Implementation of omega-automata minimization techniques in "Spot", a model checking library.

When "Spot" SAT-based minimization meets incremental SAT solving.

Paris, January 2017

Submitted by:
Alexandre GBAGUIDI AÏSSE
Student in the second year in computer engineering at EPITA
`gbagui_a@epita.fr`

Supervised by:
Alexandre Duret-Lutz
Assistant Professor at LRDE (Research and Development Laboratory of EPITA)
`adl@lrde.epita.fr`

Contents

Part I

Report

Chapter 1

Introduction

This internship took place at LRDE (Research and Development Laboratory of EPITA). It was conducted under the supervision of *Alexandre Duret-Lutz*, assistant professor, both teacher at EPITA and researcher in the laboratory.

Thanks to my father *Gérard Gbaguidi Aïsse*, senior lecturer in civil engineering, I have always been interested in research. I chose to apply for an internship at LRDE to confirm or disprove this attraction towards the world of research.

The thought of having an experience in this laboratory came naturally at the end of the first engineering year. This year covers various domains of computer engineering through many programming assignments and courses. It aims to bring every student to discover these various aspects of computer engineering and identify their favourite field of work. During this year, one of the facts that came out the most is that I like to be confronted with problems of mathematical and algorithmic character. This lead me to realize that for my last two years at EPITA I want to specialize in cognitive science and artificial intelligence (required course when working as student in the laboratory). All this reinforced my desire of applying to LRDE.

Among the laboratory's projects I chose to ask for an internship on Spot. This is for three main reasons. The first one is that it exploits *Automata Theory*, an interesting course we have been introduced to, and *Model Checking* that aroused my curiosity. The second reason is that this project is mainly written in C++, a language that I like for its expressional strength which evolves quickly at each revision. And the last one is that *Alexandre Duret-Lutz*, the supervisor of this project is one of my favourite teachers at EPITA and he really makes me want to study more of his courses.

Spot is a C++ library for Linear-time Temporal Logic (LTL), ω-automata manipulation and model checking. In addition to the C++ interface it provides some command-line tools, a Python binding and an on-line translator.

My entire internship was focused on improving SAT-based minimization of ω-automata. This minimization is the result of two publications, an initial one achieved in 2014 and an improvement made in 2015. This subject was already a topic of concern as minimizing ω-automata usually makes algorithms that work on it more efficient.

Until now, SAT-based minimization procedures were using an external SAT solver. The main goal of this internship was to choose a SAT solver, integrate it to Spot and implement different approaches of minimization (notably incremental SAT solving) in order to find out the most efficient.

This internship report will start with an introduction of the LRDE , its projects including Spot and the basics concepts related to it. Then, it will define the outline and objectives of this intership before explaining the completed work.

Chapter 2

LRDE Presentation

2.1 Line of business

The LRDE (Research and Development Laboratory of EPITA) is focused on fundamental research and development in computer science. Its main areas of expertise are:

- Image processing and pattern recognition
- Automata and verification
- Performance and genericity

Building on its solid scientific production and academic collaborations, the laboratory has industrial contracts, conducts internal research projects and participates in collaborative academic research projects.
Its members also give classes to students at EPITA from the first year of engineering.

2.2 The Laboratory

The LRDE (`https://www.lrde.epita.fr/wiki/Home`) was created in February 1998 to promote the research activity at EPITA and to allow students to be involved into important research projects.

The research activity at LRDE is focusing on subjects related to the school with the aim goal of getting recognition in the scientific domain through publications and by working together with other research centers.
One particularity of the LRDE is the will to create a bond between traditional teaching given to EPITA students and teaching through research. The point of this is to:

- participate to the production of knowledge in computer science and to promote the image of EPITA in scientific field,
- develop LRDE student's training through research and allow them to access a third cycle training.

2.3 Members

The laboratory is currently composed of thirteen permanent members, including teacher-researchers, engineers and administration.
In addition to permanent staff, the LRDE also hosts PhD students. Currently, there are five of them. During the whole duration of their doctoral studies, they work with two advisor researchers, one of the LRDE and one of another university (joint supervision in partnership).

Each year, the permanent members recruit third year students from EPITA, who will stay until the end of their studies, following a dedicated study specialization at EPITA . Hence, the laboratory hosts two generations of students that can grow to a number between ten to fifteen.

2.4 Services

The LRDE is working on four different axis:

2.4.1 Image Processing

Olena

The Olena project (`https://olena.lrde.epita.fr`) is a generic image processing library. Its objective is to implement a platform of numerical scientific computations dedicated to image processing, pattern recognition and computer vision. This environment is composed of a generic and efficient library (Milena), a set of tools for shell scripts and a visual programming interface. The project aims at offering an interpreted environment like MatLab or Mathematica. It provides many ready-to-use image data structures (regular 1D, 2D, 3D images, graph-based images, etc.) and algorithms. Milena's algorithms are built upon classical entities from the image processing field (images, points/sites, domains, neighborhoods, etc.).

Each of these parts imply its own difficulties and require the development of new solutions. For example, the library, which requires the entirety of low level features on which it relies on to be both efficient and generic — two objectives that are hard to meet at the same time in programming. Fortunately, the object oriented programming eases this problem if we avoid the classical object modeling with inheritance and polymorphism. Hence, this genericity allows the development of efficient and re-usable code - i.e. developers or practitioners can easily understand, modify, develop and extend new algorithms while retaining the core traits of Milena: genericity and efficiency. The Olena platform uses this paradigm. The project has already addressed the problem of the diversity of data and data structures.

Furthermore, the people working on this project were able to put in light the existence of conception models related to generic programming. Olena is an open source project under General Public License (GPL) version 2.

Climb

The Climb team of the laboratory has chosen to focus on the persistent question of performance and genericity, only from a different point of view.

The purpose of this research is to examine the solutions offered by languages other than C++, dynamic languages notably, and Lisp in particular. C++ has its drawbacks, it is a heavy language with an extremely complex and ambiguous syntax, the template system is actually a completely different language from standard C++ and finally it is a static language. This last point has significant implications on the application, insofar as it imposes a strict chain of Compilation → Development → Run → Debug, making for example rapid prototyping or human-machine interfacing activities difficult. It becomes therefore essential to equip the involved projects with a third language infrastructure that is rather based on scripting languages.

The Climb project aims at investigating the same domain as Olena, but starting from an opposite view. It express the same issues following an axis of dynamic genericity and compares the performance obtained by some Common Lisp compilers with those of equivalent programs written in C or C++.

2.4.2 Finite state machine manipulation

Vcsn

The Vcsn project (`https://vcsn.lrde.epita.fr`) is a finite state machine manipulation platform developed in collaboration with the ENST. Finite state machines, also called automata, are useful for language treatment and task automation. In the past, such platforms, like "FSM", were supposed to work for problems of industrial scale. Hence, for efficiency reasons, they were specialized in letter automata. On the other hand, platforms like "FSA" were based on a more abstract approach. Vcsn tries to answer both of these issues by using techniques of static and generic programming in C++.

Vcsn can then support the entirety of automata with multiplicity in any kind of semiring. Thanks to generic programming techniques, it is not necessary to code a single algorithm once for each type of automata anymore. A single abstract version is sufficient, and this without loosing efficiency. It is not necessary to handle C++ perfectly to be able to use the platform thanks to an interpreter conceived to highlight all of the system's potential. This environment should allow researchers to experiment their ideas and beginners to practice with an intuitive interface.

VCSN is an open source project under GPL license.

2.4.3 Model checking

Spot

Spot (`https://spot.lrde.epita.fr/`) is a library of algorithms for "model checking", which is a way to check that every possible behavior of a system satisfies its given properties. Spot allows to express those properties using linear-time temporal logic (LTL). It corresponds to classical propositional calculus (with its "or", "and" and "not" operators) equiped with temporal operators to express things such as "in a future time" or "anytime since now". Spot also supports arbitrary acceptance condition, transition-based acceptance and four different representation formats of ω-automata (HOA, never claims, LBTT, DSTAR). All these terms will be explained in the basic concepts (section 4).

Such formulas seen above (LTL formulas) can be translated to automata (Spot implements different algorithms) , such as verifying that the behavior of a model satisfies a formula can be reduced to operations between two automata (here again Spot implements different algorithms). This approach can be applied to different kind of systems: communication protocoles, electronic circuits, programs...

This project was born in the MoVe team at LIP6, but since 2007 it is mainly developped by the LRDE, with some occasional collaborations with LIP6. It is distributed under a GNU GPL version 3 license.

2.4.4 Speaker recognition

Speaker ID

The Speaker Recognition team is working on Machine Learning solutions applied to Speaker Recognition tasks. They propose statistical representations of speech signal which are more robust to the problem of session and channel variabilities.

A speaker must always be identified, whether he is ill, suffering from sore throats, or his current emotions bring change to his voice. To do this, all the characteristics of a voice that can change depending on any external parameter must be ignored. This is one of the issues the Speaker ID team is facing.

They participated in the evaluation campaign of speaker verification systems organized by NIST (the National Institute of Standards and Technology) which organizes competitions in various fields, both to stimulate research and to define new standards since the beginning of the project.

The work of LRDE Speaker ID team is carried out in collaboration with the Spoken Language Systems Group of the MIT Computer Science and Artificial Intelligence Laboratory (`http://groups.csail.mit.edu/sls/`).

2.5 The internship in the work company

This internship took place within the team of model checking. It was essentially focused on the improvement of the SAT-based minimization of ω-automata. This feature is a result of previous work conducted by *Alexandre Duret-Lutz* and *Souheib Baarir* explained in two publications ([5] and [6]).

Chapter 3

Spot

Spot was first presented in 2004 [11]. It was purely a library until Spot 1.0 [10], when command-line tools for LTL manipulation and translation of LTL to some generalizations of Büchi Automata have started to be distributed. Today, Spot 2.0 [2] supports more tools with arbitrary acceptance conditions as described in the Hanoi Omega Automata format (HOA) [22] and python bindings usable in interactive environments such as IPython/Jupyter [14].

3.1 Structure

The Spot project can be broken down into several parts, as shown in Figure 3.1. Orange boxes are C/C++ libraries. Red boxes are command-line program. Blue boxes are Python-related.

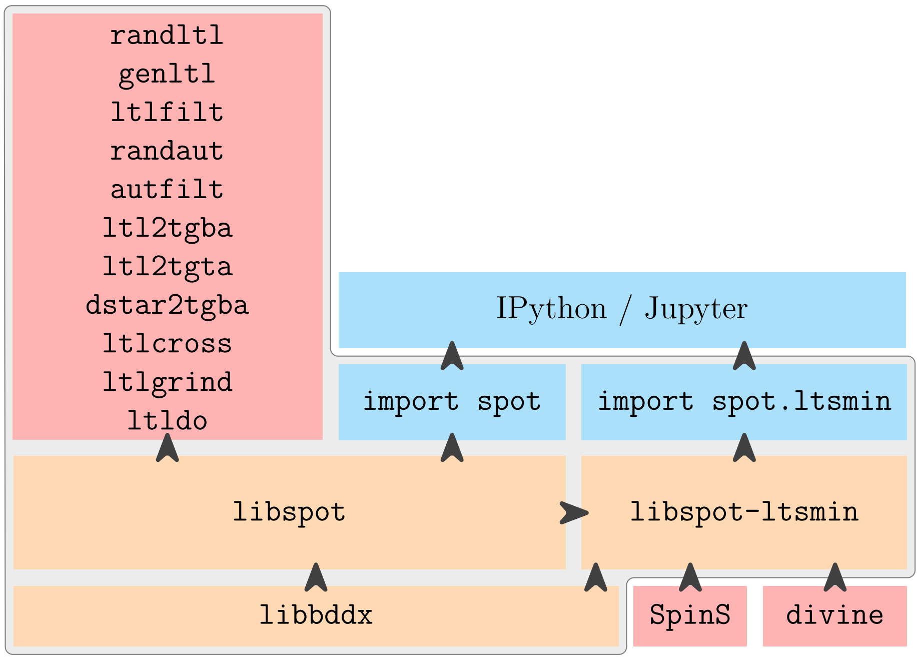


Figure 3.1: Architecture of Spot

Spot is actually split in three libraries:

- libbddx is a customized version of BuDDy for representing Binary Decision Diagrams which we use to label transitions in automata, and to implement a few algorithms.

- libspot is the main library containing all data structures and algorithms.
- libspot-ltsmin contains code to interface with state-spaces generated as shared libraries by LTSmin.

3.2 Command-line tools

Spot 2.0 installs the following eleven command-line tools, that are designed to be combined as traditional Unix tools.

SPOT 1.0 [10]	randltl	Generates random LTL/PSL formulas
	genltl	Generates LTL formulas from scalable patterns
	ltlfilt	Filter, converts, and transforms LTL/PSL formulas
	ltl2tgba	Translates LTL/PSL formulas into generalized Büchi automata [1], or deterministic parity automata (new in 2.0)
	ltl2tgta	Translates LTL/PSL formulas into Testing automata [3]
	ltlcross	Cross-compares LTL/PSL-to-automata translators to find bugs (works with arbitrary acceptance conditions since Spot 2.0)
	ltlgrind	mutates LTL/PSL formulas to help to reproduce bugs on smaller ones
	dstar2tgba	converts ltl2dstar automata into Generalized Büchi automata [5]
	randaut	generates random ω-automata
	autfilt	filters, converts and transforms ω-automata
	ltldo	runs LTL/PSL formulas through other translators, providing uniform input and output interfaces

Figure 3.2: Spot tools description

3.3 The Python Interface

Similar tasks can be performed in a more ”algorithm-friendly” environment using the Python interface. Combined with the IPython/Jupyter notebook [14] (a web application for interactive programming), this provides a nice environment for experiments, where automata and formulas are automatically displayed.

3.4 Workflow

Working on any project of the LRDE implies to follow some rules. That allows a better integration of each member. Once a patch is ready, any member of the model checking team can reread the patch and make suggestions.

3.4.1 Coding conventions

As Spot is a free software, uniformity of the code matters a lot. Some coding conventions are used so that the code looks homogeneous. Here are some points:

- UTF-8 is used for non-ASCII characters.
- tabs are not used for indentation in files, only spaces, in order to prevent issues with people assuming different tab widths.
- #include with angle-brackets refers to public Spot headers (i.e those that will be installed, or system headers that have already been installed).
- # include with double quotes refer to private headers that are distributed with Spot.

Have a look at `https://gitlab.lrde.epita.fr/spot/spot/blob/master/HACKING` for more details

3.4.2 Git Versionning Tool

The versionning tool used in Spot is Git. All development branches except 'master' and 'next' follows a particular naming convention: {initials}/{subject of work}. This allows a quick glance to identify who works on which branch and on what.

Concerning commits, large commits introducing a feature are preferred to many small commits covering the same feature. Suppose that a new feature must be implemented and needs 3 key steps. Even if each step is done in many commits during the development, at the end, it's better to squash commits so as to have only 3 large commits representing the 3 key steps.

Also, if at any moment in turns out that a previous work could have been done otherwise, any update must be applied directly to the commit concerned — each commit must actually insert code in its final form.

3.4.3 Adding Tests

Any implementation done must be tested and any feature should be documented. For the purpose on one hand to avoid regression and on the other hand to ensure the code runs as expected. All tests are located in the 'tests' folders. Most of them are written in Python (using the python bindings) or shell script.

Chapter 4

Basic concepts

Spot essentially manipulates ω-automata which is a variation of finite-state automata (FSA) that runs on infinite, rather than finite. Automata theory crops up pretty much everywhere in computer science: logic design, natural language processing, system analysis, regular expressions, etc. This chapter discusses about some basics of automata theory related to ω-automaton and then, the satisfiability problem.

4.1 Automata theory

This section consists essentially of excerpts of Spot's **concept** [7] and wikipedia ω-automaton web pages. Feel free to have a look on these pages for further details.

Automata theory [4] is the study of abstract machines and automata, as well as the computational problems that can be solved using them.

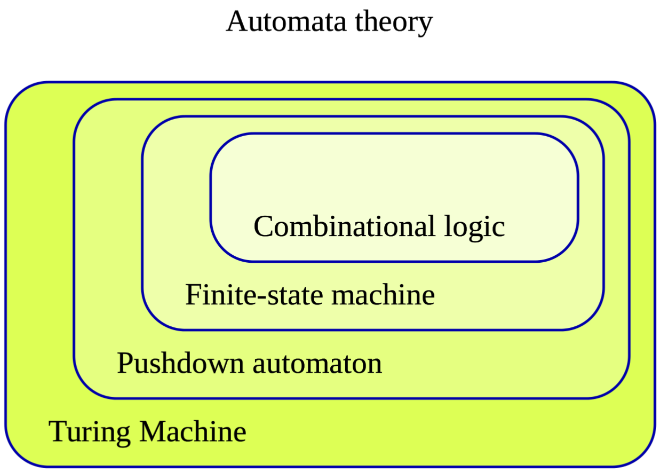


Figure 4.1: Classes of finite automata [4]

There are different classes of finate automata. The finite state machine has less computational power than some other models of computation such as the Turing machine [23]. The computational power distinction means there are computational tasks that a Turing machine can do but a finite-state automaton (FSA) cannot.

However, the ω-automata would be a simillar but parallel hierarchy.

4.1.1 Atomic proposition

An *atomic proposition* is a named Boolean variable that represents a simple property that must be true or false. It usually represents some properties of a system. They are used to construct temporal logic formulas [18] to specify properties of the system.

4.1.2 Boolean formula

A Boolean formula is formed from *atomic proposition*, the Boolean constants true and false, and standard Boolean operators like and, or, implies, xor, etc.

4.1.3 ω-words

An ω-word is a word of infinite length. In our context, each letter is used to describe the state of a system at a given time, and the sequence of letters shows the evolution of the system as the (discrete) time is incremented.

If the set **AP** of atomic propositions is fixed, an ω-word over **AP** is an infinite sequence of subsets of **AP**. In other words, there are $2^{|\mathbf{AP}|}$ possible letters to choose from, and these letters denote the set of atomic propositions that are true at a given instant.

For instance if **AP**$= \{a, b, c\}$, the infinite sequence $\{a, b\}$;$\{a\}$;$\{a, b\}$;$\{a\}$; $\{a, b\}$;$\{a\}$;... is an example of ω-word over **AP**. This particular ω-word can be interpreted as the following scenario: atomic proposition a is always true, b is true at each other instant, and c is always false.

4.1.4 ω-Automaton

An ω-automaton is used to represent sets of ω-word.

It is defined by a quintuplet $M = (AP, Q, q_o, X, \alpha)$ where:

- AP is a set of Atomic Proposition,
- Q is a finite set. The elements of Q are called states of M,
- $Q_0 \subset Q$ is the set of initial states. If the automaton is deterministic, there is only one initial state $q_0 \in Q$. Determinism is a notion defined below.
- X can be either a transition function $\delta : Q \times 2^{AP} \to Q$ (if the automaton is deterministic) or a transition relation $\Delta \subset (Q \times 2^{AP} \times Q)$ (if it is not deterministic).
- And α is a positive Boolean function over terms of the form $Fin(T)$ or $Inf(T)$ with $T \subseteq \Delta$. This is also called the *acceptance condition*.

The language of an ω-automaton is the set of ω-words it accepts.

There are many kinds of ω-automata and they mostly differ by their acceptance condition. The different types of acceptance condition, and whether the automata are deterministic or not can affect their expressive power.

4.1.5 Determinism

An automaton is said to be *deterministic* if and only if for each pair of state and symbol ($Q \times 2^{AP}$) there is one and only one transition to a next state.

A *non deterministic* automaton allows:

- many transitions labeled by the same symbol and outgoing from the same state,
- transitions labeled by the *empty word* ε,
- transitions labeled by more than one symbol.

Therefore, for each pair of state and symbol, there is no more one destination state but a set of possible destination states. Hence the $Q \times 2^{AP} \times Q$.

4.1.6 ω-Automaton run

A run of M on the input $(a_1, a_2, a_3, ...)$ is any infinite sequence $\rho = (r_0, r_1, r_2, ...)$ of states that satisfies the following conditions:

- r_0 is an element of Q_0,
- r_1 is an element of $\Delta(r_0, a_1)$,
- r_2 is an element of $\Delta(r_0, a_2)$.
  ...
- r_n is an element of $\Delta(r_{n-1}, a_n)$,

A nondeterministic ω-automaton may admit many different runs on any given input, or none at all. The input is accepted if at least one of the possible runs is accepting. Whether a run is accepting depends only on α, as for deterministic ω-automata. Every deterministic ω-automaton can be regarded as a nondeterministic ω-automaton by taking Δ to be the graph of δ. The definitions of runs and acceptance for deterministic ω-automata are then special cases of the nondeterministic cases.

4.1.7 Transition-based vs. State-based acceptance

The figure 4.2 has state-based acceptance set, runs are accepting if these visit infinitely often some state in each acceptance set, etc. Transition-based acceptance means acceptance sets are now sets of transitions and runs are accepting if the transitions they visit satisfy the acceptance condition.

Using transition based acceptance allows fore more compact automata. Here are two representations of the formula GFa (infinitely often a):

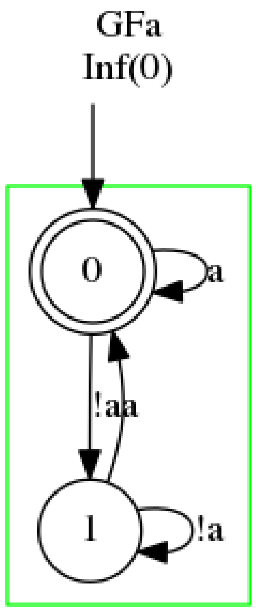


Figure 4.2: State based automaton representing formula **GFa**

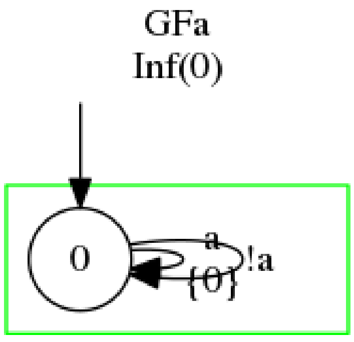


Figure 4.3: Transition based automaton representing formula **GFa**

4.1.8 Acceptance condition

An acceptance condition actually consists of two pieces: some acceptance sets, and a formula that tells how to use these acceptance sets.

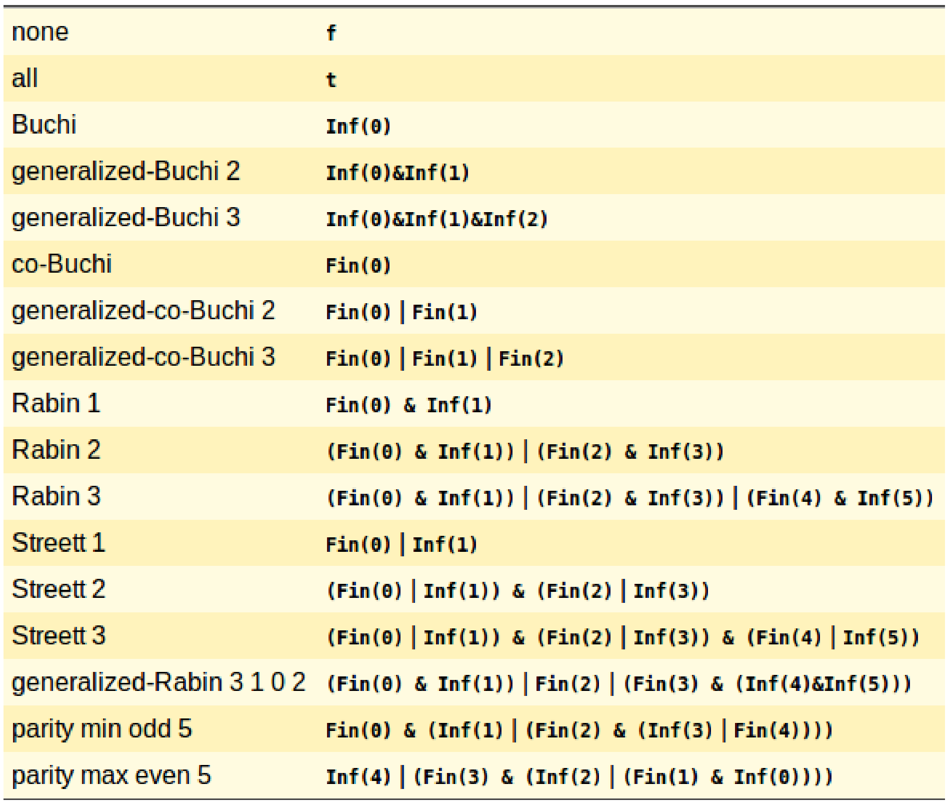
Acceptance formulas are positive Boolean formula over atoms of the form t, f, $Inf(n)$, or $Fin(n)$, where n is a non-negative integer denoting an acceptance set.

- **t** denotes the true acceptance condition: any run is accepting
- **f** denotes the false acceptance condition: no run is accepting
- **Inf(n)** means that a run is accepting if it visits infinitely often the acceptance set n

- **Fin(n)** means that a run is accepting if it visits finitely often the acceptance set n

The above atoms can be combined using only the operator & and | (also known as $\wedge$ and $\vee$), and parentheses for grouping. Note that there is no negation, but an acceptance condition can be negated swapping t and f, $\wedge$ and $\vee$, and $Fin(n)$ and $Inf(n)$.

The following table gives an overview of how some classical acceptance conditions are encoded. The first column gives a name that is more human readable (those names are defined in the HOA [22] format and are also recognized by Spot). The second column gives the encoding as a formula.



none	`f`
all	`t`
Buchi	`Inf(0)`
generalized-Buchi 2	`Inf(0)&Inf(1)`
generalized-Buchi 3	`Inf(0)&Inf(1)&Inf(2)`
co-Buchi	`Fin(0)`
generalized-co-Buchi 2	`Fin(0) | Fin(1)`
generalized-co-Buchi 3	`Fin(0) | Fin(1) | Fin(2)`
Rabin 1	`Fin(0) & Inf(1)`
Rabin 2	`(Fin(0) & Inf(1)) | (Fin(2) & Inf(3))`
Rabin 3	`(Fin(0) & Inf(1)) | (Fin(2) & Inf(3)) | (Fin(4) & Inf(5))`
Streett 1	`Fin(0) | Inf(1)`
Streett 2	`(Fin(0) | Inf(1)) & (Fin(2) | Inf(3))`
Streett 3	`(Fin(0) | Inf(1)) & (Fin(2) | Inf(3)) & (Fin(4) | Inf(5))`
generalized-Rabin 3 1 0 2	`(Fin(0) & Inf(1)) | Fin(2) | (Fin(3) & (Inf(4)&Inf(5)))`
parity min odd 5	`Fin(0) & (Inf(1) | (Fin(2) & (Inf(3) | Fin(4))))`
parity max even 5	`Inf(4) | (Fin(3) & (Inf(2) | (Fin(1) & Inf(0))))`

Figure 4.4: ω-automata acceptance conditions [7]

4.1.9 Completeness

An automaton is said to be complete if and only if for each pair of state and input, there is at least one transition to a next state.

4.2 SAT solver

Satisfiability problem is a classic of computer science.

The purpose of SAT solving is to assign each variables of a propositional formula in such a way that the formula evaluates to true. It is the canonical NP-complete problem. SAT solvers are used to solve many practical problems and this is also the case in Spot, they are used to minimize ω-automata.

4.2.1 Disjunctive Normal Form (DNF)

A logical formula is said to be in disjunctive normal form if it is a disjunction of conjunctive clauses. It can also be described as an OR of ANDS. Note that the **not** operator $\neg$ is authorized and can only be used as part of a litteral. For instance, the following formulas are in DNF:

- $(A \wedge B \wedge C) \vee (\neg A \wedge B \wedge \neg C) \vee (A \wedge \neg B \wedge C)$
- $A \vee (B \wedge C)$
- $A \vee B \vee \neg C$

To solve such a formula, as it is a disjunction of cunjunctive clauses, it can be considered as an enumeration of possible solutions. Each conjunctive clause is a solution sufficient to evaluate all the DNF to true.

4.2.2 Conjunctive Normal Forms (CNF)

Similarly, a conjunctive normal form is a logical Boolean formula that consists of conjunction of clauses — clauses being a disjunction of litterals. It can also be described as an AND of ORS and the operator $\neg$ is also authorized, only attached to a litteral.

- $\neg A \wedge (B \vee C)$
- $(A \vee B \vee C) \wedge (D \vee \neg E) \wedge (F)$
- $(A \vee B) \wedge (\neg A \vee C)$

Solving such a formula is not as easy as DNF formulas. SAT solvers can solve these CNF formulas.

4.2.3 DIMACS format

DIMACS is a file format used to define a Boolean expression written in conjunctive normal form. This file can be used as input for SAT solvers — SAT problems are encoded in a file following DIMACS format [9]. For more details, **SAT-solving in practice** [21] is a good introduction to SAT solvers.

Chapter 5

Specifications

This chapter tries to shed light on what was before this internship and why all the achieved work was needed.

5.1 Overall goal

The principal purpose of this internship is to improve an algorithm already implemented and used to minimize *deterministic* ω-automaton. The source code associated is the result of two papers:

- **SAT-based Minimization of Deterministic ω-Automata [6]** and
- **Mechanizing the Minimization of Deterministic Generalized Büchi Automata [5]**.

Those two papers written by *Souheib baarir* and *Alexandre Duret-Lutz* are themselves a generalization of **Ehlers**' SAT-based procedure [12]. Note that the first paper [6] is an extension of the second [5] which is restricted to generalized-Büchi acceptance.

5.1.1 The existing minimization

The existing minimization is based on this observation: since the reference automaton R (the inputed one) is complete and deterministic and the candidate automaton C (the mininal one to find) is also complete and deterministic, any word of Σ^ω has a unique run in R and C. Verifying that both automaton recognizes the same language can be done by ensuring that each word is accepted by R if and only if it is accepted by C. In practice, this is checked by ensuring that any cycle of the synchronous product $C \otimes R$ corresponds to cycles that are either accepting in C and R, or rejecting in both.

These previous work introduced a tool called $\textsc{Synthetize}$DTGBA(R, n, m) that can read any *deterministic* ω-automaton and synthetize (if it exists) an equivalent *deterministic* ω-automaton with a given number of states and arbitrary acceptance condition. It works this way:

- inputs a complete DTGBA R, two target numbers of state (n) and arbitrary acceptance condition (m),

- produces a DIMACS file [9] with all necessary clauses,
- calls a SAT solver to solve this problem,
- builds the resulting DTGBA C if it exists.

Using this tool, two minimization algorithms have been implemented:

Algorithm 1 A naive algorithm that calls SYNTHETIZEDTGBA(R, n, m) in a loop, with a decreasing number of states, and returns the last successfully built automaton.

```
1: procedure REDUCESTATESDTGBA($R, m = R$.NB_ACC_SETS())
2: repeat:
3:     $n \leftarrow R.nb\_states()$
4:     $C \leftarrow$ SYNTHETIZE$DTGBA(R, n-1, m)$
5:     if $C$ does not exist then return $R$
6:     $R \leftarrow C$
```

Algorithm 2 This also calls SYNTHETIZEDTGBA(R, n, m) in a loop, but attempting to find the minimum number of states using a binary search.

```
1: procedure DICHOTOMYDTGBA($R, m = R$.NB_ACC_SETS())
2:     $max\_states \leftarrow R.nb\_states() - 1$
3:     $min\_states \leftarrow 1$
4:     $S \leftarrow null$
5:     while $min\_states \leq max\_states$ do
6:         $target \leftarrow \lfloor (max\_states + min\_states)/2 \rfloor$
7:         $C \leftarrow$ SYNTHETIZE$DTGBA(R, target, m)$
8:         if $C$ does not exist then
9:             $min\_states \leftarrow target + 1$
10:        else
11:            $S \leftarrow C$
12:            $max\_states \leftarrow R.nb\_states() - 1$
13:    $R \leftarrow S$
```

Before this internship, Algorithm 1 was used by default. There are no real reasons for that except that the second algorithm was implemented later in a completely different context and has never been benchmarked and compared to the first one.

ω-Automata minimization can be seen in two ways:

- **Reduction of the number of states:** This is typically the algorithm 1 that keeps by default the same number of acceptance sets (m) and decreases n at each SYNTHETIZEDTGBA(R, n, m) call. The algorithm 2 has also the same perception. It knows the minimal automaton is between 1 (obviously, a smaller one does not exists) and $n - 1$ so instead of checking each number of states it performs a binary search with the will to be faster.

- **Rise of the accepting sets number:** This can be interpreted as the converse of a degeneralization: instead of augmenting the number of states to reduce the number of acceptance sets, we increase the number of acceptance sets in an attempt to reduce the number of states.

The figure below 5.1 is a great example from the first paper [5] that shows how smaller an automaton can become if the acceptance sets number is increased. Note that $|\mathcal{F}|$ here is m (the number of accepting sets).

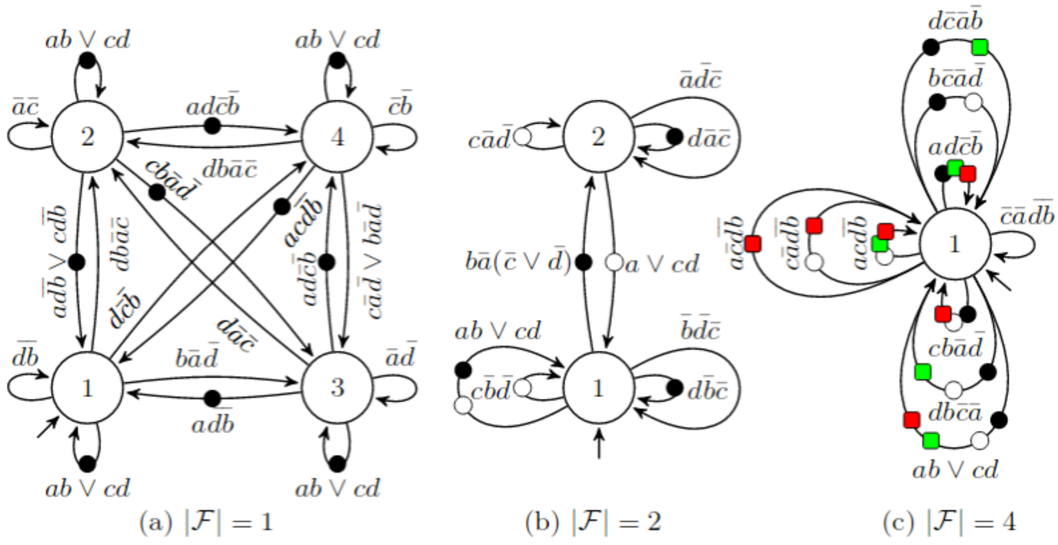


Figure 5.1: Examples of minimal DTGBA recognizing $(GFa \wedge GFb) \vee (GFc \wedge GFd)$

Finding the smallest m such that no smaller equivalent DTGBA with a larger m can be found is still an opened problem.

5.1.2 Tool chain

The figure 5.2 (from the FORTE'14 paper [5]) gives an overview of the processing chains that can be used to turn an LTL formula [18] into minimal DBA, DTBA or DTGBA. The blue area at the top describes:

Listing 5.1: bash command-line to translate a formula using ltl2tgba

```
ltl2tgba -D -x sat-minimize
```

while the purple area at the bottom corresponds to:

Listing 5.2: bash command-line to translate a formula using dstar2tgba

```
dstar2tgba -D -x stat-minimize
```

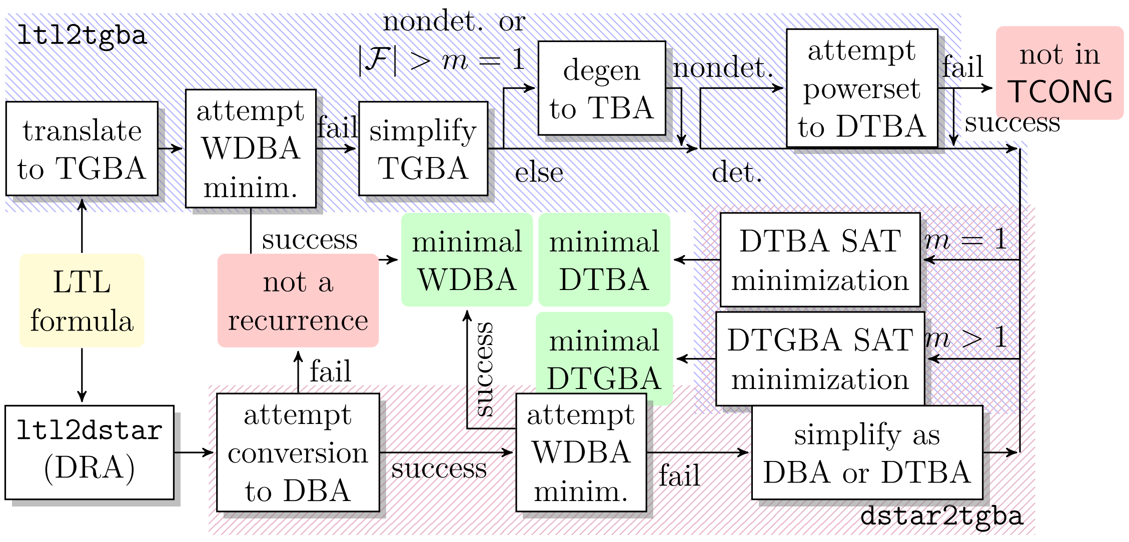


Figure 5.2: Two tool chain used to convert an LTL formula

As the SAT-based minimization only takes DTBA or DTGBA, the input automaton undergoes some transformations. In each tool, a Weak-DBA minimization is attempted. If that succeeds, a minimal weak DBA is outputed (looking for transition-based or generalized acceptance will not reduce it further). For further details, please read the FORTE'14 paper [5].

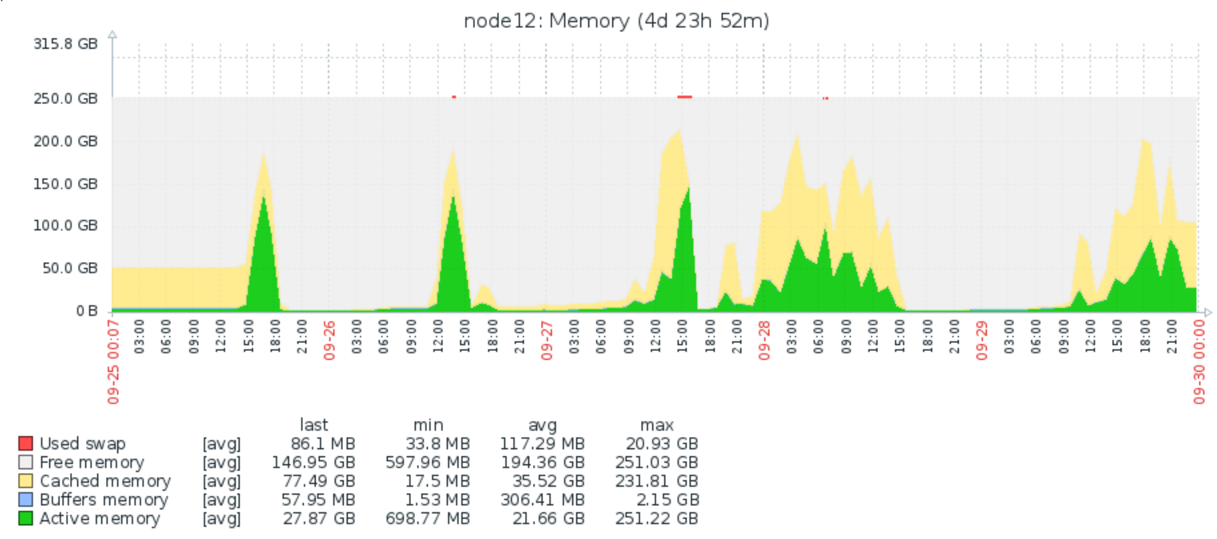
5.1.3 The limits

Consider again the default algorithm (1). At each iteration, this algorithm re-encodes the research of a smaller automaton from scratch. Iterations after iterations (from n to $n-1$, $n-1$ to $n-2$, etc.) the encoded clauses are almost the same. Therefore, it is a shame that nothing is retained, learned at each iteration. This is where incremental sat solving comes in mind.

Incremental solving is one of the recent directions in SAT solver research. This is based on an observation that in many applications of SAT solvers, the problems being solved consist of several calls to SAT solver on a sequence of SAT problem. Typically, the problems in the sequence share a large common part, making them highly interrelated. This is exactly our case! The purpose of incremental SAT solving is to recycle the work done in solving a previous problem in the sequence to solve the subsequent problems.

In order to use an incremental approach with SAT solver, it is no more possible to use DIMACS file [9]. Therefore, Spot needs to be linked to a SAT solving library. Let's remember that until now Spot requires a SAT solver to be installed on the same machine and provides a way to set it (through SPOT_SATSOLVER environment variable). obviously, being linked to a SAT solver and make calls to its functions will be more efficient than making Input/Output operations and executing another binary.

The memory consumption is also another problem. The larger the automaton is, the more variables there are. With some automata, the memory usage could grow over 150 Go which is uncommon for most users. The figure 5.3 helps to see memory usage peaks during a benchmark realized at the end of September 2016.



		last	min	avg	max
🟥 Used swap	[avg]	86.1 MB	33.8 MB	117.29 MB	20.93 GB
⬜ Free memory	[avg]	146.95 GB	597.96 MB	194.36 GB	251.03 GB
🟨 Cached memory	[avg]	77.49 GB	17.5 MB	35.52 GB	231.81 GB
🟦 Buffers memory	[avg]	57.95 MB	1.53 MB	306.41 MB	2.15 GB
🟩 Active memory	[avg]	27.87 GB	698.77 MB	21.66 GB	251.22 GB

Figure 5.3: A graph showing memory usage during a benchmark realized at the beginning of the internship

5.2 Detailed explanation of the results to be obtained

As said before, the purpose of this internship is to improve the current SAT-based minimization technique. This minimization is wanted to be faster and less greedy in memory consumption. If you look at the tool chain section above, this internship intervenes in the two SAT minimization rectangles (for $m = 1$ and $m > 1$).

The idea of testing an incremental approach for SAT-based minimization has already been raised by the Spot team. *Alexandre Duret-Lutz* had a clear idea of how to do it and I had been assigned the task of testing this idea. The objective being to improve it as mush as possible, the internship is not limited to that. It is a line of approach as many others and the results to obtain can lead to new reflections.

Algorithm 3 An incremental approach that does the same traditional encoding once and then tries to exclude one more state at each iteration of a loop. The encoding is never restarted.

```
1: procedure ReduceStatesDTGBA($R, m = R.\textsc{nb\_acc\_sets}()$)
2:     $n \leftarrow R.nb\_states()$
3:     $C \leftarrow$ SynthetizeDTGBA($R, n-1, m$)
4:     if $C$ does not exists then return $R$
5: repeat:
6:     add clauses to exclude one more state
7:     $C \leftarrow$ Try to solve the new problem and build the new automaton
8:     if $C$ does not exist then return $R$
9:     $R \leftarrow C$
```

To do so, Spot needs to be linked to a SAT solving library. The really first task is to know which library. I had to find the more suitable SAT solver that fills those requirements:

- It must have a compatible licence with Spot's one. Spot is under a GNU General Public Licence 3.
- It has to be maintained.
- It must be simple to integrate with Spot. Simple means here that the code shall be modified as little as possible so that a future update to a newer version of that solver will be simple to achieve.
- It must be easy to use as library.
- Of course, it has to be efficient. Therefore, a look to SAT solvers international competitions as well as a custom benchmark is a fundamental need.

Regarding the memory consumption, the purpose is to identify the most memory-hungry parts of the source code and come up with solutions.

The results to be obtained cannot be more precise than that. There is no precise figure estimating the speed to reach for a particular formula or the exact memory consumption to reduce, etc.

It is an internship that is part of a research work. By definition, the results are often unpredictable.

Chapter 6

Contributions to Spot

This chapter presents the different things I have done on Spot during the internship. Might it be some algorithms implemented, scripts, display arrangements, benchmarks, results analyzes, etc.
All the achieved work is presented in a chronological way, to bring out the difficulties encountered, the unexpected results that had influence on the advancement of the work.

6.1 fastSAT

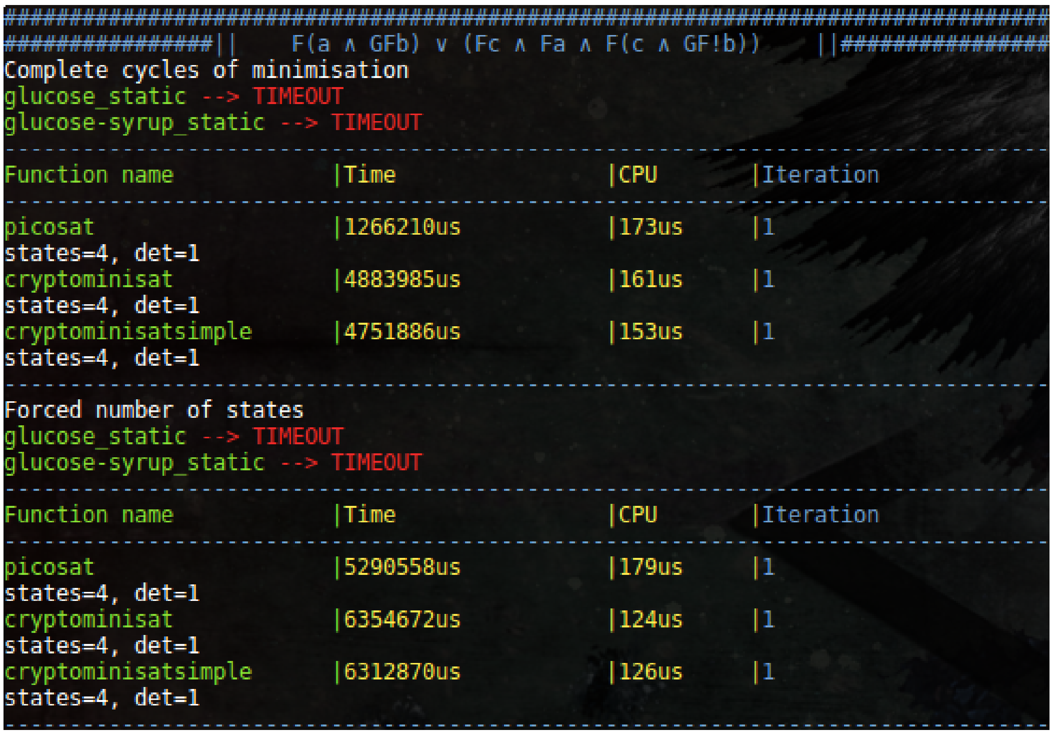
As a quick reminder, here are the required characteristics for the SAT solver:

- licence compatibility with Spot's one (GNU GPL v3),
- simplicity of integration for future updates,
- effectiveness.

The project **fastSAT** [13] was born to help to choose the SAT solver to distribute with Spot. Until now, SAT-based minimization was performed through an external SAT solver. The default one was Glucose [16] (3.0 version). Therefore, it seems logical to consider Glucose as a possible candidate. **fastSAT** [13] compared Glucose 4.0 [16] to CryptoMiniSat 5.0.1 [8] and PicoSAT 965 [20]. Note that some SAT solvers provide two versions, one parallel and one simple essentially because of the SAT competitions. In short, were compared:

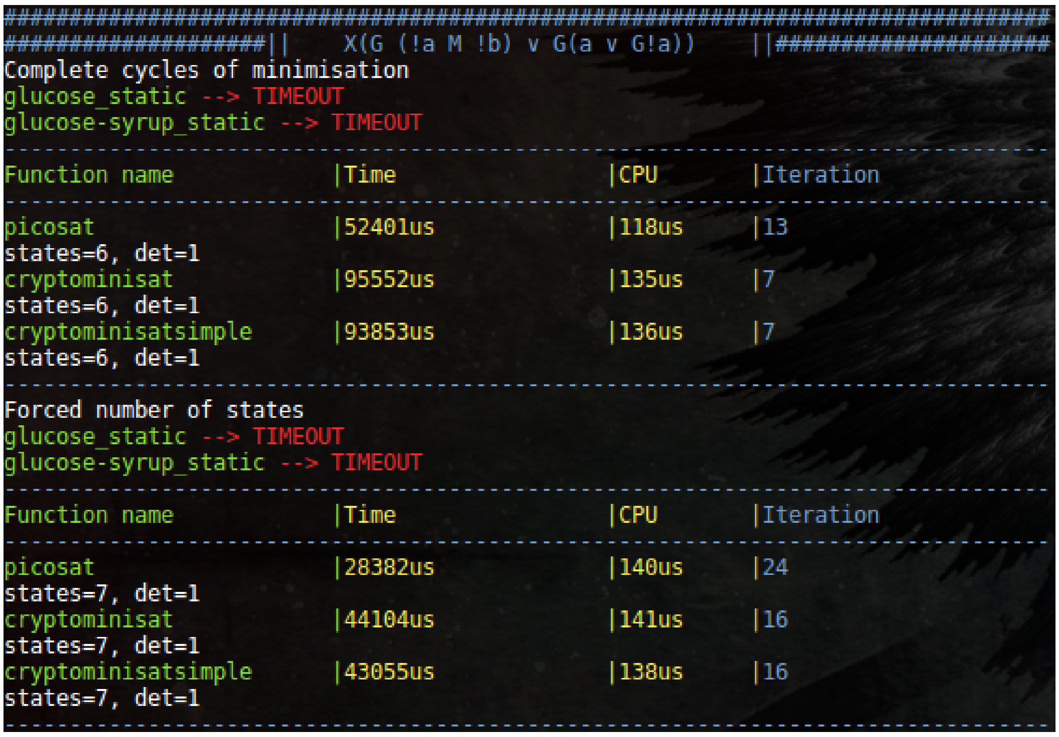
- Glucose syrup (parallel) 4.0
- Glucose simple 4.0
- CryptoMiniSat parallel 5.0.1
- CryptoMiniSat simple 5.0.1
- PicoSAT 965

The next three figures (6.1, 6.2 and 6.3) show some comparaisons for three formulas. About twenty formulas have been tested in two modes: by forcing the number of state and by doing the complete cycles of minimization. It has been executed on a computer with an **Intel(R) Core(TM) i7-4710HQ CPU @ 2.50GHz** processor and a **8 GiB** system memory. The measuring tool used is the open Google Benchmark tool [17].



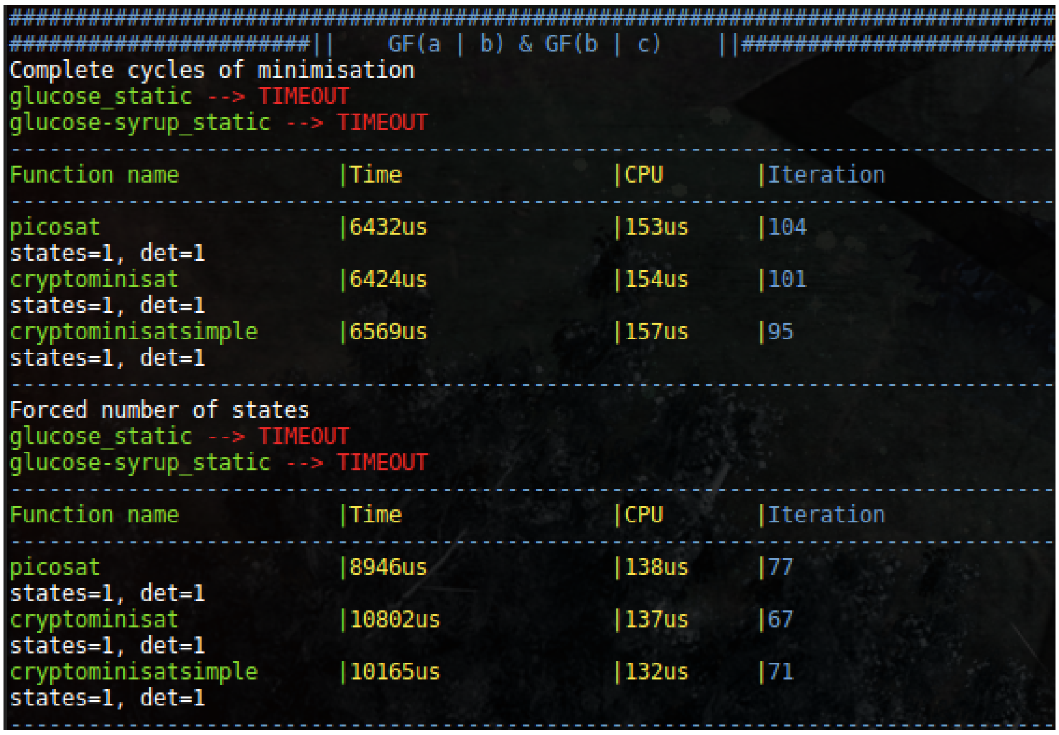
```
################################################################################
################||    F(a ∧ GFb) ∨ (Fc ∧ Fa ∧ F(c ∧ GF!b))    ||################
Complete cycles of minimisation
glucose_static --> TIMEOUT
glucose-syrup_static --> TIMEOUT
--------------------------------------------------------------------------------
Function name             |Time               |CPU        |Iteration
--------------------------------------------------------------------------------
picosat                   |1266210us          |173us      |1
states=4, det=1
cryptominisat             |4883985us          |161us      |1
states=4, det=1
cryptominisatsimple       |4751886us          |153us      |1
states=4, det=1
--------------------------------------------------------------------------------
Forced number of states
glucose_static --> TIMEOUT
glucose-syrup_static --> TIMEOUT
--------------------------------------------------------------------------------
Function name             |Time               |CPU        |Iteration
--------------------------------------------------------------------------------
picosat                   |5290558us          |179us      |1
states=4, det=1
cryptominisat             |6354672us          |124us      |1
states=4, det=1
cryptominisatsimple       |6312870us          |126us      |1
states=4, det=1
--------------------------------------------------------------------------------
```

Figure 6.1: Benchmark for formula $F(a \wedge GFb) \vee (Fc \wedge Fa \wedge F(c \wedge GF(!b)))$



```
################################################################################
####################||    X(G (!a M !b) ∨ G(a ∨ G!a))    ||######################
Complete cycles of minimisation
glucose_static --> TIMEOUT
glucose-syrup_static --> TIMEOUT
--------------------------------------------------------------------------------
Function name             |Time               |CPU        |Iteration
--------------------------------------------------------------------------------
picosat                   |52401us            |118us      |13
states=6, det=1
cryptominisat             |95552us            |135us      |7
states=6, det=1
cryptominisatsimple       |93853us            |136us      |7
states=6, det=1
--------------------------------------------------------------------------------
Forced number of states
glucose_static --> TIMEOUT
glucose-syrup_static --> TIMEOUT
--------------------------------------------------------------------------------
Function name             |Time               |CPU        |Iteration
--------------------------------------------------------------------------------
picosat                   |28382us            |140us      |24
states=7, det=1
cryptominisat             |44104us            |141us      |16
states=7, det=1
cryptominisatsimple       |43055us            |138us      |16
states=7, det=1
--------------------------------------------------------------------------------
```

Figure 6.2: Benchmark for formula $X(G(!aM!b) \vee G(a \vee G(!a)))$



```
################################################################################
#######################||    GF(a | b) & GF(b | c)    ||########################
Complete cycles of minimisation
glucose_static --> TIMEOUT
glucose-syrup_static --> TIMEOUT
--------------------------------------------------------------------------------
Function name            |Time                 |CPU        |Iteration
--------------------------------------------------------------------------------
picosat                  |6432us               |153us      |104
states=1, det=1
cryptominisat            |6424us               |154us      |101
states=1, det=1
cryptominisatsimple      |6569us               |157us      |95
states=1, det=1
--------------------------------------------------------------------------------
Forced number of states
glucose_static --> TIMEOUT
glucose-syrup_static --> TIMEOUT
--------------------------------------------------------------------------------
Function name            |Time                 |CPU        |Iteration
--------------------------------------------------------------------------------
picosat                  |8946us               |138us      |77
states=1, det=1
cryptominisat            |10802us              |137us      |67
states=1, det=1
cryptominisatsimple      |10165us              |132us      |71
states=1, det=1
--------------------------------------------------------------------------------
```

Figure 6.3: Benchmark for formula $GF(a \vee b) \wedge GF(b \vee c)$

In conclusion, among the different SAT solvers, **PicoSAT** was choosen for its strong performances. It consists of two source code files: **picosat.h** and **picosat.c** and was easily integrated and harmonised with Spot.

fastSAT [13] project is fully availlable and anyone can reproduce the benchmarks. Feel free to have a look.

6.2 New Satsolver class

There has already been a satsolver class that is instantiated at the beginning of the SAT-based minimization procedures, formerly used to initialize a temporary **cnf file** (DIMACS [9]), return it and call the external SAT solver. The file writing was directly made by those procedures.
The objective is to completely abstract the file writing. SAT-based minimization procedures will just have to instantiate a satsolver object at the beginning and make calls to its functions. Thoses functions will call the SAT solving library functions. But this class must continue to support any external SAT solver by handling temporary **cnf files**.

The figure 6.4 shows an approximate UML representation of the new satsolver class. It can either initialize PicoSAT or a cnf_stream_. The idea is to let its functions (add, comment, etc.) to decide if they call PicoSAT functions or write in the cnf_stream_. That way, SAT-based minimization procedures are not aware of what's going on behind and can repeat over and over again the same algorithms.

Of course the clause counting mechanism is provided by the new satsolver class. At the end, SAT-based procedures will just have to call the *get_solution()* method.

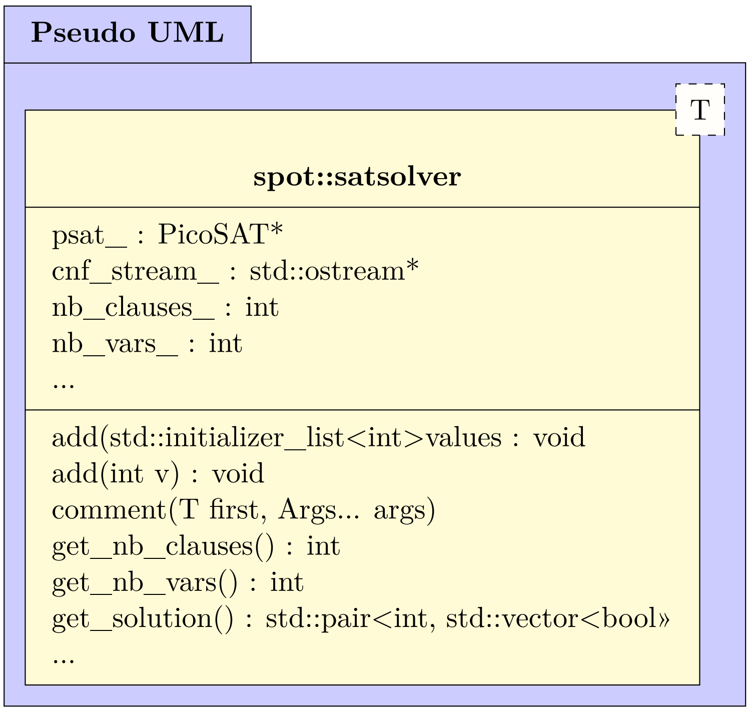

Pseudo UML
T
spot::satsolver
psat_ : PicoSAT*
cnf_stream_ : std::ostream*
nb_clauses_ : int
nb_vars_ : int
...
add(std::initializer_list<int>values : void
add(int v) : void
comment(T first, Args... args)
get_nb_clauses() : int
get_nb_vars() : int
get_solution() : std::pair<int, std::vector<bool»
...

Figure 6.4: Satsolver class UML representation

Once this was done, SAT-based procedures had already gained some speed which is entirely understandable as disk operations are slow. For instance, with this command line:

Listing 6.1: bash command-line to test a formula minimization using ltl2tgba

```
time ltl2tgba -D -x sat-minimize 'G(a -> Fb) & G(c -> Fd)'
                    --stats='states=%s, det=%d'
```

This result is obtained on a Macbook pro with **2,9GHz intel core i5** and **8 Go 1867 Mhz DDR3**:

PicoSAT as	Time result
linked SAT solving library external SAT solver (with disk operations)	0.29s user 0.08s system 98% cpu 0.371 total 0.82s user 0.09s system 98% cpu 0.925 total

6.3 First incremental approach

Just as a reminder, until now, SAT-based minimization procedures whether it is with binary search (algorithm 2) or the naive way (algorithm 1), starts the SAT encoding from scratch at each step of the cycle of minimization; that is unfortunate. This is why an incremental approach has been considered.

The algorithm 3 has been implemented. Starting with an ω-automaton of size n, if the first iteration (which encodes the research of an ω-automaton of size $n-1$) constructs an automaton of k ($k \leq n-1$) states *accessible*, then some clauses are added to forbid all the entrant transitions of the $n - k - 1$ last state. If such automaton is found, the entrant transitions of the $n - k - 2$ last state are also forbidden, etc. The last solved SAT problem corresponds to the minimal automaton.

An interesting thing, as a sideline, is that at the beginning, instead of forbidding the entrant transitions of a state, the outgoing transitions were forbidden. This did not work, because those clauses were in contradiction with the first rule of the encoding (which stated that the automaton must be *complete*), causing an absurdity. All the rules of the encoding are described in the papers [5] and [6].

After this method has been implemented (algorithm 3), a benchmark has been realized to compare it to the old default method (algorithm 1). For display reasons, only a few interesting lines of the results are displayed in the figure 6.5. Feel free to have a look in the A.1.1 section of the appendix to see all the results. For each version and each formula, two minimizations are attempted, büchi *acceptance set* and *generalized büchi acceptance set*. The best performances for each formula are colorized respectively in green and yellow.

Formulas	Time (seconds)			
	Glucose (As before)		Incr Naive	
	minDBA	minDTGBA	minDBA	minDTGBA
$F(a \wedge GFb) \vee (Fc \wedge Fa \wedge F(c \wedge GFb))$	0.02	57.65	0.01	236.36
$XXG(FaUXb)$	25.15	762.74	20.21	(killed)
$(aR(bRFc))WXGb$	(killed)	254.19	(killed)	672.80
$X(\bar{a} \wedge Fa)R(aMFb)$	2.19	46.12	1.7	132.02
$(aRFb)UX\bar{c}$	(killed , $\leq$ 11)	389.87	(killed , $\leq$ 11)	616.70

Figure 6.5: Parts of A.1.1 benchmark results showing some cases where the Old SAT-based minimization is still better

In all the lines of the figure 6.5, **glucose** is still better than our first incremental approach. There is even a case where **naive incr** never ends the minimization and is killed.

In order to better compare both version, this generated figure counts the number of times a version is better than the other with a tolerance of more or less five per cent (5%). That means: roughly equal times are skipped.

DBA			
	glu	incr1	total
glu	-	9	9
incr1	102	-	102

DTGBA			
	glu	incr1	total
glu	-	8	8
incr1	93	-	93

Figure 6.6: Summary of A.1.1 benchmark

The next two figures (6.7 and 6.8) shows two graphs generated with ggplot2 [15] (a graphing package implemented in top of the **R** statistical language). Any point between the two lines passing through the origin is in the tolerance area (more or less 5%). All the bothering points (cases where glucose win) are located in the area over both lines. Obvisouly, the first incremental approach wins when points are located under both lines.

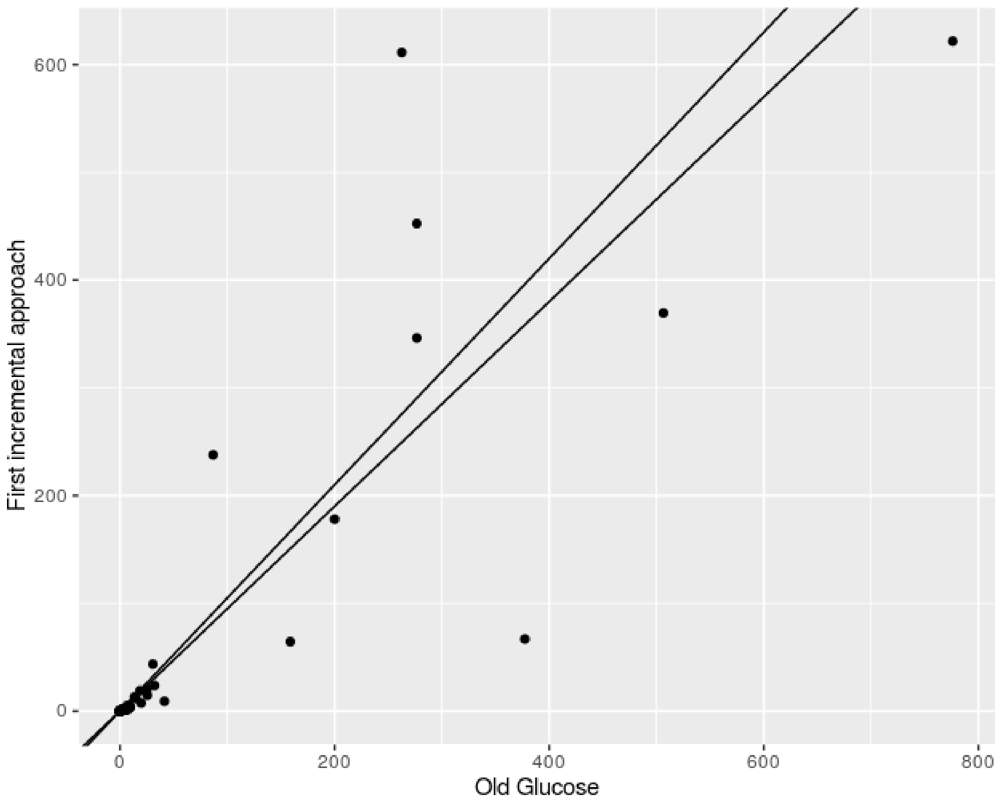


Figure 6.7: Graph comparing both minDBA time of minimization

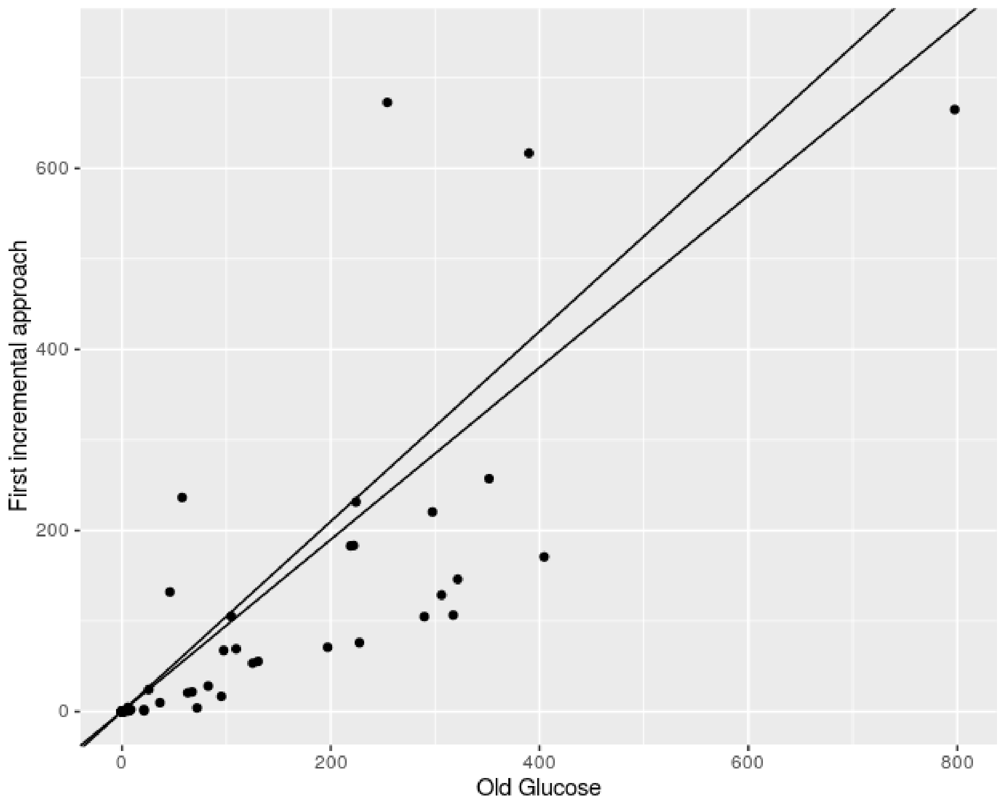


Figure 6.8: Graph comparing both minDTGBA time of minimization

In light of the above, this incremental approach does not seem to be the most appropriate method. This is really surprising. Why all that the SAT solver learned while solving the first traditional encoding did not help it to solve quickly the next similar problems?

A hypothesis raised is that may be the fact that the size of the problem is never decreased affects it? By restarting the encoding from scratch at each step, the Old SAT-based minimization also restarts with a smaller automaton. The smaller the input automaton is, the smaller the SAT problem produced is (with a decreasing number of litterals and clauses).

As SAT competitions also compares SAT solvers in incremental mode, they have defined a file format, called XCNF. As our incremental scenarios seem to offer challenges to SAT solver, we decided to provide a way to generate XCNF file corresponding to complete cycles of this first incremental minimization. We thus hope that SAT solvers contributors will think about how to improve their solver to be efficient in these incremental SAT solving scenarios.

6.4 A not fully incremental approach

With a view to verify the last hypothesis, a similar approach has been proposed. The idea is to provide the opportunity to choose how many times SAT-based minimization should work incrementally before restarting the encoding from scratch. Here times can be seen in two ways. It can be either the number of states to reduce or just a number of attemps before restarting the encoding. The second way has been chosen because gaining many states when only one is expected is unpredictable.

Here is how the algorithm looks like (s means sat_incr_steps and stands for the number of attempts).

Algorithm 4 This incremental approach attemps a traditional encoding and then tries to exclude s states incrementally before restarting the encoding from scratch.

```
 1: procedure ReduceStatesDTGBA($R, m = R$.nb_acc_sets$(), s$)
 2: repeat:
 3:     $n \leftarrow R.nb\_states()$
 4:     $C \leftarrow$ Synthetize$DTGBA(R, n-1, m)$
 5:     if $C$ does not exist then return $R$
 6:     for $i \in \{0, 1, ..., s-1\}$ do
 7:         add clauses to exclude one more state
 8:         $C \leftarrow$ Try to solve the new problem and build the new automaton
 9:         if $C$ does not exist then return $R$
10:         $R \leftarrow C$
```

Again, once this was implemented, a new benchmark was made. Different values for s has been tested: $1, 2, 4$ and 8, 2 was the best value between them. Again, for some display reasons, only this version is displayed in comparaison to the Old SAT-based minimization in the figure 6.9 and in the appendix A.1.2.

Formulas	Time (seconds)			
	Glucose (As before)		Incr Naive	
	minDBA	minDTGBA	minDBA	minDTGBA
$F(a \wedge GFb) \vee (Fc \wedge Fa \wedge F(c \wedge GF\bar{b}))$	0.02	57.65	0.01	237.43
$XXG(FaUXb)$	25.15	762.74	12.48	(killed)
$(aR(bRFc))WXGb$	(killed)	254.19	(killed)	778.38
$G(XXFaU(a \vee b \vee Fc))$	262.56	(killed)	501.21	(killed)
$X(\bar{a} \wedge Fa)R(aMFb)$	2.19	46.12	1.71	129.43
$(aRFb)UX\bar{c}$	(killed , $\leq$ 11)	389.87	(killed , $\leq$ 11)	425.21

Figure 6.9: Parts of A.1.2 benchmark results showing some cases where the Old SAT-based minimization is still better

The figure 6.10 had the same purpose of the figure 6.6, that means with a tolerance of more or less five per cent, counting for minDBA and minDTGBA the number of times each version is better than each of the others for each formula.

DBA							
	glu	incr1	incr2p1	incr2p2	incr2p4	incr2p8	total
glu	-	9	8	8	9	9	43
incr1	102	-	17	9	14	8	150
incr2p1	105	31	-	19	33	31	219
incr2p2	105	33	27	-	31	31	227
incr2p4	104	22	24	14	-	16	180
incr2p8	103	18	19	16	12	-	168

DTGBA							
	glu	incr1	incr2p1	incr2p2	incr2p4	incr2p8	total
glu	-	8	13	13	10	12	56
incr1	93	-	26	15	16	11	161
incr2p1	95	18	-	17	25	20	175
incr2p2	96	21	27	-	26	19	189
incr2p4	95	12	25	13	-	7	152
incr2p8	95	9	26	13	13	-	156

Figure 6.10: Summary of the comparaison between Old SAT-based minimization (Glucose) and the second incremental approach with different values for s : $1, 2, 4$ and 8

In the figure 6.10, it is also noticeable that the new incremental approach (algorithm 4) with $s = 2$ seems to give a better performance than the first approach. It is 33 and 21 times better than **incr1** against 9 and 15 times (in favour of **incr1**) respectively for minDBA and minDTGBA.

Up to this time, SAT solvers assumptions have not been tested. This is the subject of the next section.

6.5 Assumptions approach

In incremental SAT solving, assumptions are propositions that, once assumed, hold only for the next invocation of the solver. After that, all the assumptions expect to be assumed again. The SAT solver, when asked for a solution, consider all the basic clauses and the assumed assumptions. That is interesting because it makes possible to keep the basic rules and mix them with different hypothesis at each call.

In our case, the basic rules are obvisouly the traditional encoding and the hypothesis are the removal of states. For a better understanding let us consider the following figure (6.11). The first area represents the traditional encoding. After that, some new variables are introduced (x_1, x_2, x_3, ...), so that each of them represents an assumption that can be assumed. All the assumptions add some clauses to forbid the entrant transitions of one state and implies the previous assumption except the first one (x_1) that does not have any assumption previously declared.

Areas	File	Some explanations
1	....	Basic clauses
2	x_1 => .... x_1 => .... x_1 =>	x_1 forbid one more state
3	x_2 => .... x_2 => .... x_2 => .... x_2 => x_1	x_2 forbid one more state x_2 implies x_1
4	x_3 => .... x_3 => .... x_3 => .... x_3 => x_2	x_3 forbid one more state x_3 implies x_2
. . .	. . .	. . .

Figure 6.11:

In this way, after a first attempt (just with the basic clauses, no assumptions) succeeds, when:

- x_1 is assumed, it attempts to forbid one more state
- x_2 is assumed, it attempts to forbid two more states (the one it forbids and the one forbidden by x_1)

- x_3 is assumed, it attempts to forbid three more states (the one it forbids and the two forbidden by x_2)
- etc.

As a reminder, the difference between the first two incremental approaches. (6.3 and 6.4) is that the first one never restarts the encoding and the second restarts the encoding depending on a given parameter s. In order to conserve this possibility, the number of assumptions corresponds to this parameter as well. So this is how the algorithm works:

Algorithm 5

```
1: procedure ReduceStatesDTGBA($R, m = R.\text{nb\_acc\_sets}(), s$)
2: repeat:
3:     $n \leftarrow R.nb\_states()$
4:     $C \leftarrow$ Synthetize$DTGBA(R, n-1, m)$
5:     if $C$ does not exist then return $R$
6:     Add $s$ assumptions
7:     Assume the last assumption //which assumes every assumptions
8:     $C \leftarrow$ Try to solve the new problem and build the new automaton
9:     if $C$ does not exist then
10:        return binary_search(R, $R.size() - 1$ as max, 1 as min)
11:    else
12:        $R \leftarrow C$
13:        Go to repeat
```

Another interesting story, is that at the beginning, the first assumption approach was a bit different and did not work at all. When some assumptions have been made and a SAT problem is unsatisfiable (that means there is no solution), some SAT solvers (including PicoSAT) provide a way to ask: is the problem unsatisfiable because of this assumption or this one, etc. ? So, the encoding was exactly the same as the figure 6.11 but at the beginning each of the assumptions were assumed - not all the assumptions through the last one but really each of them. If such automaton is found, the process will restart, otherwise it loops over each assumption, asking the solver is it the one that mislead you? Or is it that one? For reasons still unknown, PicoSAT was not able to identify the assumption that failed. If you are interested, this implementation still exists in the **aga/assumesat1** branch of the Spot project `https://gitlab.lrde.epita.fr/spot/spot/tree/aga/assumesat1`.

The section A.1.3 in the appendix compares this assumption approach to the old SAT-based minimization.

6.6 Memory optimization

During the internship, benchmarks were firstly run on a cluster with Xeon E5-2620 2.00GHz cpu, 12 physical cores and 256 GO DDR3 of RAM memory. Many benchmarks have been ignored because memory was swapping. We were forced to move to a cluster with Intel(R) Xeon(R) CPU E7- 2860 @ 2.27GHz, 20 physical cores and

512 GO DDR3 to continue the experiments.

But this has attracted our attention. Call R the reference automaton (the one to minimize) and C the candidate automaton (the minimal automaton we are searching). As stated in the [5] paper page **7**, the variables used for the SAT encoding can be grouped in three categories:

- basic transitions (of C)
- accepting transitions that encodes the membership of these transitions to an acceptance set (of C)
- paths variable that encodes a path between one state to another in the product automaton ($C \otimes R$)

We realized that we were saving (using maps) some datas that could be retrieved dynamically when needed. The optimizations made relies on two facts:

- almost everything about the candidate automaton is known. Therefore we tried to save only the necessary informations,
- the litterals are continuous (just increased by one at each time).

This leads us to implement a class that will handle the variables. At the beginning, an object of this class is instantiated with some values about the candidate automaton given. This class provides the necessary functions to retrieve any litteral corresponding to a tuple. Here is an approximate UML representation.

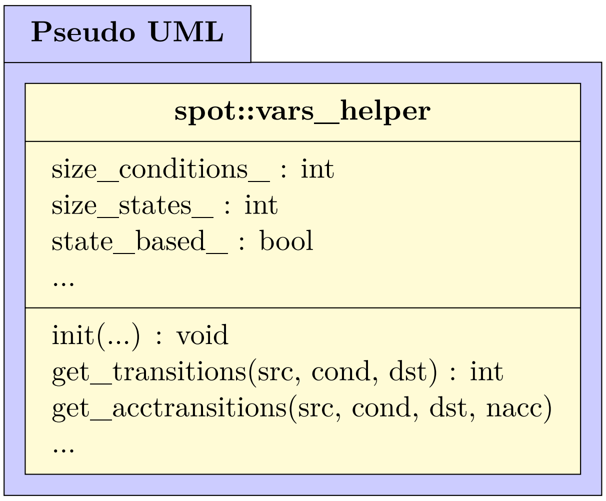


Figure 6.12: Vars_helper class UML representation

Massif [19], a heap memory profiler was used with the aim of better evaluating the memory consumption. For some display reasons, the results are shown in the A.2 section of the appendix. Note that as PicoSAT library consume also data, the current version is compared with the old one in the same conditions: by using **Glucose** as external SAT solver.

For the formula $G(F(!a) \vee (F(b)Uc))$, we now consume 23 Mb instead of 29,9 Mb. Have in mind that this formula starts the minimization with an automaton of 8 states and the more bigger the automaton is, the more memory-hungry SAT-based minimization is.

6.7 Binary search

As said before, the binary search (algorithm 2) implemented before has never been benchmarked. After the memory-usage improvements, it was logical to restart a final benchmark with all the different algorithms we have:

- old default (external Glucose) (algorithm 1),
- binary search (algorithm 2),
- naive incremental (algorithm 3),
- not fully incremental (algorithm 4),
- assumption (algorithm 5)

As a very big surprise, binary search won by litteraly knocking out the others! Again, for display reasons, it is not possible to display all the results. But this benchmark exists and can be reproduced, have a look on **bench/dtgbasat** folder of Spot project: `https://gitlab.lrde.epita.fr/spot/spot/tree/master/bench/dtgbasat`. There is a **README** file explaining how to do so.

However, this is a summary of the final benchmark:

DBA														
	glu	pic	libp	incr1	incr2p1	incr2p2	incr2p4	incr2p8	assp1	assp3	assp5	assp6	assp8	dicho	total
glu	-	19	7	9	8	8	9	9	4	7	9	6	8	3	106
pic	90	-	-	4	2	3	4	4	2	10	10	12	11	5	157
libp	106	107	-	31	18	23	32	29	33	24	29	31	32	42	537
incr1	102	103	53	-	17	9	14	8	35	35	28	33	37	45	519
incr2p1	105	108	60	31	-	19	33	31	46	32	35	35	37	49	621
incr2p2	105	105	66	33	27	-	31	31	43	37	34	35	40	46	633
incr2p4	104	103	54	22	24	14	-	16	37	31	30	30	35	46	546
incr2p8	103	103	56	18	19	16	12	-	36	33	34	34	37	46	547
assp1	109	104	60	47	45	38	47	46	-	40	27	36	40	40	679
assp3	106	99	63	56	51	52	53	51	51	-	27	27	31	41	708
assp5	103	99	62	59	56	50	55	56	45	36	-	30	40	42	733
assp6	104	98	66	60	57	54	60	60	48	39	29	-	30	44	749
assp8	102	102	63	55	50	49	54	52	43	32	25	27	-	44	698
dicho	113	106	63	55	55	52	58	56	60	65	59	61	62	-	865

DTGBA														
	glu	pic	libp	incr1	incr2p1	incr2p2	incr2p4	incr2p8	assp1	assp3	assp5	assp6	assp8	dicho	total
glu	-	19	9	8	13	13	10	12	11	9	8	8	7	10	137
pic	79	-	-	7	9	8	7	9	8	8	8	5	6	7	161
libp	95	98	-	22	21	19	25	22	23	17	16	11	17	32	418
incr1	93	96	54	-	26	15	16	11	31	18	15	10	16	33	434
incr2p1	95	94	55	18	-	17	25	20	31	16	14	10	18	31	444
incr2p2	96	96	60	21	27	-	26	19	30	16	17	12	17	33	470
incr2p4	95	95	51	12	25	13	-	7	29	14	12	10	15	36	414
incr2p8	95	95	55	9	26	13	13	-	29	17	13	12	17	33	427
assp1	93	93	59	37	48	42	44	45	-	16	15	14	19	33	558
assp3	97	98	76	63	66	68	68	63	59	-	20	13	26	38	755
assp5	97	99	76	65	65	68	72	67	65	33	-	15	39	35	796
assp6	100	101	78	73	73	71	76	75	68	37	26	-	36	41	855
assp8	101	101	73	70	71	68	75	73	62	34	27	21	-	35	811
dicho	101	103	69	64	67	66	66	65	67	65	64	63	68	-	928

Figure 6.13: Summary of the final benchmark

The section A.1.4 in the appendix shows an excerpt of this benchmark and compares this method to the Old default SAT-based minimization. In conclusion, binary search was selected as the default SAT-based minimization algorithm.

6.8 Language map

language_map is a new algorithm that identifies states that recognize the same language. It takes in input an ω-automaton and outputs a vector of integer that has exactly the same size as the automaton. The number of different values (ignoring occurences) in the vector is the total number of recognized languages. States recognizing the same language have the same value.
The algorithm 6 shows how it works. Note that R_x represents the same automaton

Algorithm 6

```
1: procedure LANGUAGE_MAP($R$)
2:     $n \leftarrow R.nb\_states()$
3:     for $i \in \{1, 2, ..., n-1\}$ do
4:         for $j \in \{0, 1, ..., i-1\}$ do
5:             if $L(R_i) \cap L(\overline{R_j}) \neq \emptyset \wedge L(\overline{R_i}) \cap L(R_j) \neq \emptyset$ then
6:                 $R_i$ and $R_j$ recognize the same language
```

as R but with the initial state set to state x, $\overline{R}$ is the complement of R and $L(R)$ is the language recognized by R.

Two automaton R_i and R_j recognizes the same language i.e. $L(R_i) = L(R_j)$ if and only if $L(R_i) \subseteq L(R_j)$ and $L(R_i) \supseteq L(R_j)$. This is checked by the line number 5 in algorithm 6.

To vizualize language_map output, highlight_languages method has also been implemented. It takes in input an ω-automaton and the associated language_map output and colorizes the states.

To come back to SAT-based minimization, by default, the binary search algorithm starts with 1 as **min** value. Our theory is that the minimal automaton cannot have less states than the total number of languages recognized by each state. Instead of using 1 for **min** value, we set it to the total number of recognized languages. This worked, it improved in general binary search algorithm but in some cases, which are far from negligible, the default binary search method was still better. This leads to a new command line option: **sat-langmap**. This option is not set by default.

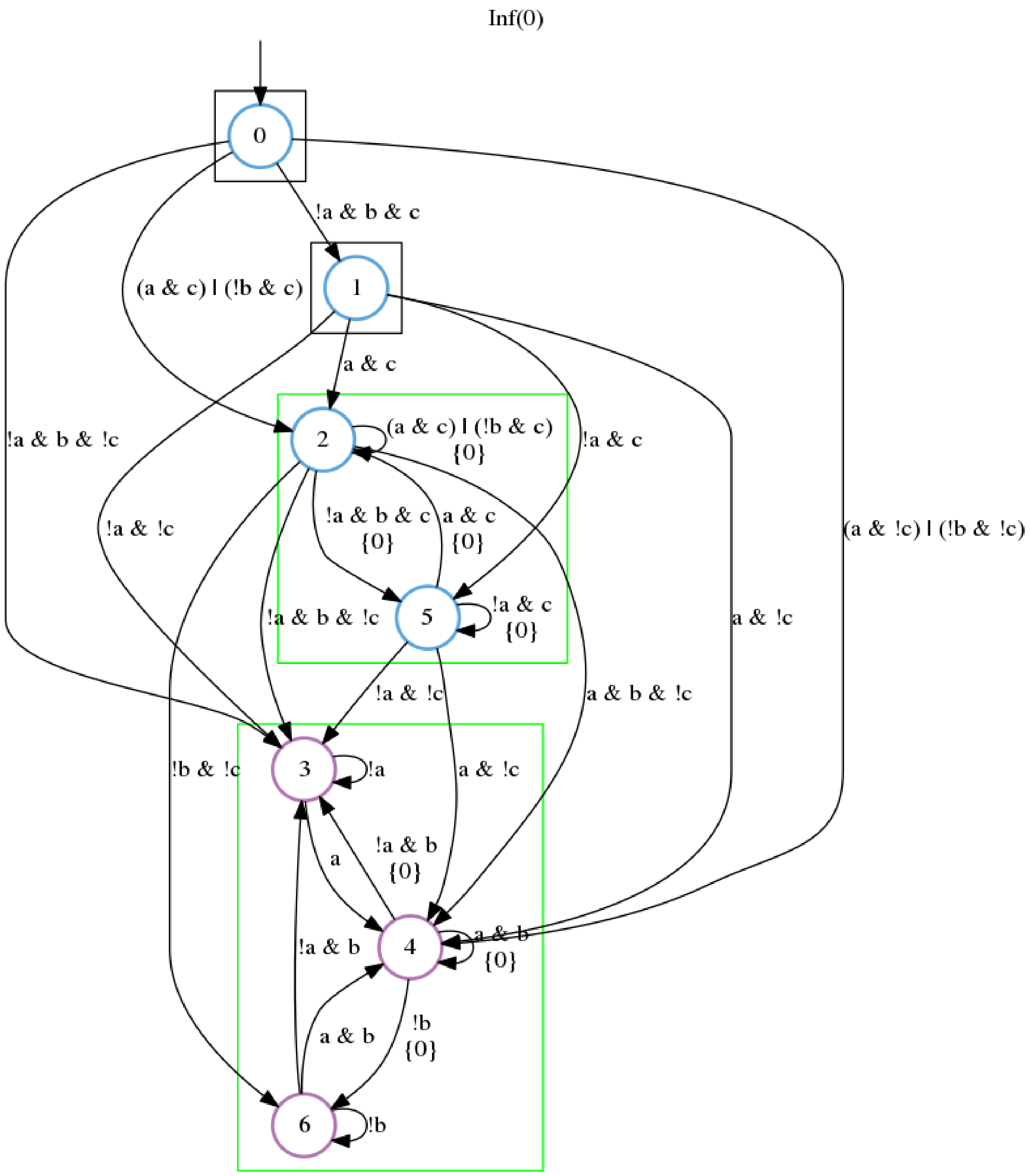


Figure 6.14: An ω-automaton colorized using **--hightlight-language** new option of **autfilt**

Chapter 7

Bibliography

References

[1] A.Duret-Lutz. “LTL translation improvements in Spot 1.0”. In: *Int. J. on Critical Computer-Based Systems* (2014), pp. 31–54.

[2] A.Duret-Lutz et al. “Spot 2.0 — a framework for LTL and omega-Automata manipulation”. In: (2016).

[3] A.Duret-Lutz A.E.Ben Salem and F.Kordon. “Model checking using generalized testing automata”. In: *Transactions on Petri Nets and Other Models of Concurrency (ToPNoC V1)* (2012), pp. 94–112.

[4] *Automata Theory*. URL: `https://en.wikipedia.org/wiki/Automata_theory`.

[5] Souheib Baarir and A.Duret-Lutz. “Mechanizing the Minimization of Deterministic Generalized Büchi Automata”. In: *FORTE* vol. 8461 (2014), pp. 266–283. URL: `https://www.lrde.epita.fr/~adl/dl/adl/baarir.14.forte.pdf`.

[6] Souheib Baarir and A.Duret-Lutz. “SAT-based Minimization of Deterministic omega-Automata”. In: *LPAR* (2015). URL: `https://www.lrde.epita.fr/~adl/dl/adl/baarir.15.lpar.pdf`.

[7] *Concepts*. URL: `https://spot.lrde.epita.fr/concepts.html`.

[8] *CryptoMiniSat*. URL: `https://github.com/msoos/cryptominisat/releases/tag/5.0.1`.

[9] *DIMACS format*. URL: `http://www.satcompetition.org/2009/format-benchmarks2009.html`.

[10] A. Duret-Lutz. “Manipulating LTL formulas using Spot 1.0”. In: *ATVA* vol. 8172.13 (Springer, 2013), pp. 442–445.

[11] A. Duret-Lutz and D.Poitrenaud. “SPOT: an Extensible Model Checking Library using Transition-based Generalized Büchi Automata”. In: *MASCOTS* 04 (2004), pp. 76–83.

[12] Rüdiger Ehlers. *Minimising deterministic Büchi automata precisely using SAT solving*. 2010.

[13] *fastSAT*. URL: `https://github.com/shumpaga/fastSAT`.

[14] F.Pérez and B.E.Granger. “IPython: a system for interactive scientific computing”. In: *Computing in Science and Engineering* (21-29, 2007). URL: http://ipython.org/.

[15] *ggplot2*. URL: http://ggplot2.org/.

[16] *Glucose*. URL: http://www.labri.fr/perso/lsimon/glucose/.

[17] *Google benchmark*. URL: https://github.com/google/benchmark.

[18] *Linear-time Temporal Logic (LTL)*. URL: https://spot.lrde.epita.fr/concepts.html#ltl.

[19] *Massif: a heap profiler*. URL: http://valgrind.org/docs/manual/ms-manual.html.

[20] *PicoSAT*. URL: http://fmv.jku.at/picosat/.

[21] *SAT-solving in practice*. URL: http://www.cse.chalmers.se/edu/year/2012/course/TDA956/Papers/satFinal.pdf.

[22] T.Babiak et al. “The Hanoi Omega-Automata format”. In: *CAV* vol. 9206.15 (Springer, 2015). URL: http://adl.github.io/hoaf/.

[23] *Turing Machine*. URL: https://en.wikipedia.org/wiki/Turing_machine.

Chapter 8

Conclusion

This internship of 5 months (from the 29th of August 2016 to the 15th of January 2017) was a very fascinating experience. As the last internship of the common core program before our specialization, its objective is to help us choosing the best academic path for the next phases.

For me, these 5 months of working have met my expectations. Working in a scientific laboratory, being in contact with computer researchers, learning from them, understanding how a laboratory works, attending seminars organized by the laboratory and taught by some external professionals, in short, being in such an environment helped me picturing with more precision how would be my future experiences as a research student.

It was an enriching internship as I have learned a lot from it. Indeed, this enabled me to understand the slight errors I was used to making while programming in C++, to learn from the existing efficient and clean code, to get valuable feedback from my supervisor, to have a better handling of Python and Shell programming. I also learned about some languages as R, perl that was new to me. Working in such a big project made me master some good behaviours while coding, especially how to make a better use of Git, which is really important when numerous people are working on a specific project at the same time. It also helped me to be more rigorous, consistant in my work, concerned about the user experience, the clarity of the code, the strength of the tests.

Obviously it also taught me a lot about the ω-automata and theoritical concepts used in Spot. Before this internship, I never heard about satisfiability problem, the existence of SAT solvers. Now, even if I do not know how they are implemented, I understand in which cases they can be really useful and how to use them.

Reciprocally, Spot SAT-based minimization has been improved. It has acquired some new algorithms, which have stirred up some new questions or axis of research. For instance, why incremental SAT solving did not meet all our expectations? However, Spot now distributes its own SAT solver, PicoSAT. Its SAT-based minimization is definitely faster than before and less memory-hungry.

This internship was a really positive experience that was concluded by my acceptance at the laboratory as a CSI student. I will be continuing to contribute on the Spot project and we have already discussed about my next subject, we want Spot to support some new algorithms concerning co-Büchi automata that have been described in some papers.

Acknowledgements

I would first like to thank my colleagues from my internship at LRDE for their wonderful collaboration. You supported me greatly and were always willing to help me. I would particularly like to single out my supervisor, Mr. Alexandre Duret-Lutz. I want to thank you for your excellent training, tips and tricks.

In addition, I would like to thank Mr. Hugues Mehou and Mr. Prodil Houanhou for their careful rereading and constructive comments. You definitely helped me to write a good report.

I would also like to thank my parents for their wise counsel and sympathetic ear. You are always there for me.

Thank you very much, everyone!

Part II

Appendix

Contents (Appendix)

Appendix A

Gross results

A.1 Benchmarks

Any benchmark added in this section follows the same chart legend:

- Column **type** shows how the initial det. aut. was obtained: T = translation produces DTGBA; P = powerset construction transforms TBA to DTBA; R = DRA to DBA.
- Column **C.** tells whether the output automaton is complete: rejecting sink states are always omitted (add 1 state when C=0 if you want the size of the complete automaton).
- For each formula, green columns correspond to best minDBA time and yellow columns to best minDTGBA time.

A.1.1 First incremental approach

The following table shows the results of the benchmark that compared the old default SAT-based minimization (using Glucose) to the first incremental approach (using PicoSAT library):

														Glucose (As before)							Incr Naive						
				DRA	DTGBA				DBA			DBAminimizer		minDBA			minDTGBA				minDBA			minDTGBA			
formula	m	type	C.	st.	st.	tr.	acc.	time	st.	tr.	time	st.	time	st.	tr.	time	st.	tr.	acc.	time	st.	tr.	time	st.	tr.	acc.	time
$G(a \vee Fb)$	1	T	1	4	2	8	1	0.00	2	8	0.00	2	0.11	2	8	0.0	2	8	1	0.0	2	8	0.0	2	8	1	0.0
$GF(a \vee b)$	1	T	1	3	1	4	1	0.00	2	8	0.00	2	0.06	2	8	0.0	1	4	1	0.0	2	8	0.0	1	4	1	0.0
$G(a \vee Fa)$	1	T	1	3	1	2	1	0.00	2	4	0.00	2	0.06	2	4	0.0	1	2	1	0.0	2	4	0.0	1	2	1	0.0
$G(a \vee F(b \vee c))$	1	T	1	4	2	16	1	0.00	2	16	0.00	2	0.06	2	16	0.0	2	16	1	0.0	2	16	0.0	2	16	1	0.0
$G(a \vee F(b \wedge c))$	1	T	1	4	2	16	1	0.00	2	16	0.00	2	0.06	2	16	0.0	2	16	1	0.0	2	16	0.0	2	16	1	0.0
$GF(a \vee b) \wedge GF(b \vee c)$	2	T	1	5	1	8	2	0.00	3	24	0.00	3	0.06	3	24	0.0	1	8	2	0.0	3	24	0.0	1	8	2	0.0
$G(Fb \wedge Fa)$	2	T	1	5	1	4	2	0.00	3	12	0.00	3	0.06	3	12	0.0	1	4	2	0.0	3	12	0.0	1	4	2	0.0
$Ga \vee Gc \vee (G(a \vee GFb) \wedge G(c \vee GF\bar{b}))$	2	P	1		6	48	1	0.01	8	64	0.01			6	48	0.52	4	32	2	1.63	6	48	0.13	4	32	2	0.2
$Ga \vee Gc \vee (G(a \vee GFb) \wedge G(c \vee GF\bar{b}))$	2	R	1	10	10	80	1	0.00	10	80	0.00	6	6.81	6	48	2.05	4	32	2	36.46	6	48	0.17	4	32	2	9.72
$F(a \wedge GFb) \vee (Fc \wedge Fa \wedge F(c \wedge GF\bar{b}))$	6	P	1		4	32	1	0.01	5	40	0.01			5	40	0.02	4	32	1	21.25	5	40	0.01	4	32	1	0.71
$F(a \wedge GFb) \vee (Fc \wedge Fa \wedge F(c \wedge GF\bar{b}))$	6	R	1	5	5	40	1	0.00	5	40	0.00	5	0.15	5	40	0.02	4	32	1	57.65	5	40	0.01	4	32	1	236.36
$GF(Fa \vee GFb \vee FG(a \vee b))$	2	P	1		2	8	1	0.00	4	16	0.00			2	8	0.0	1	4	1	0.0	2	8	0.0	1	4	1	0.0
$GF(Fa \vee GFb \vee FG(a \vee b))$	2	R	1	3	3	12	1	0.00	3	12	0.00	2	0.38	2	8	0.0	1	4	1	0.03	2	8	0.0	1	4	1	0.01
$FG(Fa \vee GFb \vee FG(a \vee b))$	2	P	1		2	8	1	0.00	4	16	0.00			2	8	0.0	1	4	1	0.0	2	8	0.0	1	4	1	0.0
$FG(Fa \vee GFb \vee FG(a \vee b))$	2	R	1	3	3	12	1	0.00	3	12	0.00	2	0.38	2	8	0.0	1	4	1	0.03	2	8	0.0	1	4	1	0.01
$FG(Fb \vee FGa \vee GFa \vee FG(a \vee b))$	2	P	1		2	8	1	0.00	4	16	0.00			2	8	0.0	1	4	1	0.0	2	8	0.0	1	4	1	0.0
$FG(Fb \vee FGa \vee GFa \vee FG(a \vee b))$	2	R	1	3	3	12	1	0.00	3	12	0.00	2	0.38	2	8	0.0	1	4	1	0.03	2	8	0.0	1	4	1	0.01
$GF(a \to XXXb)$	1	T	1	4	1	4	1	0.00	2	8	0.00	2	0.09	2	8	0.0	1	4	1	0.0	2	8	0.0	1	4	1	0.0
$G(a \to Fb) \wedge G(c \to Fd)$	2	T	1	9	4	64	2	0.00	6	96	0.00	6	0.20	6	96	0.60	4	64	2	4.83	6	96	0.13	4	64	2	0.44
$GFb \wedge GFa$	2	T	1	5	1	4	2	0.00	3	12	0.00	3	0.06	3	12	0.0	1	4	2	0.0	3	12	0.0	1	4	2	0.0
$GFc \vee GFb \vee GFa$	1	T	1	3	1	8	1	0.00	2	16	0.00	2	0.06	2	16	0.0	1	8	1	0.0	2	16	0.0	1	8	1	0.0
GFa	1	T	1	3	1	2	1	0.00	2	4	0.00	2	0.06	2	4	0.0	1	2	1	0.0	2	4	0.0	1	2	1	0.0
$Ga \wedge G(b \to Fc)$	1	T	0	4	2	8	1	0.00	2	8	0.00	2	0.06	2	8	0.0	2	8	1	0.0	2	8	0.0	2	8	1	0.0
$G(a \to Fb) \wedge G(b \to Fc)$	2	T	1	7	4	32	2	0.00	5	40	0.00	5	0.08	5	40	0.07	3	24	2	2.12	5	40	0.02	3	24	2	0.2
$G(a \to Fb) \wedge G(\bar{a} \to F\bar{b})$	2	T	1	6	3	12	2	0.00	4	16	0.00	4	0.06	4	16	0.02	3	12	2	0.09	4	16	0.01	3	12	2	0.02
$GFc \wedge GFb \wedge GFa \wedge GFd$	4	T	1	9	1	16	4	0.01	5	80	0.01	5	0.75	5	80	0.53	1	16	4	0.01	5	80	0.33	1	16	4	0.01
$GF(a \leftrightarrow XXXb)$	1	P	1		15	60	1	0.08	23	92	0.08			(killed , $\leq$ 17)			(killed , $\leq$ 8)				(killed , $\leq$ 11)			(killed , $\leq$ 8)			
$GF(a \leftrightarrow XXXb)$	1	R	1	17	17	68	1	0.00	17	68	0.00	(killed)		(killed , $\leq$ 11)			(killed , $\leq$ 8)				(killed , $\leq$ 11)			(killed , $\leq$ 8)			
$G(a \to (bUc))$	1	T	0	4	2	13	1	0.00	2	13	0.00	2	0.06	2	13	0.0	2	13	1	0.0	2	13	0.0	2	13	1	0.0
$GF(a \leftrightarrow XXb)$	1	P	1		7	28	1	0.01	11	44	0.01			6	24	6.53	4	16	1	0.11	6	24	4.88	4	16	1	0.03
$GF(a \leftrightarrow XXb)$	1	R	1	9	9	36	1	0.00	9	36	0.00	6	2.21	6	24	3.21	4	16	1	0.92	6	24	2.08	4	16	1	0.11
$G(a \to Fb)$	1	T	1	4	2	8	1	0.00	2	8	0.00	2	0.06	2	8	0.0	2	8	1	0.0	2	8	0.0	2	8	1	0.0
$G(aU(bU(\bar{a}U\bar{b})))$	1	T	1	3	1	2	1	0.00	2	4	0.00	2	0.06	2	4	0.0	1	2	1	0.0	2	4	0.0	1	2	1	0.0
$X((aMF((b \wedge c) \vee (\bar{b} \wedge \bar{c})))W(G\bar{c}Ub))$	3	P	1		13	104	1	0.01	16	128	0.01			(killed , $\leq$ 12)			(killed , $\leq$ 10)				(killed , $\leq$ 12)			(killed , $\leq$ 10)			
$X((aMF((b \wedge c) \vee (\bar{b} \wedge \bar{c})))W(G\bar{c}Ub))$	3	R	1	22	22	176	1	0.00	22	176	0.00	(killed)		(killed , $\leq$ 12)			(killed)				(killed , $\leq$ 12)			(killed)			
$X(((a \wedge b)R(\bar{a}U\bar{c}))Rb)$	1	P	0		5	33	1	0.00	6	39	0.00			6	39	0.08	5	33	1	0.03	6	39	0.06	5	33	1	0.01
$X(((a \wedge b)R(\bar{a}U\bar{c}))Rb)$	1	R	0	8	9	55	1	0.00	9	55	0.00	6	1.10	6	39	0.53	5	33	1	0.57	6	39	0.2	5	33	1	0.11
$XXG(FaUXb)$	2	P	1		9	36	1	0.01	13	52	0.01			8	32	9.44	5	20	2	71.94	8	32	3.57	5	20	2	4.09
$XXG(FaUXb)$	2	R	1	18	18	72	1	0.00	18	72	0.00	8	80.69	8	32	25.15	5	20	2	762.74	8	32	20.21	(killed)			
$(\bar{a}M\bar{b})WF\bar{c}$	2	P	1		5	40	1	0.00	6	48	0.00			4	32	0.04	3	24	1	0.21	4	32	0.01	3	24	1	0.04
$(\bar{a}M\bar{b})WF\bar{c}$	2	R	1	7	7	56	1	0.00	7	56	0.00	4	0.10	4	32	0.04	3	24	1	1.76	4	32	0.01	3	24	1	0.18
$(a \wedge Fc \wedge GFb)Rc$	2	P	0		3	20	1	0.00	4	28	0.00			4	28	0.02	3	20	1	0.06	4	28	0.01	3	20	1	0.02
$(a \wedge Fc \wedge GFb)Rc$	2	R	0	4	5	32	1	0.00	5	32	0.00	4	0.16	4	28	0.06	3	20	1	1.36	4	28	0.01	3	20	1	0.15
$(aR(bWa))WG(\bar{a}M(b \vee c))$	1	P	0		2	13	1	0.00	3	19	0.00			3	19	0.01	2	13	1	0.0	3	19	0.01	2	13	1	0.0
$(aR(bWa))WG(\bar{a}M(b \vee c))$	1	R	0	3	4	26	1	0.00	4	26	0.00	3	0.17	3	19	0.03	2	13	1	0.03	3	19	0.01	2	13	1	0.01
$(FaWb)R(\bar{a} \vee Fc)$	2	P	1		11	88	1	0.01	12	96	0.01			8	64	4.82	8	64	2	82.70	8	64	2.4	8	64	1	28.17
$(FaWb)R(\bar{a} \vee Fc)$	2	R	1	8	8	64	1	0.00	8	64	0.00	8	0.83	8	64	0.74	8	64	1	5.76	8	64	0.44	8	64	1	4.67
$X(G(\bar{a}M\bar{b}) \vee G(a \vee G\bar{a}))$	1	P	0		7	23	1	0.00	8	25	0.00			7	21	0.56	6	19	1	0.11	7	21	0.29	6	19	1	0.05
$X(G(\bar{a}M\bar{b}) \vee G(a \vee G\bar{a}))$	1	R	0	12	13	40	1	0.00	13	40	0.00	7	1.32	7	21	1.84	6	19	1	1.42	7	21	0.53	6	19	1	0.68
$FaWGb$	1	P	1		2	8	1	0.00	3	12	0.00			2	8	0.0	2	8	1	0.0	2	8	0.0	2	8	1	0.0
$FaWGb$	1	R	1	3	3	12	1	0.00	3	12	0.00	2	0.07	2	8	0.0	2	8	1	0.0	2	8	0.0	2	8	1	0.0
$Ga \vee GFb$	1	P	1		3	12	1	0.00	4	16	0.00			3	12	0.0	2	8	1	0.0	3	12	0.0	2	8	1	0.0
$Ga \vee GFb$	1	R	1	4	4	16	1	0.00	4	16	0.00	3	0.07	3	12	0.0	2	8	1	0.0	3	12	0.0	2	8	1	0.0
$aMG(F\bar{b} \vee X\bar{a})$	2	P	1		5	20	1	0.00	6	24	0.00			5	20	0.06	5	20	1	0.45	5	20	0.02	5	20	1	0.08
$aMG(F\bar{b} \vee X\bar{a})$	2	R	1	7	7	28	1	0.00	7	28	0.00	5	0.12	5	20	0.12	5	20	1	5.27	5	20	0.03	5	20	1	0.66
$G\bar{a}RXFb$	1	P	1		4	16	1	0.00	6	24	0.00			5	20	0.09	3	12	1	0.0	5	20	0.02	3	12	1	0.0
$G\bar{a}RXFb$	1	R	1	12	12	48	1	0.00	12	48	0.00	5	8.51	5	20	4.28	3	12	1	3.65	5	20	1.08	3	12	1	0.89
$XF(\bar{a} \vee GFb)$	2	P	1		4	16	1	0.00	5	20	0.00			4	16	0.02	3	12	1	0.03	4	16	0.01	3	12	1	0.01
$XF(\bar{a} \vee GFb)$	2	R	1	7	7	28	1	0.00	7	28	0.00	4	0.13	4	16	0.09	3	12	1	1.98	4	16	0.02	3	12	1	0.25
$G(F\bar{a}U\bar{a})UXa$	2	P	1		6	12	1	0.00	7	14	0.00			6	12	0.04	5	10	1	0.10	6	12	0.02	5	10	1	0.04
$G(F\bar{a}U\bar{a})UXa$	2	R	1	8	8	16	1	0.00	8	16	0.00	6	0.24	6	12	0.11	5	10	1	3.36	6	12	0.02	5	10	1	0.48
$(a \vee G(aM\bar{b}))WFc$	2	P	1		5	40	1	0.00	6	48	0.00			5	40	0.05	4	32	1	0.15	5	40	0.02	4	32	1	0.03
$(a \vee G(aM\bar{b}))WFc$	2	R	1	6	6	48	1	0.00	6	48	0.00	5	0.08	5	40	0.04	4	32	1	0.46	5	40	0.01	4	32	1	0.13
$FaWXb$	1	P	1		6	24	1	0.00	7	28	0.00			6	24	0.06	6	24	1	0.02	6	24	0.02	6	24	1	0.01
$FaWXb$	1	R	1	7	7	28	1	0.00	7	28	0.00	6	0.19	6	24	0.10	6	24	1	0.08	6	24	0.02	6	24	1	0.02
$X(aR((\bar{b} \wedge F\bar{c})MX\bar{a}))$	2	P	0		10	56	1	0.01	12	64	0.01			(killed , $\leq$ 11)			8	48	2	25.58	(killed , $\leq$ 11)			8	48	2	24.13
$X(aR((\bar{b} \wedge F\bar{c})MX\bar{a}))$	2	R	0	12	13	68	1	0.00	13	68	0.00	(killed)		(killed , $\leq$ 11)			8	48	2	222.00	(killed , $\leq$ 11)			8	48	2	183.29
$XG\bar{a}RFb$	1	P	1		2	8	1	0.00	3	12	0.00			2	8	0.0	2	8	1	0.0	2	8	0.0	2	8	1	0.0
$XG\bar{a}RFb$	1	R	1	6	6	24	1	0.00	6	24	0.00	2	0.24	2	8	0.06	2	8	1	0.06	2	8	0.02	2	8	1	0.02
$GFa \vee (b \wedge Fc)$	2	P	1		4	32	1	0.00	6	48	0.00			6	48	0.04	4	32	1	0.06	6	48	0.02	4	32	1	0.02
$GFa \vee (b \wedge Fc)$	2	R	1	6	6	48	1	0.00	6	48	0.00	6	0.12	6	48	0.04	4	32	1	2.33	6	48	0.02	4	32	1	0.21
$X(aR(FbRF\bar{b}))$	2	P	1		6	24	1	0.00	7	28	0.00			5	20	0.06	4	16	1	0.21	5	20	0.01	4	16	1	0.06
$X(aR(FbRF\bar{b}))$	2	R	1	9	9	36	1	0.00	9	36	0.00	5	0.21	5	20	0.11	4	16	1	6.87	5	20	0.02	4	16	1	0.84
$G(XaMFa)$	2	P	1		2	4	1	0.00	3	6	0.00			2	4	0.0	1	2	1	0.0	2	4	0.0	1	2	1	0.0
$G(XaMFa)$	2	R	1	4	4	8	1	0.00	4	8	0.00	2	0.09	2	4	0.0	1	2	1	0.20	2	4	0.0	1	2	1	0.03
$X(Ga \vee GFb)$	1	P	1		4	16	1	0.00	5	20	0.00			4	16	0.02	3	12	1	0.0	4	16	0.01	3	12	1	0.0
$X(Ga \vee GFb)$	1	R	1	9	9	36	1	0.00	9	36	0.00	4	0.68	4	16	0.35	3	12	1	0.26	4	16	0.06	3	12	1	0.06
$X(Ga \vee XG((\bar{b} \wedge F\bar{c}) \vee (b \wedge Gc)))$	1	P	0		9	50	1	0.00	10	54	0.00			9	50	377.41	9	50	1	306.26	9	50	66.81	9	50	1	128.67
$X(Ga \vee XG((\bar{b} \wedge F\bar{c}) \vee (b \wedge Gc)))$	1	R	0	11	12	67	1	0.00	12	67	0.00	9	357.33	9	50	86.87	9	50	1	404.49	9	50	237.87	9	50	1	170.96
$GaRFb$	1	P	1		2	8	1	0.00	3	12	0.00			3	12	0.0	2	8	1	0.0	3	12	0.0	2	8	1	0.0
$GaRFb$	1	R	1	4	4	16	1	0.00	4	16	0.00	3	0.07	3	12	0.0	2	8	1	0.0	3	12	0.0	2	8	1	0.0

														Glucose (As before)								Incr Naive						
				DRA	DTGBA				DBA			DBAminimizer		minDBA			minDTGBA				minDBA			minDTGBA			
formula	m	type	C.	st.	st.	tr.	acc.	time	st.	tr.	time	st.	time	st.	tr.	time	st.	tr.	acc.	time	st.	tr.	time	st.	tr.	acc.	time
$G(aU(b \lor X((\bar{a} \land \bar{c}) \lor (a \land c))))$	1	P	0		3	20	1	0.00	4	28	0.00			3	20	0.05	2	12	1	0.02	3	20	0.01	2	12	1	0.01
$G(aU(b \lor X((\bar{a} \land \bar{c}) \lor (a \land c))))$	1	R	0	5	6	40	1	0.00	6	40	0.00	3	0.31	3	20	0.32	2	12	1	0.46	3	20	0.07	2	12	1	0.07
$XG((G\bar{a} \land F\bar{b}) \lor (Fa \land (a \lor Gb)))$	2	P	0		5	15	1	0.00	7	19	0.00			7	19	0.26	5	15	1	0.14	7	19	0.81	5	15	1	0.07
$XG((G\bar{a} \land F\bar{b}) \lor (Fa \land (a \lor Gb)))$	2	R	0	12	13	34	1	0.00	13	34	0.00	7	3.67	7	19	1.92	5	15	1	95.24	7	19	1.17	5	15	1	16.65
$(aUX\bar{a}) \lor XG(\bar{b} \land XFc)$	2	P	0		10	74	1	0.00	12	86	0.00			8	54	6.91	6	42	1	8.65	8	54	2.72	6	42	1	2.1
$(aUX\bar{a}) \lor XG(\bar{b} \land XFc)$	2	R	0	12	13	86	1	0.00	13	86	0.00	8	14.61	8	54	7.96	6	42	1	227.55	8	54	5.2	6	42	1	76.2
$X(G\bar{a} \lor GFa)$	1	P	1		4	8	1	0.00	5	10	0.00			4	8	0.0	3	6	1	0.0	4	8	0.0	3	6	1	0.0
$X(G\bar{a} \lor GFa)$	1	R	1	9	9	18	1	0.00	9	18	0.00	4	0.11	4	8	0.19	3	6	1	0.12	4	8	0.03	3	6	1	0.03
$G(G\bar{a} \lor G\bar{b} \lor F\bar{c})$	1	P	1		4	32	1	0.02	5	40	0.02			4	32	0.10	4	32	1	0.04	4	32	0.04	4	32	1	0.03
$G(G\bar{a} \lor G\bar{b} \lor F\bar{c})$	1	R	1	7	7	56	1	0.00	7	56	0.00	4	0.10	4	32	0.15	4	32	1	0.19	4	32	0.04	4	32	1	0.03
$X(FaWb)RFc$	2	R	1	15	15	120	1	0.00	15	120	0.00	7	79.67	7	56	9.41	7	56	1	224.11	7	56	3.82	7	56	1	224.11
$\bar{a} \land ((FbUa)WXc)$	2	R	0	19	20	156	1	0.00	20	156	0.00		(killed)	(killed , $\leq$ 15)			(killed , $\leq$ 16)				(killed , $\leq$ 15)			(killed)			
$G(F\bar{a} \lor (FbUc))$	3	P	1		11	88	1	0.22	17	136	0.22			4	32	32.07	2	16	2	351.80	4	32	23.71	2	16	2	257.29
$G(F\bar{a} \lor (FbUc))$	3	R	1	23	23	184	1	0.00	23	184	0.00		(killed)	4	32	200.07	(killed)				4	32	178.09	(killed)			
$(aR(bRFc))WXGb$	1	R	1	24	24	192	1	0.00	24	192	0.00		(killed)	(killed , $\leq$ 11)			8	64	1	254.19	(killed , $\leq$ 11)			8	64	1	672.80
$X((Fa \land XFb)RXFc)$	3	R	1	34	34	272	1	0.00	34	272	0.00		(killed)	(killed , $\leq$ 13)			(killed)				(killed , $\leq$ 13)			(killed)			
$(GaR(F\bar{b}Uc))Wc$	2	P	1		12	96	1	0.01	16	128	0.01			9	72	158.67	7	56	2	219.00	9	72	64.41	7	56	2	182.89
$(GaR(F\bar{b}Uc))Wc$	2	R	1	10	10	80	1	0.00	10	80	0.00	9	9.01	9	72	25.43	7	56	2	130.60	9	72	14.63	7	56	2	55.5
$X(\bar{a} \lor G(a \land \bar{b}))RF(c \land Fb)$	2	P	1		10	80	1	0.20	12	96	0.20			(killed , $\leq$ 11)			9	72	2	125.27	(killed , $\leq$ 11)			9	72	2	53.34
$X(\bar{a} \lor G(a \land \bar{b}))RF(c \land Fb)$	2	R	1	18	18	144	1	0.00	18	144	0.00		(killed)	(killed , $\leq$ 11)			(killed , $\leq$ 16)				(killed , $\leq$ 11)			(killed)			
$G(Fb \lor Ga \lor XXF\bar{c})$	2	R	1	44	44	352	1	0.01	44	352	0.01		(killed)	(killed)			(killed)				(killed)			(killed)			
$G(F(\bar{a} \land Fa)U(bUXc))$	3	R	1	35	35	280	1	0.00	35	280	0.00		(killed)	(killed , $\leq$ 27)			(killed)				(killed)			(killed)			
$G(F\bar{a}UX(F\bar{b} \land Xb))$	3	R	1	37	37	148	1	0.00	37	148	0.00		(killed)	4	16	776.24	(killed)				4	16	621.63	(killed)			
$G(XXFaU(a \lor b \lor Fc))$	3	P	1		10	80	1	0.13	15	120	0.13			4	32	13.16	3	24	3	317.30	4	32	13.16	3	24	1	106.52
$G(XXFaU(a \lor b \lor Fc))$	3	R	1	26	26	208	1	0.00	26	208	0.00		(killed)	4	32	262.56	(killed)				4	32	611.34	(killed)			
$G(a \lor F\bar{b} \lor (cUXc))$	2	P	1		4	32	1	0.02	5	40	0.02			4	32	0.09	4	32	1	0.72	4	32	0.04	4	32	1	0.11
$G(a \lor F\bar{b} \lor (cUXc))$	2	R	1	10	10	80	1	0.00	10	80	0.00	4	11.03	4	32	6.64	4	32	1	97.47	4	32	0.85	4	32	2	67.57
$G(aUX(a \lor (F\bar{b}UXc)))$	3	R	1	80	80	640	1	0.01	80	640	0.01		(killed)	(killed)			(killed)				(killed)			(killed)			
$XF\bar{a}RF(b \lor (\bar{a} \land F\bar{c}))$	3	P	1		9	72	1	0.13	11	88	0.13			7	56	0.97	6	48	1	63.19	7	56	1.50	6	48	1	20.78
$XF\bar{a}RF(b \lor (\bar{a} \land F\bar{c}))$	3	R	1	12	12	96	1	0.00	12	96	0.00	7	0.27	7	56	1.46	6	48	1	109.47	7	56	0.18	6	48	1	69.18
$(a \land Xa)R((\bar{b} \lor XFa)Uc)$	2	R	1	19	19	152	1	0.00	19	152	0.00		(killed)	(killed , $\leq$ 12)			(killed , $\leq$ 13)				(killed , $\leq$ 12)			(killed)			
$G((a \land b) \lor Ga \lor F(\bar{c} \land XXc))$	1	R	1	27	27	216	1	0.00	27	216	0.00		(killed)	9	36	521.14	7	28	1	371.3	(killed , $\leq$ 13)			(killed)			
$G(X(Fa \land Xb)MFc)$	3	R	1	29	29	232	1	0.00	29	232	0.00	4	668.17	4	32	506.38	(killed)				4	32	369.52	(killed)			
$X(\bar{a} \land Fa)R(aMFb)$	3	P	1		9	36	1	0.01	10	40	0.01			9	36	2.19	9	36	1	46.12	9	36	1.7	9	36	1	132.02
$X(\bar{a} \land Fa)R(aMFb)$	3	R	1	11	11	44	1	0.00	11	44	0.00	9	3.18	9	36	3.39	9	36	1	297.56	9	36	1.92	9	36	1	220.48
$G(Ga \lor X(FbU(c \lor X\bar{c})))$	2	R	1	28	28	224	1	0.00	28	224	0.00		(killed)	8	64	350.49	(killed)				(killed)			(killed)			
$G(XXFaU(b \lor (\bar{b} \land F\bar{c})))$	3	P	1		10	80	1	0.28	15	120	0.28			5	40	19.80	3	24	2	289.76	5	40	7.38	3	24	2	104.72
$G(XXFaU(b \lor (\bar{b} \land F\bar{c})))$	3	R	1	26	26	208	1	0.00	26	208	0.00		(killed)	5	40	276.5	(killed)				5	40	452.45	(killed)			
$((FaUb)RFc)WX\bar{c}$	3	R	1	32	32	256	1	0.00	32	256	0.00		(killed)	(killed , $\leq$ 11)			(killed)				(killed , $\leq$ 11)			(killed)			
$G(a \lor X(aR(GFb \lor (FcUa))))$	3	R	1	42	42	336	1	0.01	42	336	0.01		(killed)	(killed)			(killed)				(killed)			(killed)			
$(F\bar{a}RF\bar{b})WG\bar{c}$	2	R	1	15	15	120	1	0.00	15	120	0.00	5	56.63	5	40	41.44	4	32	2	797.63	5	40	9.12	4	32	1	664.98
$(\bar{a} \land Fa)R(\bar{b}RXFc)$	2	R	1	24	24	192	1	0.00	24	192	0.00		(killed)	(killed , $\leq$ 11)			(killed)				(killed , $\leq$ 11)			(killed)			
$F(a \land Fb)WXc$	2	R	1	25	25	200	1	0.00	25	200	0.00		(killed)	(killed , $\leq$ 12)			(killed)				(killed , $\leq$ 12)			(killed)			
$G(G\bar{a} \lor ((FbUc)Ua))$	3	R	1	18	18	144	1	0.00	18	144	0.00	10	188.70	10	80	276.54	(killed)				10	80	346.26	(killed)			
$(Fa \land XXb)RFc$	2	R	1	43	43	344	1	0.00	43	344	0.00		(killed)	(killed , $\leq$ 28)			(killed)				(killed)			(killed)			
$GaR(F\bar{b} \land (cUb))$	2	P	0		6	36	1	0.01	10	61	0.01			8	48	14.68	5	30	2	21.12	8	48	11.63	5	30	2	2.12
$GaR(F\bar{b} \land (cUb))$	2	R	0	8	9	56	1	0.00	9	56	0.00	8	10.73	8	49	30.46	5	30	2	197.08	8	48	43.50	5	30	2	70.98
$(aRFb)UX\bar{c}$	2	R	1	13	13	104	1	0.00	13	104	0.00		(killed)	(killed , $\leq$ 11)			10	80	1	389.87	(killed , $\leq$ 11)			10	80	1	616.70
$GF(a \land XXXFb)$	2	P	1		13	52	1	0.93	19	76	0.93			3	12	18.54	1	4	2	104.7	3	12	18.54	1	4	2	104.7
$GF(a \land XXXFb)$	2	R	1	6	6	24	1	0.00	6	24	0.00		(killed)	3	12	0.14	1	4	2	6.95	3	12	0.03	1	4	2	0.8
$GF(a \land F(b \land Fc))$	3	R	1	10	10	80	1	0.00	10	80	0.00	4	1.03	4	32	3.84	1	8	3	321.75	4	32	0.79	1	8	3	146.07
$(GFb \land GFa) \lor (GFc \land GFd)$	4	P	1		5	80	1	0.02	9	144	0.02			5	80	6.35	1	16	4	67.28	5	80	1.6	1	16	4	21.81
$(GFb \land GFa) \lor (GFc \land GFd)$	4	R	1	9	9	144	1	0.00	9	144	0.00	5	2.19	5	80	2.98	(killed , $\leq$ 8)				5	80	0.81	1	16	4	739.84

A.1.2 Not fully incremental approach

The following table shows the results of the benchmark that compared the old default SAT-based minimization (using Glucose) to the second incremental approach (using PicoSAT library):

Column **type** shows how the initial det. aut. was obtained: T = translation produces DTGBA; W = WDBA minimization works; P = powerset construction transforms TBA to DTBA; R = DRA to DBA.
Column **C.** tells whether the output automaton is complete: rejecting sink states are always omitted (add 1 state when C=0 if you want the size of the complete automaton).

														Glucose (As before)								Incr param=2						
				DRA	DTGBA				DBA			DBAminimizer		minDBA			minDTGBA				minDBA			minDTGBA			
formula	m	type	C.	st.	st.	tr.	acc.	time	st.	tr.	time	st.	time	st.	tr.	time	st.	tr.	acc.	time	st.	tr.	time	st.	tr.	acc.	time
$G(a \lor Fb)$	1	T	1	4	2	8	1	0.00	2	8	0.00	2	0.11	2	8	0.0	2	8	1	0.0	2	8	0.0	2	8	1	0.0
$GF(a \lor b)$	1	T	1	3	1	4	1	0.00	2	8	0.00	2	0.06	2	8	0.0	1	4	1	0.0	2	8	0.0	1	4	1	0.0
$G(a \lor Fa)$	1	T	1	3	1	2	1	0.00	2	4	0.00	2	0.06	2	4	0.0	1	2	1	0.0	2	4	0.0	1	2	1	0.0
$G(a \lor F(b \lor c))$	1	T	1	4	2	16	1	0.00	2	16	0.00	2	0.06	2	16	0.0	2	16	1	0.0	2	16	0.0	2	16	1	0.0
$G(a \lor F(b \land c))$	1	T	1	4	2	16	1	0.00	2	16	0.00	2	0.06	2	16	0.0	2	16	1	0.0	2	16	0.0	2	16	1	0.0

Column **type** shows how the initial det. aut. was obtained: T = translation produces DTGBA; W = WDBA minimization works; P = powerset construction transforms TBA to DTBA; R = DRA to DBA.
Column **C.** tells whether the output automaton is complete: rejecting sink states are always omitted (add 1 state when C=0 if you want the size of the complete automaton).

														Glucose (As before)							Incr param=2						
				DRA	DTGBA				DBA			DBA minimizer		minDBA			minDTGBA				minDBA			minDTGBA			
formula	m	type	C.	st.	st.	tr.	acc.	time	st.	tr.	time	st.	time	st.	tr.	time	st.	tr.	acc.	time	st.	tr.	time	st.	tr.	acc.	time
$GF(a \lor b) \land GF(b \lor c)$	2	T	1	5	1	8	2	0.00	3	24	0.00	3	0.06	3	24	0.0	1	8	2	0.0	3	24	0.0	1	8	2	0.0
$G(Fb \land Fa)$	2	T	1	5	1	4	2	0.00	3	12	0.00	3	0.06	3	12	0.0	1	4	2	0.0	3	12	0.0	1	4	2	0.0
$Ga \lor Gc \lor (G(a \lor GFb) \land G(c \lor GF\bar{b}))$	2	P	1		6	48	1	0.01	8	64	0.01			6	48	0.52	4	32	2	1.63	6	48	0.13	4	32	2	0.2
$Ga \lor Gc \lor (G(a \lor GFb) \land G(c \lor GF\bar{b}))$	2	R	1	10	10	80	1	0.00	10	80	0.00	6	6.81	6	48	2.05	4	32	2	36.46	6	48	0.21	4	32	2	10.56
$F(a \land GFb) \lor (Fc \land Fa \land F(c \land GF\bar{b}))$	6	P	1		4	32	1	0.01	5	40	0.01			5	40	0.02	4	32	1	21.25	5	40	0.01	4	32	1	0.7
$F(a \land GFb) \lor (Fc \land Fa \land F(c \land GF\bar{b}))$	6	R	1	5	5	40	1	0.00	5	40	0.00	5	0.15	5	40	0.02	4	32	1	57.65	5	40	0.01	4	32	1	237.43
$GF(Fa \lor GFb \lor FG(a \lor b))$	2	P	1		2	8	1	0.00	4	16	0.00			2	8	0.0	1	4	1	0.0	2	8	0.0	1	4	1	0.0
$GF(Fa \lor GFb \lor FG(a \lor b))$	2	R	1	3	3	12	1	0.00	3	12	0.00	2	0.38	2	8	0.0	1	4	1	0.03	2	8	0.0	1	4	1	0.01
$FG(Fa \lor GFb \lor FG(a \lor b))$	2	P	1		2	8	1	0.00	4	16	0.00			2	8	0.0	1	4	1	0.0	2	8	0.0	1	4	1	0.0
$FG(Fa \lor GFb \lor FG(a \lor b))$	2	R	1	3	3	12	1	0.00	3	12	0.00	2	0.38	2	8	0.0	1	4	1	0.03	2	8	0.0	1	4	1	0.01
$FG(Fb \lor FGa \lor GFa \lor FG(a \lor b))$	2	P	1		2	8	1	0.00	4	16	0.00			2	8	0.02	1	4	1	0.0	2	8	0.01	1	4	1	0.0
$FG(Fb \lor FGa \lor GFa \lor FG(a \lor b))$	2	R	1	3	3	12	1	0.00	3	12	0.00	2	0.38	2	8	0.0	1	4	1	0.03	2	8	0.0	1	4	1	0.01
$GF(a \rightarrow XXXb)$	1	T	1	4	1	4	1	0.00	2	8	0.00	2	0.09	2	8	0.0	1	4	1	0.0	2	8	0.0	1	4	1	0.0
$G(a \rightarrow Fb) \land G(c \rightarrow Fd)$	2	T	1	9	4	64	2	0.00	6	96	0.00	6	0.20	6	96	0.60	4	64	2	4.83	6	96	0.13	4	64	2	0.43
$GFb \land GFa$	2	T	1	5	1	4	2	0.00	3	12	0.00	3	0.06	3	12	0.0	1	4	2	0.0	3	12	0.0	1	4	2	0.0
$GFc \lor GFb \lor GFa$	1	T	1	3	1	8	1	0.00	2	16	0.00	2	0.06	2	16	0.0	1	8	1	0.0	2	16	0.0	1	8	1	0.0
GFa	1	T	1	3	1	2	1	0.00	2	4	0.00	2	0.06	2	4	0.0	1	2	1	0.0	2	4	0.0	1	2	1	0.0
$Ga \land G(b \rightarrow Fc)$	1	T	0	4	2	8	1	0.00	2	8	0.00	2	0.06	2	8	0.0	2	8	1	0.0	2	8	0.0	2	8	1	0.0
$G(a \rightarrow Fb) \land G(b \rightarrow Fc)$	2	T	1	7	4	32	2	0.00	5	40	0.00	5	0.08	5	40	0.07	3	24	2	2.12	5	40	0.02	3	24	2	0.2
$G(a \rightarrow Fb) \land G(\bar{a} \rightarrow F\bar{b})$	2	T	1	6	3	12	2	0.00	4	16	0.00	4	0.06	4	16	0.0	3	12	2	0.09	4	16	0.0	3	12	2	0.02
$GFc \land GFb \land GFa \land GFd$	4	T	1	9	1	16	4	0.01	5	80	0.01	5	0.75	5	80	0.53	1	16	4	0.01	5	80	0.33	1	16	4	0.01
$GF(a \leftrightarrow XXXb)$	1	P	1		15	60	1	0.08	23	92	0.08			(killed , $\leq$ 17)			(killed , $\leq$ 8)				(killed , $\leq$ 11)			(killed , $\leq$ 8)			
$GF(a \leftrightarrow XXXb)$	1	R	1	17	17	68	1	0.00	17	68	0.00		(killed)	(killed , $\leq$ 11)			(killed , $\leq$ 8)				(killed , $\leq$ 13)			(killed , $\leq$ 8)			
$G(a \rightarrow (bUc))$	1	T	0	4	2	13	1	0.00	2	13	0.00	2	0.06	2	13	0.0	2	13	1	0.0	2	13	0.0	2	13	1	0.0
$GF(a \leftrightarrow XXb)$	1	P	1		7	28	1	0.01	11	44	0.01			6	24	6.53	4	16	1	0.11	6	24	3.42	4	16	1	0.03
$GF(a \leftrightarrow XXb)$	1	R	1	9	9	36	1	0.00	9	36	0.00	6	2.21	6	24	3.21	4	16	1	0.92	6	24	2.08	4	16	1	0.11
$G(a \rightarrow Fb)$	1	T	1	4	2	8	1	0.00	2	8	0.00	2	0.06	2	8	0.0	2	8	1	0.0	2	8	0.0	2	8	1	0.0
$G(aU(bU(\bar{a}U\bar{b})))$	1	T	1	3	1	2	1	0.00	2	4	0.00	2	0.06	2	4	0.0	1	2	1	0.0	2	4	0.0	1	2	1	0.0
$X((aMF((b \land c) \lor (\bar{b} \land \bar{c})))W(G\bar{c}Ub))$	3	P	1		13	104	1	0.01	16	128	0.01			(killed , $\leq$ 12)			(killed , $\leq$ 10)				(killed , $\leq$ 12)			(killed , $\leq$ 10)			
$X((aMF((b \land c) \lor (\bar{b} \land \bar{c})))W(G\bar{c}Ub))$	3	R	1	22	22	176	1	0.00	22	176	0.00		(killed)	(killed , $\leq$ 12)			(killed)				(killed , $\leq$ 12)			(killed)			
$X(((a \land b)R(\bar{a}U\bar{c}))Rb)$	1	P	0		5	33	1	0.00	6	39	0.00			6	39	0.08	5	33	1	0.03	6	39	0.06	5	33	1	0.01
$X(((a \land b)R(\bar{a}U\bar{c}))Rb)$	1	R	0	8	9	55	1	0.00	9	55	0.00	6	1.10	6	39	0.53	5	33	1	0.57	6	39	0.2	5	33	1	0.11
$XXG(FaUXb)$	2	P	1		9	36	1	0.01	13	52	0.01			8	32	9.44	5	20	2	71.94	8	32	1.99	5	20	2	4.15
$XXG(FaUXb)$	2	R	1	18	18	72	1	0.00	18	72	0.00	8	80.69	8	32	25.15	5	20	2	762.74	8	32	12.48	(killed)			
$(\bar{a}M\bar{b})WF\bar{c}$	2	P	1		5	40	1	0.00	6	48	0.00			4	32	0.04	3	24	1	0.21	4	32	0.01	3	24	1	0.04
$(\bar{a}M\bar{b})WF\bar{c}$	2	R	1	7	7	56	1	0.00	7	56	0.00	4	0.10	4	32	0.04	3	24	1	1.76	4	32	0.01	3	24	1	0.17
$(a \land Fc \land GFb)Rc$	2	P	0		3	20	1	0.00	4	28	0.00			4	28	0.02	3	20	1	0.06	4	28	0.01	3	20	1	0.02
$(a \land Fc \land GFb)Rc$	2	R	0	4	5	32	1	0.00	5	32	0.00	4	0.16	4	28	0.06	3	20	1	1.36	4	28	0.01	3	20	1	0.15
$(aR(bWa))WG(\bar{a}M(b \lor c))$	1	P	0		2	13	1	0.00	3	19	0.00			3	19	0.01	2	13	1	0.0	3	19	0.01	2	13	1	0.0
$(aR(bWa))WG(\bar{a}M(b \lor c))$	1	R	0	3	4	26	1	0.00	4	26	0.00	3	0.17	3	19	0.03	2	13	1	0.03	3	19	0.01	2	13	1	0.01
$(FaWb)R(\bar{a} \lor Fc)$	2	P	1		11	88	1	0.01	12	96	0.01			8	64	4.82	8	64	2	82.70	8	64	1.52	8	64	1	24.3
$(FaWb)R(\bar{a} \lor Fc)$	2	R	1	8	8	64	1	0.00	8	64	0.00	8	0.83	8	64	0.74	8	64	1	5.76	8	64	0.44	8	64	1	4.62
$X(G(\bar{a}M\bar{b}) \lor G(a \lor G\bar{a}))$	1	P	0		7	23	1	0.00	8	25	0.00			7	21	0.56	6	19	1	0.11	7	21	0.29	6	19	1	0.05
$X(G(\bar{a}M\bar{b}) \lor G(a \lor G\bar{a}))$	1	R	0	12	13	40	1	0.00	13	40	0.00	7	1.32	7	21	1.84	6	19	1	1.42	7	21	0.53	6	19	1	0.68
$FaWGb$	1	P	1		2	8	1	0.00	3	12	0.00			2	8	0.0	2	8	1	0.0	2	8	0.0	2	8	1	0.0
$FaWGb$	1	R	1	3	3	12	1	0.00	3	12	0.00	2	0.07	2	8	0.0	2	8	1	0.0	2	8	0.0	2	8	1	0.0
$Ga \lor GFb$	1	P	1		3	12	1	0.00	4	16	0.00			3	12	0.0	2	8	1	0.0	3	12	0.0	2	8	1	0.0
$Ga \lor GFb$	1	R	1	4	4	16	1	0.00	4	16	0.00	3	0.07	3	12	0.0	2	8	1	0.0	3	12	0.0	2	8	1	0.0
$aMG(F\bar{b} \lor X\bar{a})$	2	P	1		5	20	1	0.00	6	24	0.00			5	20	0.06	5	20	1	0.45	5	20	0.02	5	20	1	0.08
$aMG(F\bar{b} \lor X\bar{a})$	2	R	1	7	7	28	1	0.00	7	28	0.00	5	0.12	5	20	0.12	5	20	1	5.27	5	20	0.03	5	20	1	0.67
$G\bar{a}RXFb$	1	P	1		4	16	1	0.00	6	24	0.00			5	20	0.09	3	12	1	0.0	5	20	0.02	3	12	1	0.0
$G\bar{a}RXFb$	1	R	1	12	12	48	1	0.00	12	48	0.00	5	8.51	5	20	4.28	3	12	1	3.65	5	20	1.02	3	12	1	0.86
$XF(\bar{a} \lor GFb)$	2	P	1		4	16	1	0.00	5	20	0.00			4	16	0.02	3	12	1	0.03	4	16	0.01	3	12	1	0.01
$XF(\bar{a} \lor GFb)$	2	R	1	7	7	28	1	0.00	7	28	0.00	4	0.13	4	16	0.09	3	12	1	1.98	4	16	0.02	3	12	1	0.25
$G(F\bar{a}U\bar{a})UXa$	2	P	1		6	12	1	0.00	7	14	0.00			6	12	0.04	5	10	1	0.10	6	12	0.02	5	10	1	0.04
$G(F\bar{a}U\bar{a})UXa$	2	R	1	8	8	16	1	0.00	8	16	0.00	6	0.24	6	12	0.11	5	10	1	3.36	6	12	0.02	5	10	1	0.48
$(a \lor G(aM\bar{b}))WFc$	2	P	1		5	40	1	0.00	6	48	0.00			5	40	0.05	4	32	1	0.15	5	40	0.02	4	32	1	0.03
$(a \lor G(aM\bar{b}))WFc$	2	R	1	6	6	48	1	0.00	6	48	0.00	5	0.08	5	40	0.04	4	32	1	0.46	5	40	0.01	4	32	1	0.14
$FaWXb$	1	P	1		6	24	1	0.00	7	28	0.00			6	24	0.06	6	24	1	0.02	6	24	0.02	6	24	1	0.01
$FaWXb$	1	R	1	7	7	28	1	0.00	7	28	0.00	6	0.19	6	24	0.10	6	24	1	0.08	6	24	0.02	6	24	1	0.02
$X(aR((\bar{b} \land F\bar{c})MX\bar{a}))$	2	P	0		10	56	1	0.01	12	64	0.01			(killed , $\leq$ 11)			8	48	2	25.58	(killed , $\leq$ 11)			8	48	2	23.95
$X(aR((\bar{b} \land F\bar{c})MX\bar{a}))$	2	R	0	12	13	68	1	0.00	13	68	0.00		(killed)	(killed , $\leq$ 11)			8	48	2	222.00	(killed , $\leq$ 11)			8	48	2	182.41
$XG\bar{a}RFb$	1	P	1		2	8	1	0.00	3	12	0.00			2	8	0.0	2	8	1	0.0	2	8	0.0	2	8	1	0.0
$XG\bar{a}RFb$	1	R	1	6	6	24	1	0.00	6	24	0.00	2	0.24	2	8	0.06	2	8	1	0.06	2	8	0.02	2	8	1	0.02
$GFa \lor (b \land Fc)$	2	P	1		4	32	1	0.00	6	48	0.00			6	48	0.04	4	32	1	0.06	6	48	0.02	4	32	1	0.02
$GFa \lor (b \land Fc)$	2	R	1	6	6	48	1	0.00	6	48	0.00	6	0.12	6	48	0.04	4	32	1	2.33	6	48	0.02	4	32	1	0.21
$X(aR(FbRF\bar{b}))$	2	P	1		6	24	1	0.00	7	28	0.00			5	20	0.06	4	16	1	0.21	5	20	0.01	4	16	1	0.06
$X(aR(FbRF\bar{b}))$	2	R	1	9	9	36	1	0.00	9	36	0.00	5	0.21	5	20	0.11	4	16	1	6.87	5	20	0.03	4	16	1	0.87
$G(XaMFa)$	2	P	1		2	4	1	0.00	3	6	0.00			2	4	0.0	1	2	1	0.0	2	4	0.0	1	2	1	0.0
$G(XaMFa)$	2	R	1	4	4	8	1	0.00	4	8	0.00	2	0.09	2	4	0.0	1	2	1	0.20	2	4	0.0	1	2	1	0.03
$X(Ga \lor GFb)$	1	P	1		4	16	1	0.00	5	20	0.00			4	16	0.02	3	12	1	0.0	4	16	0.01	3	12	1	0.0
$X(Ga \lor GFb)$	1	R	1	9	9	36	1	0.00	9	36	0.00	4	0.68	4	16	0.35	3	12	1	0.26	4	16	0.06	3	12	1	0.06
$X(Ga \lor XG((\bar{b} \land F\bar{c}) \lor (b \land Gc)))$	1	P	0		9	50	1	0.00	10	54	0.00			9	50	377.41	9	50	1	306.26	9	50	67.16	9	50	1	128.68
$X(Ga \lor XG((\bar{b} \land F\bar{c}) \lor (b \land Gc)))$	1	R	0	11	12	67	1	0.00	12	67	0.00	9	357.33	9	50	86.87	9	50	1	404.49	9	50	178.60	9	50	1	173.65
$GaRFb$	1	P	1		2	8	1	0.00	3	12	0.00			3	12	0.0	2	8	1	0.0	3	12	0.0	2	8	1	0.0
$GaRFb$	1	R	1	4	4	16	1	0.00	4	16	0.00	3	0.07	3	12	0.0	2	8	1	0.0	3	12	0.0	2	8	1	0.0
$G(aU(b \lor X((\bar{a} \land \bar{c}) \lor (a \land c))))$	1	P	0		3	20	1	0.00	4	28	0.00			3	20	0.05	2	12	1	0.02	3	20	0.01	2	12	1	0.01
$G(aU(b \lor X((\bar{a} \land \bar{c}) \lor (a \land c))))$	1	R	0	5	6	40	1	0.00	6	40	0.00	3	0.31	3	20	0.32	2	12	1	0.46	3	20	0.07	2	12	1	0.07

Column **type** shows how the initial det. aut. was obtained: T = translation produces DTGBA; W = WDBA minimization works; P = powerset construction transforms TBA to DTBA; R = DRA to DBA.
Column **C.** tells whether the output automaton is complete: rejecting sink states are always omitted (add 1 state when C=0 if you want the size of the complete automaton).

														Glucose (As before)							Incr param=2						
				DRA	DTGBA				DBA			DBAminimizer		minDBA			minDTGBA				minDBA			minDTGBA			
formula	m	type	C.	st.	st.	tr.	acc.	time	st.	tr.	time	st.	time	st.	tr.	time	st.	tr.	acc.	time	st.	tr.	time	st.	tr.	acc.	time
$XG((G\bar{a} \wedge F\bar{b}) \vee (Fa \wedge (a \vee Gb)))$	2	P	0		5	15	1	0.00	7	19	0.00			7	19	0.26	5	15	1	0.14	7	19	0.81	5	15	1	0.07
$XG((G\bar{a} \wedge F\bar{b}) \vee (Fa \wedge (a \vee Gb)))$	2	R	0	12	13	34	1	0.00	13	34	0.00	7	3.67	7	19	1.92	5	15	1	95.24	7	19	1.44	5	15	1	16.18
$(aUX\bar{a}) \vee XG(\bar{b} \wedge XFc)$	2	P	0		10	74	1	0.00	12	86	0.00			8	54	6.91	6	42	1	8.65	8	54	2.73	6	42	1	2.22
$(aUX\bar{a}) \vee XG(\bar{b} \wedge XFc)$	2	R	0	12	13	86	1	0.00	13	86	0.00	8	14.61	8	54	7.96	6	42	1	227.55	8	54	4.58	6	42	1	75.16
$X(G\bar{a} \vee GFa)$	1	P	1		4	8	1	0.00	5	10	0.00			4	8	0.0	3	6	1	0.0	4	8	0.0	3	6	1	0.0
$X(G\bar{a} \vee GFa)$	1	R	1	9	9	18	1	0.00	9	18	0.00	4	0.11	4	8	0.19	3	6	1	0.12	4	8	0.03	3	6	1	0.03
$G(G\bar{a} \vee G\bar{b} \vee F\bar{c})$	1	P	1		4	32	1	0.02	5	40	0.02			4	32	0.10	4	32	1	0.04	4	32	0.04	4	32	1	0.03
$G(G\bar{a} \vee G\bar{b} \vee F\bar{c})$	1	R	1	7	7	56	1	0.00	7	56	0.00	4	0.10	4	32	0.15	4	32	1	0.19	4	32	0.04	4	32	1	0.04
$X(FaWb)RFc$	2	R	1	15	15	120	1	0.00	15	120	0.00	7	79.67	7	56	9.41	7	56	1	224.11	7	56	3.76	7	56	1	200.63
$\bar{a} \wedge ((FbUa)WXc)$	2	R	0	19	20	156	1	0.00	20	156	0.00		(killed)		(killed , $\leq$ 15)			(killed , $\leq$ 16)				(killed , $\leq$ 15)			(killed)		
$G(F\bar{a} \vee (FbUc))$	3	P	1		11	88	1	0.22	17	136	0.22			4	32	32.07	2	16	2	351.80	4	32	19.88	2	16	2	252.55
$G(F\bar{a} \vee (FbUc))$	3	R	1	23	23	184	1	0.00	23	184	0.00		(killed)	4	32	200.07		(killed)			4	32	148.42		(killed)		
$(aR(bRFc))WXGb$	1	R	1	24	24	192	1	0.00	24	192	0.00		(killed)		(killed , $\leq$ 11)		8	64	1	254.19		(killed , $\leq$ 11)		8	64	1	778.38
$X((Fa \wedge XFb)RXFc)$	3	R	1	34	34	272	1	0.00	34	272	0.00		(killed)		(killed , $\leq$ 13)			(killed)				(killed , $\leq$ 13)			(killed)		
$(GaR(F\bar{b}Uc))Wc$	2	P	1		12	96	1	0.01	16	128	0.01			9	72	158.67	7	56	2	219.00	9	72	46.36	7	56	2	170.34
$(GaR(F\bar{b}Uc))Wc$	2	R	1	10	10	80	1	0.00	10	80	0.00	9	9.01	9	72	25.43	7	56	2	130.60	9	72	14.77	7	56	2	56.05
$X(\bar{a} \vee G(a \wedge \bar{b}))RF(c \wedge Fb)$	2	P	1		10	80	1	0.20	12	96	0.20				(killed , $\leq$ 11)		9	72	2	125.27		(killed , $\leq$ 11)		9	72	2	53.79
$X(\bar{a} \vee G(a \wedge \bar{b}))RF(c \wedge Fb)$	2	R	1	18	18	144	1	0.00	18	144	0.00		(killed)		(killed , $\leq$ 11)			(killed , $\leq$ 16)				(killed , $\leq$ 11)			(killed)		
$G(Fb \vee Ga \vee XXF\bar{c})$	2	R	1	44	44	352	1	0.01	44	352	0.01		(killed)		(killed)			(killed)				(killed)			(killed)		
$G(F(\bar{a} \wedge Fa)U(bUXc))$	3	R	1	35	35	280	1	0.00	35	280	0.00		(killed)		(killed , $\leq$ 27)			(killed)				(killed)			(killed)		
$G(F\bar{a}UX(F\bar{b} \wedge Xb))$	3	R	1	37	37	148	1	0.00	37	148	0.00		(killed)	4	16	776.24		(killed)			4	16	519.86		(killed)		
$G(XXFaU(a \vee b \vee Fc))$	3	P	1		10	80	1	0.13	15	120	0.13			4	32	13.59	3	24	3	317.30	4	32	10.96	3	24	1	102.16
$G(XXFaU(a \vee b \vee Fc))$	3	R	1	26	26	208	1	0.00	26	208	0.00		(killed)	4	32	262.56		(killed)			4	32	501.21		(killed)		
$G(a \vee F\bar{b} \vee (cUXc))$	2	P	1		4	32	1	0.02	5	40	0.02			4	32	0.09	4	32	1	0.72	4	32	0.04	4	32	1	0.11
$G(a \vee F\bar{b} \vee (cUXc))$	2	R	1	10	10	80	1	0.00	10	80	0.00	4	11.03	4	32	6.64	4	32	1	97.47	4	32	0.89	4	32	2	63.3
$G(aUX(a \vee (F\bar{b}UXc)))$	3	R	1	80	80	640	1	0.01	80	640	0.01		(killed)		(killed)			(killed)				(killed)			(killed)		
$XF\bar{a}RF(b \vee (\bar{a} \wedge F\bar{c}))$	3	P	1		9	72	1	0.13	11	88	0.13			7	56	0.97	6	48	1	63.19	7	56	1.63	6	48	1	20.91
$XF\bar{a}RF(b \vee (\bar{a} \wedge F\bar{c}))$	3	R	1	12	12	96	1	0.00	12	96	0.00	7	0.27	7	56	1.46	6	48	1	109.47	7	56	0.17	6	48	1	60.81
$(a \wedge Xa)R((\bar{b} \vee XFa)Uc)$	2	R	1	19	19	152	1	0.00	19	152	0.00		(killed)		(killed , $\leq$ 12)			(killed , $\leq$ 13)				(killed , $\leq$ 12)			(killed)		
$G((a \wedge b) \vee Ga \vee F(\bar{c} \wedge XXc))$	1	R	1	27	27	216	1	0.00	27	216	0.00		(killed)	9	36	521.14	7	28	1	371.3		(killed , $\leq$ 9)			(killed)		
$G(X(Fa \wedge Xb)MFc)$	3	R	1	29	29	232	1	0.00	29	232	0.00	4	668.17	4	32	506.38		(killed)			4	32	281.55		(killed)		
$X(\bar{a} \wedge Fa)R(aMFb)$	3	P	1		9	36	1	0.01	10	40	0.01			9	36	2.19	9	36	1	46.12	9	36	1.71	9	36	1	129.43
$X(\bar{a} \wedge Fa)R(aMFb)$	3	R	1	11	11	44	1	0.00	11	44	0.00	9	3.18	9	36	3.39	9	36	1	297.56	9	36	1.93	9	36	1	221.55
$G(Ga \vee X(FbU(c \vee X\bar{c})))$	2	R	1	28	28	224	1	0.00	28	224	0.00		(killed)	8	64	350.49		(killed)				(killed)			(killed)		
$G(XXFaU(b \vee (\bar{b} \wedge F\bar{c})))$	3	P	1		10	80	1	0.28	15	120	0.28			5	40	19.80	3	24	2	289.76	5	40	6.44	3	24	2	91.66
$G(XXFaU(b \vee (\bar{b} \wedge F\bar{c})))$	3	R	1	26	26	208	1	0.00	26	208	0.00		(killed)	5	40	276.5		(killed)			5	40	388.22		(killed)		
$((FaUb)RFc)WX\bar{c}$	3	R	1	32	32	256	1	0.00	32	256	0.00		(killed)		(killed , $\leq$ 11)			(killed)				(killed , $\leq$ 11)			(killed)		
$G(a \vee X(aR(GFb \vee (FcUa))))$	3	R	1	42	42	336	1	0.01	42	336	0.01		(killed)		(killed)			(killed)				(killed)			(killed)		
$(F\bar{a}RF\bar{b})WG\bar{c}$	2	R	1	15	15	120	1	0.00	15	120	0.00	5	56.63	5	40	41.44	4	32	2	797.63	5	40	8.13	4	32	1	656.18
$(\bar{a} \wedge Fa)R(\bar{b}RXFc)$	2	R	1	24	24	192	1	0.00	24	192	0.00		(killed)		(killed , $\leq$ 11)			(killed)				(killed , $\leq$ 11)			(killed)		
$F(a \wedge Fb)WXc$	2	R	1	25	25	200	1	0.00	25	200	0.00		(killed)		(killed , $\leq$ 12)			(killed)				(killed , $\leq$ 12)			(killed)		
$G(G\bar{a} \vee ((FbUc)Ua))$	3	R	1	18	18	144	1	0.00	18	144	0.00	10	188.70	10	80	276.54		(killed)			10	80	169.76		(killed)		
$(Fa \wedge XXb)RFc$	2	R	1	43	43	344	1	0.00	43	344	0.00		(killed)		(killed , $\leq$ 28)			(killed)				(killed)			(killed)		
$GaR(F\bar{b} \wedge (cUb))$	2	P	0		6	36	1	0.01	10	61	0.01			8	48	14.68	5	30	2	21.12	8	48	11.69	5	30	2	2.15
$GaR(F\bar{b} \wedge (cUb))$	2	R	0	8	9	56	1	0.00	9	56	0.00	8	10.73	8	49	30.46	5	30	2	197.08	8	48	43.36	5	30	2	70.6
$(aRFb)UX\bar{c}$	2	R	1	13	13	104	1	0.00	13	104	0.00		(killed)		(killed , $\leq$ 11)		10	80	1	389.87		(killed , $\leq$ 11)		10	80	1	425.21
$GF(a \wedge XXXFb)$	2	P	1		13	52	1	0.93	19	76	0.93			3	12	18.54	1	4	2	104.70	3	12	13.3	1	4	2	51.26
$GF(a \wedge XXXFb)$	2	R	1	6	6	24	1	0.00	6	24	0.00		(killed)	3	12	0.14	1	4	2	6.95	3	12	0.03	1	4	2	0.7
$GF(a \wedge F(b \wedge Fc))$	3	R	1	10	10	80	1	0.00	10	80	0.00	4	1.03	4	32	3.84	1	8	3	321.75	4	32	0.79	1	8	3	87.22
$(GFb \wedge GFa) \vee (GFc \wedge GFd)$	4	P	1		5	80	1	0.02	9	144	0.02			5	80	6.35	1	16	4	67.28	5	80	1.52	1	16	4	19.82
$(GFb \wedge GFa) \vee (GFc \wedge GFd)$	4	R	1	9	9	144	1	0.00	9	144	0.00	5	2.19	5	80	2.98		(killed , $\leq$ 8)			5	80	0.68	1	16	4	724.02

A.1.3 Assumption approach

The following table shows the results of the benchmark that compared the old default SAT-based minimization (using Glucose) to the assumption approach (using PicoSAT library):

Column **type** shows how the initial det. aut. was obtained: T = translation produces DTGBA; W = WDBA minimization works; P = powerset construction transforms TBA to DTBA; R = DRA to DBA.
Column **C.** tells whether the output automaton is complete: rejecting sink states are always omitted (add 1 state when C=0 if you want the size of the complete automaton).

														Glucose (As before)							Assume param=6						
				DRA	DTGBA				DBA			DBAminimizer		minDBA			minDTGBA				minDBA			minDTGBA			
formula	m	type	C.	st.	st.	tr.	acc.	time	st.	tr.	time	st.	time	st.	tr.	time	st.	tr.	acc.	time	st.	tr.	time	st.	tr.	acc.	time
$G(a \vee Fb)$	1	T	1	4	2	8	1	0.00	2	8	0.00	2	0.11	2	8	0.0	2	8	1	0.0	2	8	0.0	2	8	1	0.0
$GF(a \vee b)$	1	T	1	3	1	4	1	0.00	2	8	0.00	2	0.06	2	8	0.0	1	4	1	0.0	2	8	0.0	1	4	1	0.0
$G(a \vee Fa)$	1	T	1	3	1	2	1	0.00	2	4	0.00	2	0.06	2	4	0.0	1	2	1	0.0	2	4	0.0	1	2	1	0.0
$G(a \vee F(b \vee c))$	1	T	1	4	2	16	1	0.00	2	16	0.00	2	0.06	2	16	0.0	2	16	1	0.0	2	16	0.0	2	16	1	0.0

Column **type** shows how the initial det. aut. was obtained: T = translation produces DTGBA; W = WDBA minimization works; P = powerset construction transforms TBA to DTBA; R = DRA to DBA.
Column **C.** tells whether the output automaton is complete: rejecting sink states are always omitted (add 1 state when C=0 if you want the size of the complete automaton).

														Glucose (As before)							Assume param=6						
				DRA	DTGBA				DBA			DBA minimizer		minDBA			minDTGBA				minDBA			minDTGBA			
formula	m	type	C.	st.	st.	tr.	acc.	time	st.	tr.	time	st.	time	st.	tr.	time	st.	tr.	acc.	time	st.	tr.	time	st.	tr.	acc.	time
$G(a \vee F(b \wedge c))$	1	T	1	4	2	16	1	0.00	2	16	0.00	2	0.06	2	16	0.0	2	16	1	0.0	2	16	0.0	2	16	1	0.0
$GF(a \vee b) \wedge GF(b \vee c)$	2	T	1	5	1	8	2	0.00	3	24	0.00	3	0.06	3	24	0.0	1	8	2	0.0	3	24	0.0	1	8	2	0.0
$G(Fb \wedge Fa)$	2	T	1	5	1	4	2	0.00	3	12	0.00	3	0.06	3	12	0.0	1	4	2	0.0	3	12	0.0	1	4	2	0.0
$Ga \vee Gc \vee (G(a \vee GFb) \wedge G(c \vee GF\bar{b}))$	2	P	1		6	48	1	0.01	8	64	0.01			6	48	0.52	4	32	2	1.63	6	48	0.17	4	32	2	0.14
$Ga \vee Gc \vee (G(a \vee GFb) \wedge G(c \vee GF\bar{b}))$	2	R	1	10	10	80	1	0.00	10	80	0.00	6	6.81	6	48	2.05	4	32	2	36.46	6	48	0.16	4	32	2	2.94
$F(a \wedge GFb) \vee (Fc \wedge Fa \wedge F(c \wedge GF\bar{b}))$	6	P	1		4	32	1	0.01	5	40	0.01			5	40	0.02	4	32	1	21.25	5	40	0.01	4	32	1	0.7
$F(a \wedge GFb) \vee (Fc \wedge Fa \wedge F(c \wedge GF\bar{b}))$	6	R	1	5	5	40	1	0.00	5	40	0.00	5	0.15	5	40	0.02	4	32	1	57.65	5	40	0.01	4	32	1	29.69
$GF(Fa \vee GFb \vee FG(a \vee b))$	2	P	1		2	8	1	0.00	4	16	0.00			2	8	0.0	1	4	1	0.0	2	8	0.0	1	4	1	0.0
$GF(Fa \vee GFb \vee FG(a \vee b))$	2	R	1	3	3	12	1	0.00	3	12	0.00	2	0.38	2	8	0.0	1	4	1	0.03	2	8	0.0	1	4	1	0.01
$FG(Fa \vee GFb \vee FG(a \vee b))$	2	P	1		2	8	1	0.00	4	16	0.00			2	8	0.0	1	4	1	0.0	2	8	0.0	1	4	1	0.0
$FG(Fa \vee GFb \vee FG(a \vee b))$	2	R	1	3	3	12	1	0.00	3	12	0.00	2	0.38	2	8	0.0	1	4	1	0.03	2	8	0.0	1	4	1	0.01
$FG(Fb \vee FGa \vee GFa \vee FG(a \vee b))$	2	P	1		2	8	1	0.00	4	16	0.00			2	8	0.0	1	4	1	0.0	2	8	0.0	1	4	1	0.0
$FG(Fb \vee FGa \vee GFa \vee FG(a \vee b))$	2	R	1	3	3	12	1	0.00	3	12	0.00	2	0.38	2	8	0.0	1	4	1	0.03	2	8	0.0	1	4	1	0.01
$GF(a \rightarrow XXXb)$	1	T	1	4	1	4	1	0.00	2	8	0.00	2	0.09	2	8	0.0	1	4	1	0.0	2	8	0.0	1	4	1	0.0
$G(a \rightarrow Fb) \wedge G(c \rightarrow Fd)$	2	T	1	9	4	64	2	0.00	6	96	0.00	6	0.20	6	96	0.60	4	64	2	4.83	6	96	0.14	4	64	2	0.43
$GFb \wedge GFa$	2	T	1	5	1	4	2	0.00	3	12	0.00	3	0.06	3	12	0.0	1	4	2	0.0	3	12	0.0	1	4	2	0.0
$GFc \vee GFb \vee GFa$	1	T	1	3	1	8	1	0.00	2	16	0.00	2	0.06	2	16	0.0	1	8	1	0.0	2	16	0.0	1	8	1	0.0
GFa	1	T	1	3	1	2	1	0.00	2	4	0.00	2	0.06	2	4	0.0	1	2	1	0.0	2	4	0.0	1	2	1	0.0
$Ga \wedge G(b \rightarrow Fc)$	1	T	0	4	2	8	1	0.00	2	8	0.00	2	0.06	2	8	0.0	2	8	1	0.0	2	8	0.0	2	8	1	0.0
$G(a \rightarrow Fb) \wedge G(b \rightarrow Fc)$	2	T	1	7	4	32	2	0.00	5	40	0.00	5	0.08	5	40	0.07	3	24	2	2.12	5	40	0.02	3	24	2	0.19
$G(a \rightarrow Fb) \wedge G(\bar{a} \rightarrow F\bar{b})$	2	T	1	6	3	12	2	0.00	4	16	0.00	4	0.06	4	16	0.0	3	12	2	0.09	4	16	0.0	3	12	2	0.02
$GFc \wedge GFb \wedge GFa \wedge GFd$	4	T	1	9	1	16	4	0.01	5	80	0.01	5	0.75	5	80	0.53	1	16	4	0.01	5	80	0.41	1	16	4	0.01
$GF(a \leftrightarrow XXXb)$	1	P	1		15	60	1	0.08	23	92	0.08			(killed , $\leq$ 17)			(killed , $\leq$ 8)				(killed , $\leq$ 16)			(killed , $\leq$ 8)			
$GF(a \leftrightarrow XXXb)$	1	R	1	17	17	68	1	0.00	17	68	0.00	(killed)		(killed , $\leq$ 11)			(killed , $\leq$ 8)				(killed)			(killed , $\leq$ 10)			
$G(a \rightarrow (bUc))$	1	T	0	4	2	13	1	0.00	2	13	0.00	2	0.06	2	13	0.0	2	13	1	0.0	2	13	0.0	2	13	1	0.0
$GF(a \leftrightarrow XXb)$	1	P	1		7	28	1	0.01	11	44	0.01			6	24	6.44	4	16	1	0.11	6	24	6.44	4	16	1	0.03
$GF(a \leftrightarrow XXb)$	1	R	1	9	9	36	1	0.00	9	36	0.00	6	2.21	6	24	3.21	4	16	1	0.92	6	24	2.97	4	16	1	0.1
$G(a \rightarrow Fb)$	1	T	1	4	2	8	1	0.00	2	8	0.00	2	0.06	2	8	0.0	2	8	1	0.0	2	8	0.0	2	8	1	0.0
$G(aU(bU(\bar{a}U\bar{b})))$	1	T	1	3	1	2	1	0.00	2	4	0.00	2	0.06	2	4	0.0	1	2	1	0.0	2	4	0.0	1	2	1	0.0
$X((aMF((b \wedge c) \vee (\bar{b} \wedge \bar{c})))W(G\bar{c}Ub))$	3	P	1		13	104	1	0.01	16	128	0.01			(killed , $\leq$ 12)			(killed , $\leq$ 10)				(killed)			(killed)			
$X((aMF((b \wedge c) \vee (\bar{b} \wedge \bar{c})))W(G\bar{c}Ub))$	3	R	1	22	22	176	1	0.00	22	176	0.00	(killed)		(killed , $\leq$ 12)			(killed)				(killed , $\leq$ 13)			(killed)			
$X(((a \wedge b)R(\bar{a}U\bar{c}))Rb)$	1	P	0		5	33	1	0.00	6	39	0.00			6	39	0.08	5	33	1	0.03	6	39	0.05	5	33	1	0.01
$X(((a \wedge b)R(\bar{a}U\bar{c}))Rb)$	1	R	0	8	9	55	1	0.00	9	55	0.00	6	1.10	6	39	0.53	5	33	1	0.57	6	39	0.22	5	33	1	0.11
$XXG(FaUXb)$	2	P	1		9	36	1	0.01	13	52	0.01			8	32	9.44	5	20	2	71.94	8	32	5.02	5	20	2	2.66
$XXG(FaUXb)$	2	R	1	18	18	72	1	0.00	18	72	0.00	8	80.69	8	32	25.15	5	20	2	762.74	8	32	11.98	5	20	2	332.96
$(\bar{a}M\bar{b})WF\bar{c}$	2	P	1		5	40	1	0.00	6	48	0.00			4	32	0.04	3	24	1	0.21	4	32	0.01	3	24	1	0.03
$(\bar{a}M\bar{b})WF\bar{c}$	2	R	1	7	7	56	1	0.00	7	56	0.00	4	0.10	4	32	0.04	3	24	1	1.76	4	32	0.01	3	24	1	0.15
$(a \wedge Fc \wedge GFb)Rc$	2	P	0		3	20	1	0.00	4	28	0.00			4	28	0.02	3	20	1	0.06	4	28	0.01	3	20	1	0.02
$(a \wedge Fc \wedge GFb)Rc$	2	R	0	4	5	32	1	0.00	5	32	0.00	4	0.16	4	28	0.06	3	20	1	1.36	4	28	0.02	3	20	1	0.14
$(aR(bWa))WG(\bar{a}M(b \vee c))$	1	P	0		2	13	1	0.00	3	19	0.00			3	19	0.01	2	13	1	0.0	3	19	0.01	2	13	1	0.0
$(aR(bWa))WG(\bar{a}M(b \vee c))$	1	R	0	3	4	26	1	0.00	4	26	0.00	3	0.17	3	19	0.03	2	13	1	0.0	3	19	0.01	2	13	1	0.0
$(FaWb)R(\bar{a} \vee Fc)$	2	P	1		11	88	1	0.01	12	96	0.01			8	64	4.82	8	64	2	82.70	8	64	1.92	8	64	1	27.46
$(FaWb)R(\bar{a} \vee Fc)$	2	R	1	8	8	64	1	0.00	8	64	0.00	8	0.83	8	64	0.74	8	64	1	5.76	8	64	0.42	8	64	1	3.55
$X(G(\bar{a}M\bar{b}) \vee G(a \vee G\bar{a}))$	1	P	0		7	23	1	0.00	8	25	0.00			7	21	0.56	6	19	1	0.11	7	21	0.33	6	19	1	0.06
$X(G(\bar{a}M\bar{b}) \vee G(a \vee G\bar{a}))$	1	R	0	12	13	40	1	0.00	13	40	0.00	7	1.32	7	21	1.84	6	19	1	1.42	7	21	1.08	6	19	1	0.21
$FaWGb$	1	P	1		2	8	1	0.00	3	12	0.00			2	8	0.0	2	8	1	0.0	2	8	0.0	2	8	1	0.0
$FaWGb$	1	R	1	3	3	12	1	0.00	3	12	0.00	2	0.07	2	8	0.0	2	8	1	0.0	2	8	0.0	2	8	1	0.0
$Ga \vee GFb$	1	P	1		3	12	1	0.00	4	16	0.00			3	12	0.0	2	8	1	0.0	3	12	0.0	2	8	1	0.0
$Ga \vee GFb$	1	R	1	4	4	16	1	0.00	4	16	0.00	3	0.07	3	12	0.0	2	8	1	0.0	3	12	0.0	2	8	1	0.0
$aMG(F\bar{b} \vee X\bar{a})$	2	P	1		5	20	1	0.00	6	24	0.00			5	20	0.06	5	20	1	0.45	5	20	0.02	5	20	1	0.08
$aMG(F\bar{b} \vee X\bar{a})$	2	R	1	7	7	28	1	0.00	7	28	0.00	5	0.12	5	20	0.12	5	20	1	5.27	5	20	0.03	5	20	2	0.49
$G\bar{a}RXFb$	1	P	1		4	16	1	0.00	6	24	0.00			5	20	0.09	3	12	1	0.0	5	20	0.02	3	12	1	0.0
$G\bar{a}RXFb$	1	R	1	12	12	48	1	0.00	12	48	0.00	5	8.51	5	20	4.28	3	12	1	3.65	5	20	0.87	3	12	1	0.83
$XF(\bar{a} \vee GFb)$	2	P	1		4	16	1	0.00	5	20	0.00			4	16	0.0	3	12	1	0.03	4	16	0.0	3	12	1	0.01
$XF(\bar{a} \vee GFb)$	2	R	1	7	7	28	1	0.00	7	28	0.00	4	0.13	4	16	0.09	3	12	1	1.98	4	16	0.01	3	12	1	0.21
$G(F\bar{a}U\bar{a})UXa$	2	P	1		6	12	1	0.00	7	14	0.00			6	12	0.04	5	10	1	0.10	6	12	0.02	5	10	1	0.02
$G(F\bar{a}U\bar{a})UXa$	2	R	1	8	8	16	1	0.00	8	16	0.00	6	0.24	6	12	0.11	5	10	1	3.36	6	12	0.02	5	10	1	0.34
$(a \vee G(aM\bar{b}))WFc$	2	P	1		5	40	1	0.00	6	48	0.00			5	40	0.05	4	32	1	0.15	5	40	0.03	4	32	1	0.03
$(a \vee G(aM\bar{b}))WFc$	2	R	1	6	6	48	1	0.00	6	48	0.00	5	0.08	5	40	0.04	4	32	1	0.46	5	40	0.02	4	32	1	0.06
$FaWXb$	1	P	1		6	24	1	0.00	7	28	0.00			6	24	0.06	6	24	1	0.02	6	24	0.03	6	24	1	0.01
$FaWXb$	1	R	1	7	7	28	1	0.00	7	28	0.00	6	0.19	6	24	0.10	6	24	1	0.08	6	24	0.02	6	24	1	0.02
$X(aR((\bar{b} \wedge F\bar{c})MX\bar{a}))$	2	P	0		10	56	1	0.01	12	64	0.01			(killed , $\leq$ 11)			8	48	2	25.58	(killed)			8	48	2	22.14
$X(aR((\bar{b} \wedge F\bar{c})MX\bar{a}))$	2	R	0	12	13	68	1	0.00	13	68	0.00	(killed)		(killed , $\leq$ 11)			8	48	2	222.00	(killed)			8	48	2	142.48
$XG\bar{a}RFb$	1	P	1		2	8	1	0.00	3	12	0.00			2	8	0.0	2	8	1	0.0	2	8	0.0	2	8	1	0.0
$XG\bar{a}RFb$	1	R	1	6	6	24	1	0.00	6	24	0.00	2	0.24	2	8	0.06	2	8	1	0.06	2	8	0.02	2	8	1	0.02
$GFa \vee (b \wedge Fc)$	2	P	1		4	32	1	0.00	6	48	0.00			6	48	0.04	4	32	1	0.06	6	48	0.02	4	32	1	0.02
$GFa \vee (b \wedge Fc)$	2	R	1	6	6	48	1	0.00	6	48	0.00	6	0.12	6	48	0.04	4	32	1	2.33	6	48	0.01	4	32	1	0.18
$X(aR(FbRF\bar{b}))$	2	P	1		6	24	1	0.00	7	28	0.00			5	20	0.06	4	16	1	0.21	5	20	0.01	4	16	1	0.03
$X(aR(FbRF\bar{b}))$	2	R	1	9	9	36	1	0.00	9	36	0.00	5	0.21	5	20	0.11	4	16	1	6.87	5	20	0.02	4	16	1	0.61
$G(XaMFa)$	2	P	1		2	4	1	0.00	3	6	0.00			2	4	0.0	1	2	1	0.0	2	4	0.0	1	2	1	0.0
$G(XaMFa)$	2	R	1	4	4	8	1	0.00	4	8	0.00	2	0.09	2	4	0.0	1	2	1	0.20	2	4	0.0	1	2	1	0.03
$X(Ga \vee GFb)$	1	P	1		4	16	1	0.00	5	20	0.00			4	16	0.0	3	12	1	0.0	4	16	0.0	3	12	1	0.0
$X(Ga \vee GFb)$	1	R	1	9	9	36	1	0.00	9	36	0.00	4	0.68	4	16	0.35	3	12	1	0.26	4	16	0.05	3	12	1	0.05
$X(Ga \vee XG((\bar{b} \wedge F\bar{c}) \vee (b \wedge Gc)))$	1	P	0		9	50	1	0.00	10	54	0.00			9	50	377.41	9	50	1	306.26	9	50	90.2	9	50	1	139.29
$X(Ga \vee XG((\bar{b} \wedge F\bar{c}) \vee (b \wedge Gc)))$	1	R	0	11	12	67	1	0.00	12	67	0.00	9	357.33	9	50	86.87	9	50	1	404.49	9	50	304.37	9	50	1	163.64
$GaRFb$	1	P	1		2	8	1	0.00	3	12	0.00			3	12	0.0	2	8	1	0.0	3	12	0.0	2	8	1	0.0
$GaRFb$	1	R	1	4	4	16	1	0.00	4	16	0.00	3	0.07	3	12	0.0	2	8	1	0.0	3	12	0.0	2	8	1	0.0
$G(aU(b \vee X((\bar{a} \wedge \bar{c}) \vee (a \wedge c))))$	1	P	0		3	20	1	0.00	4	28	0.00			3	20	0.05	2	12	1	0.02	3	20	0.01	2	12	1	0.01

Column **type** shows how the initial det. aut. was obtained: T = translation produces DTGBA; W = WDBA minimization works; P = powerset construction transforms TBA to DTBA; R = DRA to DBA.
Column **C.** tells whether the output automaton is complete: rejecting sink states are always omitted (add 1 state when C=0 if you want the size of the complete automaton).

														Glucose (As before)							Assume param=6						
				DRA	DTGBA				DBA			DBAminimizer		minDBA			minDTGBA				minDBA			minDTGBA			
formula	m	type	C.	st.	st.	tr.	acc.	time	st.	tr.	time	st.	time	st.	tr.	time	st.	tr.	acc.	time	st.	tr.	time	st.	tr.	acc.	time
$G(aU(b \vee X((\bar{a} \wedge \bar{c}) \vee (a \wedge c))))$	1	R	0	5	6	40	1	0.00	6	40	0.00	3	0.31	3	20	0.32	2	12	1	0.46	3	20	0.06	2	12	1	0.06
$XG((G\bar{a} \wedge F\bar{b}) \vee (Fa \wedge (a \vee Gb)))$	2	P	0		5	15	1	0.00	7	19	0.00			7	19	0.26	5	15	1	0.14	7	19	0.78	5	15	1	0.05
$XG((G\bar{a} \wedge F\bar{b}) \vee (Fa \wedge (a \vee Gb)))$	2	R	0	12	13	34	1	0.00	13	34	0.00	7	3.67	7	19	1.92	5	15	1	95.24	7	19	0.8	5	15	1	5.77
$(aUX\bar{a}) \vee XG(\bar{b} \wedge XFc)$	2	P	0		10	74	1	0.00	12	86	0.00			8	54	6.76	6	42	1	8.65	8	54	6.76	6	42	1	1.4
$(aUX\bar{a}) \vee XG(\bar{b} \wedge XFc)$	2	R	0	12	13	86	1	0.00	13	86	0.00	8	14.61	8	54	7.96	6	42	1	227.55	8	54	6.75	6	42	1	10.45
$X(G\bar{a} \vee GFa)$	1	P	1		4	8	1	0.00	5	10	0.00			4	8	0.0	3	6	1	0.0	4	8	0.0	3	6	1	0.0
$X(G\bar{a} \vee GFa)$	1	R	1	9	9	18	1	0.00	9	18	0.00	4	0.11	4	8	0.19	3	6	1	0.12	4	8	0.03	3	6	1	0.02
$G(G\bar{a} \vee G\bar{b} \vee F\bar{c})$	1	P	1		4	32	1	0.02	5	40	0.02			4	32	0.10	4	32	1	0.04	4	32	0.04	4	32	1	0.03
$G(G\bar{a} \vee G\bar{b} \vee F\bar{c})$	1	R	1	7	7	56	1	0.00	7	56	0.00	4	0.10	4	32	0.15	4	32	1	0.19	4	32	0.03	4	32	1	0.03
$X(FaWb)RFc$	2	R	1	15	15	120	1	0.00	15	120	0.00	7	79.67	7	56	9.41	7	56	1	224.11	7	56	1.91	7	56	1	68.24
$\bar{a} \wedge ((FbUa)WXc)$	2	R	0	19	20	156	1	0.00	20	156	0.00		(killed)	(killed , $\leq$ 15)			(killed , $\leq$ 16)				(killed)			(killed)			
$G(F\bar{a} \vee (FbUc))$	3	P	1		11	88	1	0.22	17	136	0.22			4	32	32.07	2	16	2	351.80	4	32	11.93	2	16	2	121.33
$G(F\bar{a} \vee (FbUc))$	3	R	1	23	23	184	1	0.00	23	184	0.00		(killed)	4	32	200.07	(killed)				4	32	55.73	(killed)			
$(aR(bRFc))WXGb$	1	R	1	24	24	192	1	0.00	24	192	0.00		(killed)	(killed , $\leq$ 11)			8	64	1	254.19	(killed , $\leq$ 13)			8	64	1	363.70
$X((Fa \wedge XFb)RXFc)$	3	R	1	34	34	272	1	0.00	34	272	0.00		(killed)	(killed , $\leq$ 13)			(killed)				(killed , $\leq$ 16)			(killed)			
$(GaR(F\bar{b}Uc))Wc$	2	P	1		12	96	1	0.01	16	128	0.01			9	72	158.67	7	56	2	219.00	9	72	22.14	7	56	2	100.78
$(GaR(F\bar{b}Uc))Wc$	2	R	1	10	10	80	1	0.00	10	80	0.00	9	9.01	9	72	25.43	7	56	2	130.60	9	72	15.5	7	56	2	59.12
$X(\bar{a} \vee G(a \wedge \bar{b}))RF(c \wedge Fb)$	2	P	1		10	80	1	0.20	12	96	0.20			(killed , $\leq$ 11)			9	72	2	125.27	(killed)			9	72	2	43.95
$X(\bar{a} \vee G(a \wedge \bar{b}))RF(c \wedge Fb)$	2	R	1	18	18	144	1	0.00	18	144	0.00		(killed)	(killed , $\leq$ 11)			(killed , $\leq$ 16)				(killed , $\leq$ 11)			9	72	2	494.88
$G(Fb \vee Ga \vee XXF\bar{c})$	2	R	1	44	44	352	1	0.01	44	352	0.01		(killed)	(killed)			(killed)				(killed)			(killed)			
$G(F(\bar{a} \wedge Fa)U(bUXc))$	3	R	1	35	35	280	1	0.00	35	280	0.00		(killed)	(killed , $\leq$ 27)			(killed)				(killed)			(killed)			
$G(F\bar{a}UX(F\bar{b} \wedge Xb))$	3	R	1	37	37	148	1	0.00	37	148	0.00		(killed)	4	16	776.24	(killed)				4	16	310.95	(killed)			
$G(XXFaU(a \vee b \vee Fc))$	3	P	1		10	80	1	0.13	15	120	0.13			4	32	13.59	3	24	3	317.30	4	32	5.83	3	24	2	59.32
$G(XXFaU(a \vee b \vee Fc))$	3	R	1	26	26	208	1	0.00	26	208	0.00		(killed)	4	32	262.56	(killed)				4	32	158.45	(killed)			
$G(a \vee F\bar{b} \vee (cUXc))$	2	P	1		4	32	1	0.02	5	40	0.02			4	32	0.09	4	32	1	0.72	4	32	0.03	4	32	1	0.11
$G(a \vee F\bar{b} \vee (cUXc))$	2	R	1	10	10	80	1	0.00	10	80	0.00	4	11.03	4	32	6.64	4	32	1	97.47	4	32	0.83	4	32	2	24.15
$G(aUX(a \vee (F\bar{b}UXc)))$	3	R	1	80	80	640	1	0.01	80	640	0.01		(killed)	(killed)			(killed)				(killed)			(killed)			
$XF\bar{a}RF(b \vee (\bar{a} \wedge F\bar{c}))$	3	P	1		9	72	1	0.13	11	88	0.13			7	56	0.97	6	48	1	63.19	7	56	0.35	6	48	1	10.18
$XF\bar{a}RF(b \vee (\bar{a} \wedge F\bar{c}))$	3	R	1	12	12	96	1	0.00	12	96	0.00	7	0.27	7	56	1.46	6	48	1	109.47	7	56	0.78	6	48	1	15.0
$(a \wedge Xa)R((\bar{b} \vee XFa)Uc)$	2	R	1	19	19	152	1	0.00	19	152	0.00		(killed)	(killed , $\leq$ 12)			(killed , $\leq$ 13)				(killed , $\leq$ 12)			(killed)			
$G((a \wedge b) \vee Ga \vee F(\bar{c} \wedge XXc))$	1	R	1	27	27	216	1	0.00	27	216	0.00		(killed)	9	36	521.14	7	28	1	371.3	9	36	836.70	(killed)			
$G(X(Fa \wedge Xb)MFc)$	3	R	1	29	29	232	1	0.00	29	232	0.00	4	668.17	4	32	506.38	(killed)				4	32	179.88	(killed)			
$X(\bar{a} \wedge Fa)R(aMFb)$	3	P	1		9	36	1	0.01	10	40	0.01			9	36	2.19	9	36	1	46.12	9	36	2.08	9	36	1	109.38
$X(\bar{a} \wedge Fa)R(aMFb)$	3	R	1	11	11	44	1	0.00	11	44	0.00	9	3.18	9	36	3.39	9	36	1	297.56	9	36	2.74	9	36	1	187.31
$G(Ga \vee X(FbU(c \vee X\bar{c})))$	2	R	1	28	28	224	1	0.00	28	224	0.00		(killed)	8	64	350.49	(killed)				8	64	585.42	(killed)			
$G(XXFaU(b \vee (\bar{b} \wedge F\bar{c})))$	3	P	1		10	80	1	0.28	15	120	0.28			5	40	19.80	3	24	2	289.76	5	40	5.95	3	24	2	59.59
$G(XXFaU(b \vee (\bar{b} \wedge F\bar{c})))$	3	R	1	26	26	208	1	0.00	26	208	0.00		(killed)	5	40	276.50	(killed)				5	40	255.96	(killed)			
$((FaUb)RFc)WX\bar{c}$	3	R	1	32	32	256	1	0.00	32	256	0.00		(killed)	(killed , $\leq$ 11)			(killed)				(killed , $\leq$ 11)			(killed)			
$G(a \vee X(aR(GFb \vee (FcUa))))$	3	R	1	42	42	336	1	0.01	42	336	0.01		(killed)	(killed)			(killed)				(killed)			(killed)			
$(F\bar{a}RF\bar{b})WG\bar{c}$	2	R	1	15	15	120	1	0.00	15	120	0.00	5	56.63	5	40	41.44	4	32	2	797.63	5	40	5.48	4	32	1	155.76
$(\bar{a} \wedge Fa)R(\bar{b}RXFc)$	2	R	1	24	24	192	1	0.00	24	192	0.00		(killed)	(killed , $\leq$ 11)			(killed)				(killed , $\leq$ 14)			(killed , $\leq$ 13)			
$F(a \wedge Fb)WXc$	2	R	1	25	25	200	1	0.00	25	200	0.00		(killed)	(killed , $\leq$ 12)			(killed)				(killed)			(killed)			
$G(G\bar{a} \vee ((FbUc)Ua))$	3	R	1	18	18	144	1	0.00	18	144	0.00	10	188.70	10	80	276.54	(killed)				10	80	189.93	(killed)			
$(Fa \wedge XXb)RFc$	2	R	1	43	43	344	1	0.00	43	344	0.00		(killed)	(killed , $\leq$ 28)			(killed)				(killed)			(killed)			
$GaR(F\bar{b} \wedge (cUb))$	2	P	0		6	36	1	0.01	10	61	0.01			8	48	14.68	5	30	2	21.12	8	49	19.01	5	30	2	2.24
$GaR(F\bar{b} \wedge (cUb))$	2	R	0	8	9	56	1	0.00	9	56	0.00	8	10.73	8	49	30.46	5	30	2	197.08	8	48	35.69	5	30	2	20.76
$(aRFb)UX\bar{c}$	2	R	1	13	13	104	1	0.00	13	104	0.00		(killed)	(killed , $\leq$ 11)			10	80	1	389.87	(killed)			10	80	1	591.58
$GF(a \wedge XXXFb)$	2	P	1		13	52	1	0.93	19	76	0.93			3	12	18.54	1	4	2	104.70	3	12	10.82	1	4	2	36.05
$GF(a \wedge XXXFb)$	2	R	1	6	6	24	1	0.00	6	24	0.00		(killed)	3	12	0.14	1	4	2	6.95	3	12	0.03	1	4	2	0.56
$GF(a \wedge F(b \wedge Fc))$	3	R	1	10	10	80	1	0.00	10	80	0.00	4	1.03	4	32	3.84	1	8	3	321.75	4	32	0.74	1	8	3	57.84
$(GFb \wedge GFa) \vee (GFc \wedge GFd)$	4	P	1		5	80	1	0.02	9	144	0.02			5	80	6.35	1	16	4	67.28	5	80	1.38	1	16	4	8.97
$(GFb \wedge GFa) \vee (GFc \wedge GFd)$	4	R	1	9	9	144	1	0.00	9	144	0.00	5	2.19	5	80	2.98	(killed , $\leq$ 8)				5	80	0.86	1	16	4	209.47

A.1.4 Binary Search

The following table shows the results of the benchmark that compared the old default SAT-based minimization (using Glucose) to old binary search (using PicoSAT library):

Column **type** shows how the initial det. aut. was obtained: T = translation produces DTGBA; W = WDBA minimization works; P = powerset construction transforms TBA to DTBA; R = DRA to DBA.
Column **C.** tells whether the output automaton is complete: rejecting sink states are always omitted (add 1 state when C=0 if you want the size of the complete automaton).

														Glucose (As before)							Dichotomy						
				DRA	DTGBA				DBA			DBAminimizer		minDBA			minDTGBA				minDBA			minDTGBA			
formula	m	type	C.	st.	st.	tr.	acc.	time	st.	tr.	time	st.	time	st.	tr.	time	st.	tr.	acc.	time	st.	tr.	time	st.	tr.	acc.	time
$G(a \vee Fb)$	1	T	1	4	2	8	1	0.00	2	8	0.00	2	0.11	2	8	0.0	2	8	1	0.0	2	8	0.0	2	8	1	0.0
$GF(a \vee b)$	1	T	1	3	1	4	1	0.00	2	8	0.00	2	0.06	2	8	0.0	1	4	1	0.0	2	8	0.0	1	4	1	0.0
$G(a \vee Fa)$	1	T	1	3	1	2	1	0.00	2	4	0.00	2	0.06	2	4	0.0	1	2	1	0.0	2	4	0.0	1	2	1	0.0

Column **type** shows how the initial det. aut. was obtained: T = translation produces DTGBA; W = WDBA minimization works; P = powerset construction transforms TBA to DTBA; R = DRA to DBA.
Column **C.** tells whether the output automaton is complete: rejecting sink states are always omitted (add 1 state when C=0 if you want the size of the complete automaton).

														Glucose (As before)							Dichotomy						
				DRA	DTGBA				DBA			DBA minimizer		minDBA			minDTGBA				minDBA			minDTGBA			
formula	m	type	C.	st.	st.	tr.	acc.	time	st.	tr.	time	st.	time	st.	tr.	time	st.	tr.	acc.	time	st.	tr.	time	st.	tr.	acc.	time
$G(a \vee F(b \vee c))$	1	T	1	4	2	16	1	0.00	2	16	0.00	2	0.06	2	16	0.0	2	16	1	0.0	2	16	0.0	2	16	1	0.0
$G(a \vee F(b \wedge c))$	1	T	1	4	2	16	1	0.00	2	16	0.00	2	0.06	2	16	0.0	2	16	1	0.0	2	16	0.0	2	16	1	0.0
$GF(a \vee b) \wedge GF(b \vee c)$	2	T	1	5	1	8	2	0.00	3	24	0.00	3	0.06	3	24	0.0	1	8	2	0.0	3	24	0.0	1	8	2	0.0
$G(Fb \wedge Fa)$	2	T	1	5	1	4	2	0.00	3	12	0.00	3	0.06	3	12	0.0	1	4	2	0.0	3	12	0.0	1	4	2	0.0
$Ga \vee Gc \vee (G(a \vee GFb) \wedge G(c \vee GF\bar{b}))$	2	P	1		6	48	1	0.01	8	64	0.01			6	48	0.52	4	32	2	1.63	6	48	0.1	4	32	2	0.12
$Ga \vee Gc \vee (G(a \vee GFb) \wedge G(c \vee GF\bar{b}))$	2	R	1	10	10	80	1	0.00	10	80	0.00	6	6.81	6	48	2.05	4	32	2	36.46	6	48	0.12	4	32	2	0.64
$F(a \wedge GFb) \vee (Fc \wedge Fa \wedge F(c \wedge GF\bar{b}))$	6	P	1		4	32	1	0.01	5	40	0.01			5	40	0.02	4	32	1	21.25	5	40	0.01	4	32	1	0.86
$F(a \wedge GFb) \vee (Fc \wedge Fa \wedge F(c \wedge GF\bar{b}))$	6	R	1	5	5	40	1	0.00	5	40	0.00	5	0.15	5	40	0.02	4	32	1	57.65	5	40	0.01	4	32	1	241.52
$GF(Fa \vee GFb \vee FG(a \vee b))$	2	P	1		2	8	1	0.00	4	16	0.00			2	8	0.0	1	4	1	0.0	2	8	0.0	1	4	1	0.0
$GF(Fa \vee GFb \vee FG(a \vee b))$	2	R	1	3	3	12	1	0.00	3	12	0.00	2	0.38	2	8	0.0	1	4	1	0.0	2	8	0.0	1	4	1	0.0
$FG(Fa \vee GFb \vee FG(a \vee b))$	2	P	1		2	8	1	0.00	4	16	0.00			2	8	0.0	1	4	1	0.0	2	8	0.0	1	4	1	0.0
$FG(Fa \vee GFb \vee FG(a \vee b))$	2	R	1	3	3	12	1	0.00	3	12	0.00	2	0.38	2	8	0.0	1	4	1	0.0	2	8	0.0	1	4	1	0.0
$FG(Fb \vee FGa \vee GFa \vee FG(a \vee b))$	2	P	1		2	8	1	0.00	4	16	0.00			2	8	0.0	1	4	1	0.0	2	8	0.0	1	4	1	0.0
$FG(Fb \vee FGa \vee GFa \vee FG(a \vee b))$	2	R	1	3	3	12	1	0.00	3	12	0.00	2	0.38	2	8	0.0	1	4	1	0.0	2	8	0.0	1	4	1	0.0
$GF(a \rightarrow XXXb)$	1	T	1	4	1	4	1	0.00	2	8	0.00	2	0.09	2	8	0.0	1	4	1	0.0	2	8	0.0	1	4	1	0.0
$G(a \rightarrow Fb) \wedge G(c \rightarrow Fd)$	2	T	1	9	4	64	2	0.00	6	96	0.00	6	0.20	6	96	0.60	4	64	2	4.83	6	96	0.2	4	64	2	0.53
$GFb \wedge GFa$	2	T	1	5	1	4	2	0.00	3	12	0.00	3	0.06	3	12	0.0	1	4	2	0.0	3	12	0.0	1	4	2	0.0
$GFc \vee GFb \vee GFa$	1	T	1	3	1	8	1	0.00	2	16	0.00	2	0.06	2	16	0.0	1	8	1	0.0	2	16	0.0	1	8	1	0.0
GFa	1	T	1	3	1	2	1	0.00	2	4	0.00	2	0.06	2	4	0.0	1	2	1	0.0	2	4	0.0	1	2	1	0.0
$Ga \wedge G(b \rightarrow Fc)$	1	T	0	4	2	8	1	0.00	2	8	0.00	2	0.06	2	8	0.0	2	8	1	0.0	2	8	0.0	2	8	1	0.0
$G(a \rightarrow Fb) \wedge G(b \rightarrow Fc)$	2	T	1	7	4	32	2	0.00	5	40	0.00	5	0.08	5	40	0.07	3	24	2	2.12	5	40	0.03	3	24	2	0.25
$G(a \rightarrow Fb) \wedge G(\bar{a} \rightarrow F\bar{b})$	2	T	1	6	3	12	2	0.00	4	16	0.00	4	0.06	4	16	0.02	3	12	2	0.09	4	16	0.01	3	12	2	0.02
$GFc \wedge GFb \wedge GFa \wedge GFd$	4	T	1	9	1	16	4	0.01	5	80	0.01	5	0.75	5	80	0.53	1	16	4	0.01	5	80	0.35	1	16	4	0.01
$GF(a \leftrightarrow XXXb)$	1	P	1		15	60	1	0.08	23	92	0.08			(killed , $\leq$ 17)			(killed , $\leq$ 8)				(killed , $\leq$)			(killed)			
$GF(a \leftrightarrow XXXb)$	1	R	1	17	17	68	1	0.00	17	68	0.00	(killed)		(killed , $\leq$ 11)			(killed , $\leq$ 8)				(killed)			(killed , $\leq$)			
$G(a \rightarrow (bUc))$	1	T	0	4	2	13	1	0.00	2	13	0.00	2	0.06	2	13	0.0	2	13	1	0.0	2	13	0.0	2	13	1	0.0
$GF(a \leftrightarrow XXb)$	1	P	1		7	28	1	0.01	11	44	0.01			6	24	6.53	4	16	1	0.11	6	24	3.57	4	16	1	0.03
$GF(a \leftrightarrow XXb)$	1	R	1	9	9	36	1	0.00	9	36	0.00	6	2.21	6	24	3.21	4	16	1	0.92	6	24	2.87	4	16	1	0.02
$G(a \rightarrow Fb)$	1	T	1	4	2	8	1	0.00	2	8	0.00	2	0.06	2	8	0.0	2	8	1	0.0	2	8	0.0	2	8	1	0.0
$G(aU(bU(\bar{a}U\bar{b})))$	1	T	1	3	1	2	1	0.00	2	4	0.00	2	0.06	2	4	0.0	1	2	1	0.0	2	4	0.0	1	2	1	0.0
$X((aMF((b \wedge c) \vee (\bar{b} \wedge \bar{c})))W(G\bar{c}Ub))$	3	P	1		13	104	1	0.01	16	128	0.01			(killed , $\leq$ 12)			(killed , $\leq$ 10)				(killed , $\leq$ 12)			(killed , $\leq$)			
$X((aMF((b \wedge c) \vee (\bar{b} \wedge \bar{c})))W(G\bar{c}Ub))$	3	R	1	22	22	176	1	0.00	22	176	0.00	(killed)		(killed , $\leq$ 12)			(killed)				(killed)			(killed , $\leq$)			
$X(((a \wedge b)R(\bar{a}U\bar{c}))Rb)$	1	P	0		5	33	1	0.00	6	39	0.00			6	39	0.08	5	33	1	0.03	6	39	0.08	5	33	1	0.02
$X(((a \wedge b)R(\bar{a}U\bar{c}))Rb)$	1	R	0	8	9	55	1	0.00	9	55	0.00	6	1.10	6	39	0.53	5	33	1	0.57	6	39	0.17	5	33	1	0.08
$XXG(FaUXb)$	2	P	1		9	36	1	0.01	13	52	0.01			8	32	9.44	5	20	2	71.94	8	32	1.47	5	20	2	1.96
$XXG(FaUXb)$	2	R	1	18	18	72	1	0.00	18	72	0.00	8	80.69	8	32	25.15	5	20	2	762.74	8	32	2.21	5	20	2	50.86
$(\bar{a}M\bar{b})WF\bar{c}$	2	P	1		5	40	1	0.00	6	48	0.00			4	32	0.04	3	24	1	0.21	4	32	0.01	3	24	1	0.02
$(\bar{a}M\bar{b})WF\bar{c}$	2	R	1	7	7	56	1	0.00	7	56	0.00	4	0.10	4	32	0.04	3	24	1	1.76	4	32	0.01	3	24	1	0.03
$(a \wedge Fc \wedge GFb)Rc$	2	P	0		3	20	1	0.00	4	28	0.00			4	28	0.02	3	20	1	0.06	4	28	0.01	3	20	1	0.02
$(a \wedge Fc \wedge GFb)Rc$	2	R	0	4	5	32	1	0.00	5	32	0.00	4	0.16	4	28	0.06	3	20	1	1.36	4	28	0.02	3	20	1	0.11
$(aR(bWa))WG(\bar{a}M(b \vee c))$	1	P	0		2	13	1	0.00	3	19	0.00			3	19	0.01	2	13	1	0.0	3	19	0.01	2	13	1	0.0
$(aR(bWa))WG(\bar{a}M(b \vee c))$	1	R	0	3	4	26	1	0.00	4	26	0.00	3	0.17	3	19	0.03	2	13	1	0.0	3	19	0.01	2	13	1	0.0
$(FaWb)R(\bar{a} \vee Fc)$	2	P	1		11	88	1	0.01	12	96	0.01			8	64	4.82	8	64	2	82.70	8	64	0.91	8	64	2	14.21
$(FaWb)R(\bar{a} \vee Fc)$	2	R	1	8	8	64	1	0.00	8	64	0.00	8	0.83	8	64	0.74	8	64	1	5.58	8	64	0.52	8	64	1	5.58
$X(G(\bar{a}M\bar{b}) \vee G(a \vee G\bar{a}))$	1	P	0		7	23	1	0.00	8	25	0.00			7	21	0.56	6	19	1	0.11	7	21	0.36	6	19	1	0.07
$X(G(\bar{a}M\bar{b}) \vee G(a \vee G\bar{a}))$	1	R	0	12	13	40	1	0.00	13	40	0.00	7	1.32	7	21	1.84	6	19	1	1.42	7	21	0.38	6	19	1	0.12
$FaWGb$	1	P	1		2	8	1	0.00	3	12	0.00			2	8	0.0	2	8	1	0.0	2	8	0.0	2	8	1	0.0
$FaWGb$	1	R	1	3	3	12	1	0.00	3	12	0.00	2	0.07	2	8	0.0	2	8	1	0.0	2	8	0.0	2	8	1	0.0
$Ga \vee GFb$	1	P	1		3	12	1	0.00	4	16	0.00			3	12	0.0	2	8	1	0.0	3	12	0.0	2	8	1	0.0
$Ga \vee GFb$	1	R	1	4	4	16	1	0.00	4	16	0.00	3	0.07	3	12	0.0	2	8	1	0.0	3	12	0.0	2	8	1	0.0
$aMG(F\bar{b} \vee X\bar{a})$	2	P	1		5	20	1	0.00	6	24	0.00			5	20	0.06	5	20	1	0.45	5	20	0.03	5	20	1	0.12
$aMG(F\bar{b} \vee X\bar{a})$	2	R	1	7	7	28	1	0.00	7	28	0.00	5	0.12	5	20	0.12	5	20	1	5.27	5	20	0.03	5	20	1	0.41
$G\bar{a}RXFb$	1	P	1		4	16	1	0.00	6	24	0.00			5	20	0.09	3	12	1	0.0	5	20	0.03	3	12	1	0.0
$G\bar{a}RXFb$	1	R	1	12	12	48	1	0.00	12	48	0.00	5	8.51	5	20	4.28	3	12	1	3.65	5	20	0.16	3	12	1	0.13
$XF(\bar{a} \vee GFb)$	2	P	1		4	16	1	0.00	5	20	0.00			4	16	0.02	3	12	1	0.03	4	16	0.01	3	12	1	0.01
$XF(\bar{a} \vee GFb)$	2	R	1	7	7	28	1	0.00	7	28	0.00	4	0.13	4	16	0.09	3	12	1	1.98	4	16	0.01	3	12	1	0.03
$G(F\bar{a}U\bar{a})UXa$	2	P	1		6	12	1	0.00	7	14	0.00			6	12	0.04	5	10	1	0.10	6	12	0.02	5	10	1	0.05
$G(F\bar{a}U\bar{a})UXa$	2	R	1	8	8	16	1	0.00	8	16	0.00	6	0.24	6	12	0.11	5	10	1	3.36	6	12	0.02	5	10	1	0.4
$(a \vee G(aM\bar{b}))WFc$	2	P	1		5	40	1	0.00	6	48	0.00			5	40	0.05	4	32	1	0.15	5	40	0.02	4	32	1	0.05
$(a \vee G(aM\bar{b}))WFc$	2	R	1	6	6	48	1	0.00	6	48	0.00	5	0.08	5	40	0.04	4	32	1	0.46	5	40	0.02	4	32	1	0.05
$FaWXb$	1	P	1		6	24	1	0.00	7	28	0.00			6	24	0.06	6	24	1	0.02	6	24	0.03	6	24	1	0.01
$FaWXb$	1	R	1	7	7	28	1	0.00	7	28	0.00	6	0.19	6	24	0.10	6	24	1	0.08	6	24	0.03	6	24	1	0.03
$X(aR((\bar{b} \wedge F\bar{c})MX\bar{a}))$	2	P	0		10	56	1	0.01	12	64	0.01			(killed , $\leq$ 11)			8	48	2	25.58	(killed , $\leq$ 11)			8	48	2	18.33
$X(aR((\bar{b} \wedge F\bar{c})MX\bar{a}))$	2	R	0	12	13	68	1	0.00	13	68	0.00	(killed)		(killed , $\leq$ 11)			8	48	2	222.00	(killed , $\leq$)			8	48	2	69.25
$XG\bar{a}RFb$	1	P	1		2	8	1	0.00	3	12	0.00			2	8	0.0	2	8	1	0.0	2	8	0.0	2	8	1	0.0
$XG\bar{a}RFb$	1	R	1	6	6	24	1	0.00	6	24	0.00	2	0.24	2	8	0.06	2	8	1	0.06	2	8	0.01	2	8	1	0.01
$GFa \vee (b \wedge Fc)$	2	P	1		4	32	1	0.00	6	48	0.00			6	48	0.04	4	32	1	0.06	6	48	0.03	4	32	1	0.02
$GFa \vee (b \wedge Fc)$	2	R	1	6	6	48	1	0.00	6	48	0.00	6	0.12	6	48	0.04	4	32	1	2.33	6	48	0.02	4	32	1	0.14
$X(aR(FbRF\bar{b}))$	2	P	1		6	24	1	0.00	7	28	0.00			5	20	0.06	4	16	1	0.21	5	20	0.02	4	16	1	0.03
$X(aR(FbRF\bar{b}))$	2	R	1	9	9	36	1	0.00	9	36	0.00	5	0.21	5	20	0.11	4	16	1	6.87	5	20	0.03	4	16	1	0.08
$G(XaMFa)$	2	P	1		2	4	1	0.00	3	6	0.00			2	4	0.0	1	2	1	0.0	2	4	0.0	1	2	1	0.0
$G(XaMFa)$	2	R	1	4	4	8	1	0.00	4	8	0.00	2	0.09	2	4	0.0	1	2	1	0.20	2	4	0.0	1	2	1	0.01
$X(Ga \vee GFb)$	1	P	1		4	16	1	0.00	5	20	0.00			4	16	0.02	3	12	1	0.0	4	16	0.01	3	12	1	0.0
$X(Ga \vee GFb)$	1	R	1	9	9	36	1	0.00	9	36	0.00	4	0.68	4	16	0.35	3	12	1	0.26	4	16	0.01	3	12	1	0.01
$X(Ga \vee XG((\bar{b} \wedge F\bar{c}) \vee (b \wedge Gc)))$	1	P	0		9	50	1	0.00	10	54	0.00			9	50	377.41	9	50	1	306.26	9	50	155.55	9	50	1	132.17
$X(Ga \vee XG((\bar{b} \wedge F\bar{c}) \vee (b \wedge Gc)))$	1	R	0	11	12	67	1	0.00	12	67	0.00	9	357.33	9	50	86.87	9	50	1	404.49	9	50	176.26	9	50	1	140.87
$GaRFb$	1	P	1		2	8	1	0.00	3	12	0.00			3	12	0.0	2	8	1	0.0	3	12	0.0	2	8	1	0.0
$GaRFb$	1	R	1	4	4	16	1	0.00	4	16	0.00	3	0.07	3	12	0.0	2	8	1	0.0	3	12	0.0	2	8	1	0.0

Column **type** shows how the initial det. aut. was obtained: T = translation produces DTGBA; W = WDBA minimization works; P = powerset construction transforms TBA to DTBA; R = DRA to DBA.
Column **C.** tells whether the output automaton is complete: rejecting sink states are always omitted (add 1 state when C=0 if you want the size of the complete automaton).

														Glucose (As before)								Dichotomy						
				DRA	DTGBA				DBA			DBAminimizer		minDBA			minDTGBA				minDBA			minDTGBA				
formula	m	type	C.	st.	st.	tr.	acc.	time	st.	tr.	time	st.	time	st.	tr.	time	st.	tr.	acc.	time	st.	tr.	time	st.	tr.	acc.	time
$G(aU(b \vee X((\bar{a} \wedge \bar{c}) \vee (a \wedge c))))$	1	P	0		3	20	1	0.00	4	28	0.00			3	20	0.05	2	12	1	0.02	3	20	0.02	2	12	1	0.01
$G(aU(b \vee X((\bar{a} \wedge \bar{c}) \vee (a \wedge c))))$	1	R	0	5	6	40	1	0.00	6	40	0.00	3	0.31	3	20	0.32	2	12	1	0.46	3	20	0.05	2	12	1	0.01
$XG((G\bar{a} \wedge F\bar{b}) \vee (Fa \wedge (a \vee Gb)))$	2	P	0		5	15	1	0.00	7	19	0.00			7	19	0.26	5	15	1	0.14	7	19	0.86	5	15	1	0.09
$XG((G\bar{a} \wedge F\bar{b}) \vee (Fa \wedge (a \vee Gb)))$	2	R	0	12	13	34	1	0.00	13	34	0.00	7	3.67	7	19	1.92	5	15	1	95.24	7	19	0.92	5	15	1	2.39
$(aUX\bar{a}) \vee XG(\bar{b} \wedge XFc)$	2	P	0		10	74	1	0.00	12	86	0.00			8	54	6.91	6	42	1	8.65	8	54	3.57	6	42	1	1.96
$(aUX\bar{a}) \vee XG(\bar{b} \wedge XFc)$	2	R	0	12	13	86	1	0.00	13	86	0.00	8	14.61	8	54	7.96	6	42	1	227.55	8	54	2.98	6	42	1	3.41
$X(G\bar{a} \vee GFa)$	1	P	1		4	8	1	0.00	5	10	0.00			4	8	0.0	3	6	1	0.0	4	8	0.0	3	6	1	0.0
$X(G\bar{a} \vee GFa)$	1	R	1	9	9	18	1	0.00	9	18	0.00	4	0.11	4	8	0.19	3	6	1	0.12	4	8	0.01	3	6	1	0.01
$G(G\bar{a} \vee G\bar{b} \vee F\bar{c})$	1	P	1		4	32	1	0.02	5	40	0.02			4	32	0.10	4	32	1	0.04	4	32	0.04	4	32	1	0.03
$G(G\bar{a} \vee G\bar{b} \vee F\bar{c})$	1	R	1	7	7	56	1	0.00	7	56	0.00	4	0.10	4	32	0.15	4	32	1	0.19	4	32	0.03	4	32	1	0.03
$X(FaWb)RFc$	2	R	1	15	15	120	1	0.00	15	120	0.00	7	79.67	7	56	9.41	7	56	1	224.11	7	56	0.34	7	56	1	12.36
$\bar{a} \wedge ((FbUa)WXc)$	2	R	0	19	20	156	1	0.00	20	156	0.00	(killed)		(killed , $\leq$ 15)			(killed , $\leq$ 16)				(killed)			(killed)			
$G(F\bar{a} \vee (FbUc))$	3	P	1		11	88	1	0.22	17	136	0.22			4	32	32.07	2	16	2	351.80	4	32	2.03	2	16	3	18.71
$G(F\bar{a} \vee (FbUc))$	3	R	1	23	23	184	1	0.00	23	184	0.00	(killed)		4	32	200.07	(killed)				4	32	7.89	(killed)			
$(aR(bRFc))WXGb$	1	R	1	24	24	192	1	0.00	24	192	0.00	(killed)		(killed , $\leq$ 11)			8	64	1	254.19	(killed , $\leq$)			8	64	1	26.85
$X((Fa \wedge XFb)RXFc)$	3	R	1	34	34	272	1	0.00	34	272	0.00	(killed)		(killed , $\leq$ 13)			(killed)				(killed , $\leq$)			(killed)			
$(GaR(F\bar{b}Uc))Wc$	2	P	1		12	96	1	0.01	16	128	0.01			9	72	158.67	7	56	2	219.00	9	72	29.39	7	56	2	47.89
$(GaR(F\bar{b}Uc))Wc$	2	R	1	10	10	80	1	0.00	10	80	0.00	9	9.01	9	72	25.43	7	56	2	130.60	9	72	14.77	7	56	2	36.83
$X(\bar{a} \vee G(a \wedge \bar{b}))RF(c \wedge Fb)$	2	P	1		10	80	1	0.20	12	96	0.20			(killed , $\leq$ 11)			9	72	2	125.27	(killed , $\leq$)			9	72	2	66.8
$X(\bar{a} \vee G(a \wedge \bar{b}))RF(c \wedge Fb)$	2	R	1	18	18	144	1	0.00	18	144	0.00	(killed)		(killed , $\leq$ 11)			(killed , $\leq$ 16)				(killed , $\leq$ 11)			9	72	2	116.55
$G(Fb \vee Ga \vee XXF\bar{c})$	2	R	1	44	44	352	1	0.01	44	352	0.01	(killed)		(killed)			(killed)				5	40	688.36	(killed)			
$G(F(\bar{a} \wedge Fa)U(bUXc))$	3	R	1	35	35	280	1	0.00	35	280	0.00	(killed)		(killed , $\leq$ 27)			(killed)				10	80	198.63	(killed)			
$G(F\bar{a}UX(F\bar{b} \wedge Xb))$	3	R	1	37	37	148	1	0.00	37	148	0.00	(killed)		4	16	776.24	(killed)				4	16	36.96	(killed)			
$G(XXFaU(a \vee b \vee Fc))$	3	P	1		10	80	1	0.13	15	120	0.13			4	32	13.59	3	24	3	317.30	4	32	0.85	3	24	1	12.67
$G(XXFaU(a \vee b \vee Fc))$	3	R	1	26	26	208	1	0.00	26	208	0.00	(killed)		4	32	262.56	(killed)				4	32	17.5	(killed)			
$G(a \vee F\bar{b} \vee (cUXc))$	2	P	1		4	32	1	0.02	5	40	0.02			4	32	0.09	4	32	1	0.72	4	32	0.04	4	32	1	0.14
$G(a \vee F\bar{b} \vee (cUXc))$	2	R	1	10	10	80	1	0.00	10	80	0.00	4	11.03	4	32	6.64	4	32	1	97.47	4	32	0.14	4	32	2	5.49
$G(aUX(a \vee (F\bar{b}UXc)))$	3	R	1	80	80	640	1	0.01	80	640	0.01	(killed)		(killed)			(killed)				(killed)			(killed)			
$XF\bar{a}RF(b \vee (\bar{a} \wedge F\bar{c}))$	3	P	1		9	72	1	0.13	11	88	0.13			7	56	0.97	6	48	1	63.19	7	56	1.29	6	48	1	6.84
$XF\bar{a}RF(b \vee (\bar{a} \wedge F\bar{c}))$	3	R	1	12	12	96	1	0.00	12	96	0.00	7	0.27	7	56	1.46	6	48	1	109.47	7	56	0.51	6	48	1	12.32
$(a \wedge Xa)R((\bar{b} \vee XFa)Uc)$	2	R	1	19	19	152	1	0.00	19	152	0.00	(killed)		(killed , $\leq$ 12)			(killed , $\leq$ 13)				(killed , $\leq$ 14)			9	72	2	285.56
$G((a \wedge b) \vee Ga \vee F(\bar{c} \wedge XXc))$	1	R	1	27	27	216	1	0.00	27	216	0.00	(killed)		9	36	521.14	7	28	1	371.30	9	36	111.71	7	28	1	62.45
$G(X(Fa \wedge Xb)MFc)$	3	R	1	29	29	232	1	0.00	29	232	0.00	4	668.17	4	32	506.38	(killed)				4	32	25.21	(killed)			
$X(\bar{a} \wedge Fa)R(aMFb)$	3	P	1		9	36	1	0.01	10	40	0.01			9	36	2.19	9	36	1	46.12	9	36	1.46	9	36	1	165.60
$X(\bar{a} \wedge Fa)R(aMFb)$	3	R	1	11	11	44	1	0.00	11	44	0.00	9	3.18	9	36	3.39	9	36	1	297.56	9	36	1.95	9	36	1	161.05
$G(Ga \vee X(FbU(c \vee X\bar{c})))$	2	R	1	28	28	224	1	0.00	28	224	0.00	(killed)		8	64	350.49	(killed)				8	64	39.2	(killed)			
$G(XXFaU(b \vee (\bar{b} \wedge F\bar{c})))$	3	P	1		10	80	1	0.28	15	120	0.28			5	40	19.80	3	24	2	289.76	5	40	1.01	3	24	2	15.95
$G(XXFaU(b \vee (\bar{b} \wedge F\bar{c})))$	3	R	1	26	26	208	1	0.00	26	208	0.00	(killed)		5	40	276.50	(killed)				5	40	19.0	(killed)			
$((FaUb)RFc)WX\bar{c}$	3	R	1	32	32	256	1	0.00	32	256	0.00	(killed)		(killed , $\leq$ 11)			(killed)				(killed , $\leq$)			(killed)			
$G(a \vee X(aR(GFb \vee (FcUa))))$	3	R	1	42	42	336	1	0.01	42	336	0.01	(killed)		(killed)			(killed)				6	48	139.48	(killed)			
$(F\bar{a}RF\bar{b})WG\bar{c}$	2	R	1	15	15	120	1	0.00	15	120	0.00	5	56.63	5	40	41.44	4	32	2	797.63	5	40	0.8	4	32	1	30.52
$(\bar{a} \wedge Fa)R(\bar{b}RXFc)$	2	R	1	24	24	192	1	0.00	24	192	0.00	(killed)		(killed , $\leq$ 11)			(killed)				(killed , $\leq$)			7	56	1	127.42
$F(a \wedge Fb)WXc$	2	R	1	25	25	200	1	0.00	25	200	0.00	(killed)		(killed , $\leq$ 12)			(killed)				(killed , $\leq$ 12)			(killed , $\leq$)			
$G(G\bar{a} \vee ((FbUc)Ua))$	3	R	1	18	18	144	1	0.00	18	144	0.00	10	188.70	10	80	276.54	(killed)				10	80	174.09	(killed)			
$(Fa \wedge XXb)RFc$	2	R	1	43	43	344	1	0.00	43	344	0.00	(killed)		(killed , $\leq$ 28)			(killed)				(killed , $\leq$ 21)			(killed)			
$GaR(F\bar{b} \wedge (cUb))$	2	P	0		6	36	1	0.01	10	61	0.01			8	48	14.68	5	30	2	21.12	8	49	11.47	5	30	2	2.98
$GaR(F\bar{b} \wedge (cUb))$	2	R	0	8	9	56	1	0.00	9	56	0.00	8	10.73	8	49	30.46	5	30	2	197.08	8	48	24.69	5	30	2	16.32
$(aRFb)UX\bar{c}$	2	R	1	13	13	104	1	0.00	13	104	0.00	(killed)		(killed , $\leq$ 11)			10	80	1	389.87	11	88	842.13	10	80	1	500.42
$GF(a \wedge XXXFb)$	2	P	1		13	52	1	0.93	19	76	0.93			3	12	18.54	1	4	2	104.70	3	12	2.25	1	4	2	5.86
$GF(a \wedge XXXFb)$	2	R	1	6	6	24	1	0.00	6	24	0.00	(killed)		3	12	0.14	1	4	2	6.95	3	12	0.01	1	4	2	0.11
$GF(a \wedge F(b \wedge Fc))$	3	R	1	10	10	80	1	0.00	10	80	0.00	4	1.03	4	32	3.84	1	8	3	321.75	4	32	0.14	1	8	3	11.86
$(GFb \wedge GFa) \vee (GFc \wedge GFd)$	4	P	1		5	80	1	0.02	9	144	0.02			5	80	6.35	1	16	4	67.28	5	80	0.94	1	16	4	1.45
$(GFb \wedge GFa) \vee (GFc \wedge GFd)$	4	R	1	9	9	144	1	0.00	9	144	0.00	5	2.19	5	80	2.98	(killed , $\leq$ 8)				5	80	0.61	1	16	4	34.87

A.2 Heap memory profiling

The following figures (A.1 and A.2) show memory consumption of SAT-based minimization of an input automaton produced with this formula: $G(F(!a)|(F(b)Uc))$.

In both cases, **Glucose** is used as external SAT solver. We now consume 23 Mb (figure A.2) instead of 29,9 (figure A.1). Have in mind that this formula starts the minimization with an automaton of 8 states and the bigger the automaton is, the more memory-hungry SAT-based minimization is.

A.2.1 Before Optimizations

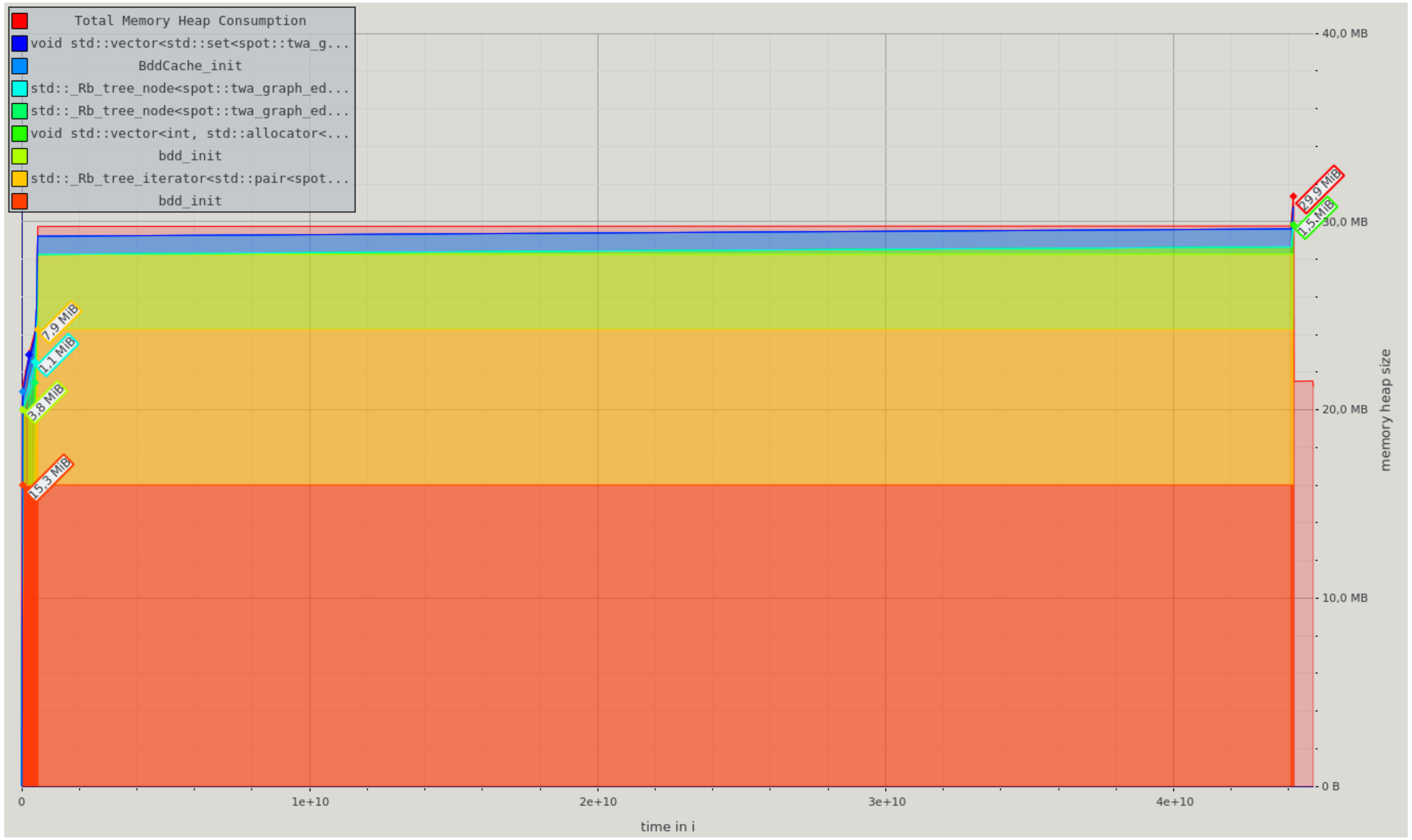


Figure A.1: Memory heap consumption before optimizations

A.2.2 After Optimizations

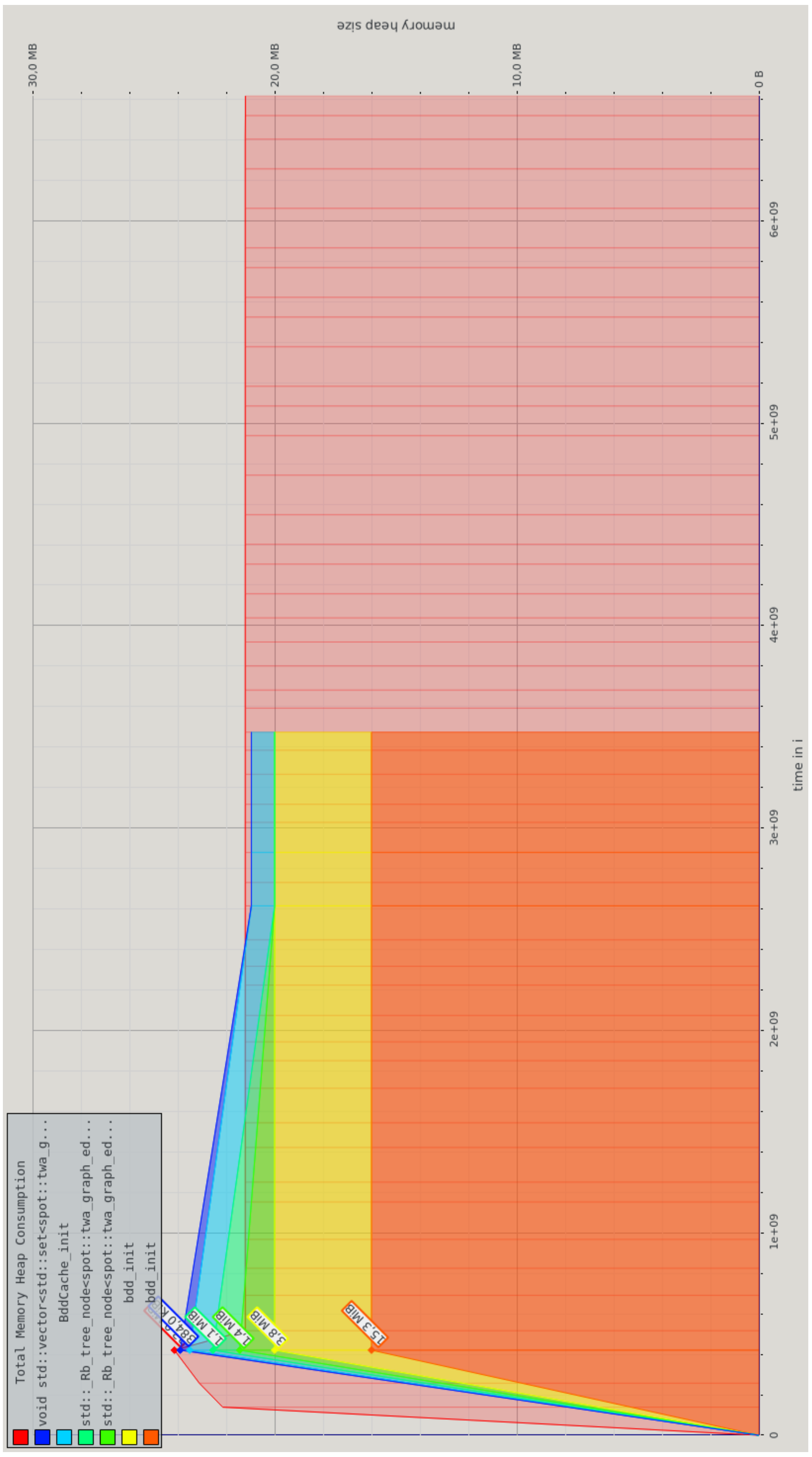


Figure A.2: Memory heap consumption after optimizations

List of Figures