for SAT. In: *Artif. Intell.* 112 (1999), Nr. 1-2, p. 213–232. – URL https://doi.org/10.1016/S0004-3702(99)00048-X

[Horrocks et al. 2006] Horrocks, Ian ; Hustadt, Ullrich ; Sattler, Ulrike ; Schmidt, Renate: Computational Modal Logic. In: *Handbook of Modal Logic*. (Blackburn et al. 2006). – URL https://hal.inria.fr/inria-00120237. – ISBN 978-0444516909

[Huang et al. 2013] Huang, Jinbo ; Li, Jason J. ; Renz, Jochen: Decomposition and tractability in qualitative spatial and temporal reasoning. In: *Artificial Intelligence* 195 (2013), p. 140–164. – URL https://doi.org/10.1016/j.artint.2012.09.009

[Hustadt et al. 1999] Hustadt, Ullrich ; Schmidt, Renate A. ; Weidenbach, Christoph: MSPASS: Subsumption Testing with SPASS. In: Lambrix, Patrick (Editor) ; Borgida, Alexander (Editor) ; Lenzerini, Maurizio (Editor) ; Möller, Ralf (Editor) ; Patel-Schneider, Peter F. (Editor): *Proceedings of the 1999 International Workshop on Description Logics (DL'99), Linköping, Sweden, July 30 - August 1, 1999* Volume 22, CEUR-WS.org, 1999, p. 1–2. – URL http://ceur-ws.org/Vol-22/schmidt.ps

[Iser et al. 2013] Iser, Markus ; Sinz, Carsten ; Taghdiri, Mana: Minimizing Models for Tseitin-Encoded SAT Instances. In: (Järvisalo and Gelder 2013), p. 224–232. – URL https://doi.org/10.1007/978-3-642-39071-5_17. – ISBN 978-3-642-39070-8

[Jackson 2006] Jackson, Daniel: *Software Abstractions - Logic, Language, and Analysis*. MIT Press, 2006. – URL http://mitpress.mit.edu/catalog/item/default.asp?ttype=2&tid=10928. – ISBN 978-0-262-10114-1

[Janota et al. 2016] Janota, Mikolás ; Klieber, William ; Marques-Silva, Joao ; Clarke, Edmund M.: Solving QBF with counterexample guided refinement. In: *Artif. Intell.* 234 (2016), p. 1–25. – URL https://doi.org/10.1016/j.artint.2016.01.004

[Järvisalo and Gelder 2013] Järvisalo, Matti (Editor) ; Gelder, Allen V. (Editor): *Theory and Applications of Satisfiability Testing - SAT 2013 - 16th International Conference, Helsinki, Finland, July 8-12, 2013. Proceedings.* Volume *7962*. Springer, 2013. (Lecture Notes in Computer Science). – URL https://doi.org/10.1007/978-3-642-39071-5. – ISBN 978-3-642-39070-8

[Järvisalo et al. 2012] Järvisalo, Matti ; Le Berre, Daniel ; Roussel, Olivier ; Simon, Laurent: The International SAT Solver Competitions. In: *AI Magazine* 33 (2012), Nr. 1. – URL http://www.aaai.org/ojs/index.php/aimagazine/article/view/2395

[Kaminski and Tebbi 2013] Kaminski, Mark ; Tebbi, Tobias: InKreSAT: Modal Reasoning via Incremental Reduction to SAT. In: Bonacina, Maria P. (Editor): *Automated Deduction - CADE-24 - 24th International Conference on Automated Deduction, Lake Placid, NY, USA, June 9-14, 2013. Proceedings* Volume 7898, Springer, 2013, p. 436–442. – URL https://doi.org/10.1007/978-3-642-38574-2_31. – ISBN 978-3-642-38573-5

[Kautz and Selman 1996] Kautz, Henry A. ; Selman, Bart: Pushing the Envelope: Planning, Propositional Logic and Stochastic Search. In: Clancey, William J. (Editor) ; Weld, Daniel S. (Editor): *Proceedings of the Thirteenth National Conference on Artificial Intelligence and Eighth Innovative Applications of Artificial Intelligence Conference, AAAI 96, IAAI 96, Portland, Oregon, August 4-8, 1996, Volume 2.*, AAAI Press / The MIT Press, 1996, p. 1194–1201. – URL http://www.aaai.org/Library/AAAI/1996/aaai96-177.php

[Khasidashvili et al. 2015] Khasidashvili, Zurab ; Korovin, Konstantin ; Tsarkov, Dmitry: EPR-based k-induction with Counterexample Guided Abstraction Refinement. In: Gottlob, Georg (Editor) ; Sutcliffe, Geoff (Editor) ; Voronkov, Andrei (Editor): *Global Conference on Artificial Intelligence, GCAI 2015, Tbilisi, Georgia, October 16-19, 2015* Volume 36, EasyChair, 2015, p. 137–150. – URL http://www.easychair.org/publications/paper/245322

[Koshimura et al. 2009] Koshimura, Miyuki ; Nabeshima, Hidetomo ; Fujita, Hiroshi ; Hasegawa, Ryuzo: Minimal Model Generation With Respect To An Atom Set. In: *in International Workshop on First-Order Theorem Proving*, CiteSeer, 2009, p. 49–59. – URL http://ceur-ws.org/Vol-556/paper06.pdf

[Kovács and Voronkov 2013] Kovács, Laura ; Voronkov, Andrei: First-Order Theorem Proving and Vampire. In: Sharygina, Natasha (Editor) ; Veith, Helmut (Editor): *Computer Aided Verification - 25th International Conference, CAV 2013, Saint Petersburg, Russia, July 13-19, 2013. Proceedings* Volume 8044, Springer, 2013, p. 1–35. – URL https://doi.org/10.1007/978-3-642-39799-8_1. – ISBN 978-3-642-39798-1

[Kripke 1959] Kripke, Saul: A Completeness Theorem in Modal Logic. In: *J. Symb. Log.* 24 (1959), Nr. 1, p. 1–14. – URL http://dx.doi.org/10.2307/2964568

[Kroening 2009] Kroening, Daniel: Software Verification. In: (Biere et al. 2009), p. 505–532. – URL https://doi.org/10.3233/978-1-58603-929-5-505. – ISBN 978-1-58603-929-5

[Kullmann 2009] Kullmann, Oliver (Editor): *Theory and Applications of Satisfiability Testing - SAT 2009, 12th International Conference, SAT 2009, Swansea, UK, June 30 - July 3, 2009. Proceedings.* Volume *5584*. Springer, 2009. (Lecture Notes in Computer Science). – URL https://doi.org/10.1007/978-3-642-02777-2. – ISBN 978-3-642-02776-5

[Kupferman 2018] Kupferman, Orna: *Automata Theory and Model Checking.* Volume 1. p. 107–151. see (Clarke et al. 2018). – URL https://doi.org/10.1007/978-3-319-10575-8_4. – ISBN 978-3-319-10575-8

[Ladner 1977] Ladner, Richard E.: The Computational Complexity of Provability in Systems of Modal Propositional Logic. In: *SIAM J. Comput.* 6 (1977), Nr. 3, p. 467–480. – URL https://doi.org/10.1137/0206033

[Lagniez et al. 2016a] Lagniez, Jean-Marie ; Le Berre, Daniel ; de Lima, Tiago ; Montmirail, Valentin: À propos de la vérification de modèles en logique modale K. In: *Actes des 10es Journées d'Intelligence Artificielle Fondamentale (JIAF 2016)*, ., June 2016, p. 149–157. – URL https://www.supagro.fr/jfpc_jiaf_2016/Articles.IAF.2016/Lagniez_IAF_2016.pdf

[Lagniez et al. 2016b] Lagniez, Jean-Marie ; Le Berre, Daniel ; de Lima, Tiago ; Montmirail, Valentin: On Checking Kripke Models for Modal Logic K. In: *5th Workshop on Practical Aspects of Automated Reasoning (PAAR@IJCAR 2016)*, Springer, June 2016, p. 69–81. – URL http://ceur-ws.org/Vol-1635/#paper-07

[Lagniez et al. 2017a] Lagniez, Jean-Marie ; Le Berre, Daniel ; de Lima, Tiago ; Montmirail, Valentin: A Recursive Shortcut for CEGAR: Application To The Modal Logic K Satisfiability Problem. In: *Proceedings of the 26th IJCAI International Joint Conference on Artificial Intelligence (IJCAI 2017)*, ijcai.org, August 2017, p. 674–680. – URL https://www.ijcai.org/proceedings/2017/94

[Lagniez et al. 2017b] Lagniez, Jean-Marie ; Le Berre, Daniel ; de Lima, Tiago ; Montmirail, Valentin: Un raccourci récursif pour CEGAR : Application au problème de satisfiabilité en logique modale K. In: *Actes des 11es Journées d'Intelligence Artificielle Fondamentale (JIAF 2017)*, ., July 2017, p. 169–176. – URL https://pfia2017.greyc.fr/share/actes/IAF/Lagniez_IAF_2017.pdf

[Lagniez et al. 2018a] Lagniez, Jean-Marie ; Le Berre, Daniel ; de Lima, Tiago ; Montmirail, Valentin: An Assumption-Based Approach for Solving The Minimal S5-Satisfiability Problem. In: *Proceedings of the 9th International Joint Conference on Automated Reasoning (IJCAR 2018)*, Springer, July 2018, p. 1–18. – URL https://doi.org/10.1007/978-3-319-94205-6_1

[Lagniez et al. 2018b] Lagniez, Jean-Marie ; Le Berre, Daniel ; de Lima, Tiago ; Montmirail, Valentin: Space-Awareness in a SAT-Based Approach For PSPACE Modal Logics. In: *Principles of Knowledge Representation and Reasoning: Proceedings of the Fifteenth International Conference, KR 2018, Tempe, Arizona, USA, 30 October - 2 November, 2018, Proceedings*, Springer, October 2018, p. to appear. – URL http://reasoning.eas.asu.edu/kr2018/

[Lagniez et al. 2018c] Lagniez, Jean-Marie ; Le Berre, Daniel ; de Lima, Tiago ; Montmirail, Valentin: Une approche SAT incrémentale pour le problème de satisfiabilité minimale en logique modale S5. In: *Actes des 14es journées Francophones de Programmation par Contraintes (JFPC 2018)*, ., June 2018, p. 1–10. – URL https://home.mis.u-picardie.fr/~evenement/JFPC2018/articles/JFPC_2018_papier_1.pdf

[Le Berre and Parrain 2010] Le Berre, Daniel ; Parrain, Anne: The SAT4J Library, release 2.2. In: *JSAT* 7 (2010), Nr. 2-3, p. 59–6. – URL http://jsat.ewi.tudelft.nl/content/volume7/JSAT7_4_LeBerre.pdf

[Le Berre and Rapicault 2018] Le Berre, Daniel ; Rapicault, Pascal: Boolean-Based Dependency Management for the Eclipse Ecosystem. In: *International Journal on Artificial Intelligence Tools* 27 (2018), Nr. 1, p. 1–23. – URL https://doi.org/10.1142/S0218213018400031

[Le Berre et al. 2003] Le Berre, Daniel ; Simon, Laurent ; Tacchella, Armando: Challenges in the QBF Arena: the SAT'03 Evaluation of QBF Solvers. In: (Giunchiglia and Tacchella 2003), p. 468–485. – URL https://doi.org/10.1007/978-3-540-24605-3_35. – ISBN 3-540-20851-8

[Lewis and Langford 1932] Lewis, Clarence Irving ; Langford, Cooper Harold: *Symbolic logic*. New York, The Century Co, 1932 (Century philosophy series). – URL https://archive.org/details/symboliclogic00carr

[Li and Anbulagan 1997] Li, Chu M. ; Anbulagan: Heuristics Based on Unit Propagation for Satisfiability Problems. In: *Proceedings of the Fifteenth International Joint Conference on Artificial Intelligence, IJCAI 97, Nagoya, Japan, August 23-29, 1997, 2 Volumes*, Morgan Kaufmann, 1997, p. 366–371. – URL http://ijcai.org/Proceedings/97-1/Papers/057.pdf

[Li and Manyà 2009] Li, Chu M. ; Manyà, Felip: MaxSAT, Hard and Soft Constraints. In: (Biere et al. 2009), p. 613–631. – URL http://dx.doi.org/10.3233/978-1-58603-929-5-613. – ISBN 978-1-58603-929-5

[Li 2006] Li, Sanjiang: On Topological Consistency and Realization. In: *Constraints* 11 (2006), Jan, Nr. 1, p. 31–51. – URL https://doi.org/10.1007/s10601-006-6847-9. – ISSN 1572-9354

[Li and Ying 2003] Li, Sanjiang ; Ying, Mingsheng: Region Connection Calculus: Its models and composition table. In: *Artificial Intelligence* 145 (2003), Nr. 1-2, p. 121–146. – URL https://doi.org/10.1016/S0004-3702(02)00372-7

[Liang et al. 2015] Liang, Jia H. ; Ganesh, Vijay ; Zulkoski, Ed ; Zaman, Atulan ; Czarnecki, Krzysztof: Understanding VSIDS Branching Heuristics in Conflict-Driven Clause-Learning SAT Solvers. In: Piterman, Nir (Editor): *Hardware and Software: Verification and Testing - 11th International Haifa Verification Conference, HVC 2015, Haifa, Israel, November 17-19, 2015, Proceedings* Volume 9434, Springer, 2015, p. 225–241. – URL https://doi.org/10.1007/978-3-319-26287-1_14. – ISBN 978-3-319-26286-4

[Liberatore 2000] Liberatore, Paolo: On the complexity of choosing the branching literal in DPLL. In: *Artif. Intell.* 116 (2000), Nr. 1-2, p. 315–326. – URL https://doi.org/10.1016/S0004-3702(99)00097-1

[Long 2017] Long, Zhiguo: *Qualitative Spatial And Temporal Representation And Reasoning: Efficiency in Time And Space*, Faculty of Engineering and Information Technology, University of Technology Sydney (UTS), Ph.D. thesis, January 2017. – URL http://hdl.handle.net/10453/90055

[Long et al. 2016] Long, Zhiguo ; Sioutis, Michael ; Li, Sanjiang: Efficient Path Consistency Algorithm for Large Qualitative Constraint Networks. In: Kambhampati, Subbarao (Editor): *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence, IJCAI 2016, New York, NY, USA, 9-15 July 2016*, IJCAI/AAAI Press, 2016, p. 1202–1208. – URL http://www.ijcai.org/Abstract/16/174. – ISBN 978-1-57735-770-4

[Mackworth 1977] Mackworth, Alan K.: Consistency in networks of relations. In: *Artificial Intelligence* 8 (1977), Nr. 1, p. 99 – 118. – URL http://www.sciencedirect.com/science/article/pii/0004370277900078. – ISSN 0004-3702

[Maher et al. 2017] Maher, Stephen J. ; Fischer, Tobias ; Gally, Tristan ; Gamrath, Gerald ; Gleixner, Ambros ; Gottwald, Robert L. ; Hendel, Gregor ; Koch, Thorsten ; Lübbecke, Marco E. ; Miltenberger, Matthias ; Müller, Benjamin ; Pfetsch, Marc E. ; Puchert, Christian ; Rehfeldt, Daniel ; Schenker, Sebastian ; Schwarz, Robert ; Serrano, Felipe ; Shinano, Yuji ; Weninger, Dieter ; Witt, Jonas T. ; Witzig, Jakob: The

SCIP Optimization Suite 4.0 / ZIB. Takustr. 7, 14195 Berlin : ZIB, 2017 (17-12). – Research Report. – URL http://nbn-resolving.de/urn:nbn:de:0297-zib-62170. Document Type: ZIB-Report

[Manquinho and Roussel 2006] Manquinho, Vasco M. ; Roussel, Olivier: The First Evaluation of Pseudo-Boolean Solvers (PB'05). In: *JSAT* 2 (2006), Nr. 1-4, p. 103–143. – URL http://sat.inesc-id.pt/~vmm/research/papers/jsat06_2.pdf

[Marques-Silva and Janota 2014] Marques-Silva, João ; Janota, Mikolás: On the Query Complexity of Selecting Few Minimal Sets. In: *Electronic Colloquium on Computational Complexity (ECCC)* 21 (2014), p. 31. – URL http://eccc.hpi-web.de/report/2014/031

[Marques-Silva and Lynce 2011] Marques-Silva, João ; Lynce, Inês: On Improving MUS Extraction Algorithms. In: (Sakallah and Simon 2011), p. 159–173. – URL https://doi.org/10.1007/978-3-642-21581-0_14. – ISBN 978-3-642-21580-3

[Marques-Silva and Sakallah 1999] Marques-Silva, Joao ; Sakallah, Karem A.: GRASP: A Search Algorithm for Propositional Satisfiability. In: *IEEE Trans. Computers* 48 (1999), Nr. 5, p. 506–521. – URL https://doi.org/10.1109/12.769433

[Massacci 1999] Massacci, Fabio: Design and Results of the Tableaux-99 Non-classical (Modal) Systems Comparison. In: Murray, Neil V. (Editor): *Automated Reasoning with Analytic Tableaux and Related Methods, International Conference, TABLEAUX '99, Saratoga Springs, NY, USA, June 7-11, 1999, Proceedings* Volume 1617, Springer, 1999, p. 14–18. – URL https://doi.org/10.1007/3-540-48754-9_2. – ISBN 3-540-66086-0

[Massacci 2000] Massacci, Fabio: Single Step Tableaux for Modal Logics. In: *J. Autom. Reasoning* 24 (2000), Nr. 3, p. 319–364. – URL https://doi.org/10.1023/A:1006155811656

[Massacci and Donini 2000] Massacci, Fabio ; Donini, Francesco M.: Design and Results of TANCS-2000 Non-classical (Modal) Systems Comparison. In: Dyckhoff, Roy (Editor): *Proc. of TABLEAUX'00* Volume 1847, Springer, 2000, p. 52–56. – URL https://doi.org/10.1007/10722086_4. – ISBN 3-540-67697-X

[Mencía et al. 2015] Mencía, Carlos ; Previti, Alessandro ; Marques-Silva, João: Literal-Based MCS Extraction. In: Yang, Qiang (Editor) ; Wooldridge, Michael (Editor): *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, IJCAI 2015, Buenos Aires, Argentina, July 25-31, 2015*, AAAI Press, 2015, p. 1973–1979. – URL http://ijcai.org/Abstract/15/280. – ISBN 978-1-57735-738-4

[Merchez et al. 2001] Merchez, Sylvain ; Lecoutre, Christophe ; Boussemart, Frédéric: AbsCon: A Prototype to Solve CSPs with Abstraction. In: Walsh, Toby (Editor): *Principles and Practice of Constraint Programming - CP 2001, 7th International Conference, CP 2001, Paphos, Cyprus, November 26 - December 1, 2001, Proceedings* Volume 2239, Springer, 2001, p. 730–744. – URL https://doi.org/10.1007/3-540-45578-7_59. – ISBN 3-540-42863-1

[Mitchell et al. 1992] Mitchell, David G. ; Selman, Bart ; Levesque, Hector J.: Hard and Easy Distributions of SAT Problems. In: (Swartout 1992), p. 459–465. – URL http://www.aaai.org/Library/AAAI/1992/aaai92-071.php. – ISBN 0-262-51063-4

[Mordan and Mutilin 2016] Mordan, Vitaly O. ; Mutilin, Vadim S.: Checking several requirements at once by CEGAR. In: *Programming and Computer Software* 42 (2016), Nr. 4, p. 225–238. – URL https://doi.org/10.1134/S0361768816040058

[Moskewicz et al. 2001] Moskewicz, Matthew W. ; Madigan, Conor F. ; Zhao, Ying ; Zhang, Lintao ; Malik, Sharad: Chaff: Engineering an Efficient SAT Solver. In: *Proceedings of the 38th Design Automation Conference, DAC 2001, Las Vegas, NV, USA, June 18-22, 2001*, ACM, 2001, p. 530–535. – URL http://doi.acm.org/10.1145/378239.379017. – ISBN 1-58113-297-2

[Nalon et al. 2015] Nalon, Cláudia ; Hustadt, Ullrich ; Dixon, Clare: A Modal-Layered Resolution Calculus for K. In: Nivelle, Hans de (Editor): *Automated Reasoning with Analytic Tableaux and Related Methods - 24th International Conference, TABLEAUX 2015, Wrocław, Poland, September 21-24, 2015. Proceedings* Volume 9323, Springer, 2015, p. 185–200. – URL https://doi.org/10.1007/978-3-319-24312-2_13. – ISBN 978-3-319-24311-5

[Nalon et al. 2016] Nalon, Cláudia ; Hustadt, Ullrich ; Dixon, Clare: K_SP : A Resolution-Based Prover for Multimodal K. In: Olivetti, Nicola (Editor) ; Tiwari, Ashish (Editor): *Automated Reasoning - 8th International Joint Conference, IJCAR 2016, Coimbra, Portugal, June 27 - July 2, 2016, Proceedings* Volume 9706, Springer, 2016, p. 406–415. – URL https://doi.org/10.1007/978-3-319-40229-1_28. – ISBN 978-3-319-40228-4

[Nguyen 1999] Nguyen, Linh A.: A New Space Bound for the Modal Logics K4, KD4 and S4. In: Kutylowski, Miroslaw (Editor) ; Pacholski, Leszek (Editor) ; Wierzbicki, Tomasz (Editor): *Mathematical Foundations of Computer Science 1999, 24th International Symposium, MFCS'99, Szklarska Poreba, Poland, September 6-10, 1999, Proceedings* Volume 1672, Springer, 1999, p. 321–331. – URL https://doi.org/10.1007/3-540-48340-3_29. – ISBN 3-540-66408-4

[Ore 1944] Ore, Øystein: Galois Connexions. In: *Transactions of the American Mathematical Society* 55 (1944), p. 493–513. – URL https://doi.org/10.1090/S0002-9947-1944-0010555-7

[O'Sullivan et al. 2007] O'Sullivan, Barry ; Papadopoulos, Alexandre ; Faltings, Boi ; Pu, Pearl: Representative Explanations for Over-Constrained Problems. In: *Proceedings of the Twenty-Second AAAI Conference on Artificial Intelligence, July 22-26, 2007, Vancouver, British Columbia, Canada*, AAAI Press, 2007, p. 323–328. – URL http://www.aaai.org/Library/AAAI/2007/aaai07-050.php. – ISBN 978-1-57735-323-2

[Pan and Vardi 2004] Pan, Guoqiang ; Vardi, Moshe Y.: Symbolic Decision Procedures for QBF. In: (Wallace 2004), p. 453–467. – URL https://doi.org/10.1007/978-3-540-30201-8_34. – ISBN 3-540-23241-9

[Papadimitriou 1994] Papadimitriou, Christos H.: *Computational complexity*. Addison-Wesley, 1994. – URL https://dl.acm.org/citation.cfm?id=1074100.1074233. – ISBN 978-0-201-53082-7

[Patel-Schneider and Sebastiani 2003] Patel-Schneider, Peter F. ; Sebastiani, Roberto: A New General Method to Generate Random Modal Formulae for Testing Decision Procedures. In: *J. Artif. Intell. Res.* 18 (2003), p. 351–389. – URL https://arxiv.org/pdf/1106.5261.pdf

[Pelleau 2015] Pelleau, Marie: *Abstract Domains in Constraint Programming*. 1st. Amsterdam, The Netherlands, The Netherlands : Elsevier Science Publishers B. V., 2015. – URL https://www.sciencedirect.com/science/book/9781785480102. – ISBN 1785480103, 9781785480102

[Petrick and Bacchus 2002] Petrick, Ronald P. A. ; Bacchus, Fahiem: A Knowledge-Based Approach to Planning with Incomplete Information and Sensing. In: Ghallab, Malik (Editor) ; Hertzberg, Joachim (Editor) ; Traverso, Paolo (Editor): *Proceedings of the Sixth International Conference on Artificial Intelligence Planning Systems, April 23-27, 2002, Toulouse, France*, AAAI, 2002, p. 212–222. – URL http://www.aaai.org/Library/AIPS/2002/aips02-022.php. – ISBN 1-57735-142-8

[Plaisted and Greenbaum 1986] Plaisted, David A. ; Greenbaum, Steven: A Structure-Preserving Clause Form Translation. In: *J. Symb. Comput.* 2 (1986), Nr. 3, p. 293–304. – URL https://doi.org/10.1016/S0747-7171(86)80028-1

[Randell et al. 1992] Randell, David A. ; Cui, Zhan ; Cohn, Anthony G.: A Spatial Logic based on Regions and Connection. In: Nebel, Bernhard (Editor) ; Rich, Charles (Editor) ; Swartout, William R. (Editor): *Proceedings of the 3rd International Conference on Principles of Knowledge Representation and Reasoning (KR'92). Cambridge, MA, October 25-29, 1992.*, Morgan Kaufmann, 1992, p. 165–176. – URL http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.39.486&rep=rep1&type=pdf. – ISBN 1-55860-262-3

[dos Reis Morgado et al. 2014] Reis Morgado, António José dos ; Ignatiev, Alexey S. ; Marques Silva, João: MSCG: Robust Core-Guided MaxSAT Solving. In: *Journal on Satisfiability Boolean Modeling and Computation* 9 (2014), p. 129–134. – URL https://satassociation.org/jsat/index.php/jsat/article/view/127

[Renz 2002] Renz, Jochen: A Canonical Model of the Region Connection Calculus. In: *Journal of Applied Non-Classical Logics* 12 (2002), Nr. 3-4, p. 469–494. – URL https://doi.org/10.3166/jancl.12.469-494

[Renz and Ligozat 2005] Renz, Jochen ; Ligozat, Gérard: Weak Composition for Qualitative Spatial and Temporal Reasoning. In: Beek, Peter van (Editor): *Principles and Practice of Constraint Programming - CP 2005, 11th International Conference, CP 2005, Sitges, Spain, October 1-5, 2005, Proceedings* Volume 3709, Springer, 2005, p. 534–548. – URL https://doi.org/10.1007/11564751_40. – ISBN 3-540-29238-1

[Renz and Nebel 1999] Renz, Jochen ; Nebel, Bernhard: On the Complexity of Qualitative Spatial Reasoning: A Maximal Tractable Fragment of the Region Connection Calculus. In: *Artificial Intelligence* 108 (1999), Nr. 1-2, p. 69–123. – URL https://doi.org/10.1016/S0004-3702(99)00002-8

[Renz and Nebel 2001] Renz, Jochen ; Nebel, Bernhard: Efficient Methods for Qualitative Spatial Reasoning. In: *Journal of Artificial Intelligence Research* 15 (2001), p. 289–318. – URL https://doi.org/10.1613/jair.872

[Rintanen 2009] Rintanen, Jussi: Planning and SAT. In: (Biere et al. 2009), p. 483–504. – URL https://doi.org/10.3233/978-1-58603-929-5-483. – ISBN 978-1-58603-929-5

[Robinson 1965] Robinson, John A.: A Machine-Oriented Logic Based on the Resolution Principle. In: *J. ACM* 12 (1965), Nr. 1, p. 23–41. – URL http://doi.acm.org/10.1145/321250.321253

[Robinson and Voronkov 2001] Robinson, John A. (Editor) ; Voronkov, Andrei (Editor): *Handbook of Automated Reasoning (in 2 volumes)*. Elsevier and MIT Press, 2001. – URL https://www.sciencedirect.com/science/book/9780444508133. – ISBN 0-444-50813-9

[Rossi et al. 2006] Rossi, Francesca (Editor) ; Beek, Peter van (Editor) ; Walsh, Toby (Editor): *Foundations of Artificial Intelligence*. Volume 2: *Handbook of Constraint Programming*. Elsevier, 2006. – URL http://www.sciencedirect.com/science/bookseries/15746526/2. – ISBN 978-0-444-52726-4

[Roussel and Lecoutre 2009] Roussel, Olivier ; Lecoutre, Christophe: XML Representation of Constraint Networks: Format XCSP 2.1. In: *CoRR* abs/0902.2362 (2009). – URL http://arxiv.org/abs/0902.2362

[Roussel and Manquinho 2009] Roussel, Olivier ; Manquinho, Vasco M.: Pseudo-Boolean and Cardinality Constraints. In: (Biere et al. 2009), p. 695–733. – URL https://doi.org/10.3233/978-1-58603-929-5-695. – ISBN 978-1-58603-929-5

[Sahlqvist 1975] Sahlqvist, Henrik: Completeness and Correspondence In The First and Second Order Semantics For Modal Logic. 82 (1975), p. 110 – 143. – URL http://www.sciencedirect.com/science/article/pii/S0049237X08707286. – ISSN 0049-237X

[Sakai and Nabeshima 2015] Sakai, Masahiko ; Nabeshima, Hidetomo: Construction of an ROBDD for a PB-Constraint in Band Form and Related Techniques for PB-Solvers. In: *IEICE Transactions* 98-D (2015), Nr. 6, p. 1121–1127. – URL http://search.ieice.org/bin/summary.php?id=e98-d_6_1121

[Sakallah and Simon 2011] Sakallah, Karem A. (Editor) ; Simon, Laurent (Editor): *Theory and Applications of Satisfiability Testing - SAT 2011 - 14th International Conference, SAT 2011, Ann Arbor, MI, USA, June 19-22, 2011. Proceedings*. Volume *6695*. Springer, 2011. (Lecture Notes in Computer Science). – URL https://doi.org/10.1007/978-3-642-21581-0. – ISBN 978-3-642-21580-3

[Savelsbergh 1997] Savelsbergh, Martin W. P.: A Branch-and-Price Algorithm for the Generalized Assignment Problem. In: *Operations Research* 45 (1997), Nr. 6, p. 831–841. – URL https://doi.org/10.1287/opre.45.6.831

[Savický and Vomlel 2009] Savický, Petr ; Vomlel, Jirí: Triangulation Heuristics for BN2O Networks. In: Sossai, Claudio (Editor) ; Chemello, Gaetano (Editor): *Symbolic and Quantitative Approaches to Reasoning with Uncertainty, 10th European Conference, ECSQARU 2009, Verona, Italy, July 1-3, 2009. Proceedings* Volume 5590, Springer, 2009, p. 566–577. – URL https://doi.org/10.1007/978-3-642-02906-6_49. – ISBN 978-3-642-02905-9

[Schmidt and Tishkovsky 2008] Schmidt, Renate A. ; Tishkovsky, Dmitry: A General Tableau Method for Deciding Description Logics, Modal Logics and Related First-Order Fragments. In: Armando, Alessandro (Editor) ; Baumgartner, Peter (Editor) ; Dowek, Gilles (Editor): *Automated Reasoning, 4th International Joint Conference, IJCAR 2008, Sydney, Australia, August 12-15, 2008, Proceedings* Volume 5195, Springer, 2008, p. 194–209. – URL https://doi.org/10.1007/978-3-540-71070-7_17. – ISBN 978-3-540-71069-1

[Schmidt-Schauß and Smolka 1991] Schmidt-Schauss, Manfred ; Smolka, Gert: Attributive Concept Descriptions with Complements. In: *Artif. Intell.* 48 (1991), Nr. 1, p. 1–26. – URL https://doi.org/10.1016/0004-3702(91)90078-X

[Sebastiani and McAllester 1997] Sebastiani, Roberto ; McAllester, David: New Upper Bounds for Satisfiability in Modal Logics the Case-study of Modal K / IRST, Trento, Italy.

Citeseerx, October 1997. – Technical Report 9710-15. – URL http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.42.8332.

[Sebastiani and Tacchella 2009] Sebastiani, Roberto ; Tacchella, Armando: SAT Techniques for Modal and Description Logics. In: (Biere et al. 2009), p. 781–824. – URL https://doi.org/10.3233/978-1-58603-929-5-781. – ISBN 978-1-58603-929-5

[Sebastiani and Vescovi 2009] Sebastiani, Roberto ; Vescovi, Michele: Automated Reasoning in Modal and Description Logics via SAT Encoding: the Case Study of K(m)/ALC-Satisfiability. In: *J. Artif. Intell. Res.* 35 (2009), p. 343–389. – URL https://doi.org/10.1613/jair.2675

[Seipp and Helmert 2013] Seipp, Jendrik ; Helmert, Malte: Counterexample-Guided Cartesian Abstraction Refinement. In: Borrajo, Daniel (Editor) ; Kambhampati, Subbarao (Editor) ; Oddi, Angelo (Editor) ; Fratini, Simone (Editor): *Proc. of ICAPS'13*, AAAI, 2013, p. 347–351. – URL http://www.aaai.org/ocs/index.php/ICAPS/ICAPS13/paper/view/6034. – ISBN 978-1-57735-609-7

[Selman 2004] Selman, Bart: Algorithmic Adventures at the Interface of Computer Science, Statistical Physics, and Combinatorics. In: (Wallace 2004), p. 9–12. – URL https://doi.org/10.1007/978-3-540-30201-8_3. – ISBN 3-540-23241-9

[Selman et al. 1992] Selman, Bart ; Levesque, Hector J. ; Mitchell, David G.: A New Method for Solving Hard Satisfiability Problems. In: (Swartout 1992), p. 440–446. – URL http://www.aaai.org/Library/AAAI/1992/aaai92-068.php. – ISBN 0-262-51063-4

[Simon et al. 2005] Simon, Laurent ; Le Berre, Daniel ; Hirsch, Edward A.: The SAT2002 Competition. In: *Annals of Mathematics and A.I.* 43 (2005), Nr. 1, p. 307–342. – URL https://doi.org/10.1007/s10472-005-0424-6

[Sinz 2004] Sinz, Carsten: Visualizing the Internal Structure of SAT Instances (Preliminary Report). In: Proceeding, Online (Editor): *SAT 2004 - The Seventh International Conference on Theory and Applications of Satisfiability Testing, 10-13 May 2004, Vancouver, BC, Canada, Online Proceedings* Volume 3569, Springer, 2004, p. 1–6. – URL http://www.satisfiability.org/SAT04/programme/117.pdf

[Sinz and Dieringer 2005] Sinz, Carsten ; Dieringer, Edda-Maria: DPvis - A Tool to Visualize the Structure of SAT Instances. In: Bacchus, Fahiem (Editor) ; Walsh, Toby (Editor): *Theory and Applications of Satisfiability Testing, 8th International Conference, SAT 2005, St. Andrews, UK, June 19-23, 2005, Proceedings* Volume 3569, Springer, 2005, p. 257–268. – URL https://doi.org/10.1007/11499107_19. – ISBN 3-540-26276-8

[Sioutis and Condotta 2014] Sioutis, Michael ; Condotta, Jean-François: Tackling Large Qualitative Spatial Networks of Scale-Free-Like Structure. In: Likas, Aristidis (Editor) ; Blekas, Konstantinos (Editor) ; Kalles, Dimitris (Editor): *Artificial Intelligence: Methods and Applications - 8th Hellenic Conference on AI, SETN 2014, Ioannina, Greece, May 15-17, 2014. Proceedings* Volume 8445, Springer, 2014, p. 178–191. – URL https://doi.org/10.1007/978-3-319-07064-3_15. – ISBN 978-3-319-07063-6

[Sioutis et al. 2016] Sioutis, Michael ; Condotta, Jean-François ; Koubarakis, Manolis: An Efficient Approach for Tackling Large Real World Qualitative Spatial Networks.

In: *International Journal on Artificial Intelligence Tools* 25 (2016), Nr. 2, p. 1–33. – URL https://doi.org/10.1142/S0218213015500311

[Sioutis and Koubarakis 2012] Sioutis, Michael ; Koubarakis, Manolis: Consistency of Chordal RCC-8 Networks. In: *IEEE 24th International Conference on Tools with Artificial Intelligence, ICTAI 2012, Athens, Greece, November 7-9, 2012*, IEEE Computer Society, 2012, p. 436–443. – URL https://doi.org/10.1109/ICTAI.2012.66. – ISBN 978-1-4799-0227-9

[Smullyan 1966] Smullyan, Raymond M.: Trees and Nest Structures. In: *J. Symb. Log.* 31 (1966), Nr. 3, p. 303–321. – URL https://doi.org/10.2307/2270448

[Soare 2016] Soare, Robert I.: *Turing Computability - Theory and Applications*. Springer, 2016 (Theory and Applications of Computability). – URL https://doi.org/10.1007/978-3-642-31933-4. – ISBN 978-3-642-31932-7

[Soh and Inoue 2010] Soh, Takehide ; Inoue, Katsumi: Identifying Necessary Reactions in Metabolic Pathways by Minimal Model Generation. In: Coelho, Helder (Editor) ; Studer, Rudi (Editor) ; Wooldridge, Michael (Editor): *ECAI 2010 - 19th European Conference on Artificial Intelligence, Lisbon, Portugal, August 16-20, 2010, Proceedings* Volume 215, IOS Press, 2010, p. 277–282. – URL http://www.booksonline.iospress.nl/Content/View.aspx?piid=17757. – ISBN 978-1-60750-605-8

[Sörensson and Biere 2009] Sörensson, Niklas ; Biere, Armin: Minimizing Learned Clauses. In: (Kullmann 2009), p. 237–243. – URL https://doi.org/10.1007/978-3-642-02777-2_23. – ISBN 978-3-642-02776-5

[Stallman and Sussman 1977] Stallman, Richard M. ; Sussman, Gerald J.: Forward Reasoning and Dependency-Directed Backtracking in a System for Computer-Aided Circuit Analysis. In: *Artificial Intelligence* 9 (1977), Nr. 2, p. 135–196. – URL https://doi.org/10.1016/0004-3702(77)90029-7

[Stevenson et al. 2008] Stevenson, Lynn ; Britz, Katarina ; Hörne, Tertia: KT and S4 Satisfiability in a Constraint Logic Environment. In: Ho, Tu B. (Editor) ; Zhou, Zhi-Hua (Editor): *PRICAI 2008: Trends in Artificial Intelligence, 10th Pacific Rim International Conference on Artificial Intelligence, Hanoi, Vietnam, December 15-19, 2008. Proceedings* Volume 5351, Springer, 2008, p. 370–381. – URL https://doi.org/10.1007/978-3-540-89197-0_35. – ISBN 978-3-540-89196-3

[Stockmeyer 1976] Stockmeyer, Larry J.: The Polynomial-Time Hierarchy. In: *Theor. Comput. Sci.* 3 (1976), Nr. 1, p. 1–22. – URL https://doi.org/10.1016/0304-3975(76)90061-X

[Sutcliffe and Suttner 1997] Sutcliffe, Geoff ; Suttner, Christian B.: The CADE-13 ATP System Competition. In: *J. Autom. Reasoning* 18 (1997), Nr. 2, p. 137–138. – URL https://doi.org/10.1023/A:1005839515219

[de Swart 1998] Swart, Harrie C. M. de (Editor): *Automated Reasoning with Analytic Tableaux and Related Methods, International Conference, TABLEAUX '98, Oisterwijk, The Netherlands, May 5-8, 1998, Proceedings*. Volume *1397*. Springer, 1998. (Lecture Notes in Computer Science). – URL https://doi.org/10.1007/3-540-69778-0. – ISBN 3-540-64406-7

[Swartout 1992] Swartout, William R. (Editor): *Proceedings of the 10th National Conference on Artificial Intelligence. San Jose, CA, July 12-16, 1992*. AAAI Press / The MIT Press, 1992. – URL http://www.aaai.org/Conferences/AAAI/aaai92.php. – ISBN 0-262-51063-4

[Szczepanski 2012] Szczepanski, Nicolas: Méthodes Efficaces de Raisonnement en Logique Modale. In: *Master Thesis* (2012). – URL http://www.cril.univ-artois.fr/~szczepanski/publi_en.html

[Tallon et al. 2004] Tallon, Jean-Marc ; Vergnaud, Jean-Christophe ; Zamir, Shmuel: Communication among Agents: A Way to Revise Beliefs in KD45 Kripke Structures. In: *Journal of Applied Non-Classical Logics* 14 (2004), Nr. 4, p. 477–500. – URL https://doi.org/10.3166/jancl.14.477-500

[Tentrup 2017] Tentrup, Leander: On Expansion and Resolution in CEGAR Based QBF Solving. In: Majumdar, Rupak (Editor) ; Kuncak, Viktor (Editor): *Computer Aided Verification - 29th International Conference, CAV 2017, Heidelberg, Germany, July 24-28, 2017, Proceedings, Part II* Volume 10427, Springer, 2017, p. 475–494. – URL https://doi.org/10.1007/978-3-319-63390-9_25. – ISBN 978-3-319-63389-3

[Tsarkov and Horrocks 2006] Tsarkov, Dmitry ; Horrocks, Ian: FaCT++ Description Logic Reasoner: System Description. In: Furbach, Ulrich (Editor) ; Shankar, Natarajan (Editor): *Automated Reasoning, Third International Joint Conference, IJCAR 2006, Seattle, WA, USA, August 17-20, 2006, Proceedings* Volume 4130, Springer, 2006, p. 292–297. – URL https://doi.org/10.1007/11814771_26. – ISBN 3-540-37187-7

[Tseitin 1983] Tseitin, G. S.: On the Complexity of Derivation in Propositional Calculus. In: *Auto. of Reaso. 2: Classical Papers on Computational Logic 1967–1970*. Springer, 1983, p. 466–483. – URL http://www.decision-procedures.org/handouts/Tseitin70.pdf

[Turing 1938] Turing, Alan M.: *Systems of Logic Based on Ordinals*, Princeton University, NJ, USA, Ph.D. thesis, 1938. – URL https://doi.org/10.1112/plms/s2-45.1.161

[Turing 1950] Turing, Alan M.: Computing Machinery and Intelligence. In: *Mind* 59 (1950), Nr. 236, p. 433–460. – URL http://www.jstor.org/stable/2251299. – ISSN 00264423, 14602113

[Urquhart 1981] Urquhart, Alasdair: Decidability and the finite model property. In: *J. Philosophical Logic* 10 (1981), Nr. 3, p. 367–370. – URL https://doi.org/10.1007/BF00293428

[Van Benthem 2010] Van Benthem, Johan: *Modal Logic For Open Minds*. Volume 1. Center for the Study of Language and Inf, 2010. – 350 p. – URL http://fenrong.net/teaching/mljvb.pdf. – ISBN 978-1575865980

[Voronkov 1999] Voronkov, Andrei: KЭ: A Theorem Prover For K. In: Ganzinger, Harald (Editor): *Automated Deduction - CADE-16, 16th International Conference on Automated Deduction, Trento, Italy, July 7-10, 1999, Proceedings* Volume 1632, Springer, 1999, p. 383–387. – URL https://doi.org/10.1007/3-540-48660-7_35. – ISBN 3-540-66222-7

[Wallace 2004] Wallace, Mark (Editor): *Principles and Practice of Constraint Programming - CP 2004, 10th International Conference, CP 2004, Toronto, Canada, September 27 - October 1, 2004, Proceedings*. Volume *3258*. Springer, 2004. (Lecture Notes in Computer Science). – URL https://doi.org/10.1007/b100482. – ISBN 3-540-23241-9

[Wang et al. 2007] Wang, Chao ; Gupta, Aarti ; Ivancic, Franjo: Induction in CEGAR for Detecting Counterexamples. In: *Formal Methods in Computer-Aided Design, 7th International Conference, FMCAD 2007, Austin, Texas, USA, November 11-14, 2007, Proceedings*, IEEE Computer Society, 2007, p. 77–84. – URL https://doi.org/10.1109/FAMCAD.2007.21. – ISBN 0-7695-3023-0

[Weidenbach et al. 2009] Weidenbach, Christoph ; Dimova, Dilyana ; Fietzke, Arnaud ; Kumar, Rohit ; Suda, Martin ; Wischnewski, Patrick: SPASS Version 3.5. In: Schmidt, Renate A. (Editor): *Automated Deduction - CADE-22, 22nd International Conference on Automated Deduction, Montreal, Canada, August 2-7, 2009. Proceedings* Volume 5663, Springer, 2009, p. 140–145. – URL https://doi.org/10.1007/978-3-642-02959-2_10. – ISBN 978-3-642-02958-5

[Westphal et al. 2009] Westphal, Matthias ; Wölfl, Stefan ; Gantner, Zeno: GQR: A Fast Solver for Binary Qualitative Constraint Networks. In: *Benchmarking of Qualitative Spatial and Temporal Reasoning Systems, Papers from the 2009 AAAI Spring Symposium, Technical Report SS-09-02, Stanford, California, USA, March 23-25, 2009*, AAAI, 2009, p. 51–52. – URL http://www.aaai.org/Library/Symposia/Spring/2009/ss09-02-011.php

[Zhang 1997] Zhang, Hantao: SATO: An Efficient Propositional Prover. In: McCune, William (Editor): *Automated Deduction - CADE-14, 14th International Conference on Automated Deduction, Townsville, North Queensland, Australia, July 13-17, 1997, Proceedings* Volume 1249, Springer, 1997, p. 272–275. – URL https://doi.org/10.1007/3-540-63104-6_28. – ISBN 3-540-63104-6

[Zhang et al. 2001] Zhang, Lintao ; Madigan, Conor F. ; Moskewicz, Matthew W. ; Malik, Sharad: Efficient Conflict Driven Learning in Boolean Satisfiability Solver. In: Ernst, Rolf (Editor): *Proceedings of the 2001 IEEE/ACM International Conference on Computer-Aided Design, ICCAD 2001, San Jose, CA, USA, November 4-8, 2001*, IEEE Computer Society, 2001, p. 279–285. – URL https://doi.org/10.1109/ICCAD.2001.968634. – ISBN 0-7803-7249-2

Résumé

Ces quinze dernières années, des progrès spectaculaires dans le cadre du test de cohérence de formules propositionnelles (SAT) ont permis à une grande variété de problèmes de satisfaction et d'optimisation d'être traités à l'aide d'un moteur de résolution générique, le solveur SAT. Au delà des seuls solveurs SAT, de nombreux cadres utilisant des techniques issues du monde SAT ont pu bénéficier de ces améliorations : on citera par exemple SAT modulo Theories (SMT) ou Answer Set Programming (ASP). Cependant, il existe encore de nombreux formalismes pour lesquels le test de cohérence ne passe pas à l'échelle. C'est le cas par exemple dès que l'on rajoute des opérateurs modaux dans une formule logique. Le but de cette thèse est de concevoir un outil de résolution efficace pour tester la cohérence de formules logiques dans le cadre de la logique modale, problème que nous appellerons Modal SAT. Une première approche sera d'étudier les diverses façons de réduire le problème Modal SAT à un problème pour lequel il existe des solveurs efficaces en pratique : par exemple SAT, SMT ou ASP. Une autre approche sera de concevoir un solveur "ad hoc" pour Modal SAT en adaptant les principes et techniques des meilleurs solveurs cités plus haut. L'évaluation de solveurs requiert un ensemble varié de benchmarks, idéalement représentant des problèmes Modal SAT réels (par opposition aux problèmes générés aléatoirement ou aux exemples académiques). Un aspect important de la thèse sera de collecter et classer les problèmes Modal SAT disponibles dans la communauté et d'en créer de nouveaux.

Mots-clés: logiques modales, évaluation, raisonnement automatique

Abstract

In the past fifteen years, dramatic improvements in the context of testing propositional formulas consistency (SAT) have enabled a wide variety of satisfaction and optimization problems to be processed using a generic resolution engine: the SAT solver. Beyond SAT solvers, many fields using SAT world technics have benefited from these improvements: for example SAT modulo Theories (SMT) or Answer Set Programming (ASP). However, there are still many formalisms for which the coherence test does not pass the scale. This is the case for example when we add modal operators in a logical formula. The aim of this thesis is to design an effective resolution tool to test the consistency of logical formulas within modal logic, problems that we will call Modal SAT. One approach is to study the various ways to reduce the problem to a SAT problem for which there are effective solvers in practice: for example SAT, SMT or ASP. Another approach will be to design an "ad hoc" solver for Modal SAT adapting the principles and technics of the best solvers mentioned above. Checking solvers requires a diverse set of benchmarks, ideally representing actual Modal SAT problems (as opposed to randomly generated problems or academic examples). An important aspect of the thesis will be to collect and classify Modal SAT problems available in the community and create new ones.

Keywords: modal logics, benchmarks, automated reasoning