A Proof Builder for Max-SAT

Matthieu Py, Mohamed Sami Cherif, Djamal Habet

Aix-Marseille Université, Université de Toulon, CNRS, LIS, Marseille, France

SAT 2021

SAT Problem

SAT Problem

Given a CNF formula ϕ, solving the SAT problem consists in determining if there exists an assignment of the variables which satisfies ϕ.

Let $\phi = (\overline{x_1} \vee x_3) \wedge (x_1) \wedge (\overline{x_1} \vee x_2) \wedge (\overline{x_2} \vee \overline{x_3})$.

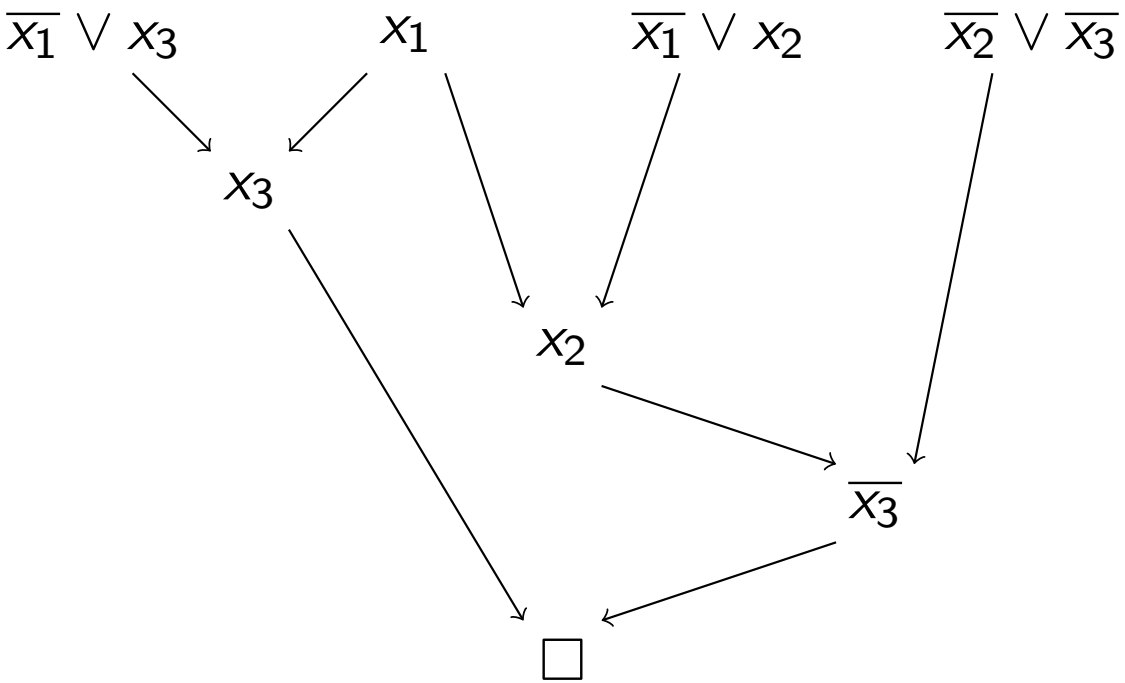
- Assignment $x_1 = 0$, $x_2 = x_3 = 1$:
  - satisfies $(\overline{x_1} \vee x_3)$ and $(\overline{x_1} \vee x_2)$
  - falsifies (x_1) and $(\overline{x_2} \vee \overline{x_3})$ and therefore ϕ
- How to certify that ϕ is unsatisfiable?

  $\rightarrow$ With a resolution refutation

Resolution Refutation

Resolution Refutation

Sequence of resolution steps from the initial formula to the empty clause.



Max-SAT Problem

Max-SAT Problem

Given a CNF formula ϕ, solving the Max-SAT problem consists in determining the maximum (resp. minimum) number of clauses which can be satisfied (resp. falsified) by an assignment of the variables.

Let $\phi = (\overline{x_1} \vee x_3) \wedge (x_1) \wedge (\overline{x_1} \vee x_2) \wedge (\overline{x_2} \vee \overline{x_3})$.

- Assignment $x_1 = 0$, $x_2 = x_3 = 1$:
  - satisfies two clauses $(\overline{x_1} \vee x_3)$ and $(\overline{x_1} \vee x_2)$
  - falsifies two clauses (x_1) and $(\overline{x_2} \vee \overline{x_3})$
- Assignment $x_1 = x_2 = x_3 = 0$:
  - satisfies three clauses $(\overline{x_1} \vee x_3)$, $(\overline{x_1} \vee x_2)$ and $(\overline{x_2} \vee \overline{x_3})$
  - falsifies one clause (x_1)
  - is optimal

Contributions

How to certify the Max-SAT optimum of ϕ ?

$\rightarrow$ By exhibiting a transformation from ϕ into an equivalent $\phi' \wedge \underbrace{\square \wedge \cdots \wedge \square}_{opt(\phi)}$ with ϕ' satisfiable and a model for ϕ'.

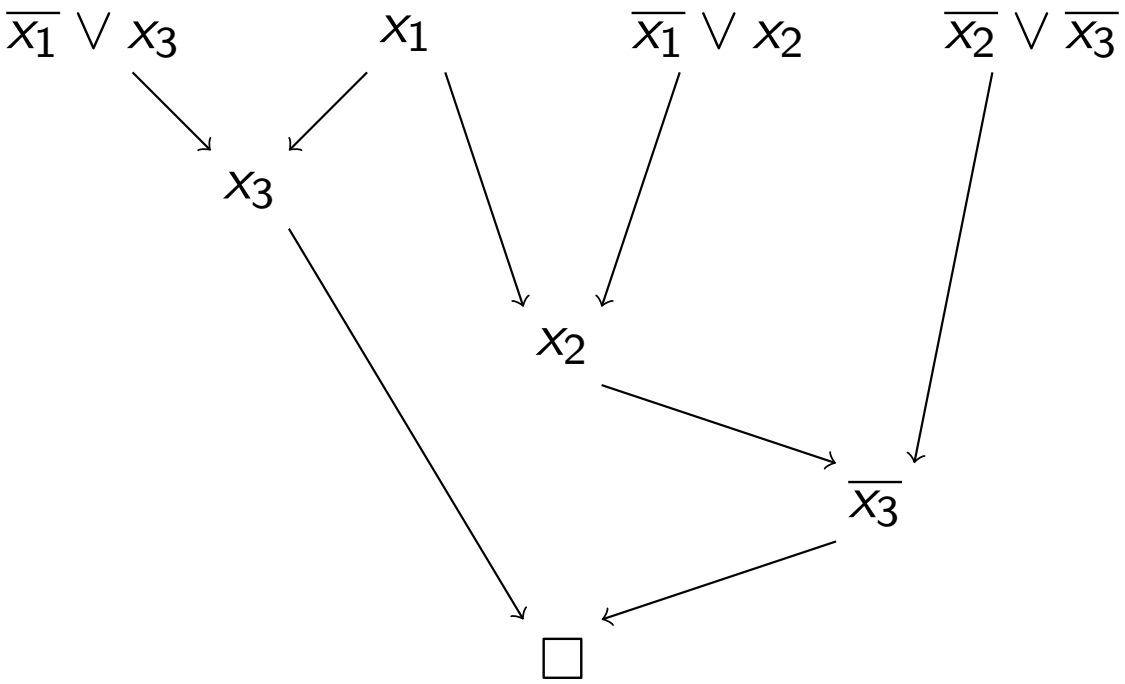
Contributions

- MS-Builder: a tool to generate certificates for the Max-SAT problem
- MS-Checker: a tool to check certificates for the Max-SAT problem

Resolution Refutation

How to generate empty clauses for Max-SAT?

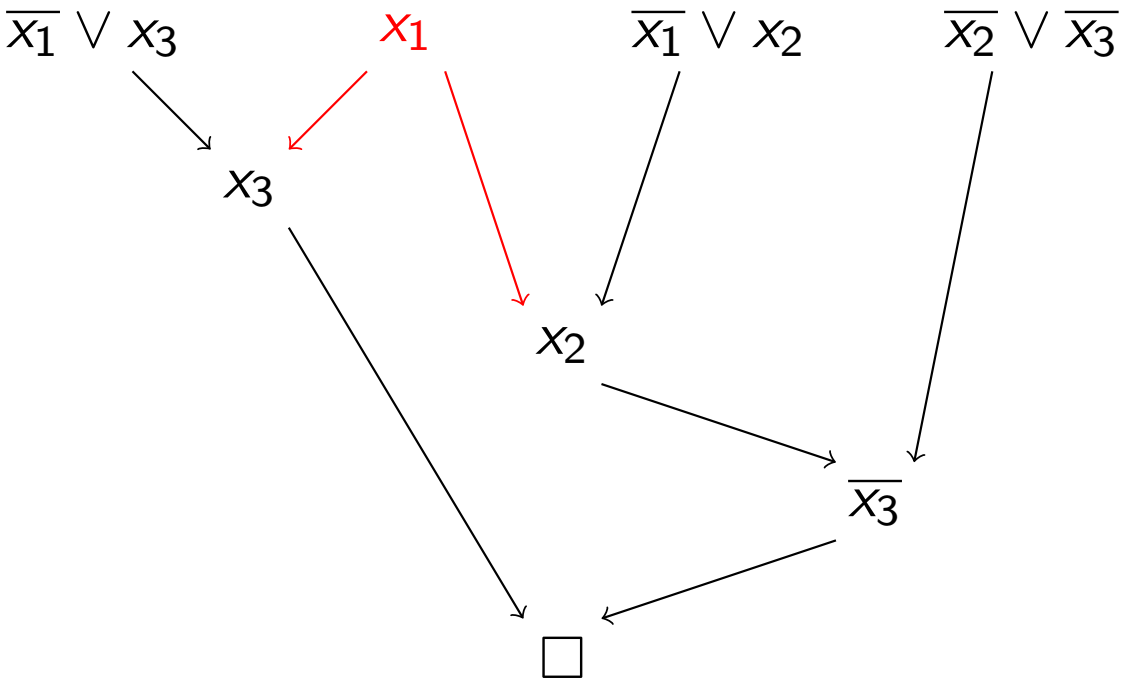
→ By adapting resolution refutations for Max-SAT and applying it on the formula.



Resolution Refutation

Read-Once

A resolution refutation is read-once if each clause is used at most one time as a premise of a resolution step.

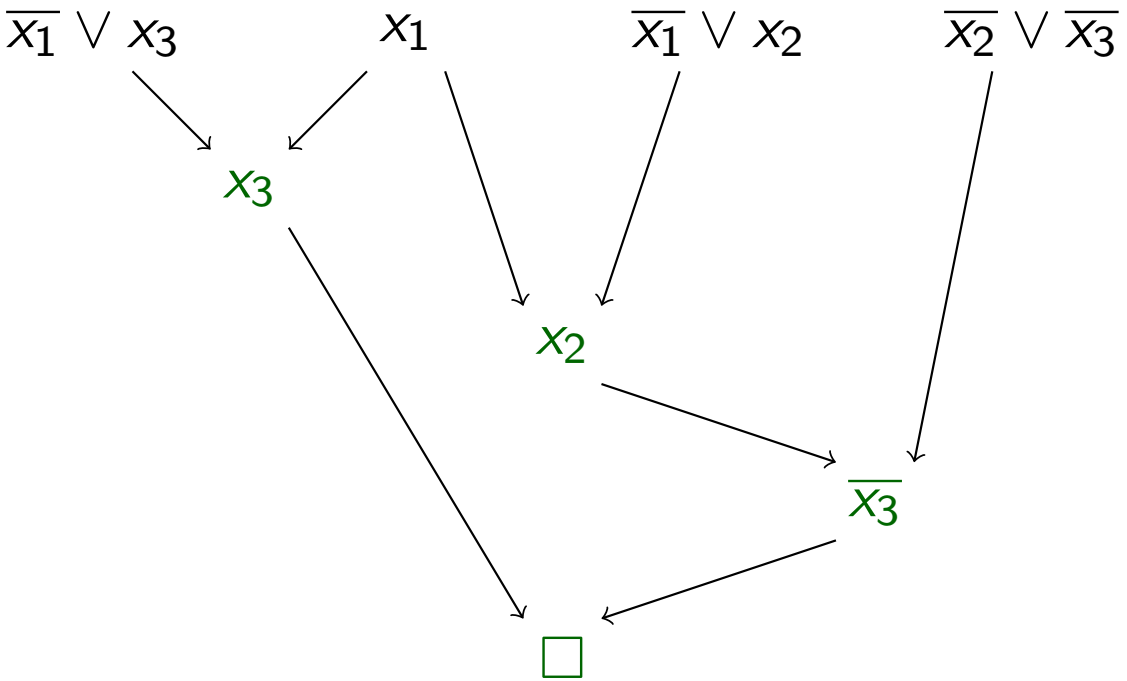


- Read-Once : No
- Tree-Like : Yes
- Regular : Yes

Resolution Refutation

Tree-like

A resolution refutation is tree-like if each derived clause is used at most one time as a premise of a resolution step.

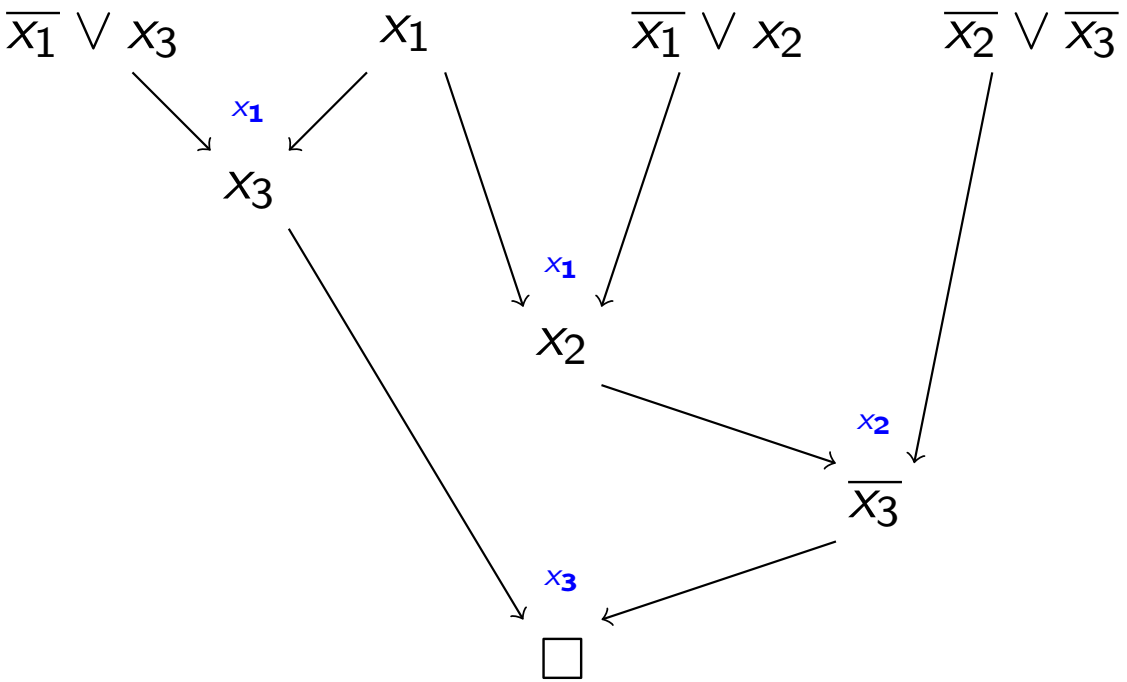


- Read-Once : No
- Tree-Like : Yes
- Regular : Yes

Resolution Refutation

Regular

A resolution refutation is regular if each path from the top to the bottom contains at most one resolution per variable.



- Read-Once : No
- Tree-Like : Yes
- Regular : Yes

Resolution & Max-SAT Resolution

Resolution Rule (Robinson, 1965)

$$\frac{c_1 = (x \vee A) \quad c_2 = (\overline{x} \vee B)}{c_3 = (A \vee B)}$$

Max-SAT Resolution Rule (Bonet & al, 07)(Heras & Larrosa, 06)

$$\frac{(x \vee y_1 \vee \ldots \vee y_s), (\overline{x} \vee z_1 \vee \ldots \vee z_t)}{(y_1 \vee \ldots \vee y_s \vee z_1 \vee \ldots \vee z_t), cc_1, \ldots, cc_t, cc_{t+1}, \ldots, cc_{t+s}}$$

with

$$cc_1 = (x \vee y_1 \vee \ldots \vee y_s \vee \overline{z_1}),$$

$$\ldots$$

$$cc_t = (x \vee y_1 \vee \ldots \vee y_s \vee z_1 \vee \ldots \vee z_{t-1} \vee \overline{z_t}),$$

$$cc_{t+1} = (\overline{x} \vee z_1 \vee \ldots \vee z_t \vee \overline{y_1}),$$

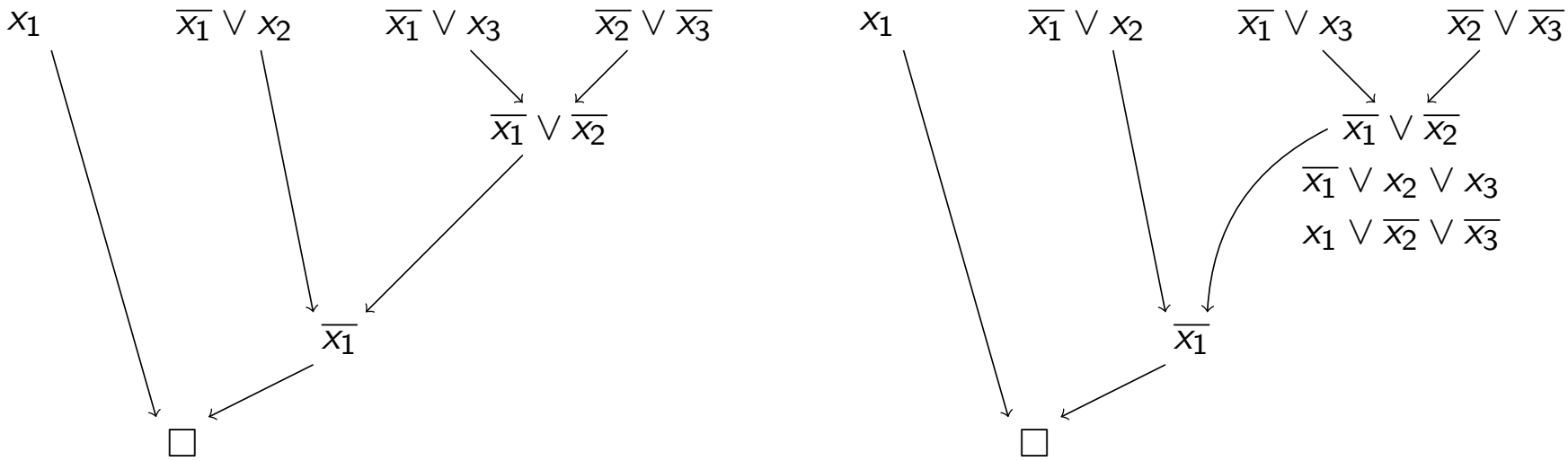
$$\ldots$$

$$cc_{t+s} = (\overline{x} \vee z_1 \vee \ldots \vee z_t \vee y_1 \vee \ldots \vee y_{s-1} \vee \overline{y_s})$$

From a resolution refutation to a Max-SAT refutation

How to adapt a resolution refutation into a Max-SAT refutation?

$\rightarrow$ Easy and well-known if the resolution refutation is read-once [1]

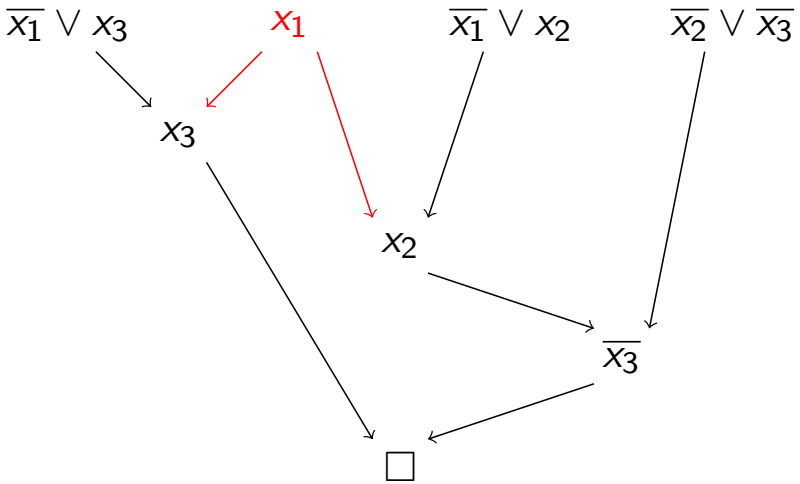


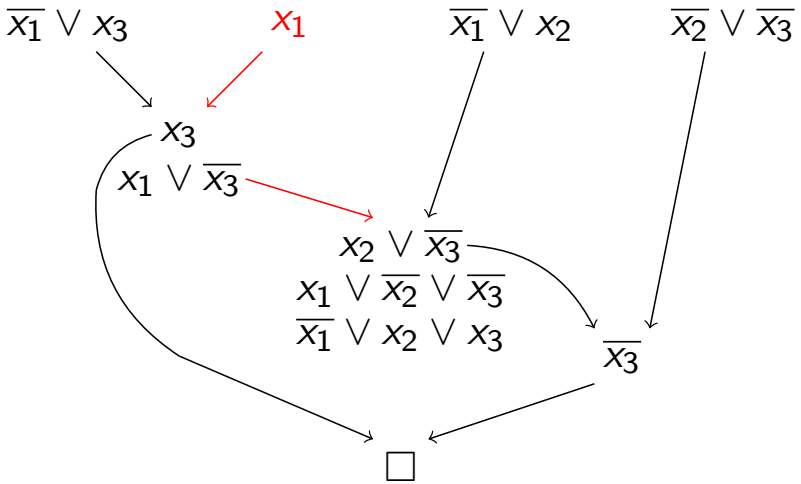
[1] Bonet, Levy & Manyà, 2007 ; Heras & Marques-Silva, 2011

From a Resolution Refutation to a Max-SAT Refutation

How to adapt a resolution refutation into a Max-SAT refutation?

$\rightarrow$ Recent work in the case of non read-once resolution refutation [2]





[2] Py, Cherif & Habet, 2020

Adaptation from SAT to Max-SAT refutations

Resolution Refutation	Size of the Max-SAT Refutation
Read-once	Linear [3]
Tree-like regular	Linear [4]
Tree-like	Linear [4]
Semi-tree-like	Linear [4]
Unrestricted	Exponential [4]

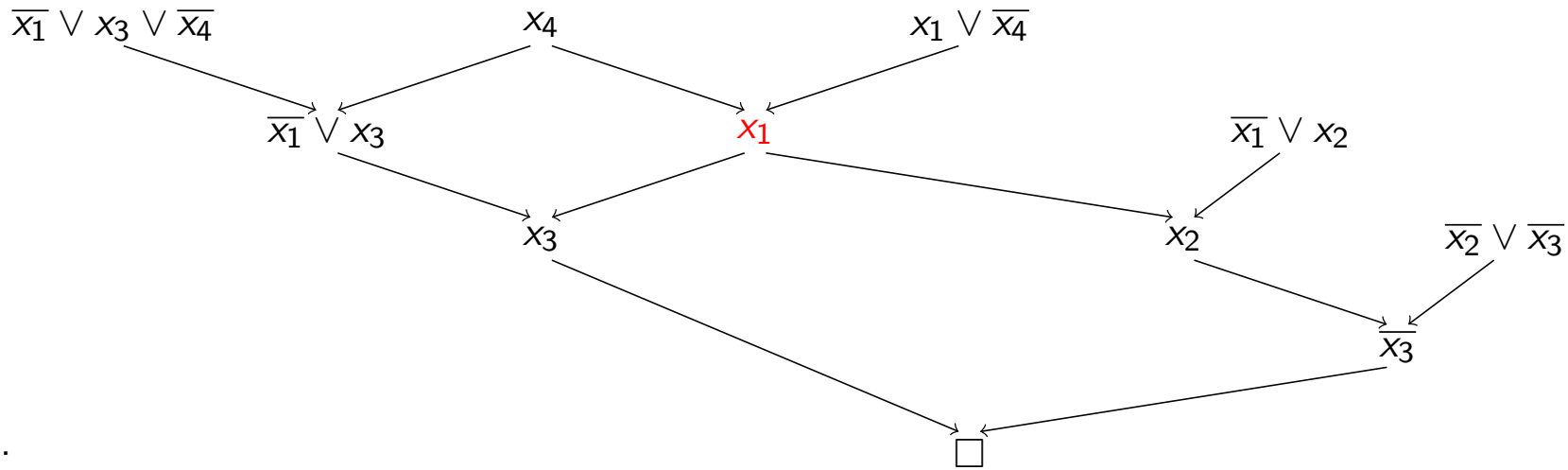
Split rule

Let $c_1 = (A)$ with A a disjunction of literals and x a variable. The split rule deduces two clauses from c_1 as follows:

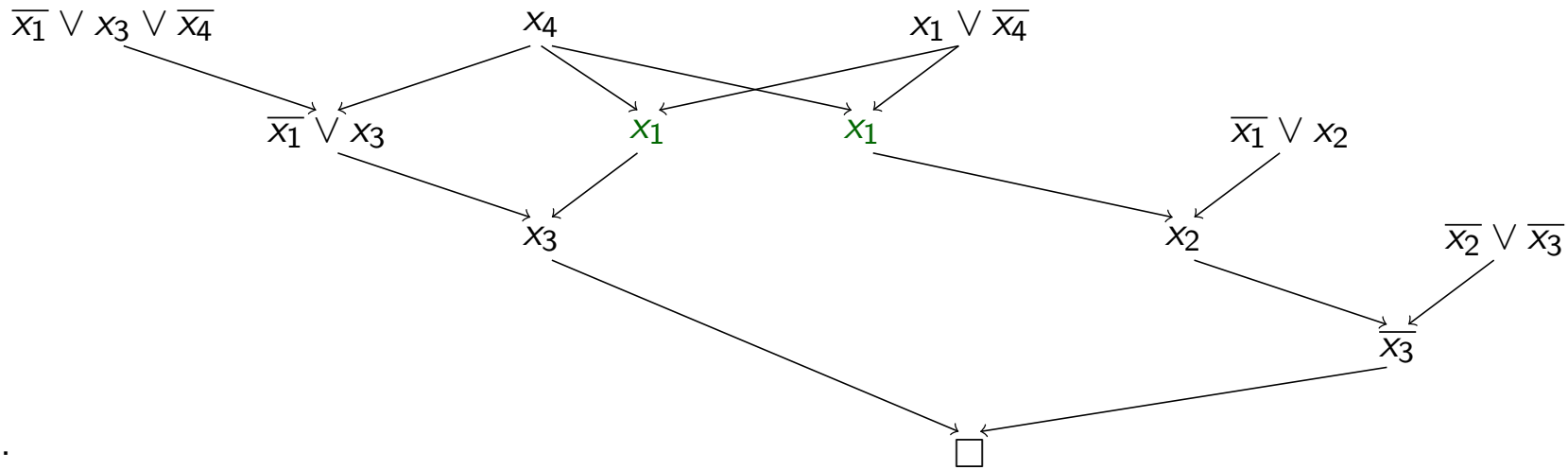
$$\frac{c_1 = (A)}{c_2 = (x \vee A) \quad c_3 = (\overline{x} \vee A)}$$

[3] Bonet, Levy & Manyà, 2007 ; Heras & Marques-Silva, 2011
[4] Py, Cherif & Habet, 2020

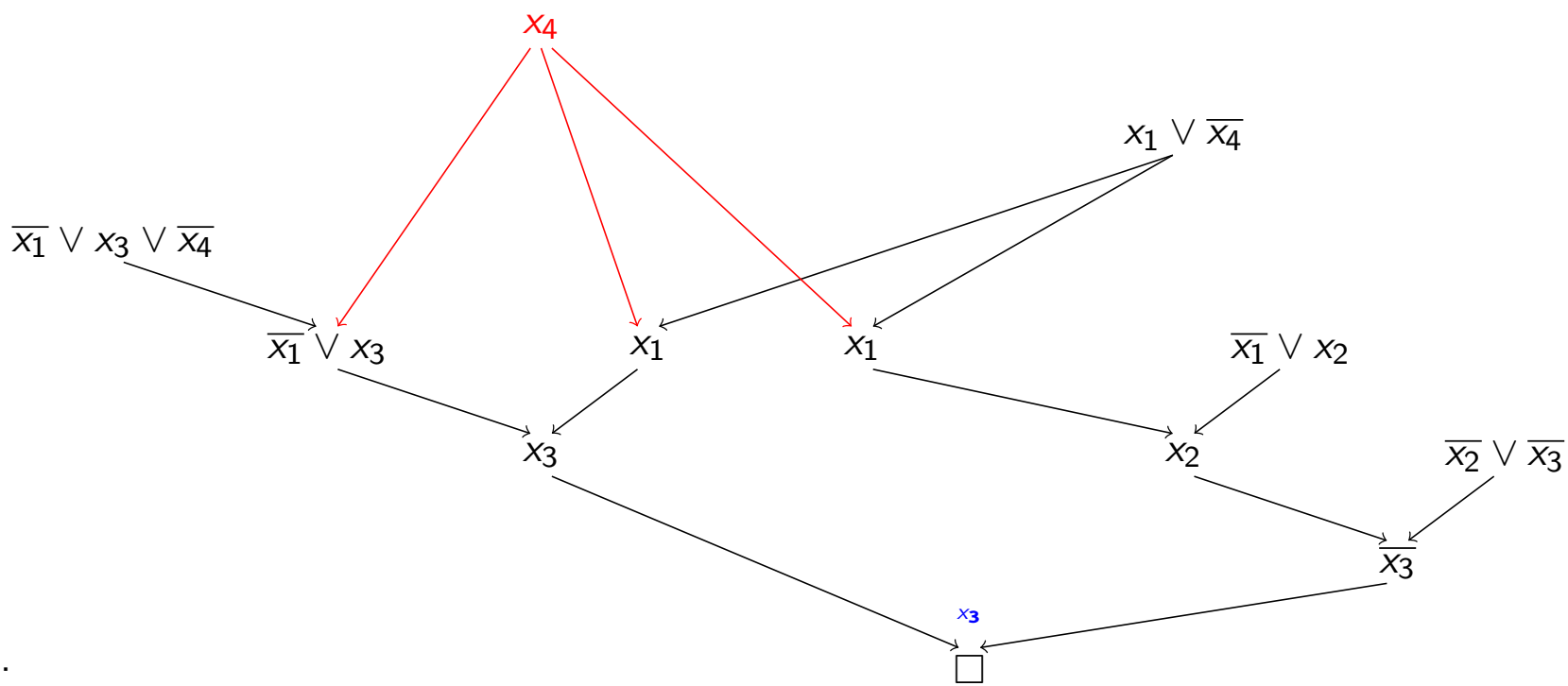
Adaptation From SAT to Max-SAT Refutations [Py, Cherif and Habet, 2020]



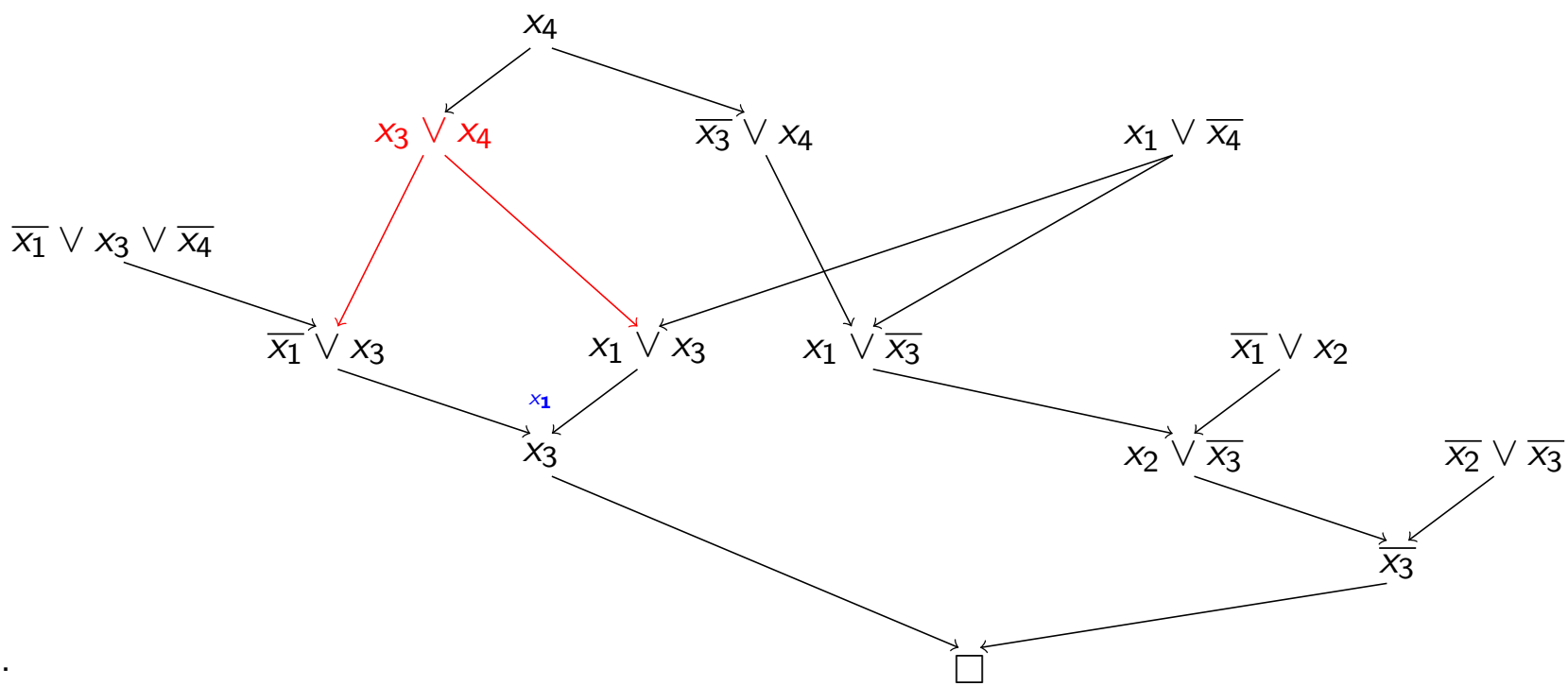
Adaptation From SAT to Max-SAT Refutations [Py, Cherif and Habet, 2020]



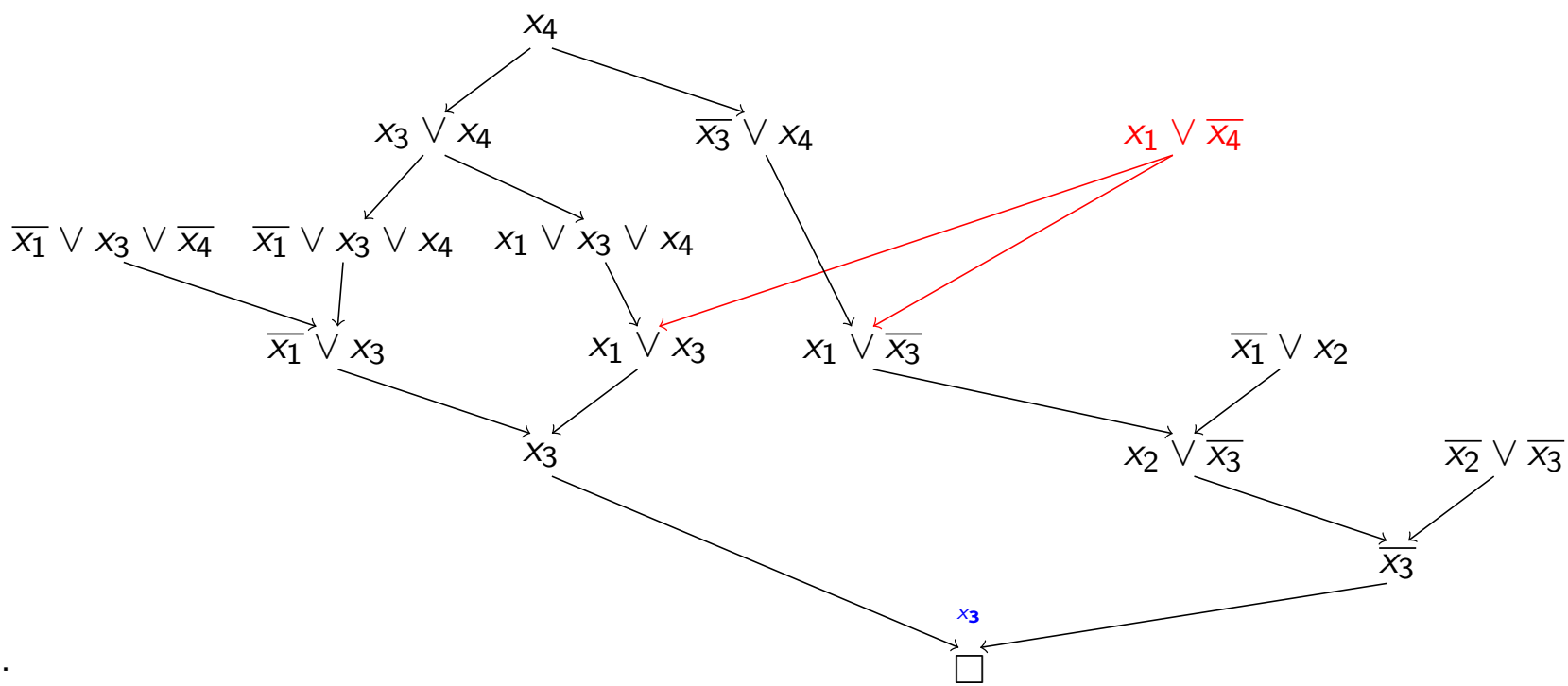
Adaptation From SAT to Max-SAT Refutations [Py, Cherif and Habet, 2020]



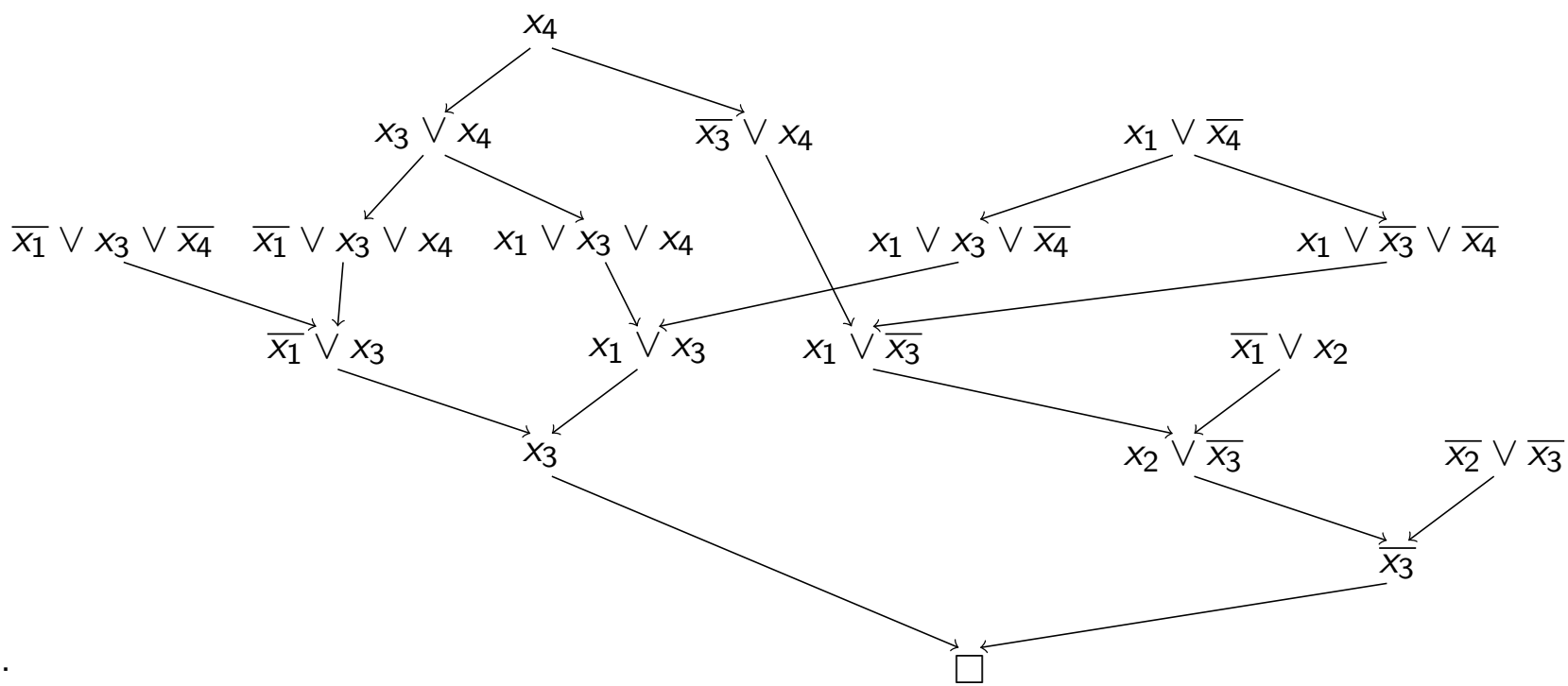
Adaptation From SAT to Max-SAT Refutations [Py, Cherif and Habet, 2020]



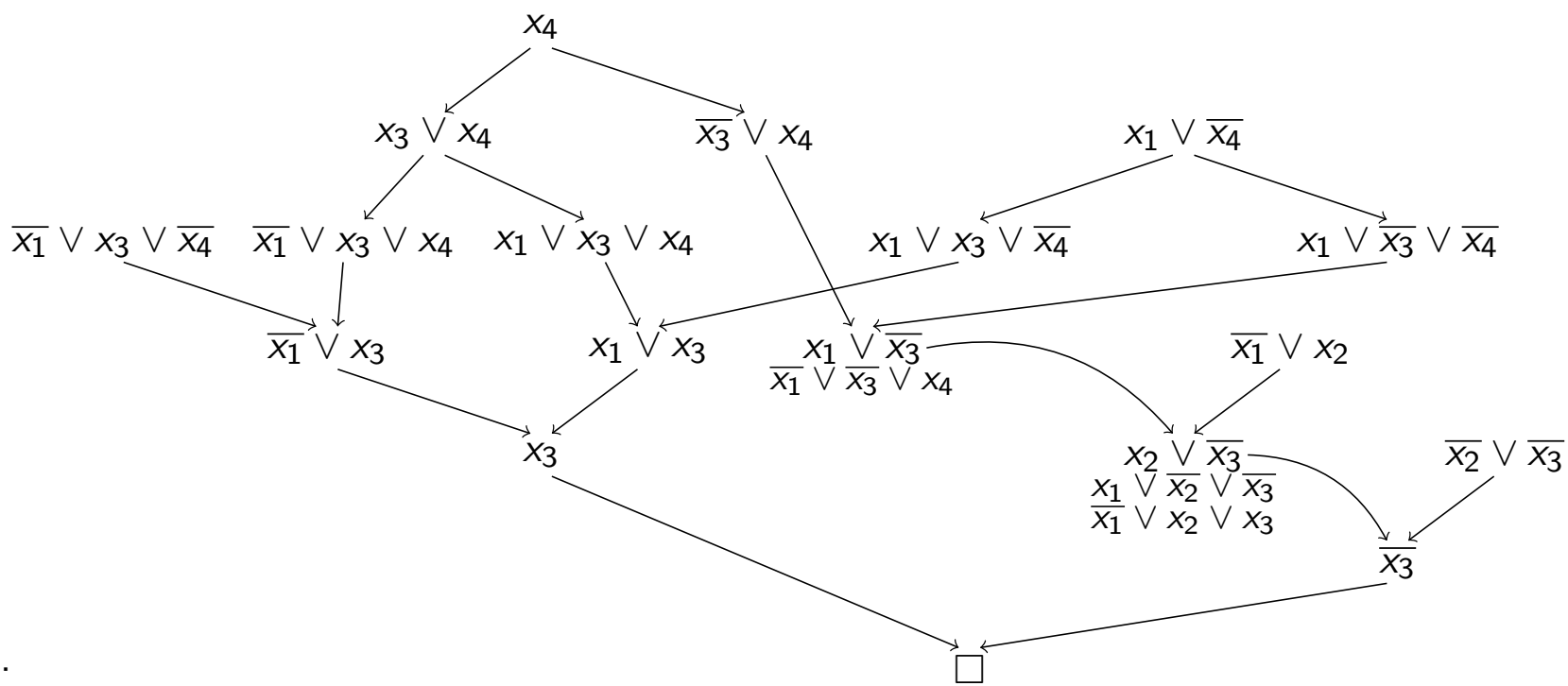
Adaptation from SAT to Max-SAT Refutations [Py, Cherif and Habet, 2020]



Adaptation from SAT to Max-SAT Refutations [Py, Cherif and Habet, 2020]



Adaptation from SAT to Max-SAT Refutations [Py, Cherif and Habet, 2020]



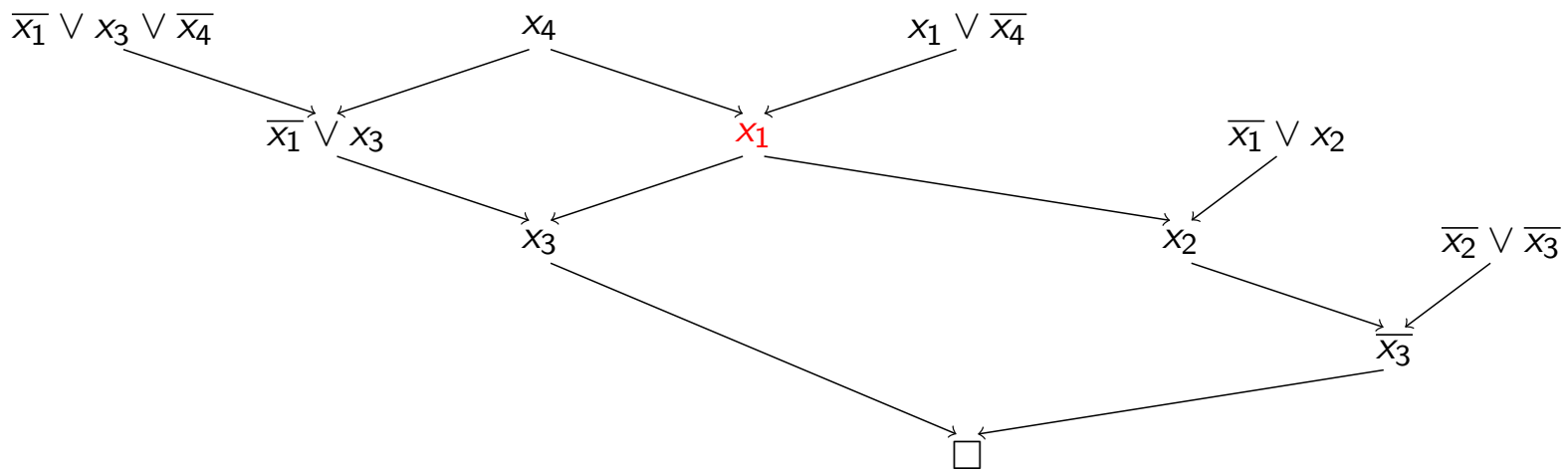
MS-Builder Algorithm

Algorithm 1 MS-Builder Algorithm

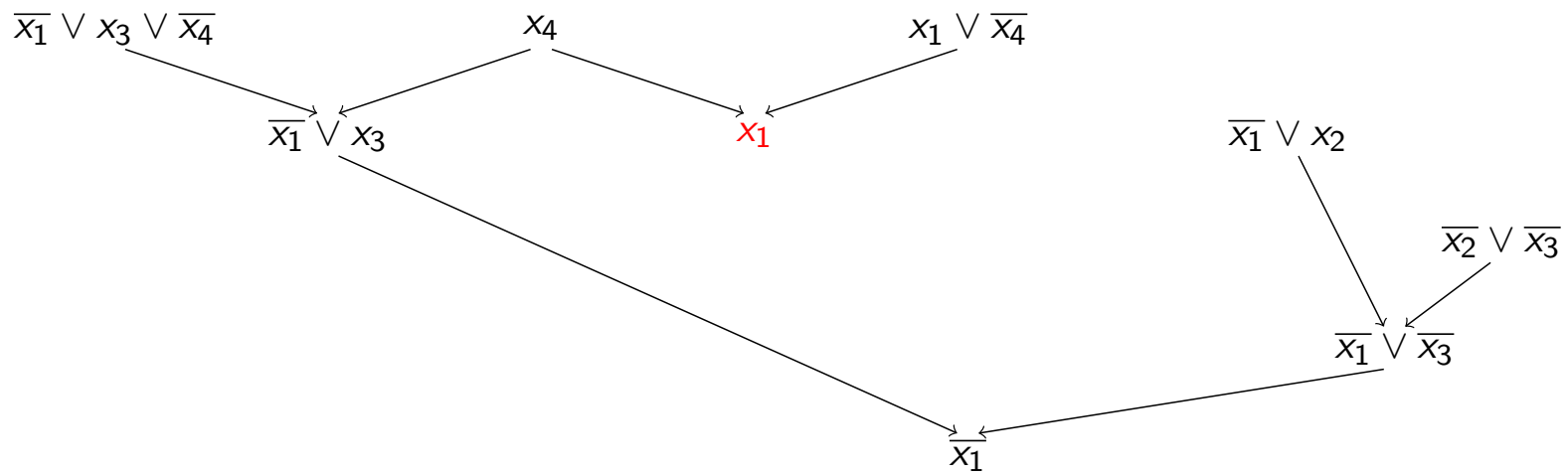
Require: CNF formula ϕ
Ensure: certificate C for ϕ

1: $(T, opt) \leftarrow (\emptyset, 0)$
2: **while** ϕ is unsatisfiable **do**
3: $RP \leftarrow$ compute_resolution_refutation(ϕ)
4: $MRP \leftarrow$ adapt_resolution_refutation_for_MaxSAT(RP)
5: $\phi \leftarrow$ apply_maxsat_refutation(ϕ, MRP)
6: $(\phi, opt) \leftarrow$ remove_empty_clauses(ϕ, opt)
7: $T \leftarrow T.MRP$
8: **end while**
9: $I \leftarrow$ compute_model(ϕ)
10: **return** (T, opt, I)

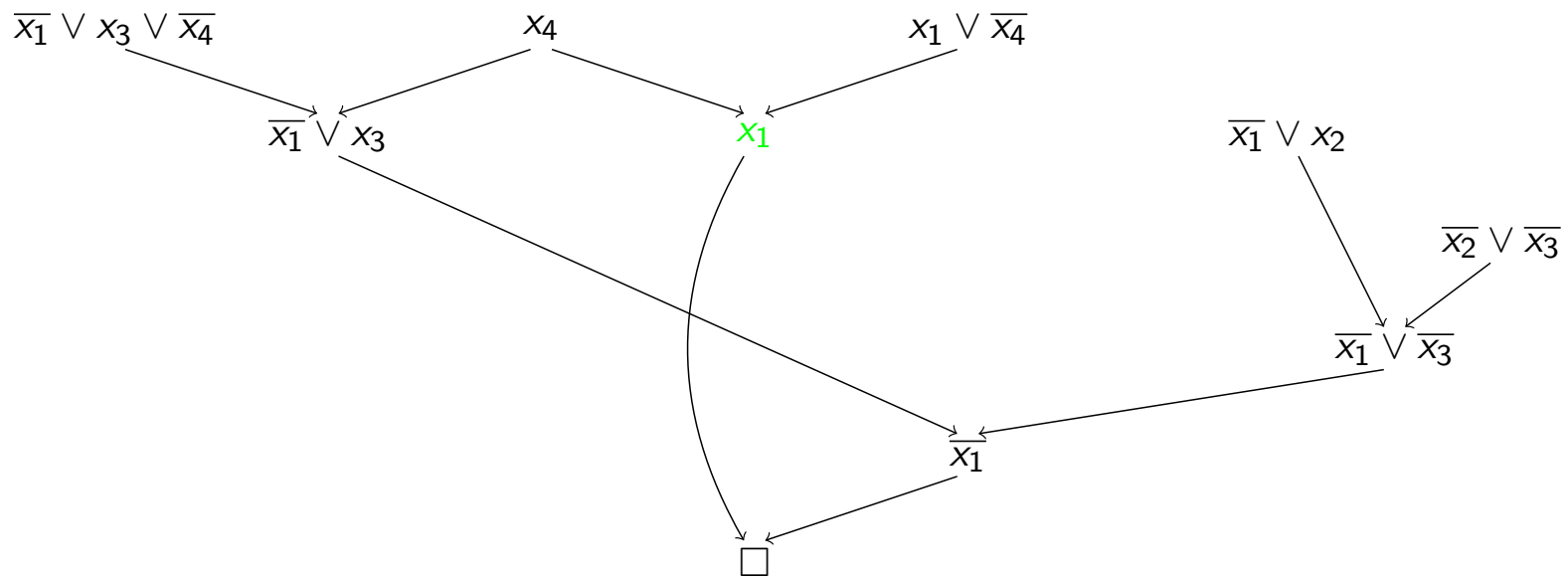
Heuristic to Fix Unit-Propagation



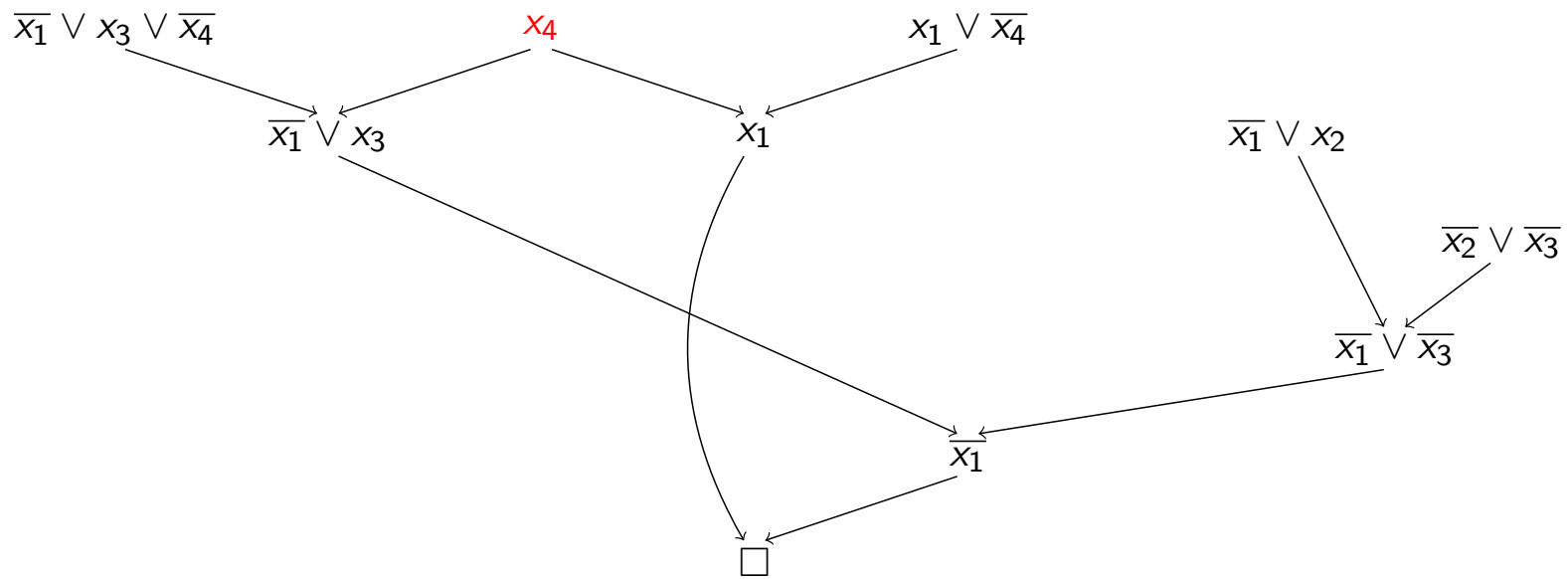
Heuristic to Fix Unit-Propagation



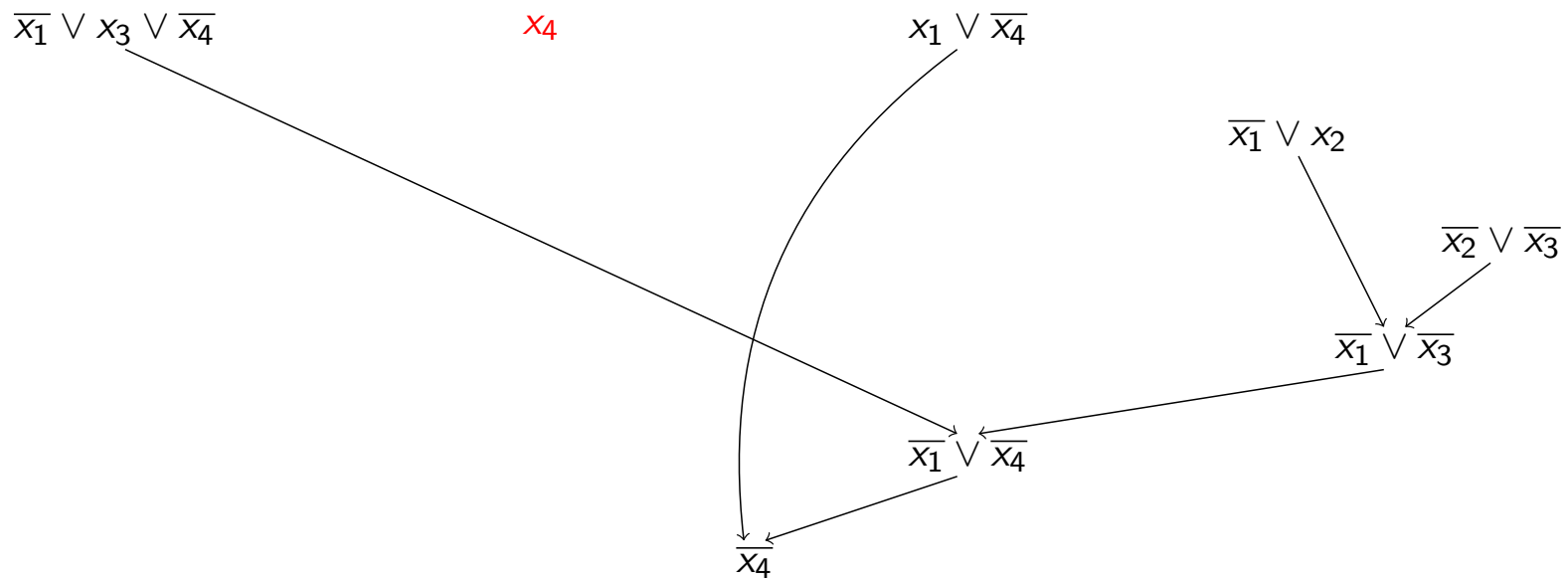
Heuristic to Fix Unit-Propagation



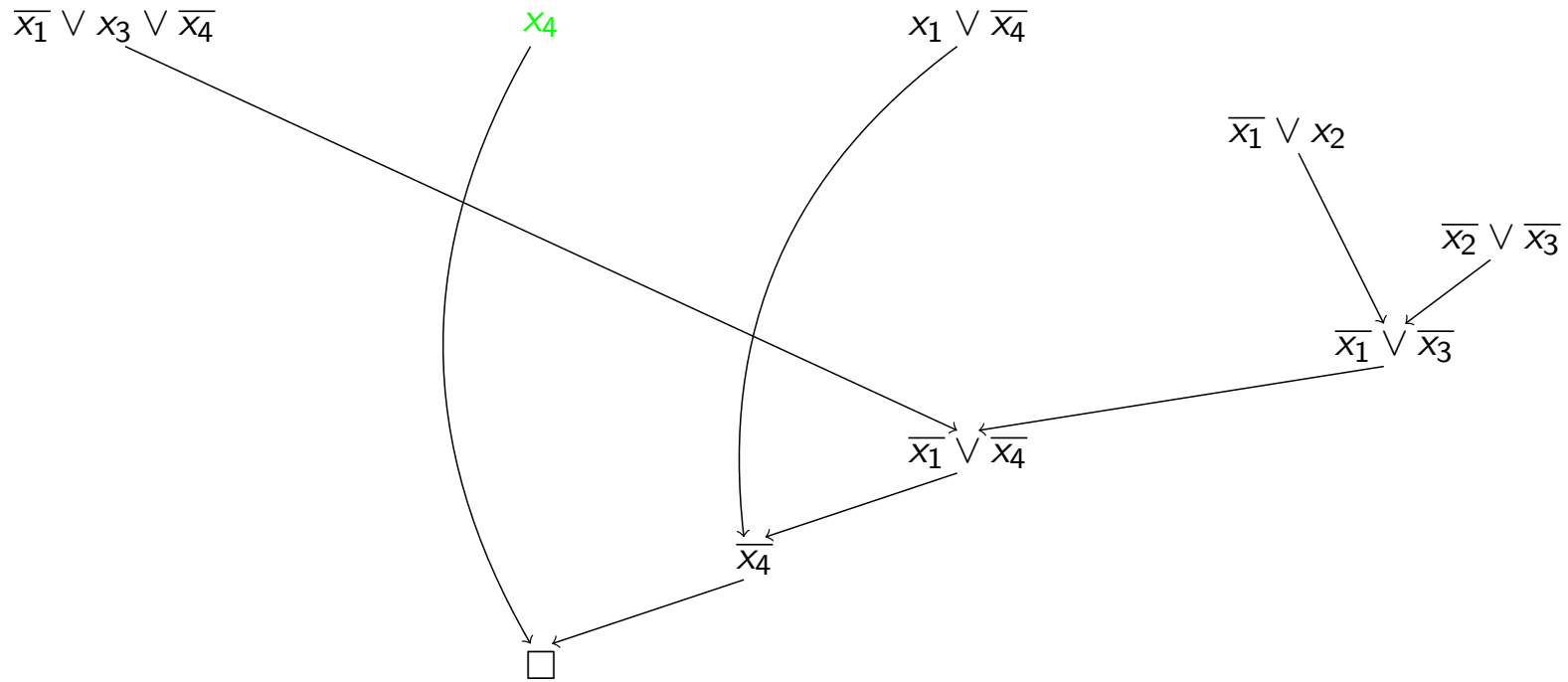
Heuristic to Fix Unit-Propagation



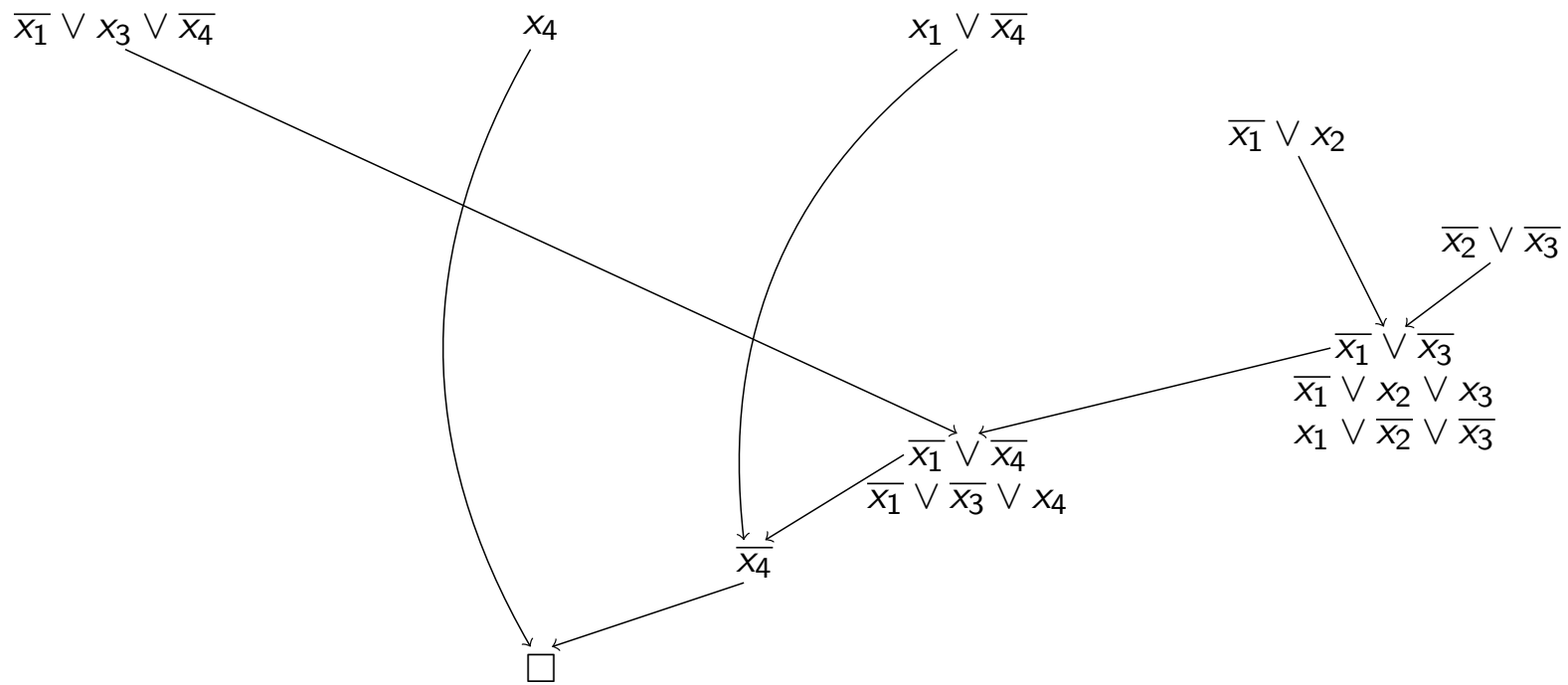
Heuristic to Fix Unit-Propagation



Heuristic to Fix Unit-Propagation



Heuristic to Fix Unit-Propagation

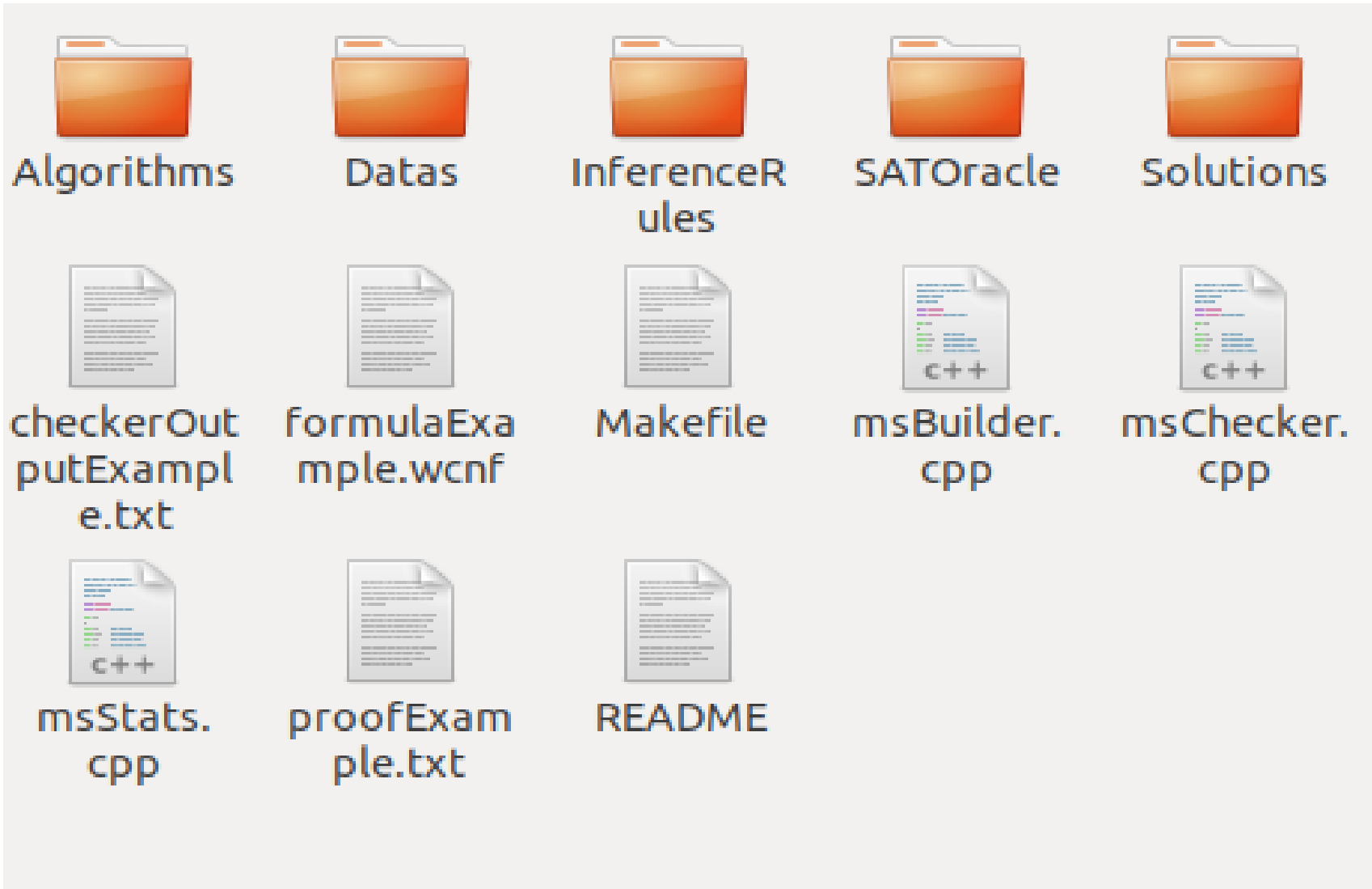


Implementation

- C++ language
- SAT Oracle: Booleforce (+Tracecheck) [5]
- Source code available on https://pageperso.lis-lab.fr/matthieu.py/en/software.html
- Experiments on the unweighted partial instances of the 2020 Max-SAT Evaluation [6]

[5] Armin Biere, http://fmv.jku.at/booleforce/
[6] https://maxsat-evaluations.github.io/2020/

Files of the Tools



Formula File

```
c Example of CNF formula
1 -1 3 0
1 1 0
1 -1 2 0
1 -2 -3 0
```

Certificate File

```
c resolution proof was semi-read-once
t msres < 1.000000 -1 3 | 1.000000 -2 -3 >
t msres < 1.000000 -1 -2 | 1.000000 -1 2 >
t msres < 1.000000 1 | 1.000000 -1 >
o 1
v 010
c executionTime 0.0024 seconds
```

```
Proof has been checked with success in 0.000474 seconds.
```

Execution Time for MS-Builder and MS-Checker

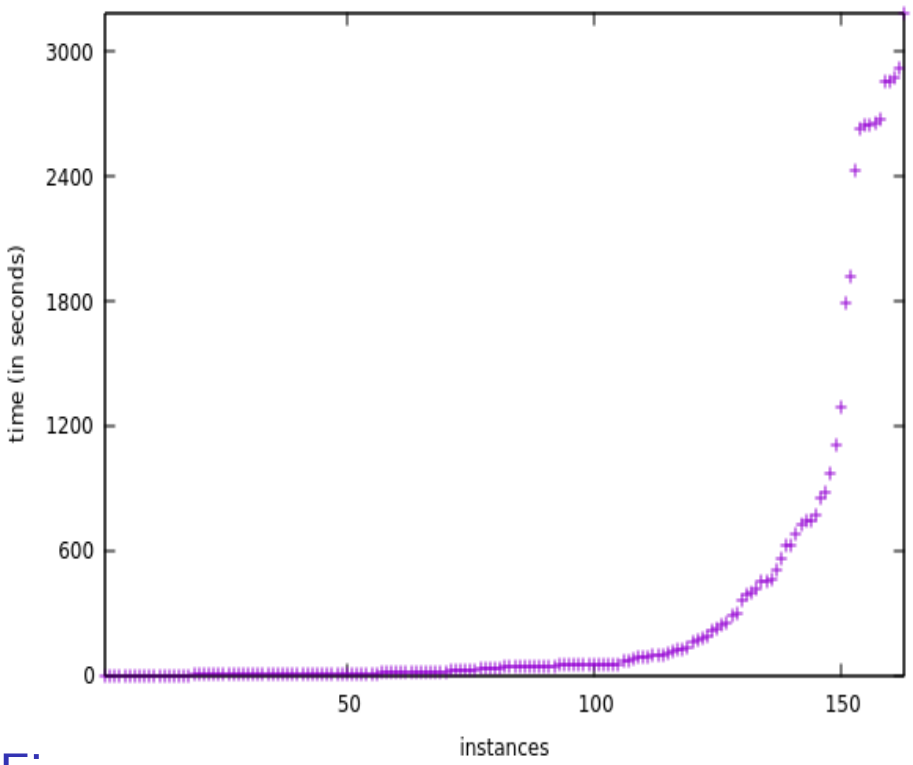


Figure: Running time (in seconds) for building complete proofs

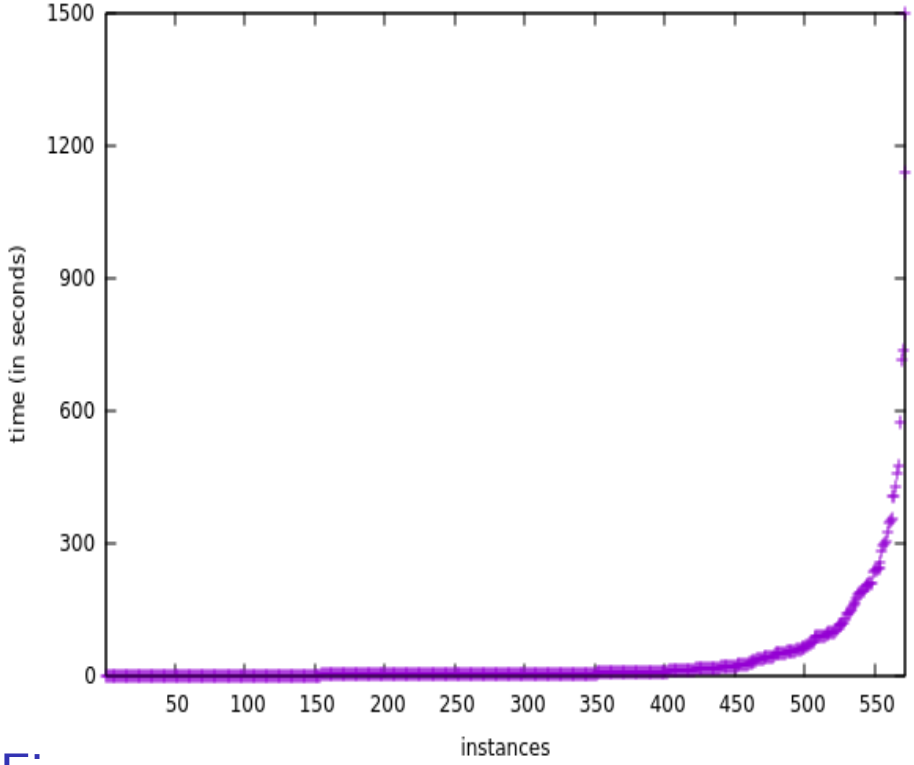


Figure: Running time (in seconds) for checking proof

Proof Percentages and Sizes

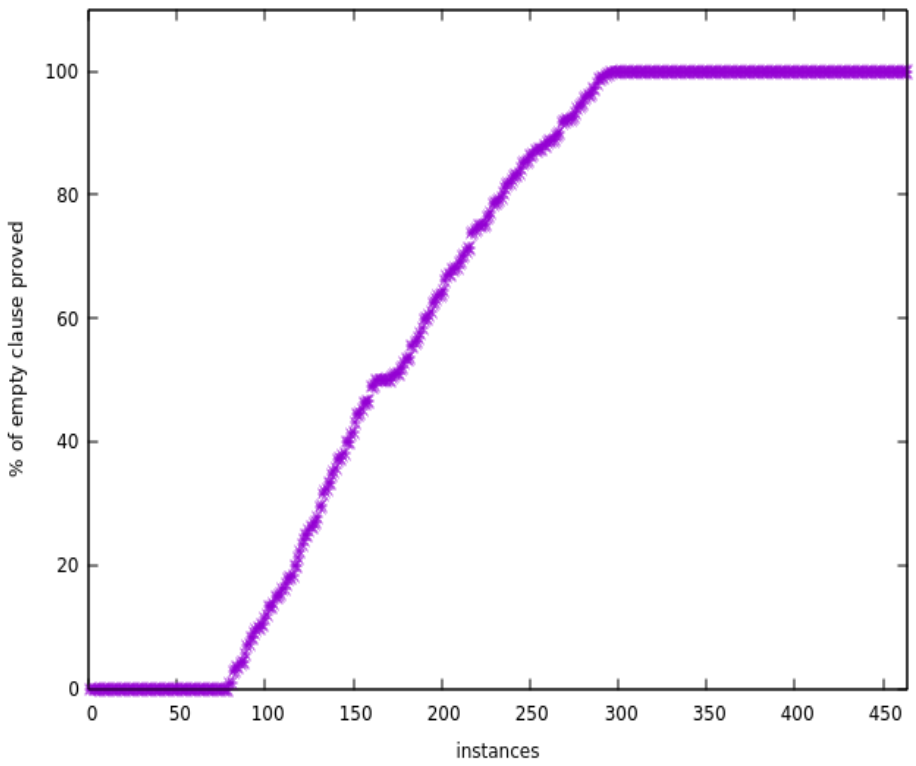


Figure: Percentage of proved □ per instance

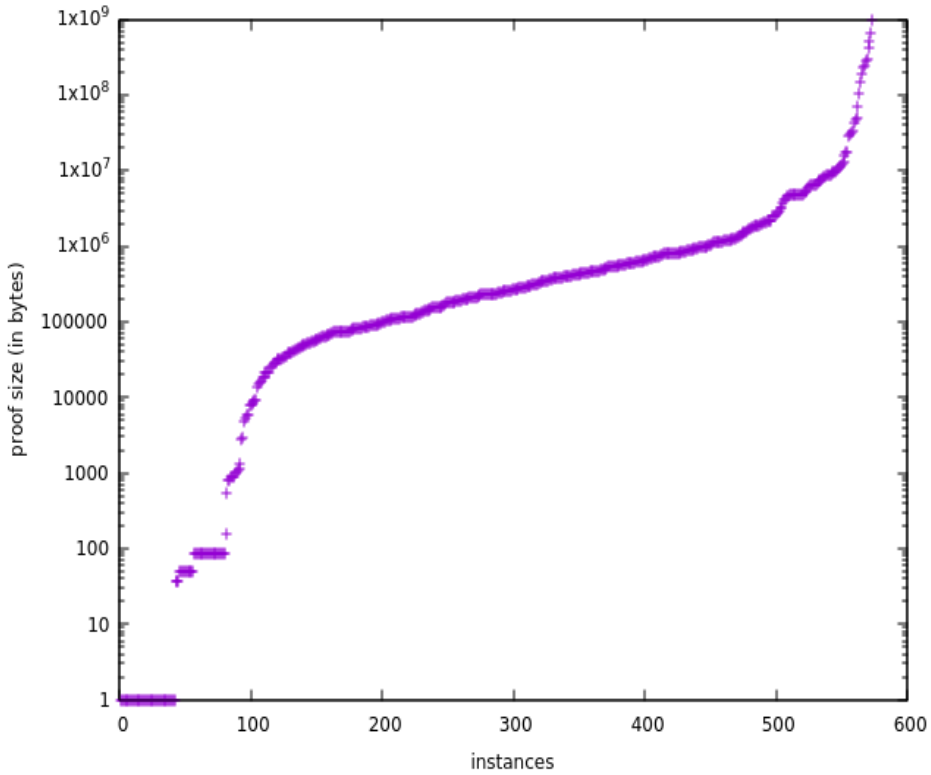


Figure: Size of proof per instance (in logarithmic scale)

Encountered Resolution Refutations

Type of resolution refutation	Number	Percentage
read-once	169,239	83.7 %
read-once after UP-fixing	24,556	12.1 %
tree-like regular	2,879	1.4 %
tree-like	1,795	0.9 %
unrestricted	3,799	1.9 %

Table: Encountered types of resolution refutations in the whole benchmark

Conclusions

Contributions

- MS-Builder: a tool to generate certificates for the Max-SAT problem
- MS-Checker: a tool to check certificates for the Max-SAT problem

Future Work

- Improve theoretical background for MS-Builder
- Improve implementation of MS-Builder & MS-Checker
- Adapt both tools for Weighted (Partial) Max-SAT