Building Bridges between Symbolic Computation and Satisfiability Checking

Erika Ábrahám
RWTH Aachen University, Germany

in cooperation with
Florian Corzilius, Gereon Kremer, Stefan Schupp and others

ISSAC'15, 7 July 2015

Photo: Prior Park, Bath / flickr Liam Gladdy

What is this talk about?

Satisfiability problem

The satisfiability problem is the problem of deciding whether a logical formula is satisfiable.

We focus on the automated solution of the satisfiability problem for first-order logic over arithmetic theories, especially on similarities and differences in

- symbolic computation and
- SAT and SMT solving.

1940

Decision procedures for first-order logic over arithmetic theories in mathematical logic

Computer architecture development

1960

1970

1980

2000

2010

Decision procedures for first-order logic over arithmetic theories in mathematical logic

1940 Computer architecture development

CAS

1960 Computer algebra systems

1970

CAD

1980

Partial CAD

Virtual substitution

2000

2010

Decision procedures for first-order logic over arithmetic theories in mathematical logic

1940 Computer architecture development

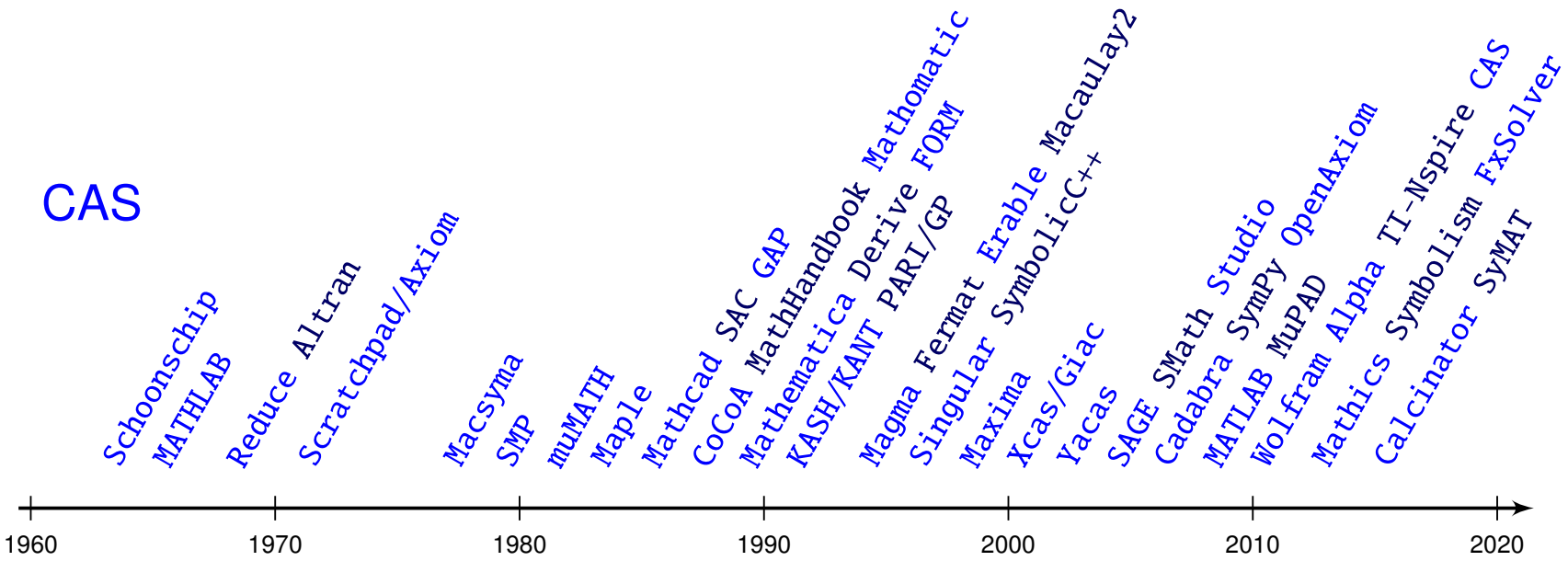
CAS	SAT (propositional logic)
	Enumeration
1960 Computer algebra systems	DP (resolution) [Davis, Putnam'60]
	DPLL (propagation) [Davis,Putnam,Logemann,Loveland'62]
1970	NP-completeness [Cook'71]
CAD	Conflict-directed backjumping
1980	
Partial CAD	
Virtual substitution	CDCL [GRASP'97] [zChaff'04]
2000	Watched literals
	Clause learning/forgetting
	Variable ordering heuristics
	Restarts
2010	

Decision procedures for first-order logic over arithmetic theories in mathematical logic

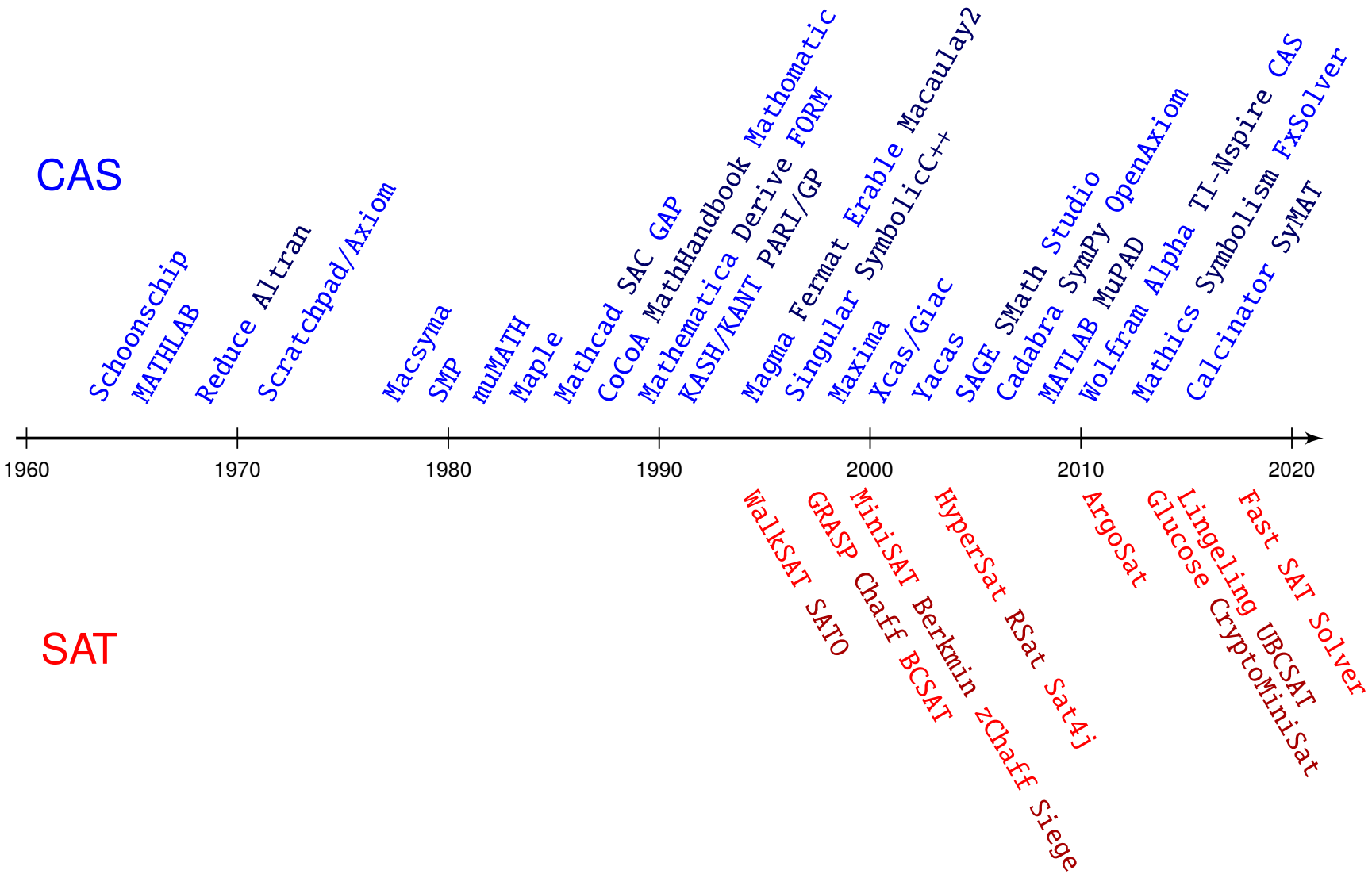
1940 Computer architecture development

	CAS	SAT (propositional logic)	SMT (SAT modulo theories)
1960	Computer algebra systems	Enumeration DP (resolution) [Davis, Putnam'60]	
1970		DPLL (propagation) [Davis,Putnam,Logemann,Loveland'62] NP-completeness [Cook'71]	Decision procedures for combined theories [Shostak'79] [Nelson, Oppen'79]
1980	CAD	Conflict-directed backjumping	
	Partial CAD		
2000	Virtual substitution	CDCL [GRASP'97] [zChaff'04] Watched literals Clause learning/forgetting Variable ordering heuristics Restarts	DPLL(T) Equalities and uninterpreted functions Bit-vectors Array theory Arithmetic
2010			

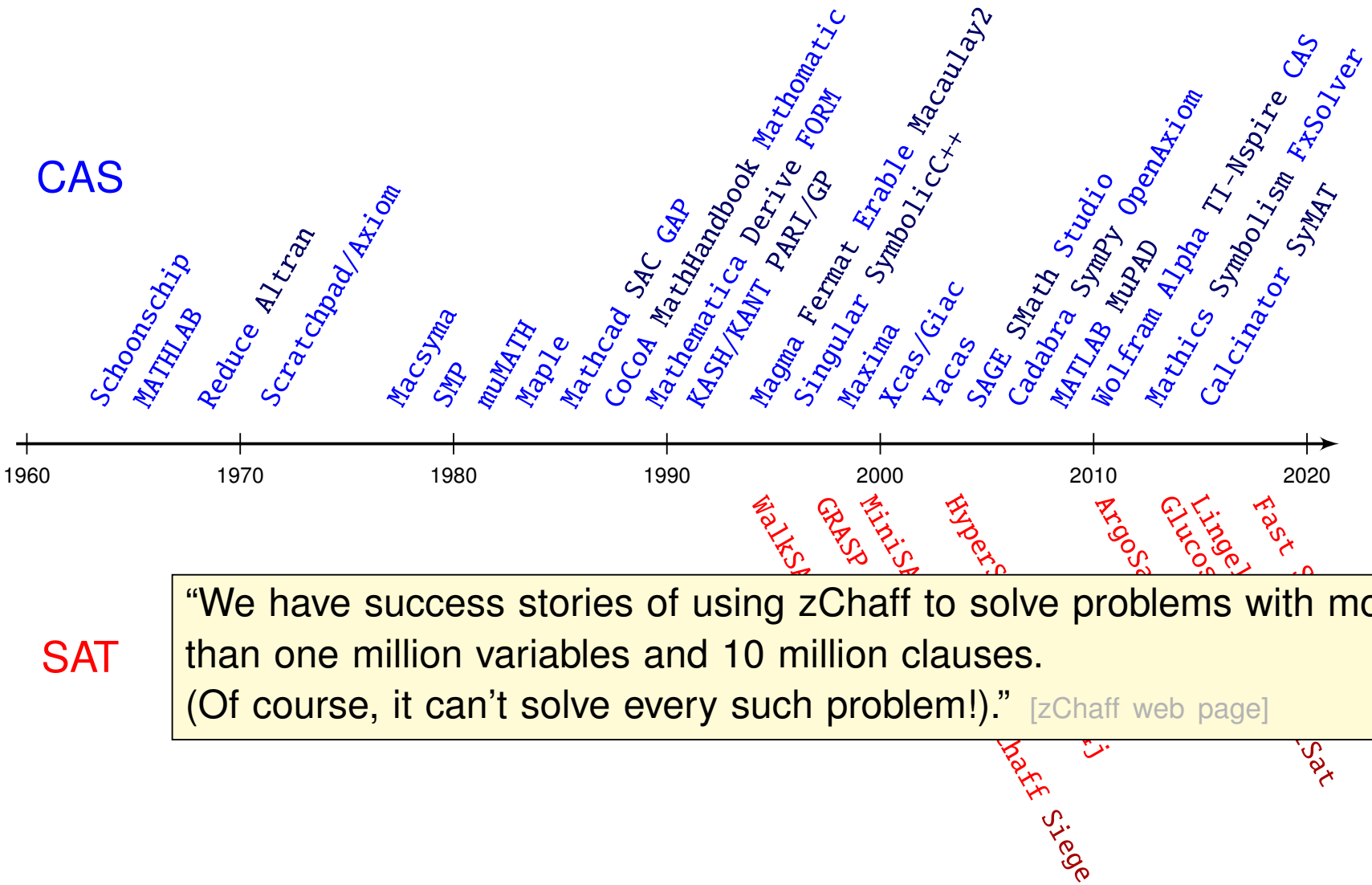
Tool development



Tool development

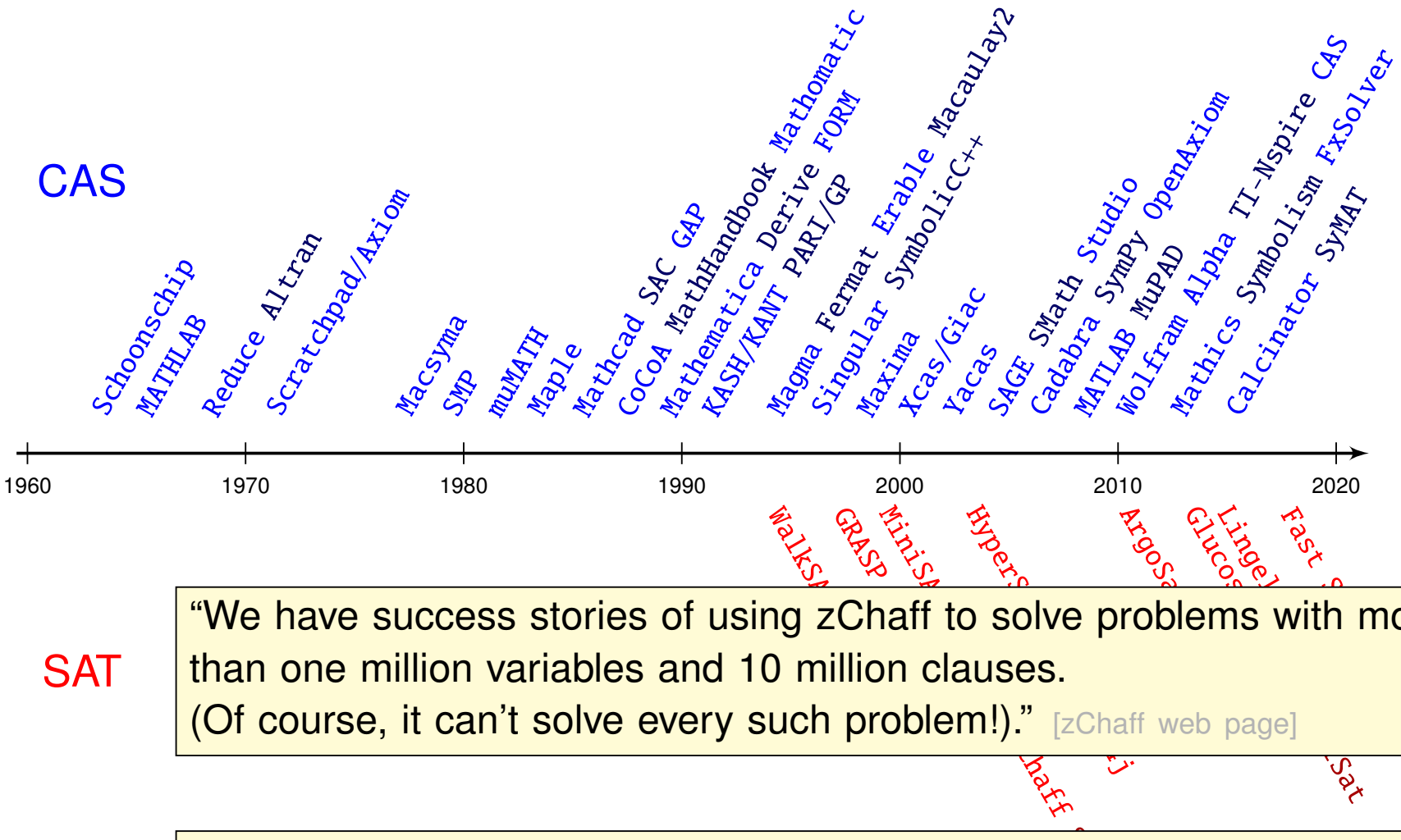


Tool development



"We have success stories of using zChaff to solve problems with more than one million variables and 10 million clauses.
(Of course, it can't solve every such problem!)." [zChaff web page]

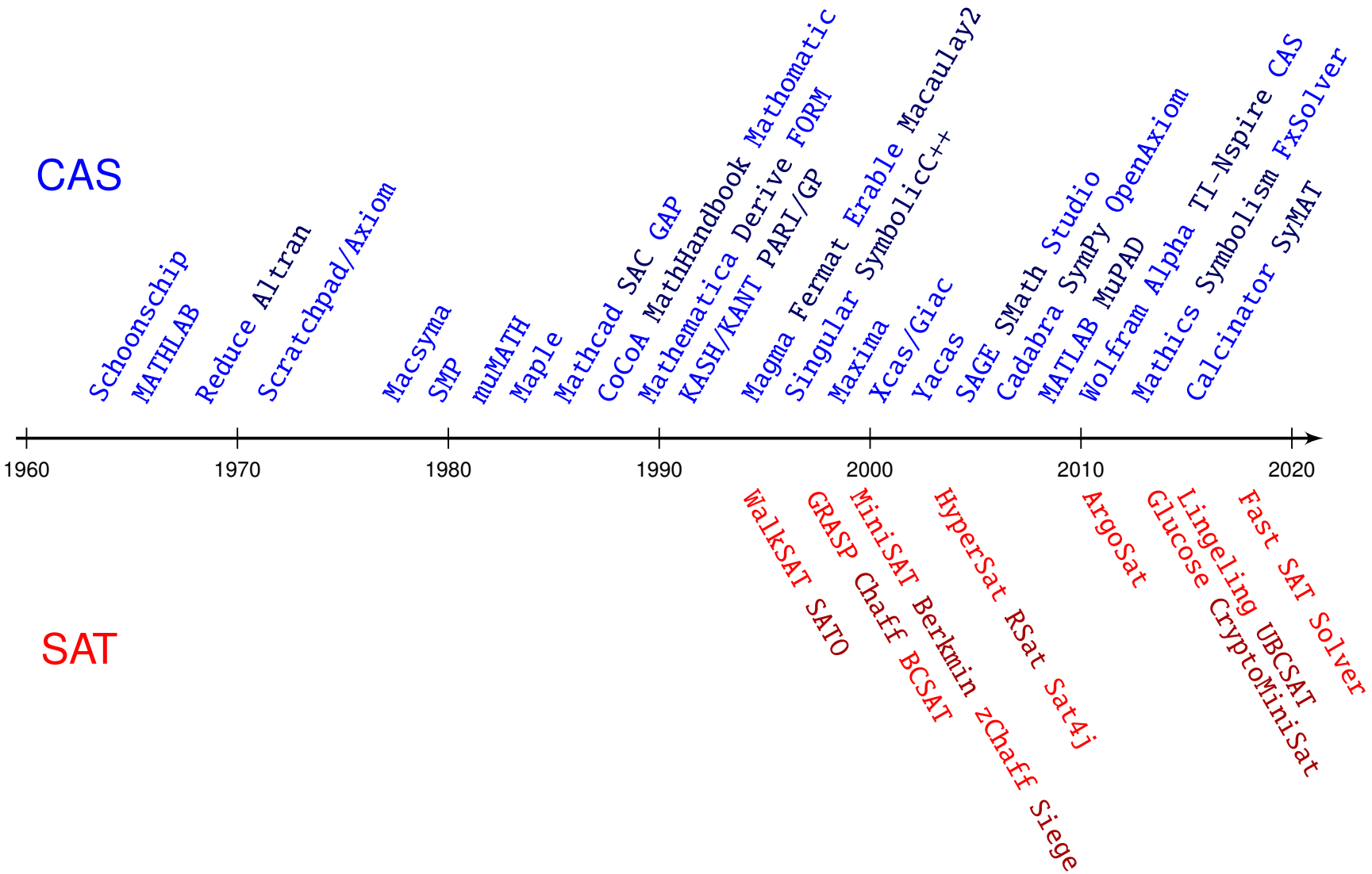
Tool development



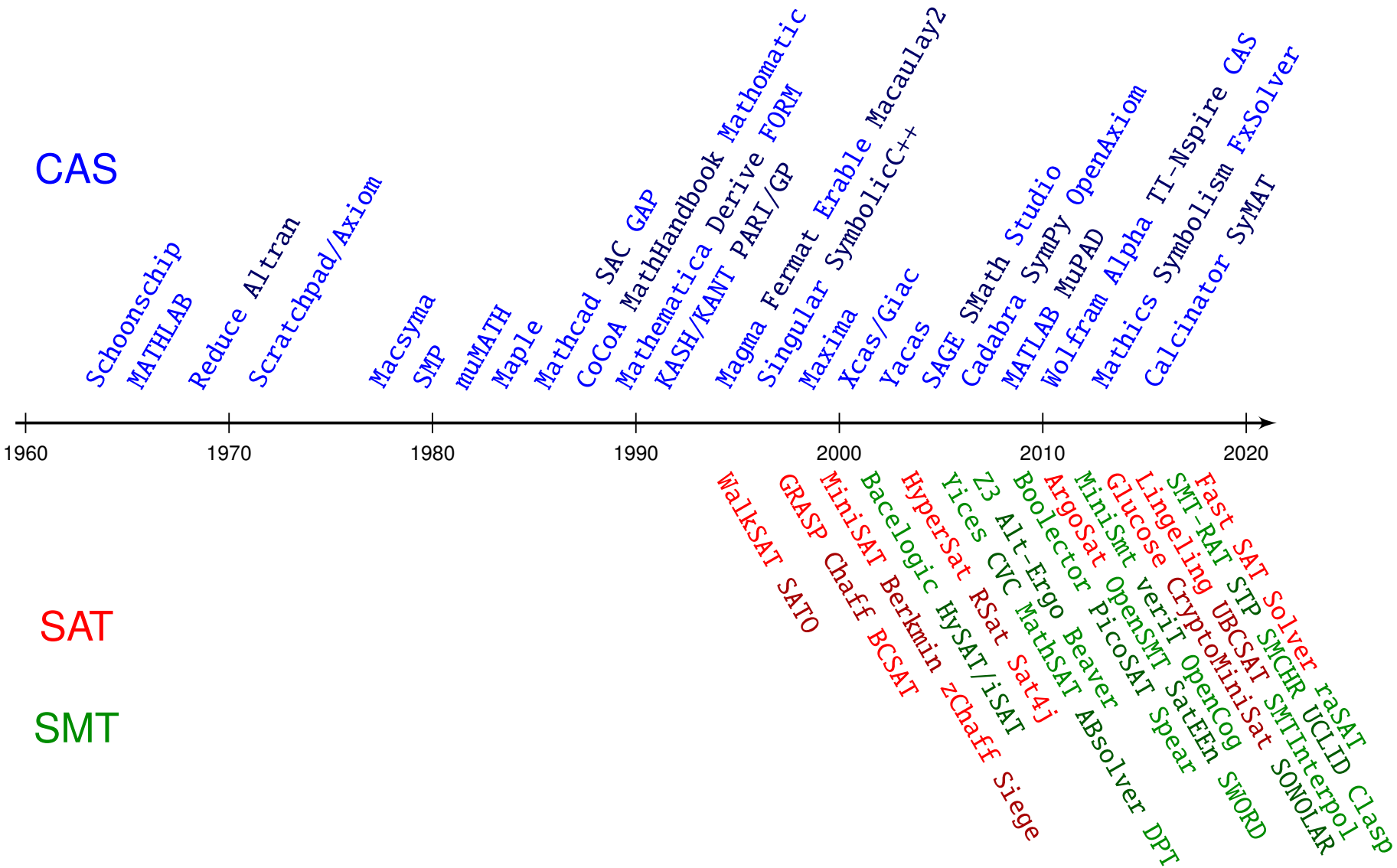
"We have success stories of using zChaff to solve problems with more than one million variables and 10 million clauses.
(Of course, it can't solve every such problem!)." [zChaff web page]

" The efficiency of our programs allowed us to solve over one hundred open quasigroup problems in design theory." [SATO web page]

Tool development



Tool development



Satisfiability checking for propositional logic

Success story: SAT-solving

- Practical problems with millions of variables are solvable.
- Frequently used in different research areas for, e.g., analysis, synthesis and optimisation.
- Also massively used in industry for, e.g., digital circuit design and verification.

Satisfiability checking for propositional logic

Success story: SAT-solving

- Practical problems with millions of variables are solvable.
- Frequently used in different research areas for, e.g., analysis, synthesis and optimisation.
- Also massively used in industry for, e.g., digital circuit design and verification.

Community support:

- Standardised input language, lots of benchmarks available.
- Competitions since 2002.

  2014 SAT Competition: 3 categories, 79 participants with 137 solvers.

  SAT Live! forum as community platform, dedicated conferences, journals, etc.

Input in CNF: Tseitin's encoding

Every formula can be converted to an equi-satisfiable formula in conjunctive normal form (CNF) in linear time and space if (a linear number of) new variables are admitted.

Input in CNF: Tseitin's encoding

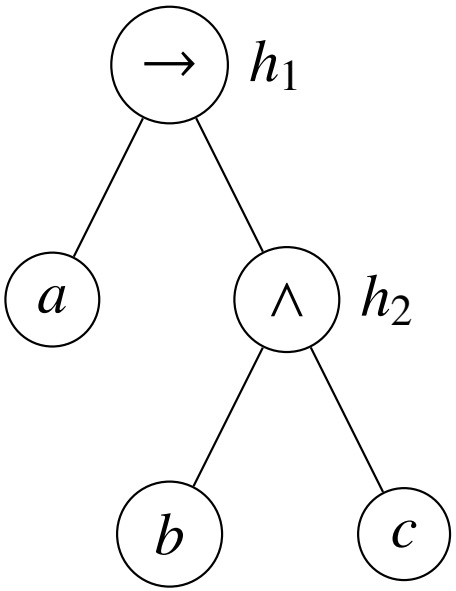
Every formula can be converted to an equi-satisfiable formula in conjunctive normal form (CNF) in linear time and space if (a linear number of) new variables are admitted.

$$(a \rightarrow (b \wedge c))$$

Input in CNF: Tseitin's encoding

Every formula can be converted to an equi-satisfiable formula in conjunctive normal form (CNF) in linear time and space if (a linear number of) new variables are admitted.

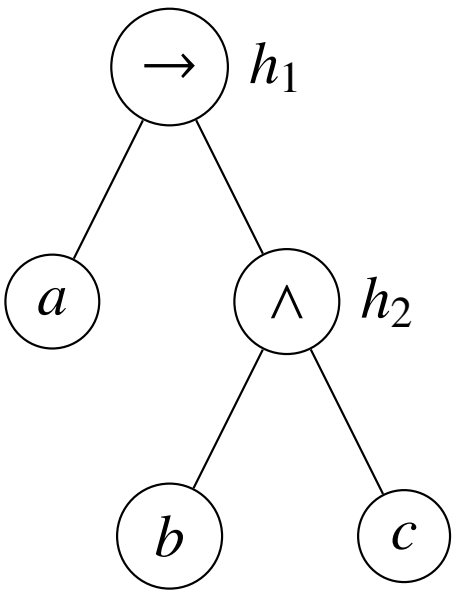
$$(a \rightarrow (b \wedge c))$$



Input in CNF: Tseitin's encoding

Every formula can be converted to an equi-satisfiable formula in conjunctive normal form (CNF) in linear time and space if (a linear number of) new variables are admitted.

$$(a \rightarrow (b \wedge c))$$



$$\begin{aligned} & (h_1 \leftrightarrow (a \rightarrow h_2)) \\ \wedge \quad & (h_2 \leftrightarrow (b \wedge c)) \\ \wedge \quad & (h_1) \end{aligned}$$

SAT solving: Resolution

Assumption: conjunctive normal form (CNF)

SAT solving: Resolution

Assumption: conjunctive normal form (CNF)

Derivation rule form:

$$\frac{antecendent_1 \quad \ldots \quad antecendent_n}{consequent}\, Rule\_name$$

SAT solving: Resolution

Assumption: conjunctive normal form (CNF)

Derivation rule form:

$$\frac{antecendent_1 \quad \ldots \quad antecendent_n}{consequent}\, Rule\_name$$

$$\frac{(l_1 \vee \ldots \vee l_n \vee x) \quad (l'_1 \vee \ldots \vee l'_m \vee \neg x)}{(l_1 \vee \ldots \vee l_n \vee l'_1 \vee \ldots \vee l'_m)}\, Rule_{res}$$

SAT solving: Resolution

Assumption: conjunctive normal form (CNF)

Derivation rule form:

$$\frac{antecendent_1 \quad \ldots \quad antecendent_n}{consequent}\ Rule\_name$$

$$\frac{(l_1 \vee \ldots \vee l_n \vee x) \quad (l'_1 \vee \ldots \vee l'_m \vee \neg x)}{(l_1 \vee \ldots \vee l_n \vee l'_1 \vee \ldots \vee l'_m)}\ Rule_{res}$$

$$C_x \vee C_{\neg x} \vee C$$

is equi-satisfiable to

$$Resolvents(C_x, C_{\neg x}) \vee C$$

SAT solving: Resolution

$$
\begin{array}{lllllllllll}
c_1: & (& \neg a & \vee & & & & d & \vee & e &) \\
c_2: & (& \neg a & \vee & & & & d & \vee & \neg e &) \\
c_3: & (& \neg a & \vee & & & & \neg d & \vee & e &) \\
c_4: & (& \neg a & \vee & & & & \neg d & \vee & \neg e &) \\
c_5: & (& a & \vee & b & & & & & &) \\
c_6: & (& a & \vee & \neg b & & & & & &) \\
c_7: & (& & & b & \vee & c & & & &) \\
c_8: & (& & & \neg b & \vee & \neg c & & & &)
\end{array}
$$

SAT solving: Resolution

$$
\begin{array}{lllllll}
c_1: & (\ \neg a & \vee & & d & \vee & e\ ) \\
c_2: & (\ \neg a & \vee & & d & \vee & \neg e\ ) \\
c_3: & (\ \neg a & \vee & & \neg d & \vee & e\ ) \\
c_4: & (\ \neg a & \vee & & \neg d & \vee & \neg e\ ) \\
c_5: & (\ a & \vee & b & & & \ ) \\
c_6: & (\ a & \vee & \neg b & & & \ ) \\
c_7: & (\ & & b & \vee & c & \ ) \\
c_8: & (\ & & \neg b & \vee & \neg c & \ )
\end{array}
$$

$$
\dfrac{\dfrac{\dfrac{c_1 : (\neg a \vee d \vee e)}{c_2 : (\neg a \vee d \vee \neg e)}}{(\neg a \vee d)} \quad \dfrac{\dfrac{c_3 : (\neg a \vee \neg d \vee e)}{c_4 : (\neg a \vee \neg d \vee \neg e)}}{(\neg a \vee \neg d)}}{(\neg a)} \quad \dfrac{\dfrac{c_5 : (a \vee b)}{c_6 : (a \vee \neg b)}}{(a)}}{()}
$$

SAT solving: Resolution

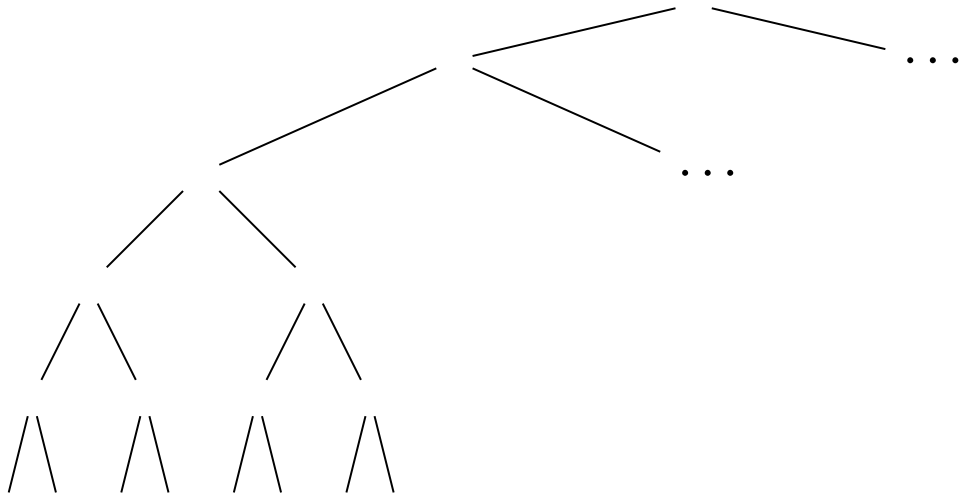
$$
\begin{array}{lllllll}
c_1: & (\ \neg a & \vee & & d & \vee & e\ ) \\
c_2: & (\ \neg a & \vee & & d & \vee & \neg e\ ) \\
c_3: & (\ \neg a & \vee & & \neg d & \vee & e\ ) \\
c_4: & (\ \neg a & \vee & & \neg d & \vee & \neg e\ ) \\
c_5: & (\ a & \vee & b & & & \ ) \\
c_6: & (\ a & \vee & \neg b & & & \ ) \\
c_7: & (\ & & b & \vee & c & \ ) \\
c_8: & (\ & & \neg b & \vee & \neg c & \ )
\end{array}
$$

$$
\dfrac{\dfrac{\dfrac{c_1 : (\neg a \vee d \vee e) \quad c_2 : (\neg a \vee d \vee \neg e)}{(\neg a \vee d)} \quad \dfrac{c_3 : (\neg a \vee \neg d \vee e) \quad c_4 : (\neg a \vee \neg d \vee \neg e)}{(\neg a \vee \neg d)}}{(\neg a)} \quad \dfrac{c_5 : (a \vee b) \quad c_6 : (a \vee \neg b)}{(a)}}{()}
$$

Problem: combinatorial blowup

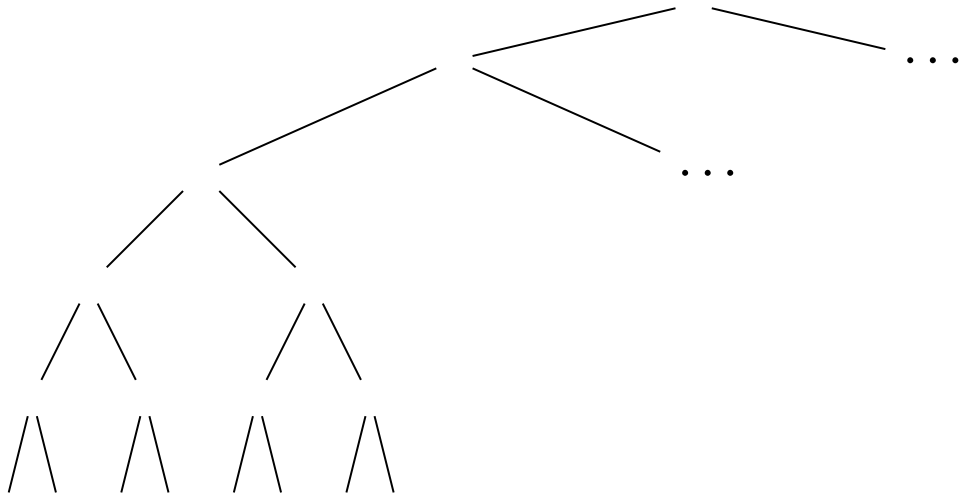
SAT solving: Enumeration

$$
\begin{array}{lllllllll}
c_1: & (& \neg a & \vee & & & d & \vee & e &) \\
c_2: & (& \neg a & \vee & & & d & \vee & \neg e &) \\
c_3: & (& \neg a & \vee & & & \neg d & \vee & e &) \\
c_4: & (& \neg a & \vee & & & \neg d & \vee & \neg e &) \\
c_5: & (& a & \vee & b & & & & &) \\
c_6: & (& a & \vee & \neg b & & & & &) \\
c_7: & (& & & b & \vee & c & & &) \\
c_8: & (& & & \neg b & \vee & \neg c & & &)
\end{array}
$$



SAT solving: Enumeration

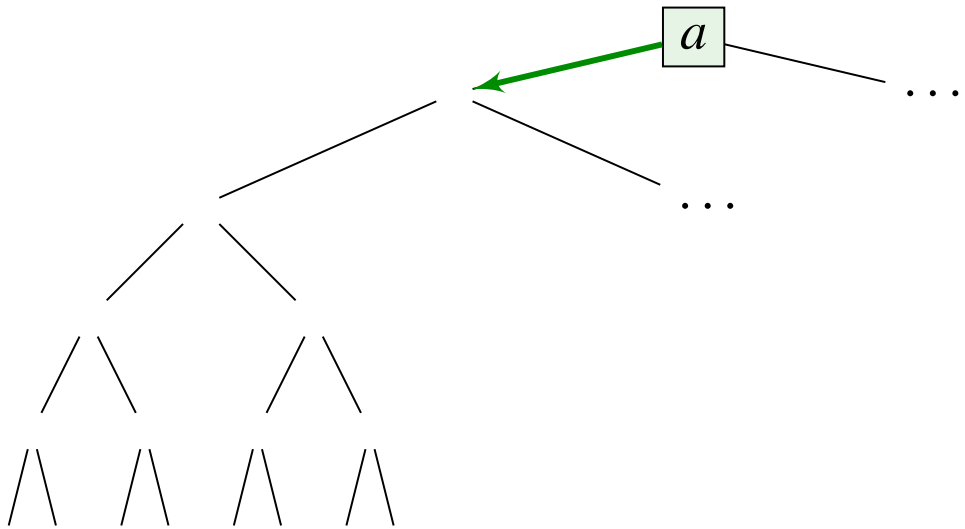
$$
\begin{array}{lllllllll}
c_1: & (& \neg a & \vee & & & d & \vee & e &) \\
c_2: & (& \neg a & \vee & & & d & \vee & \neg e &) \\
c_3: & (& \neg a & \vee & & & \neg d & \vee & e &) \\
c_4: & (& \neg a & \vee & & & \neg d & \vee & \neg e &) \\
c_5: & (& a & \vee & b & & & & &) \\
c_6: & (& a & \vee & \neg b & & & & &) \\
c_7: & (& & & b & \vee & c & & &) \\
c_8: & (& & & \neg b & \vee & \neg c & & &)
\end{array}
$$



Decision

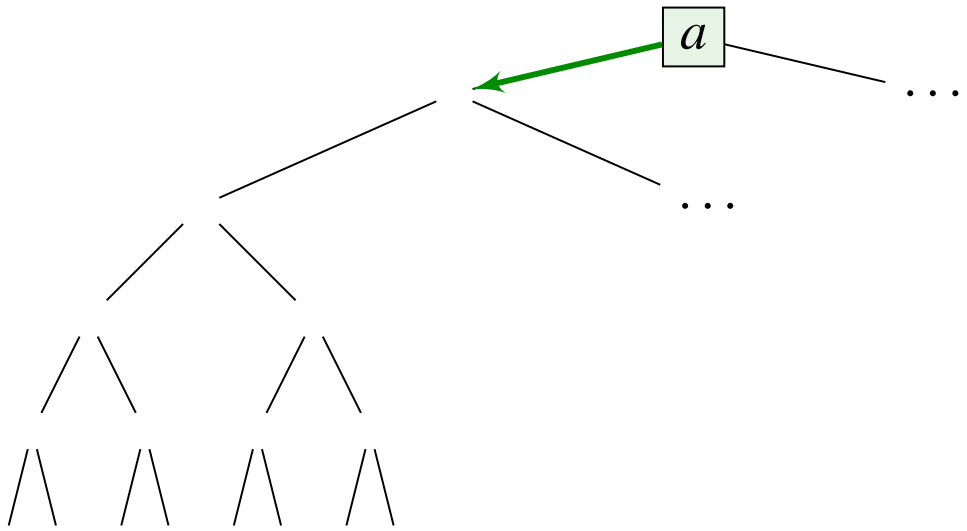
SAT solving: Enumeration

$$
\begin{array}{lllllllll}
c_1: & (& \neg a & \vee & & d & \vee & e &) \\
c_2: & (& \neg a & \vee & & d & \vee & \neg e &) \\
c_3: & (& \neg a & \vee & & \neg d & \vee & e &) \\
c_4: & (& \neg a & \vee & & \neg d & \vee & \neg e &) \\
c_5: & (& a & \vee & b & & & &) \\
c_6: & (& a & \vee & \neg b & & & &) \\
c_7: & (& & & b & \vee & c & &) \\
c_8: & (& & & \neg b & \vee & \neg c & &)
\end{array}
$$



SAT solving: Enumeration

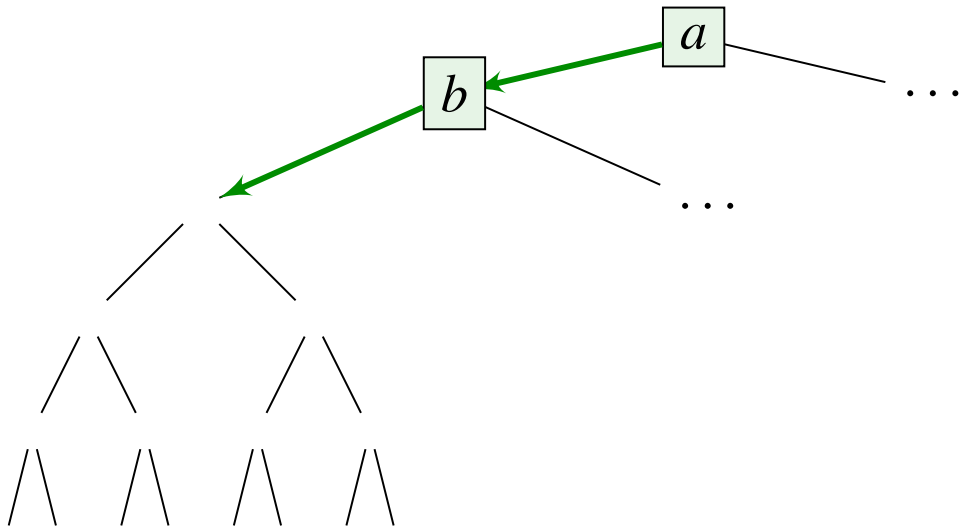
$$
\begin{array}{lllllll}
c_1: & (\ \neg a & \vee & & d & \vee & e\ ) \\
c_2: & (\ \neg a & \vee & & d & \vee & \neg e\ ) \\
c_3: & (\ \neg a & \vee & & \neg d & \vee & e\ ) \\
c_4: & (\ \neg a & \vee & & \neg d & \vee & \neg e\ ) \\
c_5: & (\ a & \vee & b & & &) \\
c_6: & (\ a & \vee & \neg b & & &) \\
c_7: & (\ & & b & \vee & c & &) \\
c_8: & (\ & & \neg b & \vee & \neg c & &)
\end{array}
$$



Decision

SAT solving: Enumeration

$$
\begin{array}{lllllll}
c_1: & (\ \neg a & \vee & & d & \vee & e\ ) \\
c_2: & (\ \neg a & \vee & & d & \vee & \neg e\ ) \\
c_3: & (\ \neg a & \vee & & \neg d & \vee & e\ ) \\
c_4: & (\ \neg a & \vee & & \neg d & \vee & \neg e\ ) \\
c_5: & (\ a & \vee & b & & &) \\
c_6: & (\ a & \vee & \neg b & & &) \\
c_7: & (\ & & b & \vee & c &) \\
c_8: & (\ & & \neg b & \vee & \neg c &)
\end{array}
$$



SAT solving: Enumeration

$c_1 : (\ \neg a \vee d \vee e\ )$

$c_2 : (\ \neg a \vee d \vee \neg e\ )$

$c_3 : (\ \neg a \vee \neg d \vee e\ )$

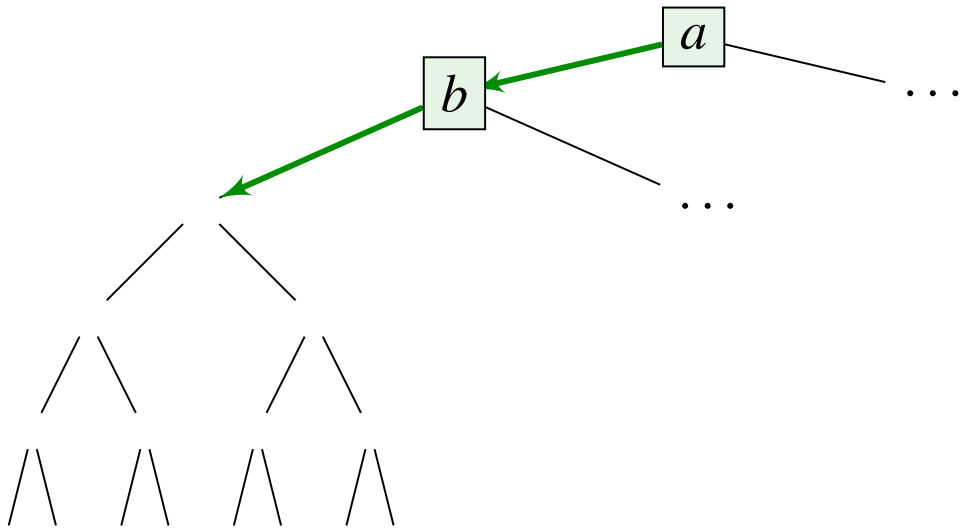
$c_4 : (\ \neg a \vee \neg d \vee \neg e\ )$

$c_5 : (\ a \vee b\ )$

$c_6 : (\ a \vee \neg b\ )$

$c_7 : (\ b \vee c\ )$

$c_8 : (\ \neg b \vee \neg c\ )$



Decision

SAT solving: Enumeration

$c_1 : (\ \neg a \vee d \vee e\ )$

$c_2 : (\ \neg a \vee d \vee \neg e\ )$

$c_3 : (\ \neg a \vee \neg d \vee e\ )$

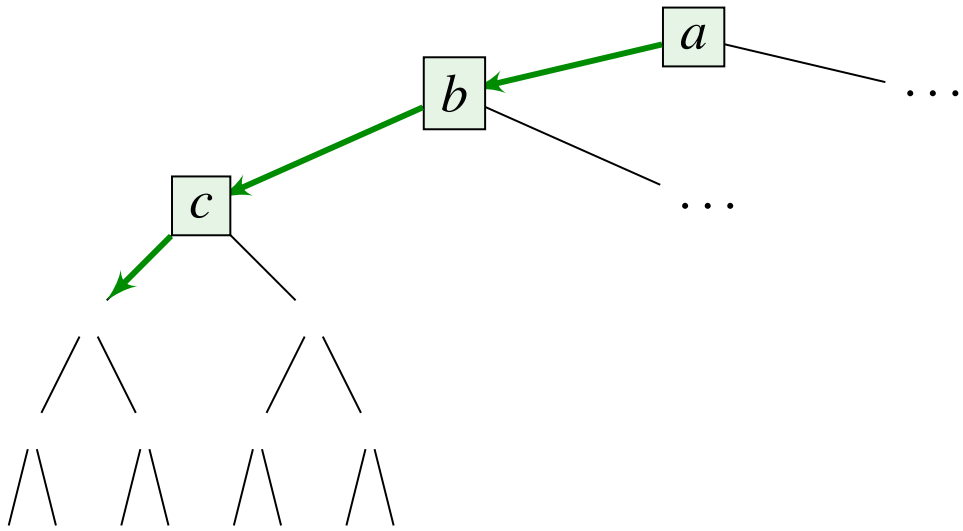
$c_4 : (\ \neg a \vee \neg d \vee \neg e\ )$

$c_5 : (\ a \vee b\ )$

$c_6 : (\ a \vee \neg b\ )$

$c_7 : (\ b \vee c\ )$

$c_8 : (\ \neg b \vee \neg c\ )$



SAT solving: Enumeration

$c_1 : (\ \neg a \vee d \vee e\ )$

$c_2 : (\ \neg a \vee d \vee \neg e\ )$

$c_3 : (\ \neg a \vee \neg d \vee e\ )$

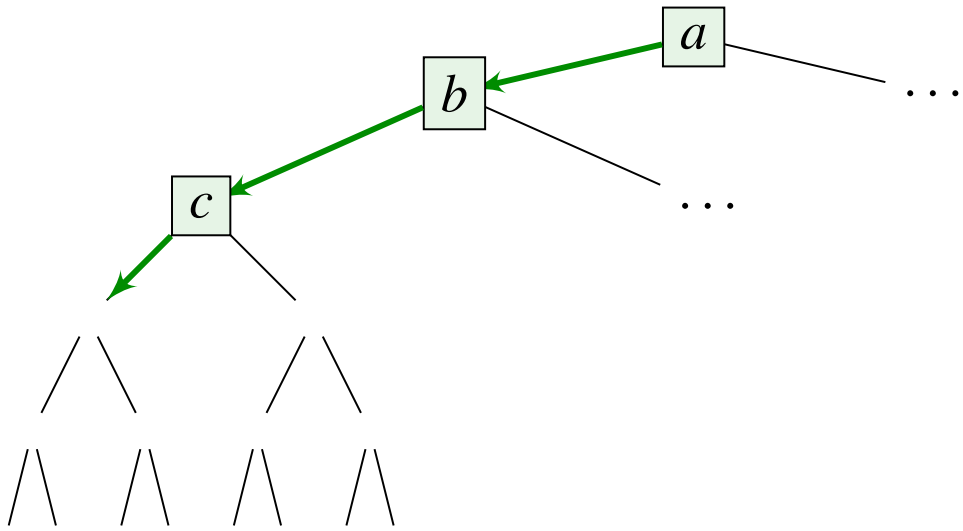
$c_4 : (\ \neg a \vee \neg d \vee \neg e\ )$

$c_5 : (\ a \vee b\ )$

$c_6 : (\ a \vee \neg b\ )$

$c_7 : (\ b \vee c\ )$

$c_8 : (\ \neg b \vee \neg c\ )$



Decision

SAT solving: Enumeration

$c_1 : (\ \neg a \vee d \vee e\ )$

$c_2 : (\ \neg a \vee d \vee \neg e\ )$

$c_3 : (\ \neg a \vee \neg d \vee e\ )$

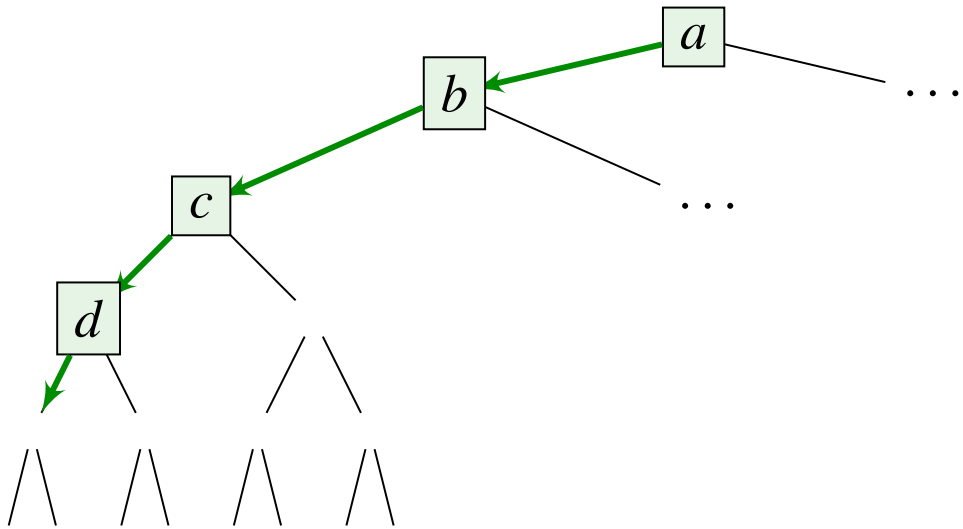
$c_4 : (\ \neg a \vee \neg d \vee \neg e\ )$

$c_5 : (\ a \vee b\ )$

$c_6 : (\ a \vee \neg b\ )$

$c_7 : (\ b \vee c\ )$

$c_8 : (\ \neg b \vee \neg c\ )$



SAT solving: Enumeration

$c_1 : (\ \neg a \vee d \vee e\ )$

$c_2 : (\ \neg a \vee d \vee \neg e\ )$

$c_3 : (\ \neg a \vee \neg d \vee e\ )$

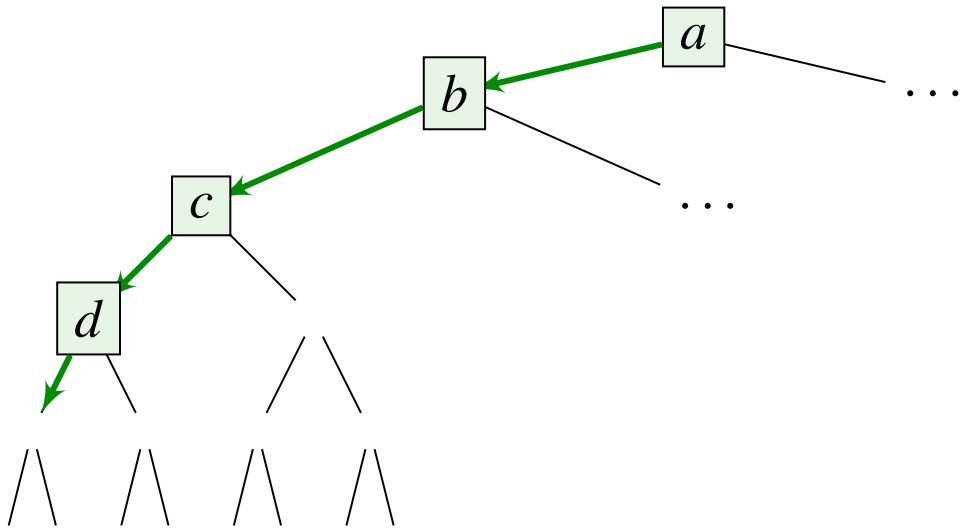
$c_4 : (\ \neg a \vee \neg d \vee \neg e\ )$

$c_5 : (\ a \vee b\ )$

$c_6 : (\ a \vee \neg b\ )$

$c_7 : (\ b \vee c\ )$

$c_8 : (\ \neg b \vee \neg c\ )$



Decision

SAT solving: Enumeration

$c_1 : (\ \neg a \lor d \lor e\ )$

$c_2 : (\ \neg a \lor d \lor \neg e\ )$

$c_3 : (\ \neg a \lor \neg d \lor e\ )$

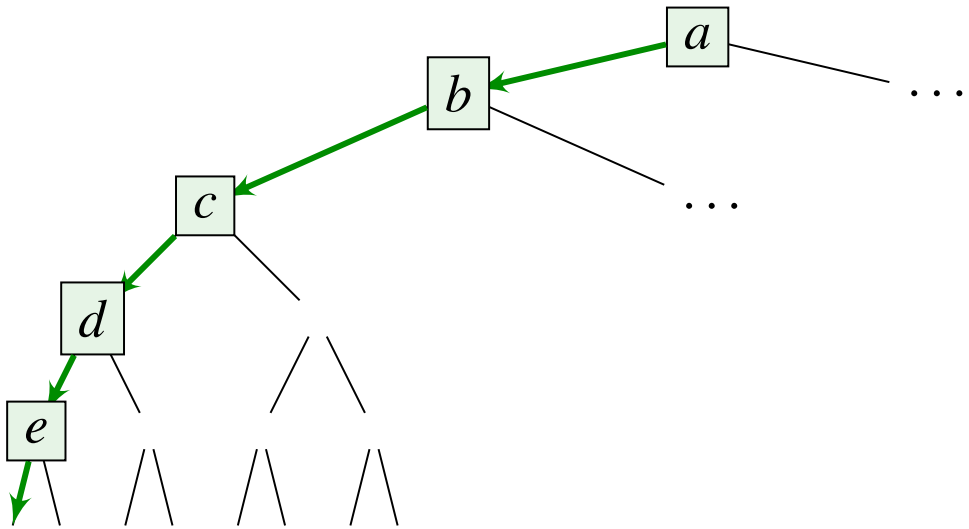
$c_4 : (\ \neg a \lor \neg d \lor \neg e\ )$

$c_5 : (\ a \lor b\ )$

$c_6 : (\ a \lor \neg b\ )$

$c_7 : (\ b \lor c\ )$

$c_8 : (\ \neg b \lor \neg c\ )$



SAT solving: Enumeration

$c_1 : (\ \neg a \vee d \vee e\ )$

$c_2 : (\ \neg a \vee d \vee \neg e\ )$

$c_3 : (\ \neg a \vee \neg d \vee e\ )$

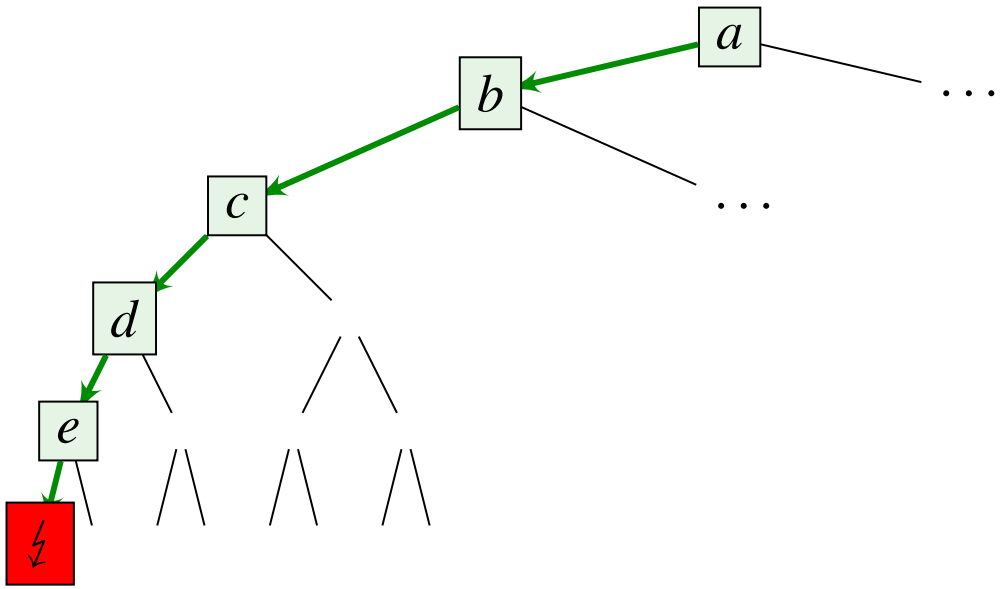
$c_4 : (\ \neg a \vee \neg d \vee \neg e\ )$

$c_5 : (\ a \vee b\ )$

$c_6 : (\ a \vee \neg b\ )$

$c_7 : (\ b \vee c\ )$

$c_8 : (\ \neg b \vee \neg c\ )$



Conflict

SAT solving: Enumeration

$c_1 : (\ \neg a \vee d \vee e\ )$

$c_2 : (\ \neg a \vee d \vee \neg e\ )$

$c_3 : (\ \neg a \vee \neg d \vee e\ )$

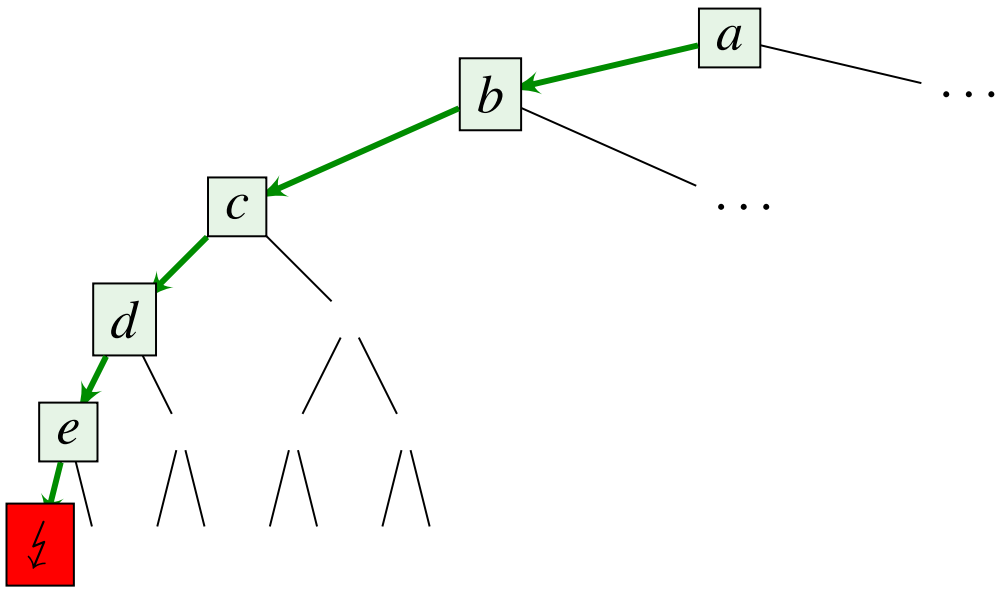
$c_4 : (\ \neg a \vee \neg d \vee \neg e\ )$

$c_5 : (\ a \vee b\ )$

$c_6 : (\ a \vee \neg b\ )$

$c_7 : (\ b \vee c\ )$

$c_8 : (\ \neg b \vee \neg c\ )$



Backjumping

SAT solving: Enumeration

$c_1 : (\ \neg a \vee d \vee e\ )$

$c_2 : (\ \neg a \vee d \vee \neg e\ )$

$c_3 : (\ \neg a \vee \neg d \vee e\ )$

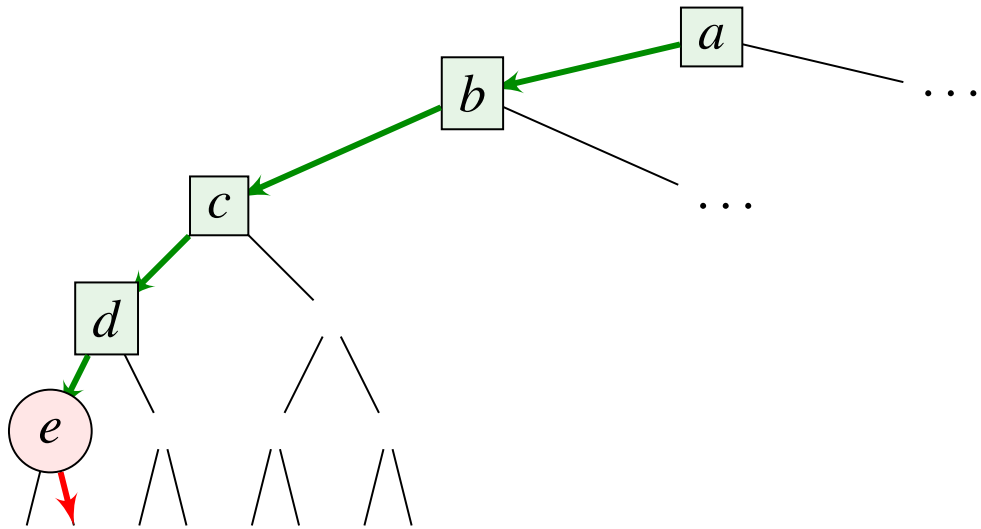
$c_4 : (\ \neg a \vee \neg d \vee \neg e\ )$

$c_5 : (\ a \vee b\ )$

$c_6 : (\ a \vee \neg b\ )$

$c_7 : (\ b \vee c\ )$

$c_8 : (\ \neg b \vee \neg c\ )$



SAT solving: Enumeration

$c_1 : (\ \neg a \vee d \vee e\ )$

$c_2 : (\ \neg a \vee d \vee \neg e\ )$

$c_3 : (\ \neg a \vee \neg d \vee e\ )$

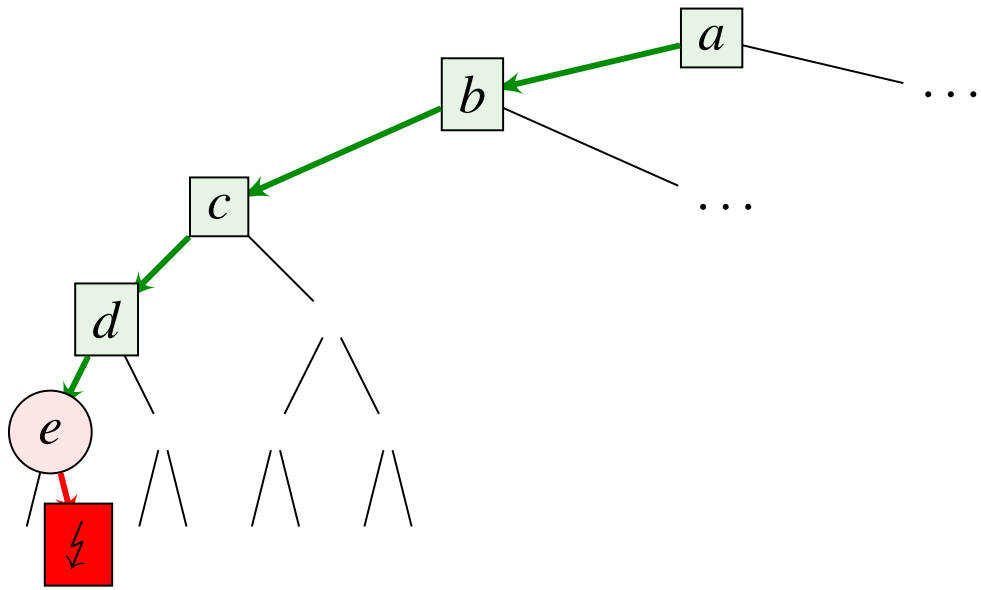
$c_4 : (\ \neg a \vee \neg d \vee \neg e\ )$

$c_5 : (\ a \vee b\ )$

$c_6 : (\ a \vee \neg b\ )$

$c_7 : (\ b \vee c\ )$

$c_8 : (\ \neg b \vee \neg c\ )$



Conflict

SAT solving: Enumeration

$c_1 : (\ \neg a \vee d \vee e\ )$

$c_2 : (\ \neg a \vee d \vee \neg e\ )$

$c_3 : (\ \neg a \vee \neg d \vee e\ )$

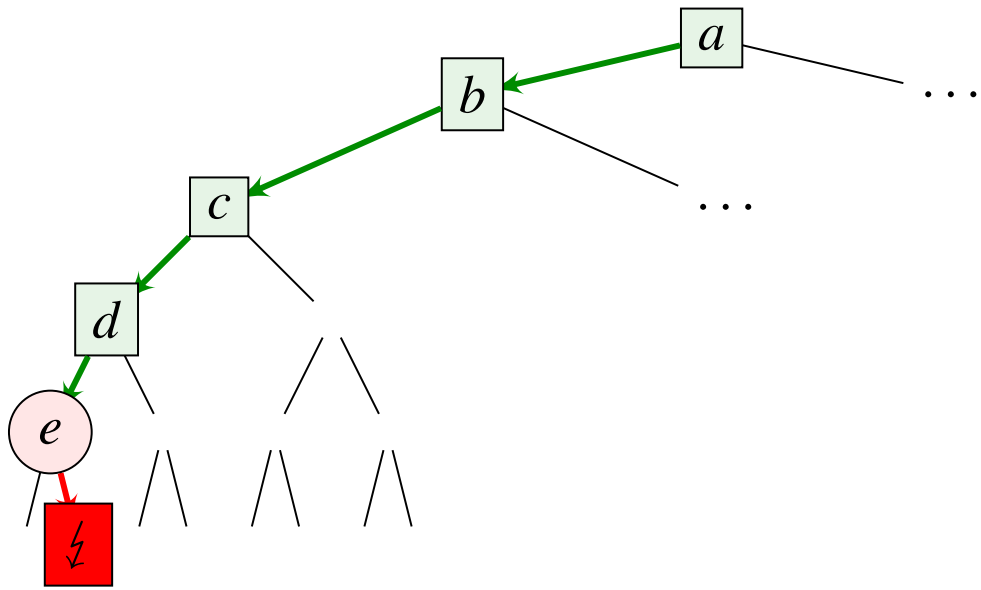
$c_4 : (\ \neg a \vee \neg d \vee \neg e\ )$

$c_5 : (\ a \vee b\ )$

$c_6 : (\ a \vee \neg b\ )$

$c_7 : (\ b \vee c\ )$

$c_8 : (\ \neg b \vee \neg c\ )$



Backjumping

SAT solving: Enumeration

$c_1 : (\ \neg a \vee d \vee e\ )$

$c_2 : (\ \neg a \vee d \vee \neg e\ )$

$c_3 : (\ \neg a \vee \neg d \vee e\ )$

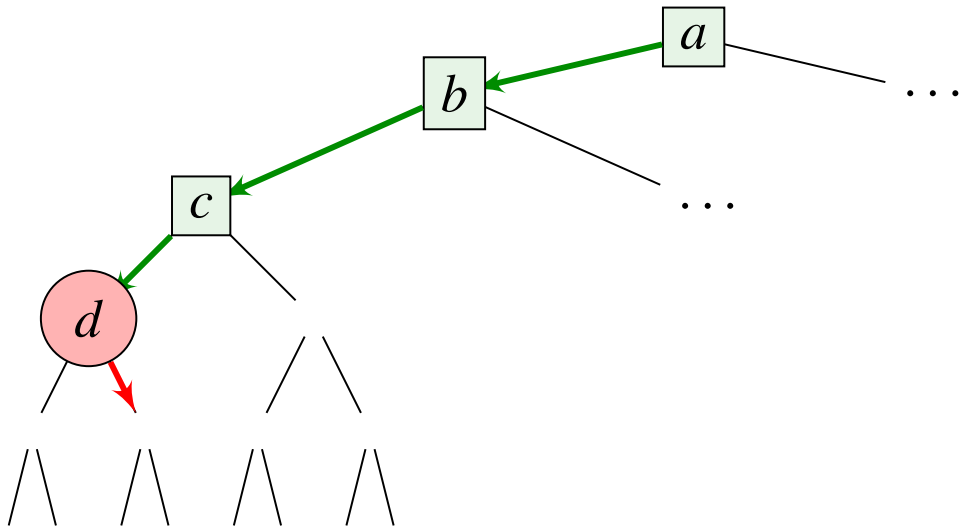
$c_4 : (\ \neg a \vee \neg d \vee \neg e\ )$

$c_5 : (\ a \vee b\ )$

$c_6 : (\ a \vee \neg b\ )$

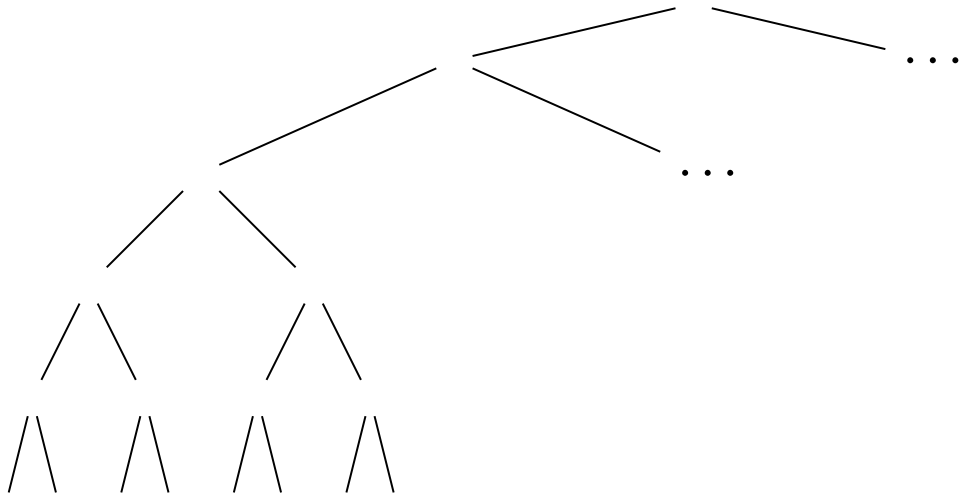
$c_7 : (\ b \vee c\ )$

$c_8 : (\ \neg b \vee \neg c\ )$



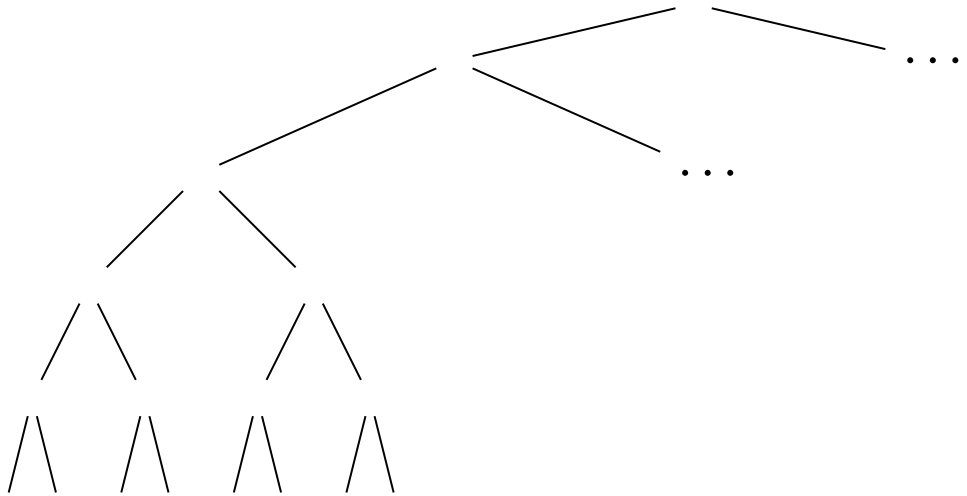
SAT solving: The DPLL algorithm

$$
\begin{array}{lllllllll}
c_1: & (& \neg a & \vee & & & d & \vee & e &) \\
c_2: & (& \neg a & \vee & & & d & \vee & \neg e &) \\
c_3: & (& \neg a & \vee & & & \neg d & \vee & e &) \\
c_4: & (& \neg a & \vee & & & \neg d & \vee & \neg e &) \\
c_5: & (& a & \vee & b & & & & &) \\
c_6: & (& a & \vee & \neg b & & & & &) \\
c_7: & (& & & b & \vee & c & & &) \\
c_8: & (& & & \neg b & \vee & \neg c & & &)
\end{array}
$$



SAT solving: The DPLL algorithm

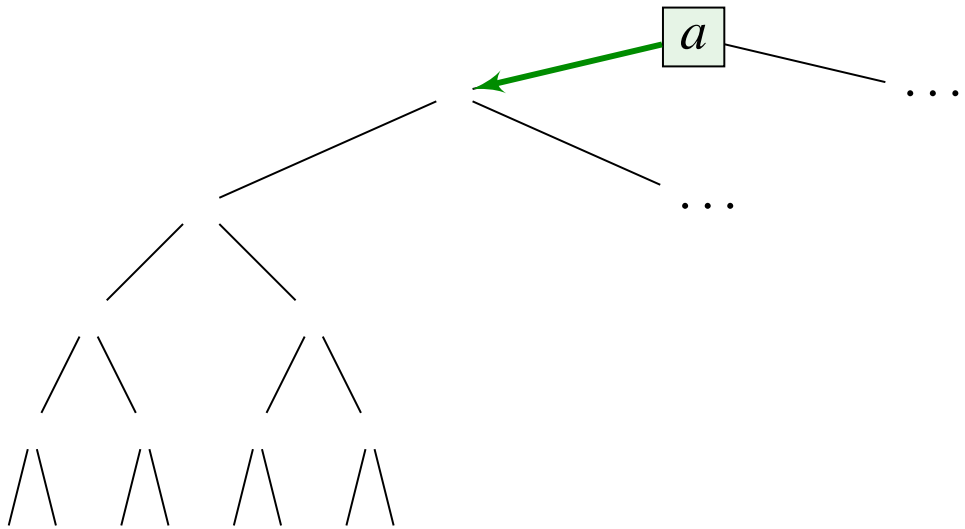
$$
\begin{array}{lllllllll}
c_1: & (& \neg a & \vee & & & d & \vee & e &) \\
c_2: & (& \neg a & \vee & & & d & \vee & \neg e &) \\
c_3: & (& \neg a & \vee & & & \neg d & \vee & e &) \\
c_4: & (& \neg a & \vee & & & \neg d & \vee & \neg e &) \\
c_5: & (& a & \vee & b & & & & &) \\
c_6: & (& a & \vee & \neg b & & & & &) \\
c_7: & (& & & b & \vee & c & & &) \\
c_8: & (& & & \neg b & \vee & \neg c & & &)
\end{array}
$$



Decision

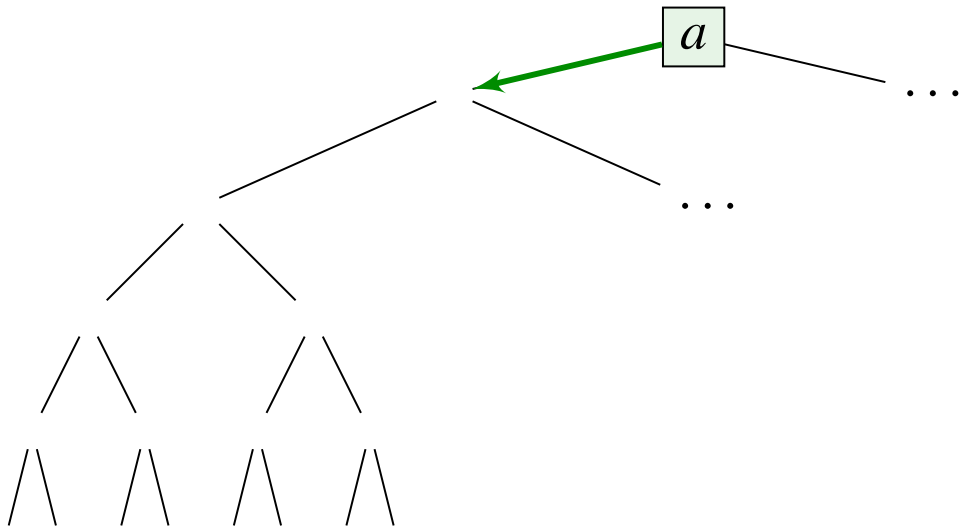
SAT solving: The DPLL algorithm

$$
\begin{array}{lllllllll}
c_1: & (& \neg a & \vee & & d & \vee & e &) \\
c_2: & (& \neg a & \vee & & d & \vee & \neg e &) \\
c_3: & (& \neg a & \vee & & \neg d & \vee & e &) \\
c_4: & (& \neg a & \vee & & \neg d & \vee & \neg e &) \\
c_5: & (& a & \vee & b & & & &) \\
c_6: & (& a & \vee & \neg b & & & &) \\
c_7: & (& & & b & \vee & c & &) \\
c_8: & (& & & \neg b & \vee & \neg c & &)
\end{array}
$$



SAT solving: The DPLL algorithm

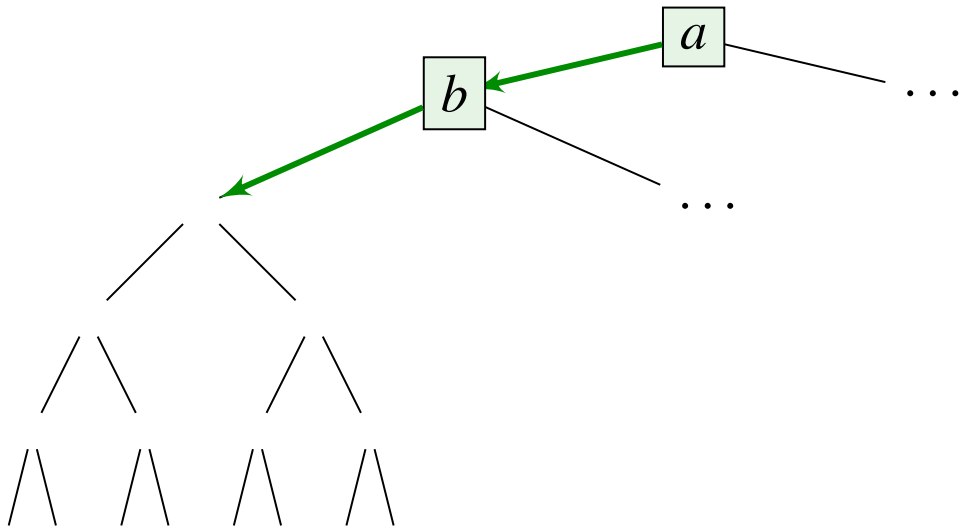
$$
\begin{array}{lllllllll}
c_1: & (& \neg a & \vee & & d & \vee & e &) \\
c_2: & (& \neg a & \vee & & d & \vee & \neg e &) \\
c_3: & (& \neg a & \vee & & \neg d & \vee & e &) \\
c_4: & (& \neg a & \vee & & \neg d & \vee & \neg e &) \\
c_5: & (& a & \vee & b & & & &) \\
c_6: & (& a & \vee & \neg b & & & &) \\
c_7: & (& & & b & \vee & c & &) \\
c_8: & (& & & \neg b & \vee & \neg c & &)
\end{array}
$$



Decision

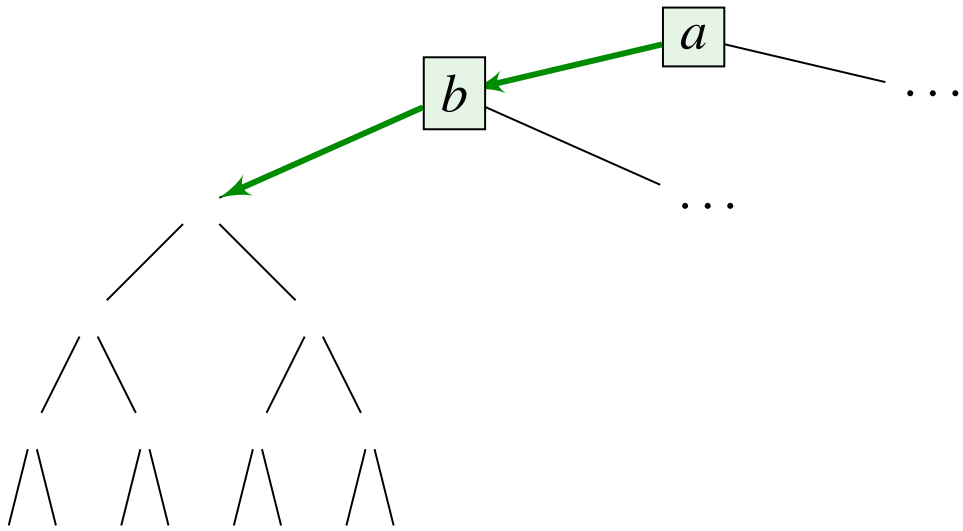
SAT solving: The DPLL algorithm

$$
\begin{array}{lllllllll}
c_1: & (& \neg a & \vee & & d & \vee & e &) \\
c_2: & (& \neg a & \vee & & d & \vee & \neg e &) \\
c_3: & (& \neg a & \vee & & \neg d & \vee & e &) \\
c_4: & (& \neg a & \vee & & \neg d & \vee & \neg e &) \\
c_5: & (& a & \vee & b & & & &) \\
c_6: & (& a & \vee & \neg b & & & &) \\
c_7: & (& & & b & \vee & c & &) \\
c_8: & (& & & \neg b & \vee & \neg c & &)
\end{array}
$$



SAT solving: The DPLL algorithm

$$
\begin{array}{lll}
c_1 : & (\neg a \vee & d \vee e) \\
c_2 : & (\neg a \vee & d \vee \neg e) \\
c_3 : & (\neg a \vee & \neg d \vee e) \\
c_4 : & (\neg a \vee & \neg d \vee \neg e) \\
c_5 : & (a \vee b &) \\
c_6 : & (a \vee \neg b &) \\
c_7 : & (b \vee c &) \\
c_8 : & (\neg b \vee \neg c &)
\end{array}
$$



Pure literal detection

SAT solving: The DPLL algorithm

$c_1 : (\ \neg a \vee d \vee e\ )$

$c_2 : (\ \neg a \vee d \vee \neg e\ )$

$c_3 : (\ \neg a \vee \neg d \vee e\ )$

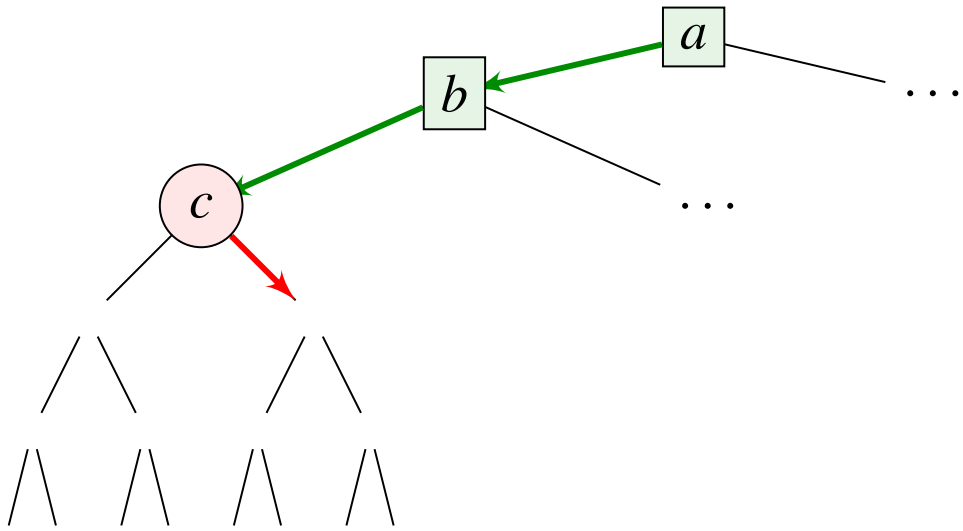
$c_4 : (\ \neg a \vee \neg d \vee \neg e\ )$

$c_5 : (\ a \vee b\ )$

$c_6 : (\ a \vee \neg b\ )$

$c_7 : (\ b \vee c\ )$

$c_8 : (\ \neg b \vee \neg c\ )$



SAT solving: The DPLL algorithm

$c_1 : (\ \neg a \vee d \vee e\ )$

$c_2 : (\ \neg a \vee d \vee \neg e\ )$

$c_3 : (\ \neg a \vee \neg d \vee e\ )$

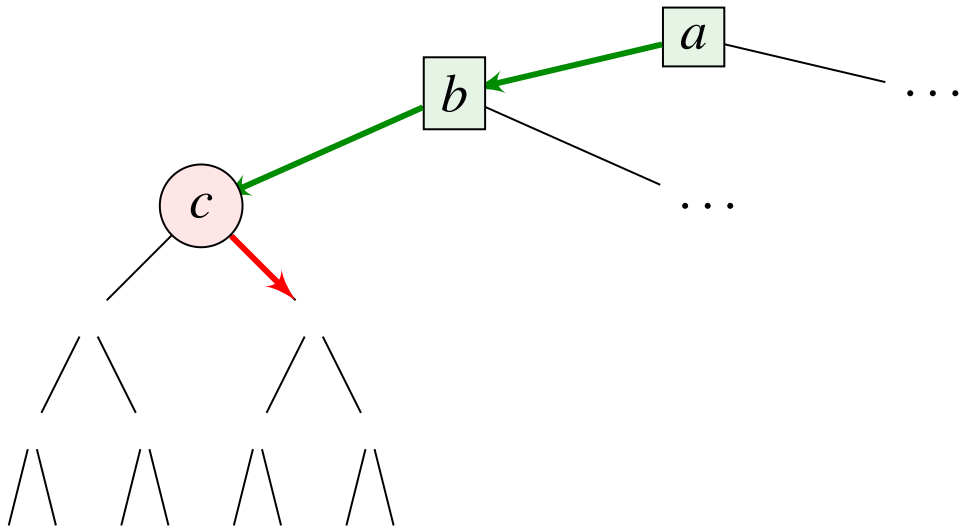
$c_4 : (\ \neg a \vee \neg d \vee \neg e\ )$

$c_5 : (\ a \vee b\ )$

$c_6 : (\ a \vee \neg b\ )$

$c_7 : (\ b \vee c\ )$

$c_8 : (\ \neg b \vee \neg c\ )$



Decision

SAT solving: The DPLL algorithm

$c_1 : (\ \neg a \vee d \vee e\ )$

$c_2 : (\ \neg a \vee d \vee \neg e\ )$

$c_3 : (\ \neg a \vee \neg d \vee e\ )$

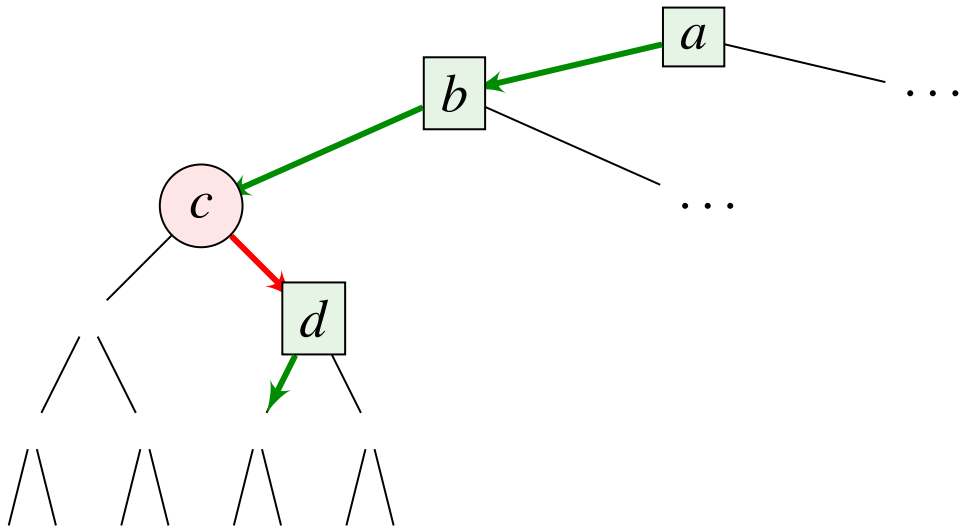
$c_4 : (\ \neg a \vee \neg d \vee \neg e\ )$

$c_5 : (\ a \vee b\ )$

$c_6 : (\ a \vee \neg b\ )$

$c_7 : (\ b \vee c\ )$

$c_8 : (\ \neg b \vee \neg c\ )$



SAT solving: The DPLL algorithm

$c_1 : (\ \neg a \vee d \vee e\ )$

$c_2 : (\ \neg a \vee d \vee \neg e\ )$

$c_3 : (\ \neg a \vee \neg d \vee e\ )$

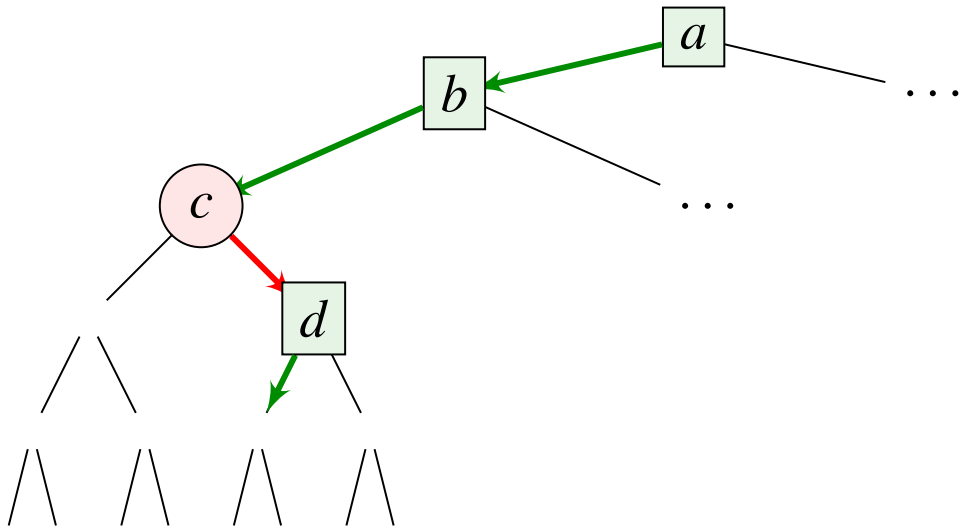
$c_4 : (\ \neg a \vee \neg d \vee \neg e\ )$

$c_5 : (\ a \vee b\ )$

$c_6 : (\ a \vee \neg b\ )$

$c_7 : (\ b \vee c\ )$

$c_8 : (\ \neg b \vee \neg c\ )$



Boolean constraint propagation

SAT solving: The DPLL algorithm

$c_1 : (\ \neg a \vee d \vee e\ )$

$c_2 : (\ \neg a \vee d \vee \neg e\ )$

$c_3 : (\ \neg a \vee \neg d \vee e\ )$

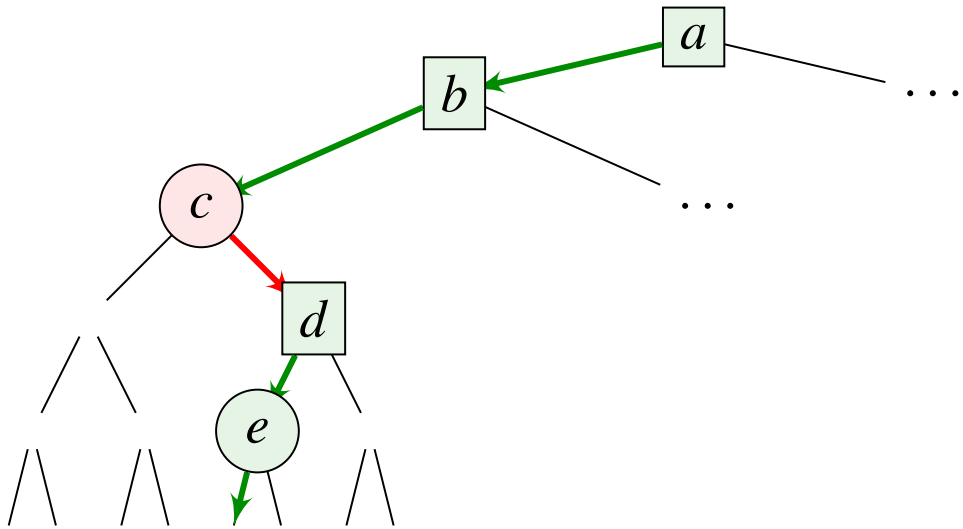
$c_4 : (\ \neg a \vee \neg d \vee \neg e\ )$

$c_5 : (\ a \vee b\ )$

$c_6 : (\ a \vee \neg b\ )$

$c_7 : (\ b \vee c\ )$

$c_8 : (\ \neg b \vee \neg c\ )$



SAT solving: The DPLL algorithm

$c_1 : (\ \neg a \vee d \vee e\ )$

$c_2 : (\ \neg a \vee d \vee \neg e\ )$

$c_3 : (\ \neg a \vee \neg d \vee e\ )$

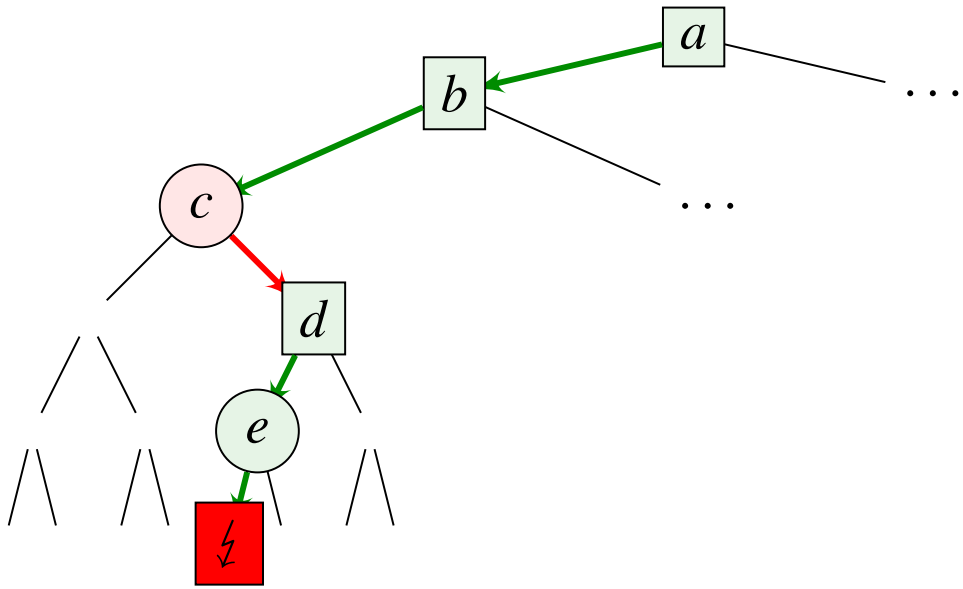
$c_4 : (\ \neg a \vee \neg d \vee \neg e\ )$

$c_5 : (\ a \vee b\ )$

$c_6 : (\ a \vee \neg b\ )$

$c_7 : (\ b \vee c\ )$

$c_8 : (\ \neg b \vee \neg c\ )$



Conflict

SAT solving: The DPLL algorithm

$c_1 : (\ \neg a \vee d \vee e\ )$

$c_2 : (\ \neg a \vee d \vee \neg e\ )$

$c_3 : (\ \neg a \vee \neg d \vee e\ )$

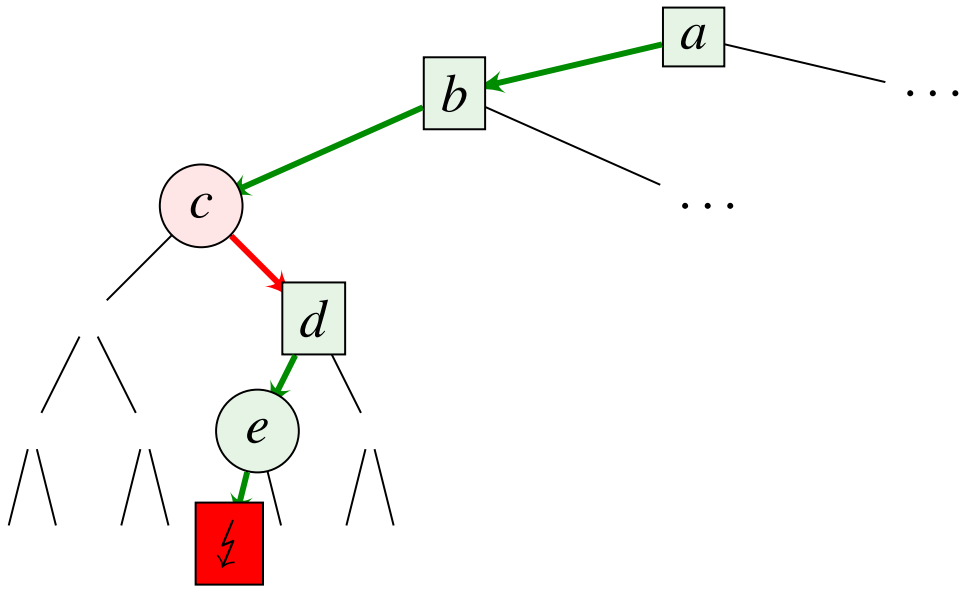
$c_4 : (\ \neg a \vee \neg d \vee \neg e\ )$

$c_5 : (\ a \vee b\ )$

$c_6 : (\ a \vee \neg b\ )$

$c_7 : (\ b \vee c\ )$

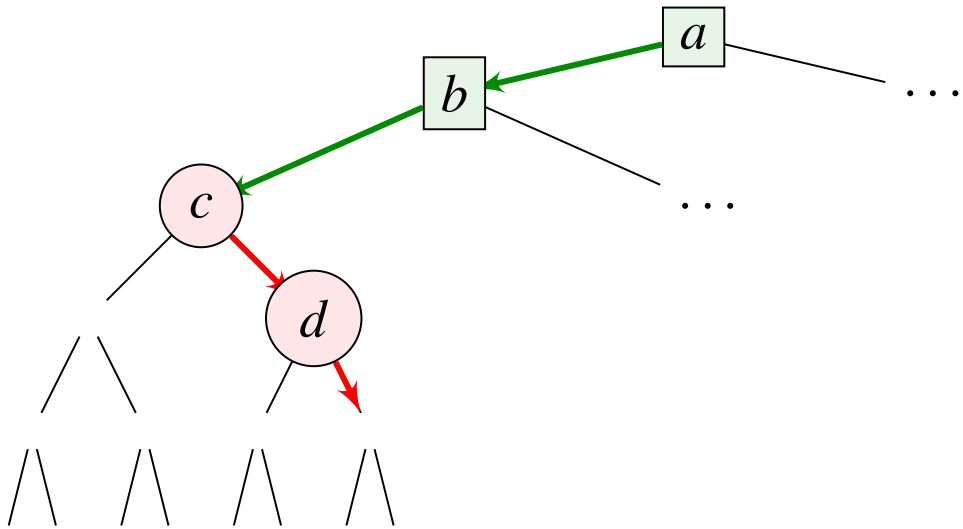
$c_8 : (\ \neg b \vee \neg c\ )$



Backjumping

SAT solving: The DPLL algorithm

$$
\begin{array}{lllllllll}
c_1: & (\ \neg a & \vee & & & d & \vee & e &) \\
c_2: & (\ \neg a & \vee & & & d & \vee & \neg e &) \\
c_3: & (\ \neg a & \vee & & & \neg d & \vee & e &) \\
c_4: & (\ \neg a & \vee & & & \neg d & \vee & \neg e &) \\
c_5: & (\ a & \vee & b & & & & &) \\
c_6: & (\ a & \vee & \neg b & & & & &) \\
c_7: & (\ & & b & \vee & c & & &) \\
c_8: & (\ & & \neg b & \vee & \neg c & & &)
\end{array}
$$



SAT solving: The DPLL algorithm

$c_1 : (\ \neg a \vee d \vee e\ )$

$c_2 : (\ \neg a \vee d \vee \neg e\ )$

$c_3 : (\ \neg a \vee \neg d \vee e\ )$

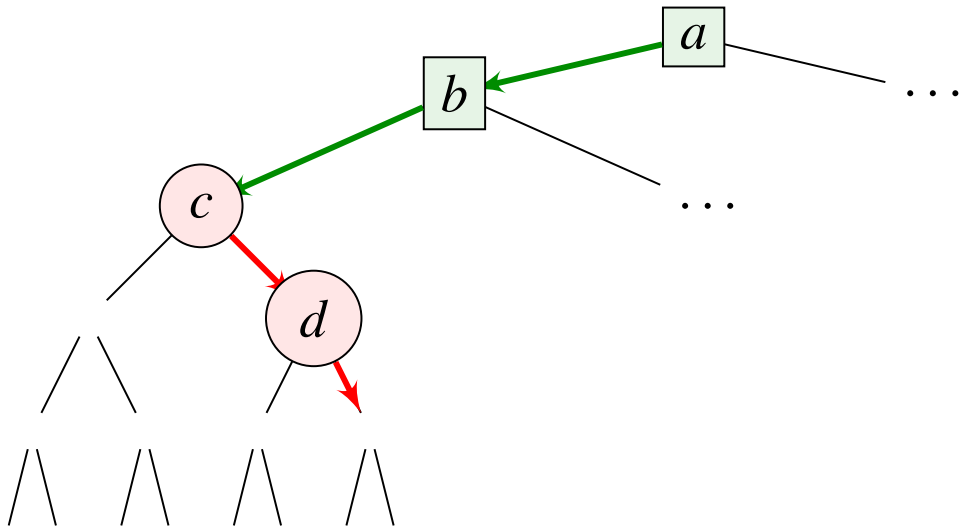
$c_4 : (\ \neg a \vee \neg d \vee \neg e\ )$

$c_5 : (\ a \vee b\ )$

$c_6 : (\ a \vee \neg b\ )$

$c_7 : (\ b \vee c\ )$

$c_8 : (\ \neg b \vee \neg c\ )$



Boolean constraint propagation

SAT solving: The DPLL algorithm

$c_1 : (\ \neg a \vee d \vee e\ )$

$c_2 : (\ \neg a \vee d \vee \neg e\ )$

$c_3 : (\ \neg a \vee \neg d \vee e\ )$

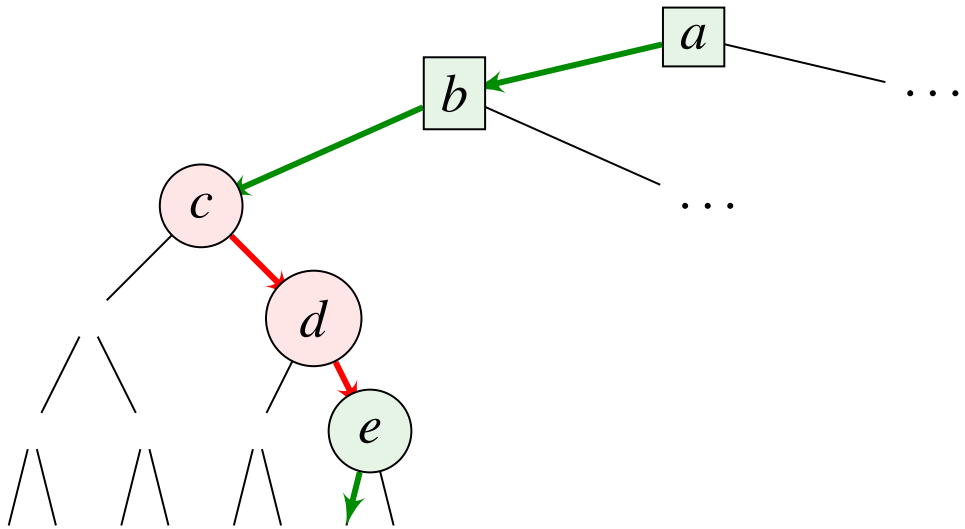
$c_4 : (\ \neg a \vee \neg d \vee \neg e\ )$

$c_5 : (\ a \vee b\ )$

$c_6 : (\ a \vee \neg b\ )$

$c_7 : (\ b \vee c\ )$

$c_8 : (\ \neg b \vee \neg c\ )$



SAT solving: The DPLL algorithm

$c_1 : (\ \neg a \vee d \vee e\ )$

$c_2 : (\ \neg a \vee d \vee \neg e\ )$

$c_3 : (\ \neg a \vee \neg d \vee e\ )$

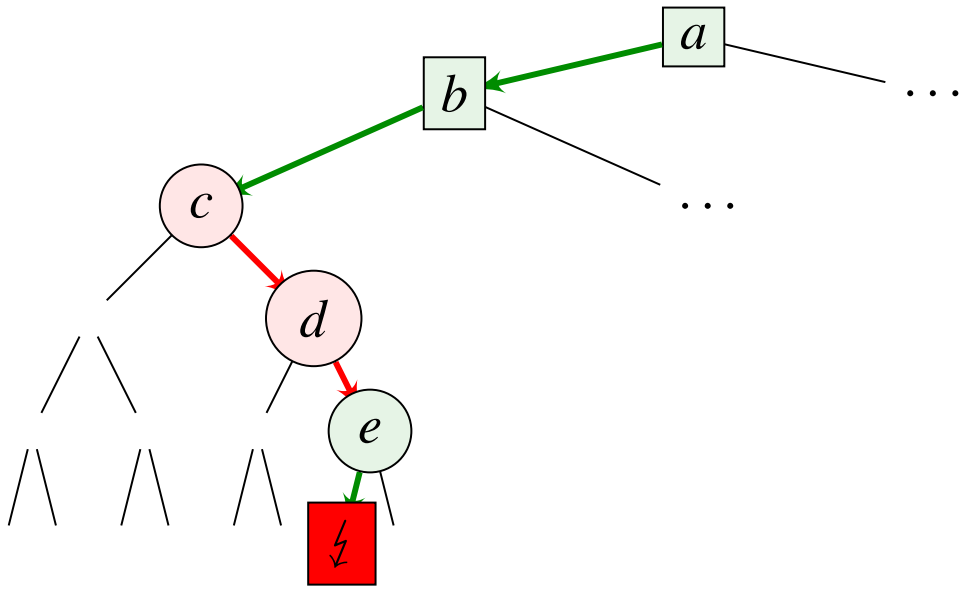
$c_4 : (\ \neg a \vee \neg d \vee \neg e\ )$

$c_5 : (\ a \vee b\ )$

$c_6 : (\ a \vee \neg b\ )$

$c_7 : (\ b \vee c\ )$

$c_8 : (\ \neg b \vee \neg c\ )$



Conflict

SAT solving: The DPLL algorithm

$c_1 : (\ \neg a \vee d \vee e\ )$

$c_2 : (\ \neg a \vee d \vee \neg e\ )$

$c_3 : (\ \neg a \vee \neg d \vee e\ )$

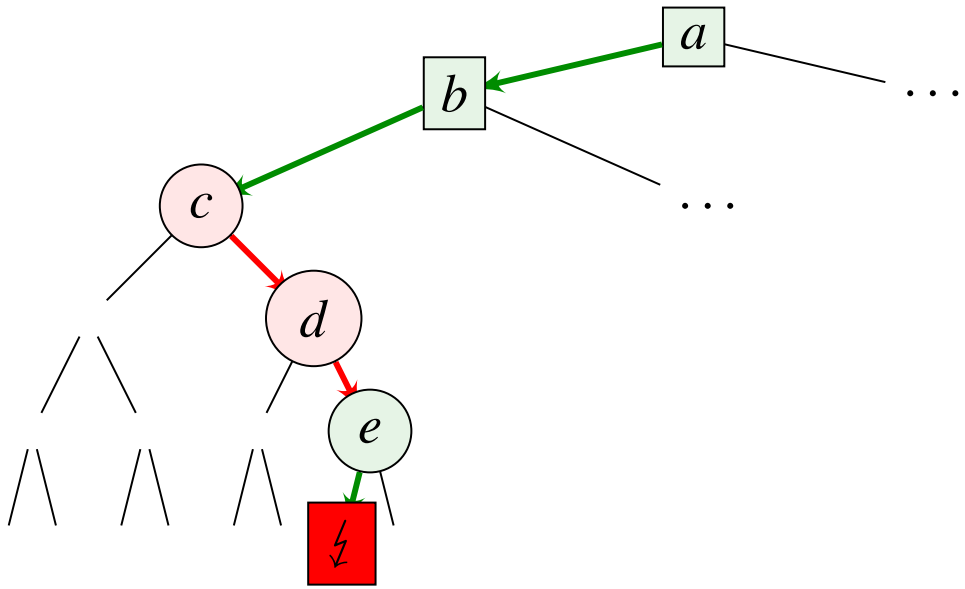
$c_4 : (\ \neg a \vee \neg d \vee \neg e\ )$

$c_5 : (\ a \vee b\ )$

$c_6 : (\ a \vee \neg b\ )$

$c_7 : (\ b \vee c\ )$

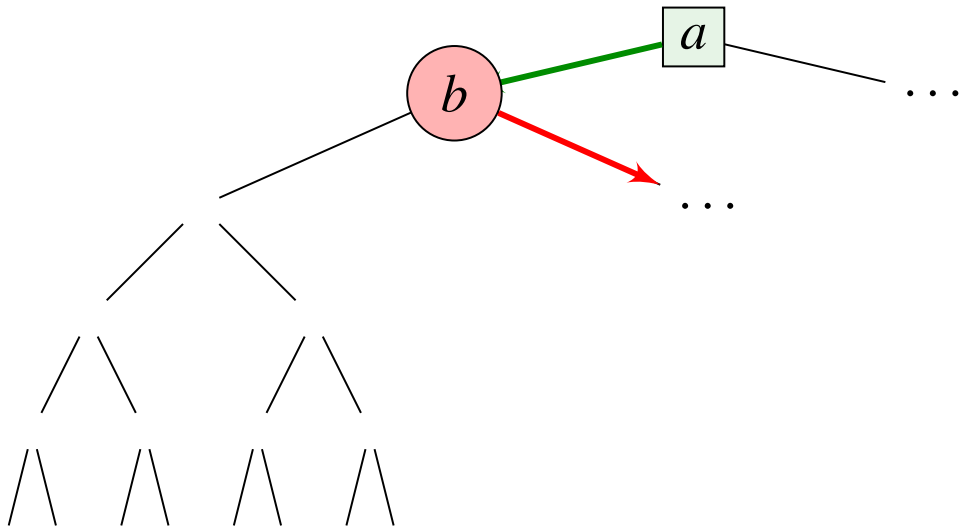
$c_8 : (\ \neg b \vee \neg c\ )$



Backjumping

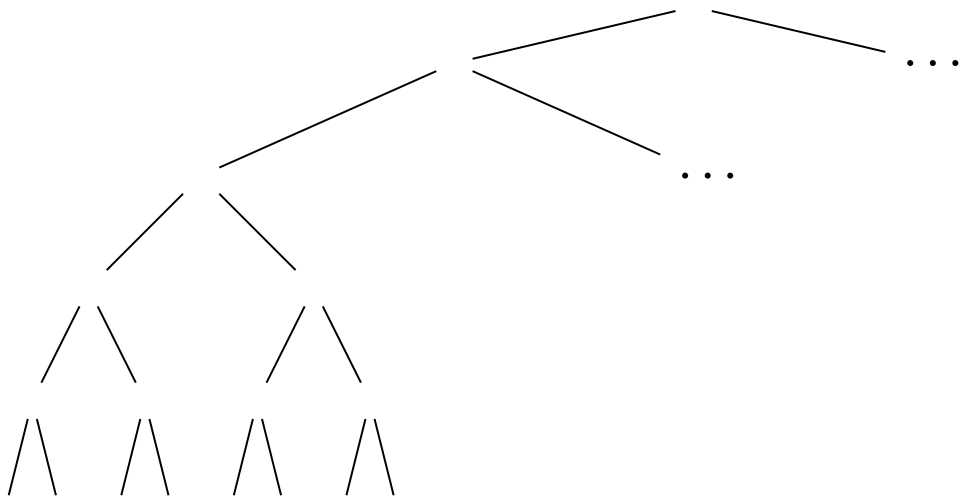
SAT solving: The DPLL algorithm

$$
\begin{array}{lllllll}
c_1: & (\ \neg a & \vee & & d & \vee & e\ ) \\
c_2: & (\ \neg a & \vee & & d & \vee & \neg e\ ) \\
c_3: & (\ \neg a & \vee & & \neg d & \vee & e\ ) \\
c_4: & (\ \neg a & \vee & & \neg d & \vee & \neg e\ ) \\
c_5: & (\ a & \vee & b & & &)\\
c_6: & (\ a & \vee & \neg b & & &)\\
c_7: & (\ & & b & \vee & c & &)\\
c_8: & (\ & & \neg b & \vee & \neg c & &)
\end{array}
$$



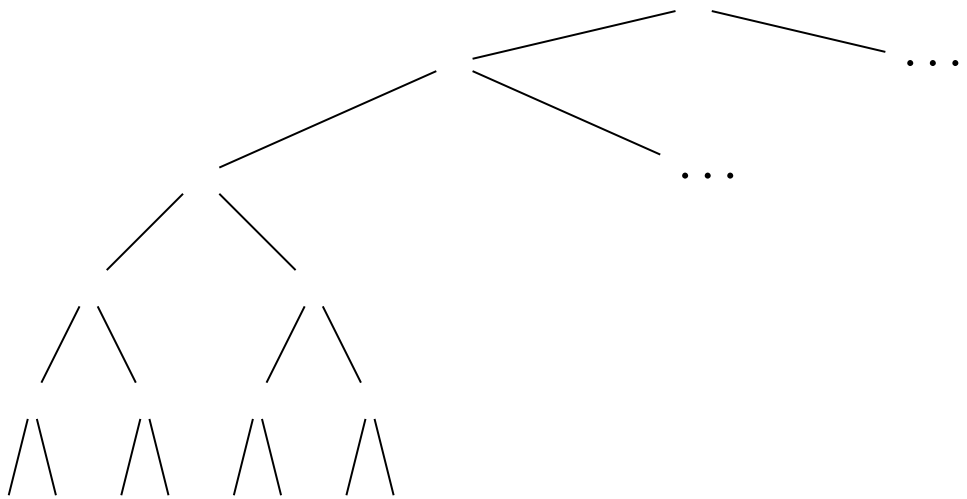
SAT solving: Conflict-directed clause learning

$$
\begin{array}{lllllllll}
c_1: & (& \neg a & \vee & & & d & \vee & e &) \\
c_2: & (& \neg a & \vee & & & d & \vee & \neg e &) \\
c_3: & (& \neg a & \vee & & & \neg d & \vee & e &) \\
c_4: & (& \neg a & \vee & & & \neg d & \vee & \neg e &) \\
c_5: & (& a & \vee & b & & & & &) \\
c_6: & (& a & \vee & \neg b & & & & &) \\
c_7: & (& & & b & \vee & c & & &) \\
c_8: & (& & & \neg b & \vee & \neg c & & &)
\end{array}
$$



SAT solving: Conflict-directed clause learning

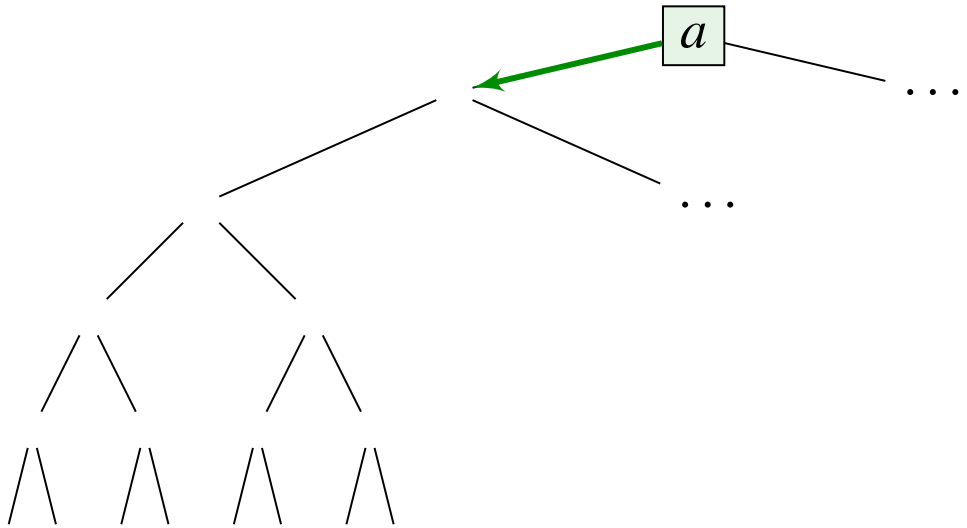
$$
\begin{aligned}
c_1 &: (\ \neg a \ \vee \qquad\qquad d \ \vee \ e\ ) \\
c_2 &: (\ \neg a \ \vee \qquad\qquad d \ \vee \ \neg e\ ) \\
c_3 &: (\ \neg a \ \vee \qquad\qquad \neg d \ \vee \ e\ ) \\
c_4 &: (\ \neg a \ \vee \qquad\qquad \neg d \ \vee \ \neg e\ ) \\
c_5 &: (\ a \ \vee \ b\ ) \\
c_6 &: (\ a \ \vee \ \neg b\ ) \\
c_7 &: (\ b \ \vee \ c\ ) \\
c_8 &: (\ \neg b \ \vee \ \neg c\ )
\end{aligned}
$$



Decision

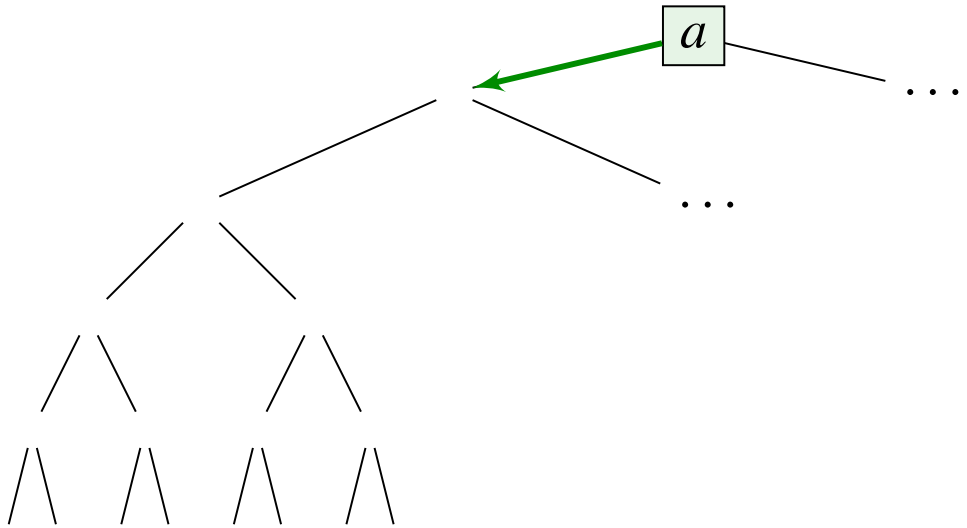
SAT solving: Conflict-directed clause learning

$$
\begin{array}{lllllllll}
c_1: & (& \neg a & \vee & & d & \vee & e &) \\
c_2: & (& \neg a & \vee & & d & \vee & \neg e &) \\
c_3: & (& \neg a & \vee & & \neg d & \vee & e &) \\
c_4: & (& \neg a & \vee & & \neg d & \vee & \neg e &) \\
c_5: & (& a & \vee & b & & & &) \\
c_6: & (& a & \vee & \neg b & & & &) \\
c_7: & (& & & b & \vee & c & &) \\
c_8: & (& & & \neg b & \vee & \neg c & &)
\end{array}
$$



SAT solving: Conflict-directed clause learning

$$
\begin{array}{lllllllll}
c_1: & (& \neg a & \vee & & d & \vee & e &) \\
c_2: & (& \neg a & \vee & & d & \vee & \neg e &) \\
c_3: & (& \neg a & \vee & & \neg d & \vee & e &) \\
c_4: & (& \neg a & \vee & & \neg d & \vee & \neg e &) \\
c_5: & (& a & \vee & b & & & &) \\
c_6: & (& a & \vee & \neg b & & & &) \\
c_7: & (& & & b & \vee & c & &) \\
c_8: & (& & & \neg b & \vee & \neg c & &)
\end{array}
$$



Decision

SAT solving: Conflict-directed clause learning

$c_1 : (\ \neg a \vee d \vee e\ )$

$c_2 : (\ \neg a \vee d \vee \neg e\ )$

$c_3 : (\ \neg a \vee \neg d \vee e\ )$

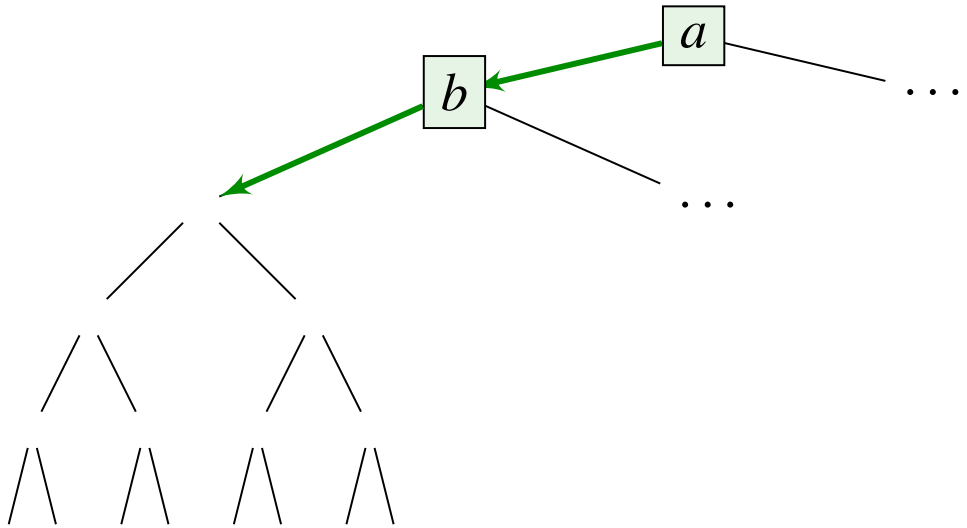
$c_4 : (\ \neg a \vee \neg d \vee \neg e\ )$

$c_5 : (\ a \vee b\ )$

$c_6 : (\ a \vee \neg b\ )$

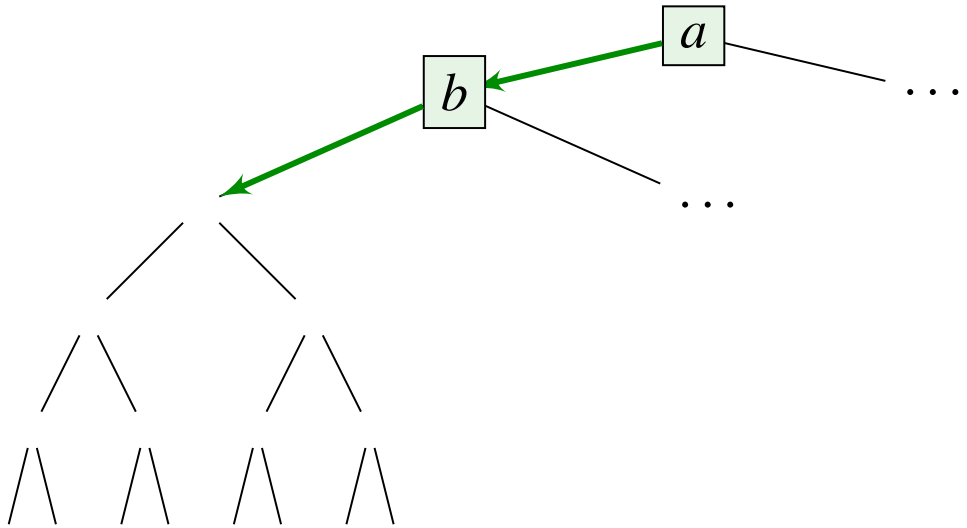
$c_7 : (\ b \vee c\ )$

$c_8 : (\ \neg b \vee \neg c\ )$



SAT solving: Conflict-directed clause learning

$$
\begin{array}{lllllll}
c_1: & (\ \neg a & \vee & & d & \vee & e\ ) \\
c_2: & (\ \neg a & \vee & & d & \vee & \neg e\ ) \\
c_3: & (\ \neg a & \vee & & \neg d & \vee & e\ ) \\
c_4: & (\ \neg a & \vee & & \neg d & \vee & \neg e\ ) \\
c_5: & (\ a & \vee & b & & & \ ) \\
c_6: & (\ a & \vee & \neg b & & & \ ) \\
c_7: & (\ & & b & \vee & c & \ ) \\
c_8: & (\ & & \neg b & \vee & \neg c & \ )
\end{array}
$$



Boolean constraint propagation

SAT solving: Conflict-directed clause learning

$c_1 : (\ \neg a \vee d \vee e\ )$

$c_2 : (\ \neg a \vee d \vee \neg e\ )$

$c_3 : (\ \neg a \vee \neg d \vee e\ )$

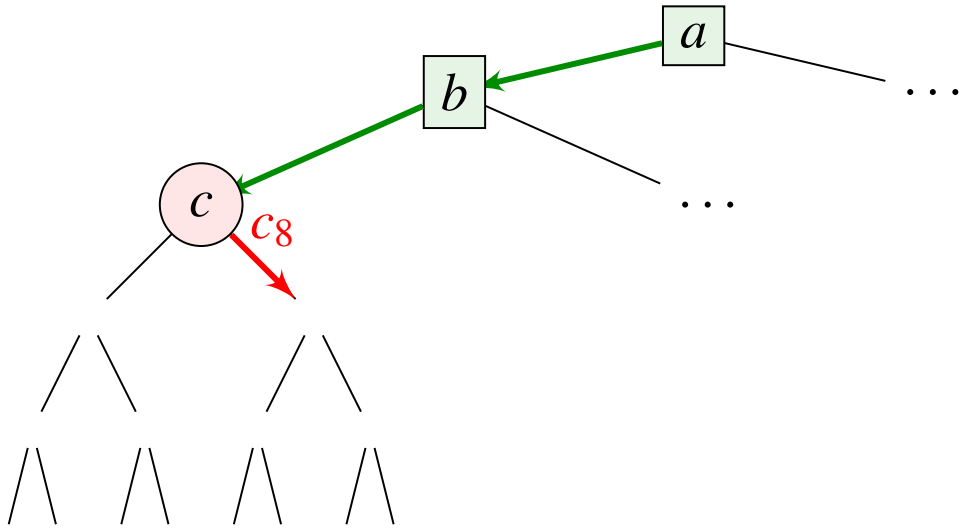
$c_4 : (\ \neg a \vee \neg d \vee \neg e\ )$

$c_5 : (\ a \vee b\ )$

$c_6 : (\ a \vee \neg b\ )$

$c_7 : (\ b \vee c\ )$

$c_8 : (\ \neg b \vee \neg c\ )$



SAT solving: Conflict-directed clause learning

$c_1 : (\ \neg a \vee d \vee e\ )$

$c_2 : (\ \neg a \vee d \vee \neg e\ )$

$c_3 : (\ \neg a \vee \neg d \vee e\ )$

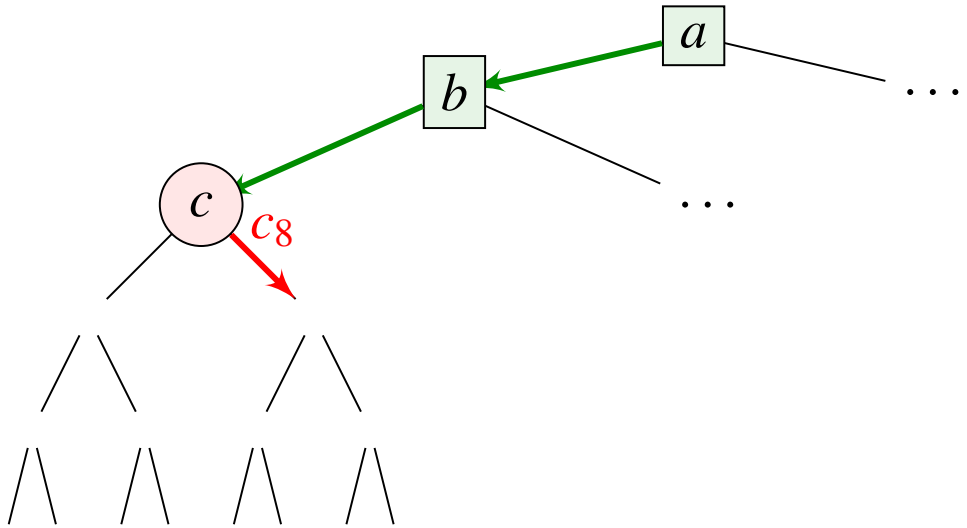
$c_4 : (\ \neg a \vee \neg d \vee \neg e\ )$

$c_5 : (\ a \vee b\ )$

$c_6 : (\ a \vee \neg b\ )$

$c_7 : (\ b \vee c\ )$

$c_8 : (\ \neg b \vee \neg c\ )$



Decision

SAT solving: Conflict-directed clause learning

$c_1 : (\ \neg a \vee d \vee e\ )$

$c_2 : (\ \neg a \vee d \vee \neg e\ )$

$c_3 : (\ \neg a \vee \neg d \vee e\ )$

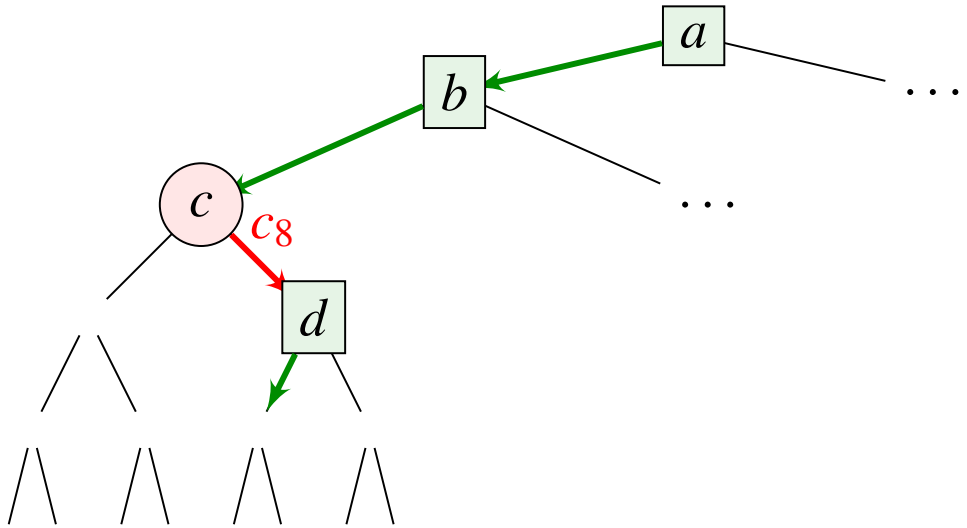
$c_4 : (\ \neg a \vee \neg d \vee \neg e\ )$

$c_5 : (\ a \vee b\ )$

$c_6 : (\ a \vee \neg b\ )$

$c_7 : (\ b \vee c\ )$

$c_8 : (\ \neg b \vee \neg c\ )$



SAT solving: Conflict-directed clause learning

$c_1 : (\ \neg a \vee d \vee e\ )$

$c_2 : (\ \neg a \vee d \vee \neg e\ )$

$c_3 : (\ \neg a \vee \neg d \vee e\ )$

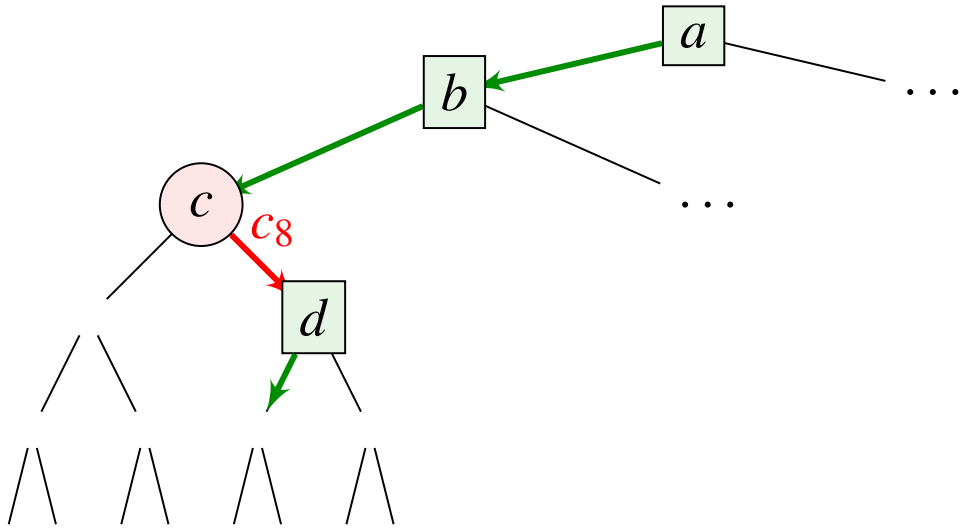
$c_4 : (\ \neg a \vee \neg d \vee \neg e\ )$

$c_5 : (\ a \vee b\ )$

$c_6 : (\ a \vee \neg b\ )$

$c_7 : (\ b \vee c\ )$

$c_8 : (\ \neg b \vee \neg c\ )$



Boolean constraint propagation

SAT solving: Conflict-directed clause learning

$c_1 : (\ \neg a \vee d \vee e\ )$

$c_2 : (\ \neg a \vee d \vee \neg e\ )$

$c_3 : (\ \neg a \vee \neg d \vee e\ )$

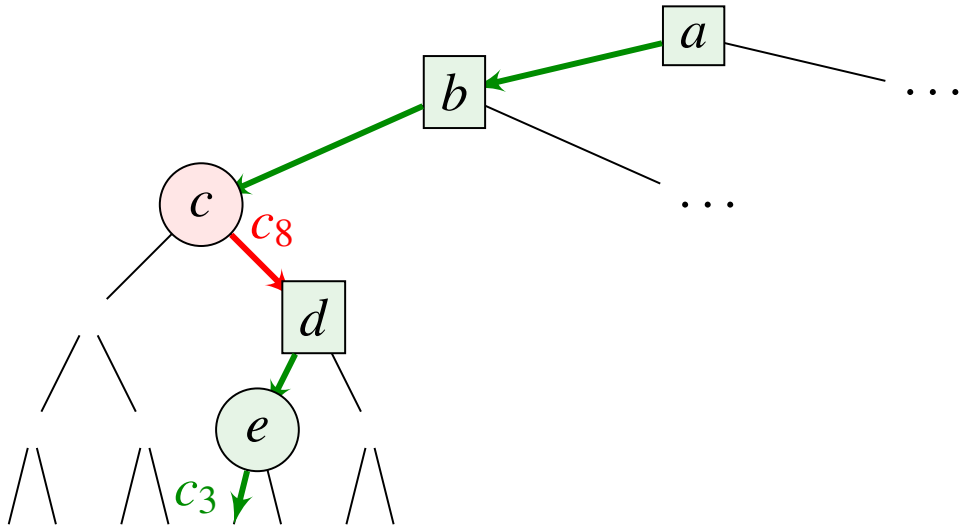
$c_4 : (\ \neg a \vee \neg d \vee \neg e\ )$

$c_5 : (\ a \vee b\ )$

$c_6 : (\ a \vee \neg b\ )$

$c_7 : (\ b \vee c\ )$

$c_8 : (\ \neg b \vee \neg c\ )$



SAT solving: Conflict-directed clause learning

$c_1 : (\ \neg a \vee d \vee e\ )$

$c_2 : (\ \neg a \vee d \vee \neg e\ )$

$c_3 : (\ \neg a \vee \neg d \vee e\ )$

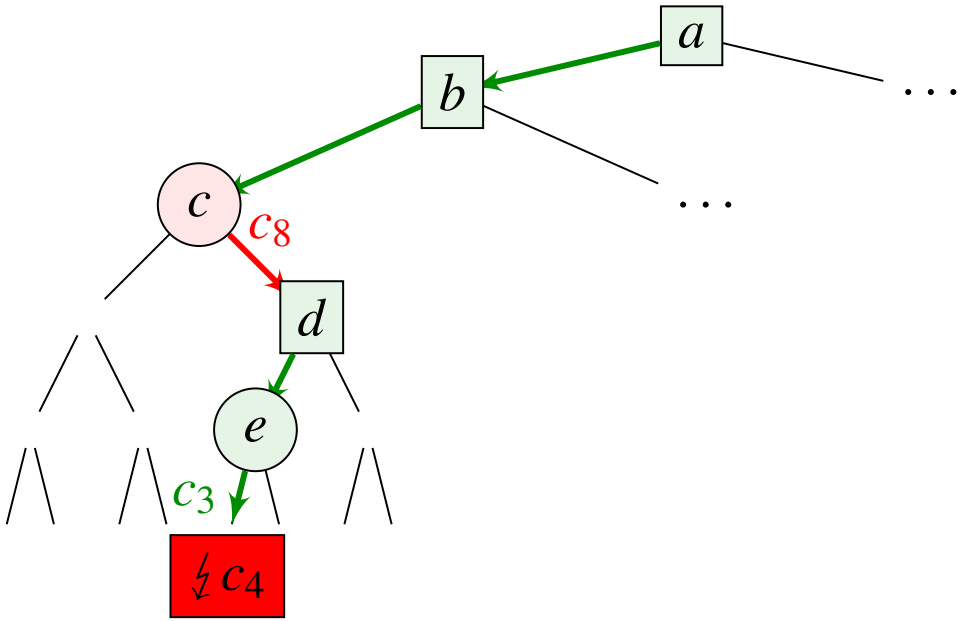
$c_4 : (\ \neg a \vee \neg d \vee \neg e\ )$

$c_5 : (\ a \vee b\ )$

$c_6 : (\ a \vee \neg b\ )$

$c_7 : (\ b \vee c\ )$

$c_8 : (\ \neg b \vee \neg c\ )$



Conflict

SAT solving: Conflict-directed clause learning

$c_1 : (\ \neg a \vee d \vee e\ )$

$c_2 : (\ \neg a \vee d \vee \neg e\ )$

$c_3 : (\ \neg a \vee \neg d \vee e\ )$

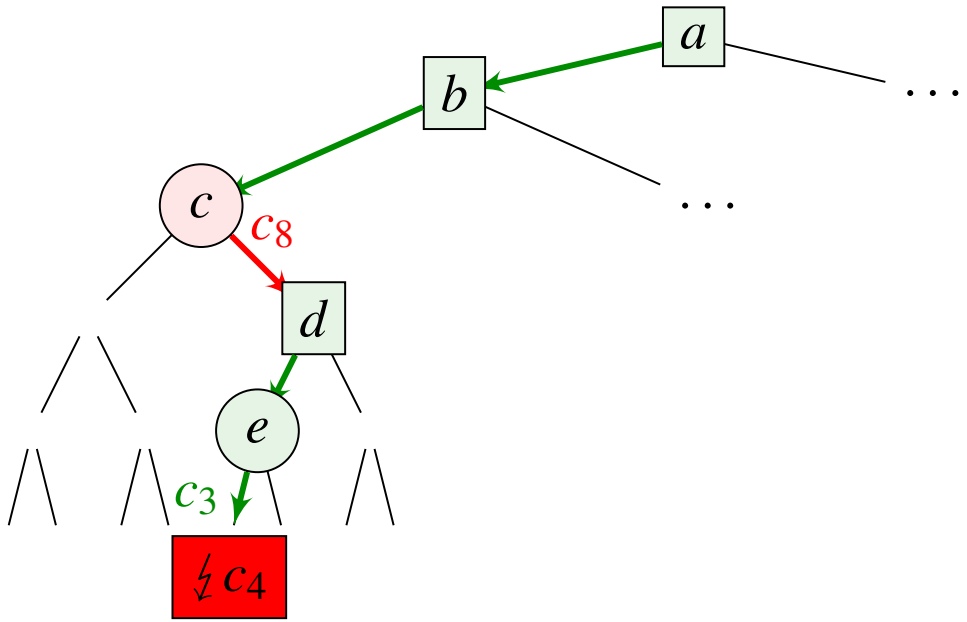
$c_4 : (\ \neg a \vee \neg d \vee \neg e\ )$

$c_5 : (\ a \vee b\ )$

$c_6 : (\ a \vee \neg b\ )$

$c_7 : (\ b \vee c\ )$

$c_8 : (\ \neg b \vee \neg c\ )$



Conflict resolution and backtracking

SAT solving: Conflict-directed clause learning

$c_1 : (\neg a \vee d \vee e)$

$c_2 : (\neg a \vee d \vee \neg e)$

$c_3 : (\neg a \vee \neg d \vee e)$

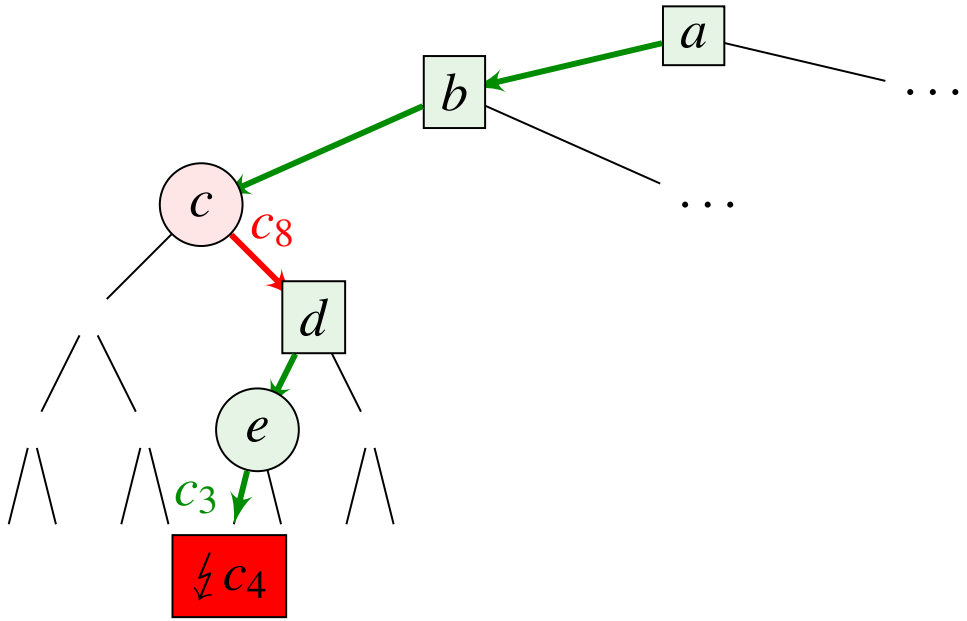
$c_4 : (\neg a \vee \neg d \vee \neg e)$

$c_5 : (a \vee b)$

$c_6 : (a \vee \neg b)$

$c_7 : (b \vee c)$

$c_8 : (\neg b \vee \neg c)$



$$\frac{c_4 : (\neg a \vee \neg d \vee \neg e) \quad c_3 : (\neg a \vee \neg d \vee e)}{c_9 : (\neg a \vee \neg d)}$$

SAT solving: Conflict-directed clause learning

$c_1 : (\neg a \vee d \vee e)$

$c_2 : (\neg a \vee d \vee \neg e)$

$c_3 : (\neg a \vee \neg d \vee e)$

$c_4 : (\neg a \vee \neg d \vee \neg e)$

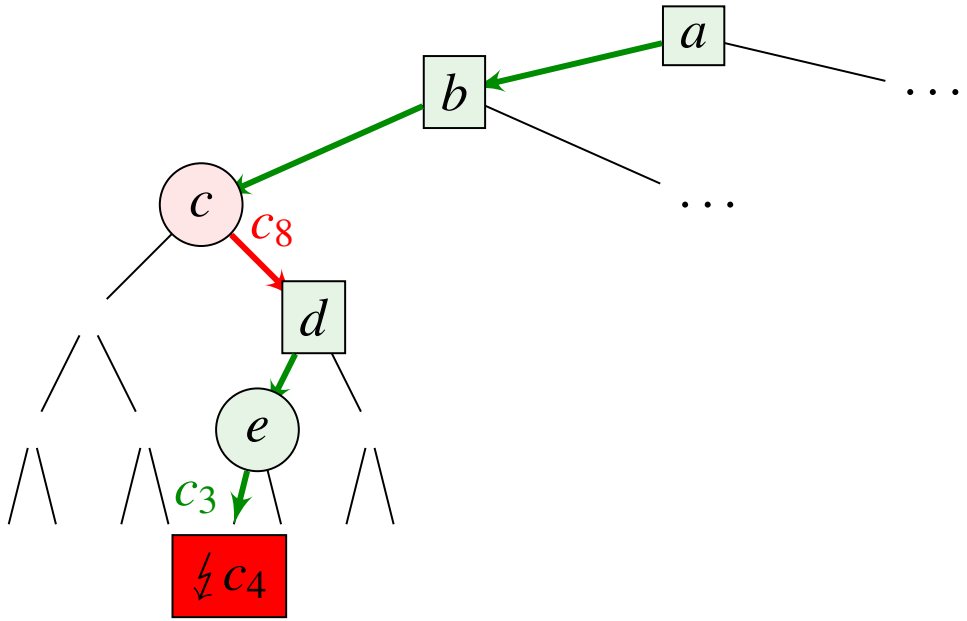
$c_5 : (a \vee b)$

$c_6 : (a \vee \neg b)$

$c_7 : (b \vee c)$

$c_8 : (\neg b \vee \neg c)$

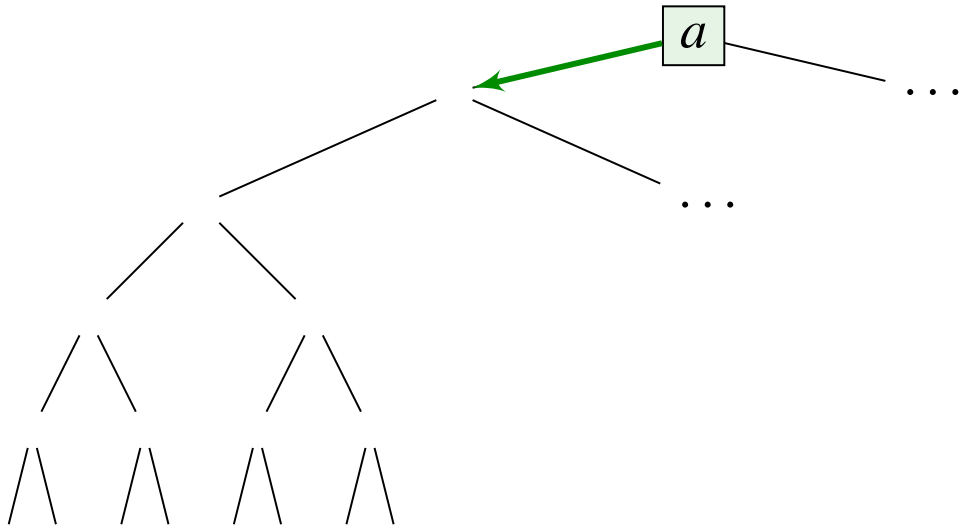
$c_9 : (\neg a \vee \neg d)$



$$\frac{c_4 : (\neg a \vee \neg d \vee \neg e) \quad c_3 : (\neg a \vee \neg d \vee e)}{c_9 : (\neg a \vee \neg d)}$$

SAT solving: Conflict-directed clause learning

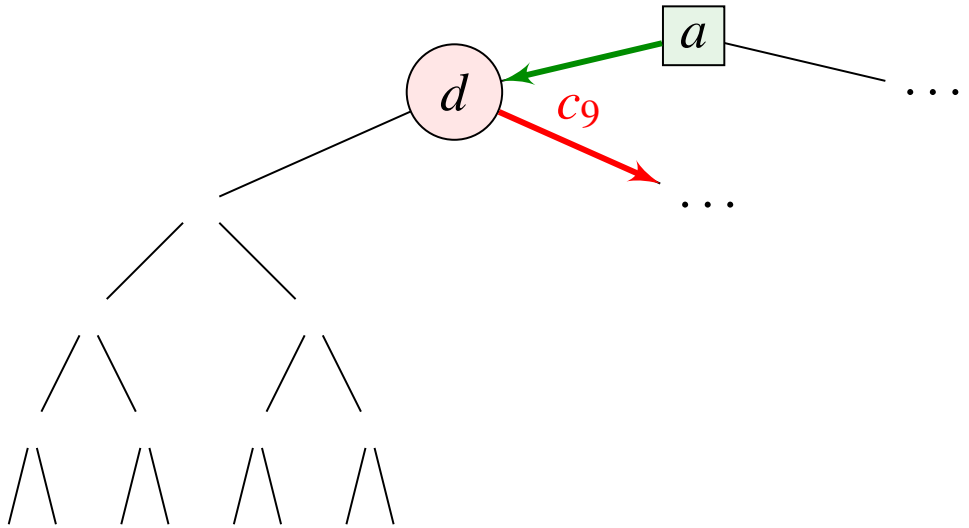
$$
\begin{array}{lllllllll}
c_1: & (\ & \neg a & \vee & & d & \vee & e &) \\
c_2: & (\ & \neg a & \vee & & d & \vee & \neg e &) \\
c_3: & (\ & \neg a & \vee & & \neg d & \vee & e &) \\
c_4: & (\ & \neg a & \vee & & \neg d & \vee & \neg e &) \\
c_5: & (\ & a & \vee & b & & & &) \\
c_6: & (\ & a & \vee & \neg b & & & &) \\
c_7: & (\ & & & b & \vee & c & &) \\
c_8: & (\ & & & \neg b & \vee & \neg c & &) \\
c_9: & (\ & \neg a & \vee & & & \neg d & &)
\end{array}
$$



$$
\frac{c_4 : (\neg a \vee \neg d \vee \neg e) \quad c_3 : (\neg a \vee \neg d \vee e)}{c_9 : (\neg a \vee \neg d)}
$$

SAT solving: Conflict-directed clause learning

$$
\begin{array}{lllllllll}
c_1: & (\ & \neg a & \vee & & & d & \vee & e &) \\
c_2: & (\ & \neg a & \vee & & & d & \vee & \neg e &) \\
c_3: & (\ & \neg a & \vee & & & \neg d & \vee & e &) \\
c_4: & (\ & \neg a & \vee & & & \neg d & \vee & \neg e &) \\
c_5: & (\ & a & \vee & b & & & & &) \\
c_6: & (\ & a & \vee & \neg b & & & & &) \\
c_7: & (\ & & & b & \vee & c & & &) \\
c_8: & (\ & & & \neg b & \vee & \neg c & & &) \\
c_9: & (\ & \neg a & \vee & & & \neg d & & &)
\end{array}
$$



$$\frac{c_4 : (\neg a \vee \neg d \vee \neg e) \quad c_3 : (\neg a \vee \neg d \vee e)}{c_9 : (\neg a \vee \neg d)}$$

Satisfiability modulo theories solving

- Propositional logic is sometimes too weak for modelling.
- We need more expressive logics and decision procedures for them.
- Logics:
  quantifier-free fragments of first-order logic over various theories.
- Our focus: SAT-modulo-theories (SMT) solving.

Satisfiability modulo theories solving

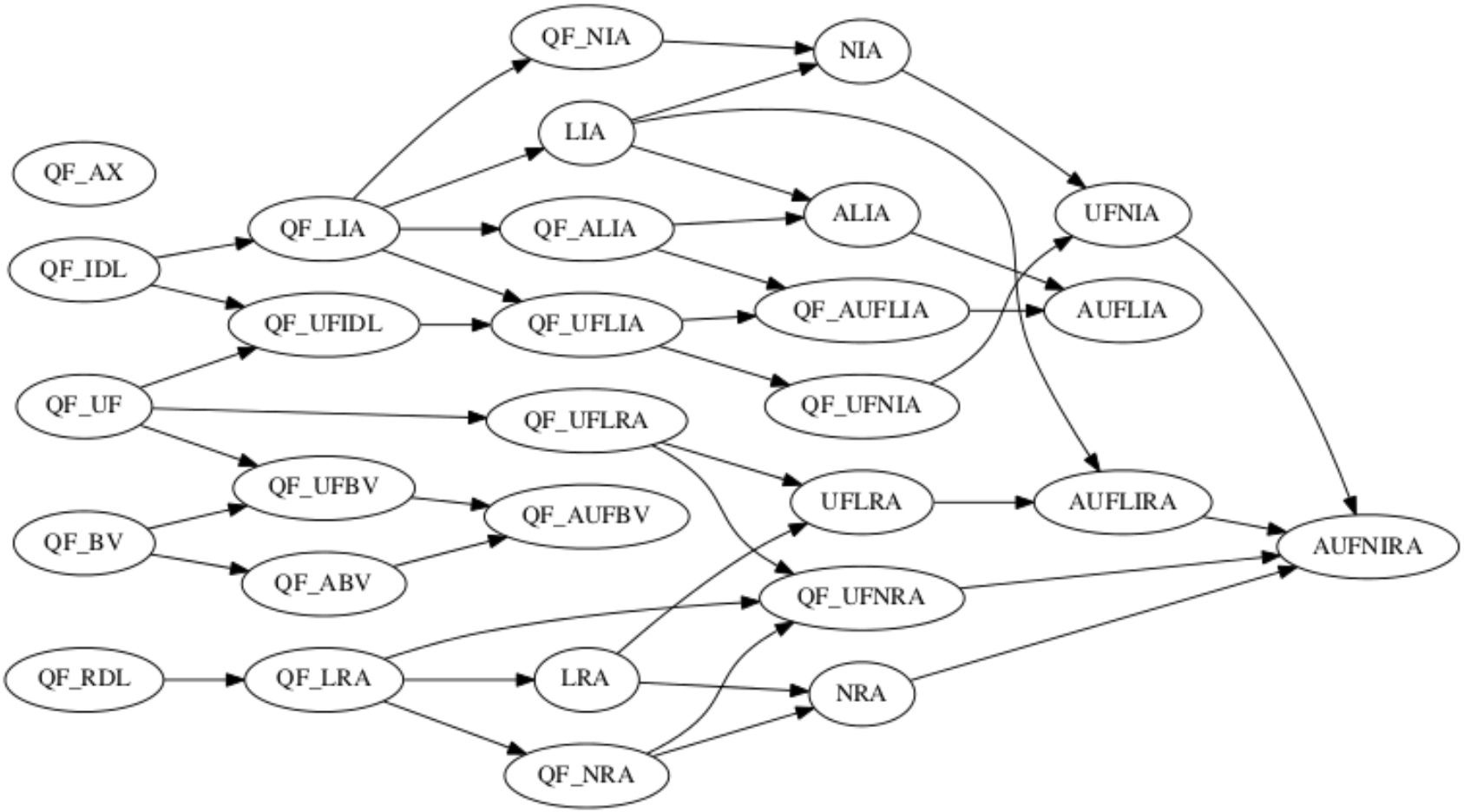
- Propositional logic is sometimes too weak for modelling.
- We need more expressive logics and decision procedures for them.
- Logics:
  quantifier-free fragments of first-order logic over various theories.
- Our focus: SAT-modulo-theories (SMT) solving.
- SMT-LIB as standard input language since 2004.
- Competitions since 2005.
- SMT-COMP 2014 competition:
  - 32 logical categories, 20 solvers.
  - Linear real arithmetic (since 2005): 6 solvers.
  - Non-linear real arithmetic (since 2010): 4 solvers.
  - 67426 benchmark instances.

Satisfiability modulo theories solving

- Propositional logic is sometimes too weak for modelling.
- We need more expressive logics and decision procedures for them.
- Logics:
  quantifier-free fragments of first-order logic over various theories.
- Our focus: SAT-modulo-theories (SMT) solving.
- SMT-LIB as standard input language since 2004.
- Competitions since 2005.
- SMT-COMP 2014 competition:
  - 32 logical categories, 20 solvers.
  - Linear real arithmetic (since 2005): 6 solvers.
  - Non-linear real arithmetic (since 2010): 4 solvers.
  - 67426 benchmark instances.

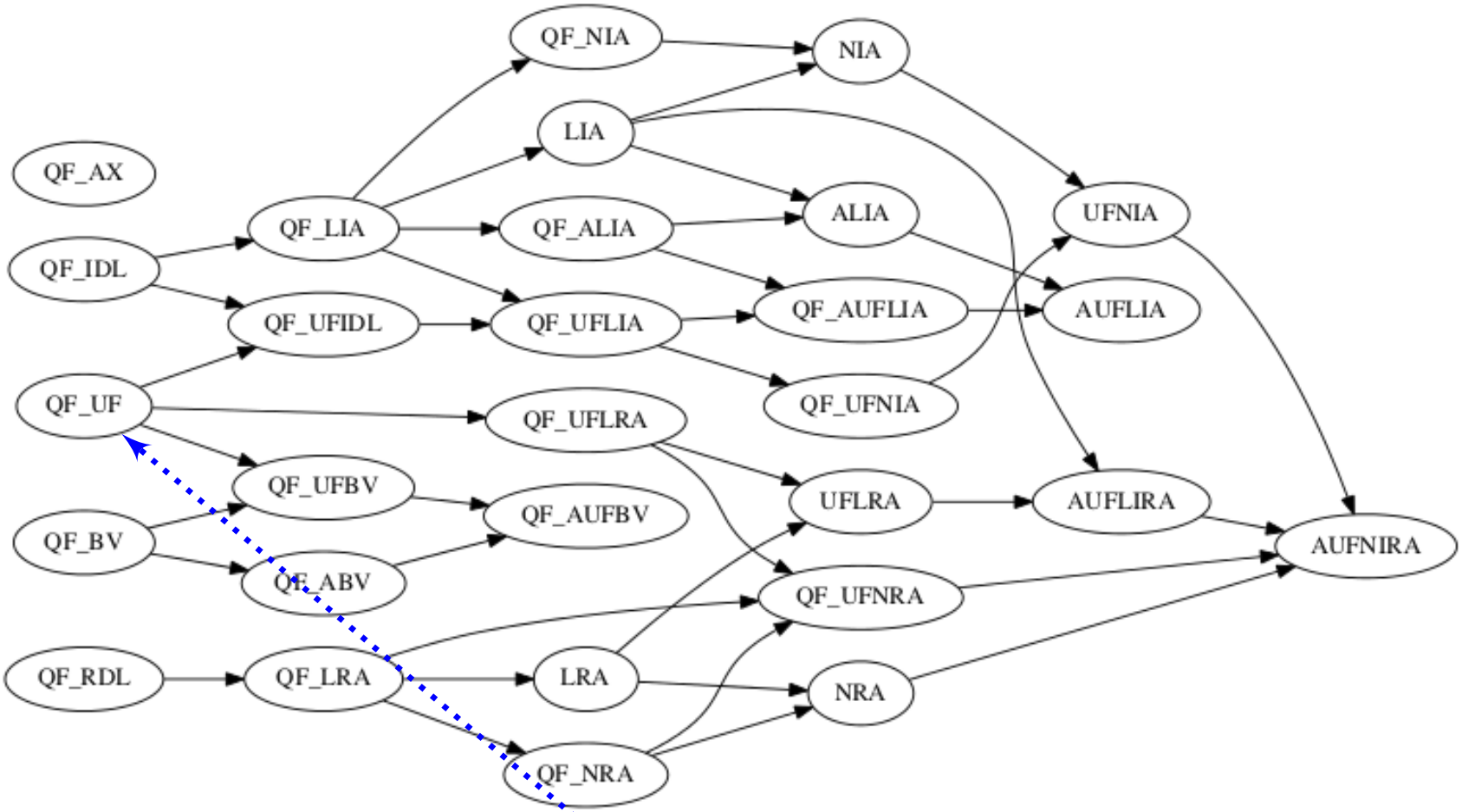
SMT applications: verification (model checking, static analysis, termination analysis); test case generation; controller synthesis; predicate abstraction; equivalence checking; scheduling; planning; product design automation and optimisation, ...

SMT-LIB theories



Source: `http://smtlib.cs.uiowa.edu/logics.shtml`

SMT-LIB theories

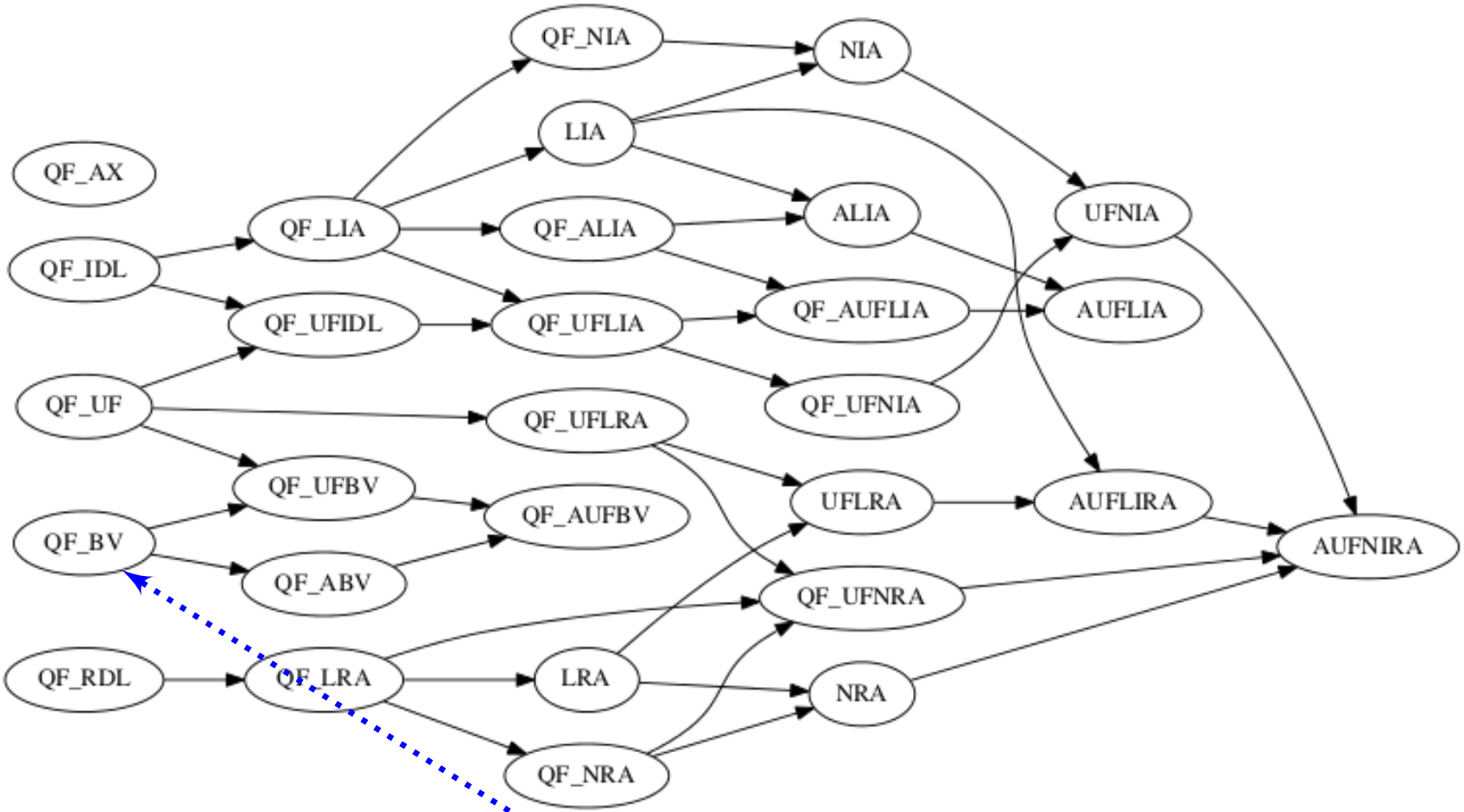


Quantifier-free equality logic with uninterpreted functions

$$(a = c \wedge b = d) \rightarrow f(a,b) = f(c,d)$$

Source: `http://smtlib.cs.uiowa.edu/logics.shtml`

SMT-LIB theories

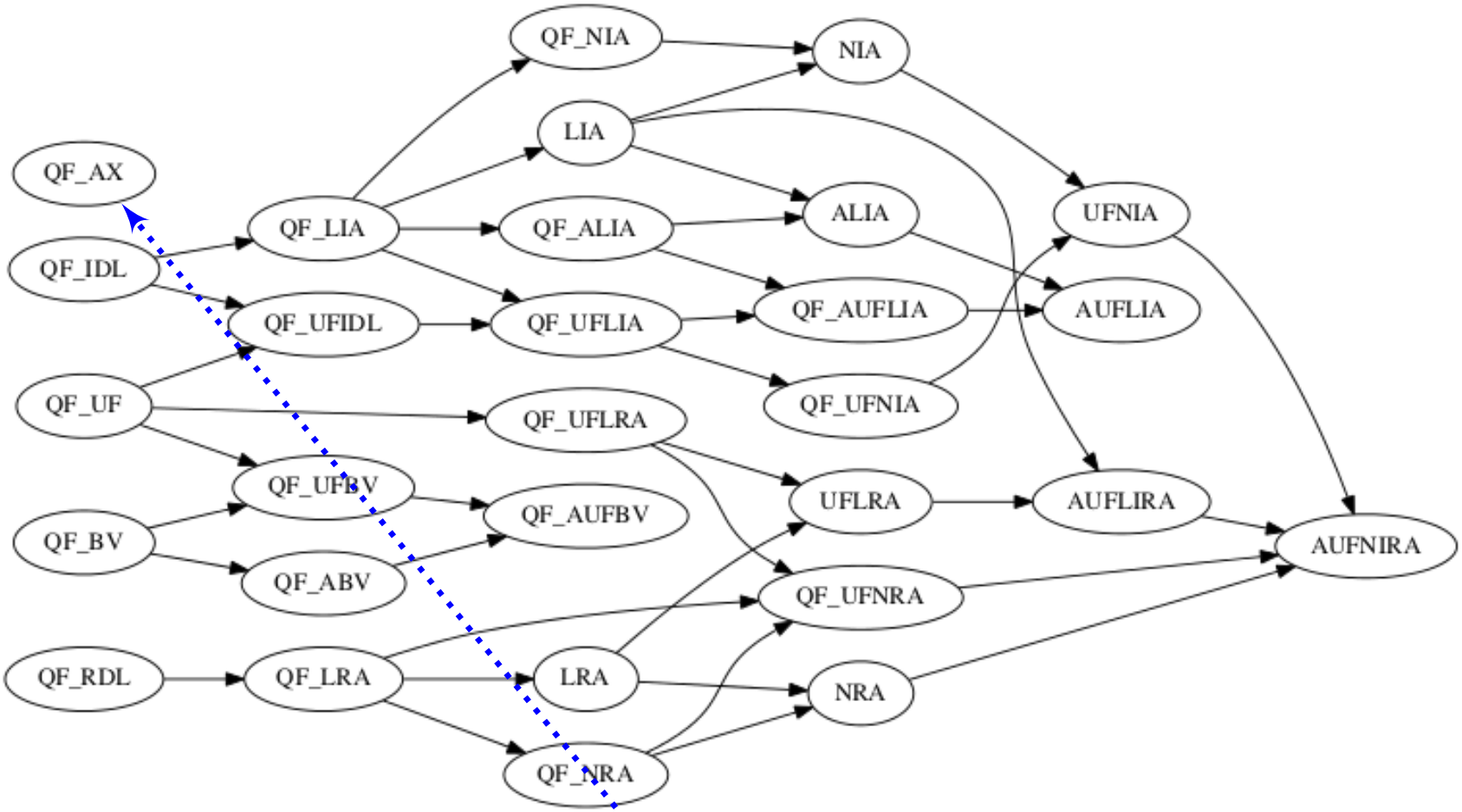


Quantifier-free bit-vector arithmetic

$$(a|b) \leq (a\&b)$$

Source: `http://smtlib.cs.uiowa.edu/logics.shtml`

SMT-LIB theories

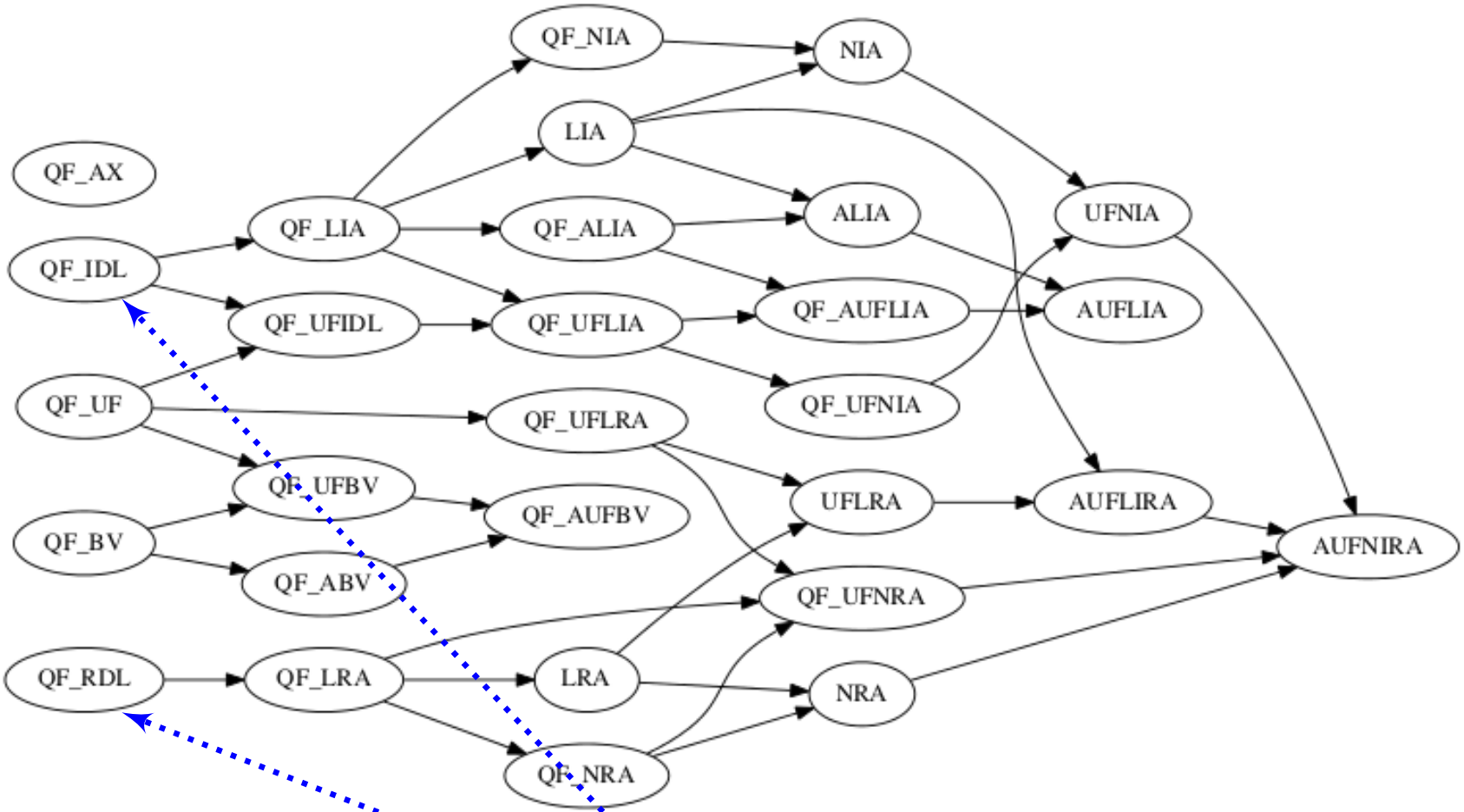


Quantifier-free array theory

$$i = j \rightarrow \mathit{read}(\mathit{write}(a, i, v), j) = v$$

Source: `http://smtlib.cs.uiowa.edu/logics.shtml`

SMT-LIB theories

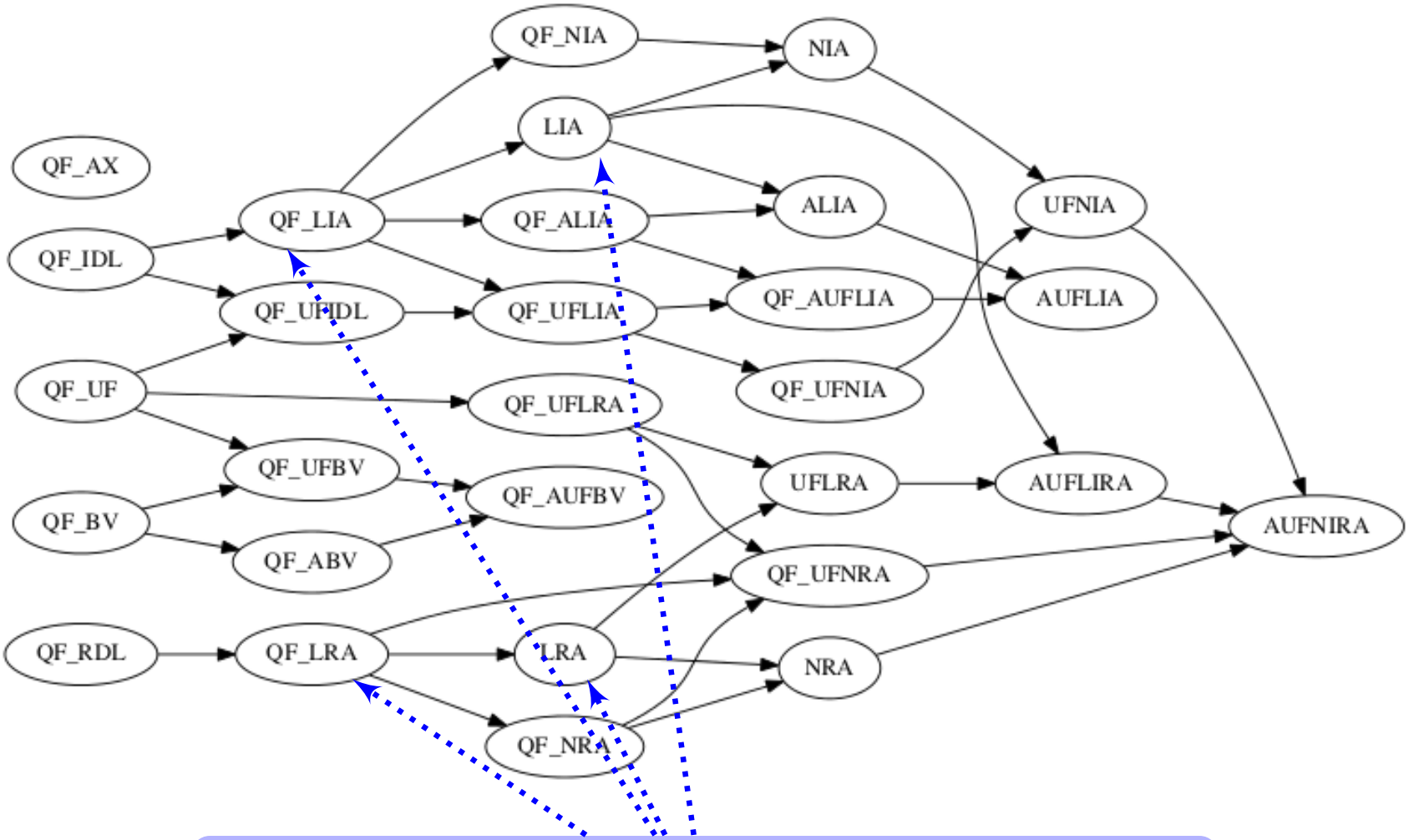


Quantifier-free integer/rational difference logic

$$x - y \sim 0, \sim \in \{<, \leq, =, \geq, >\}$$

Source: `http://smtlib.cs.uiowa.edu/logics.shtml`

SMT-LIB theories

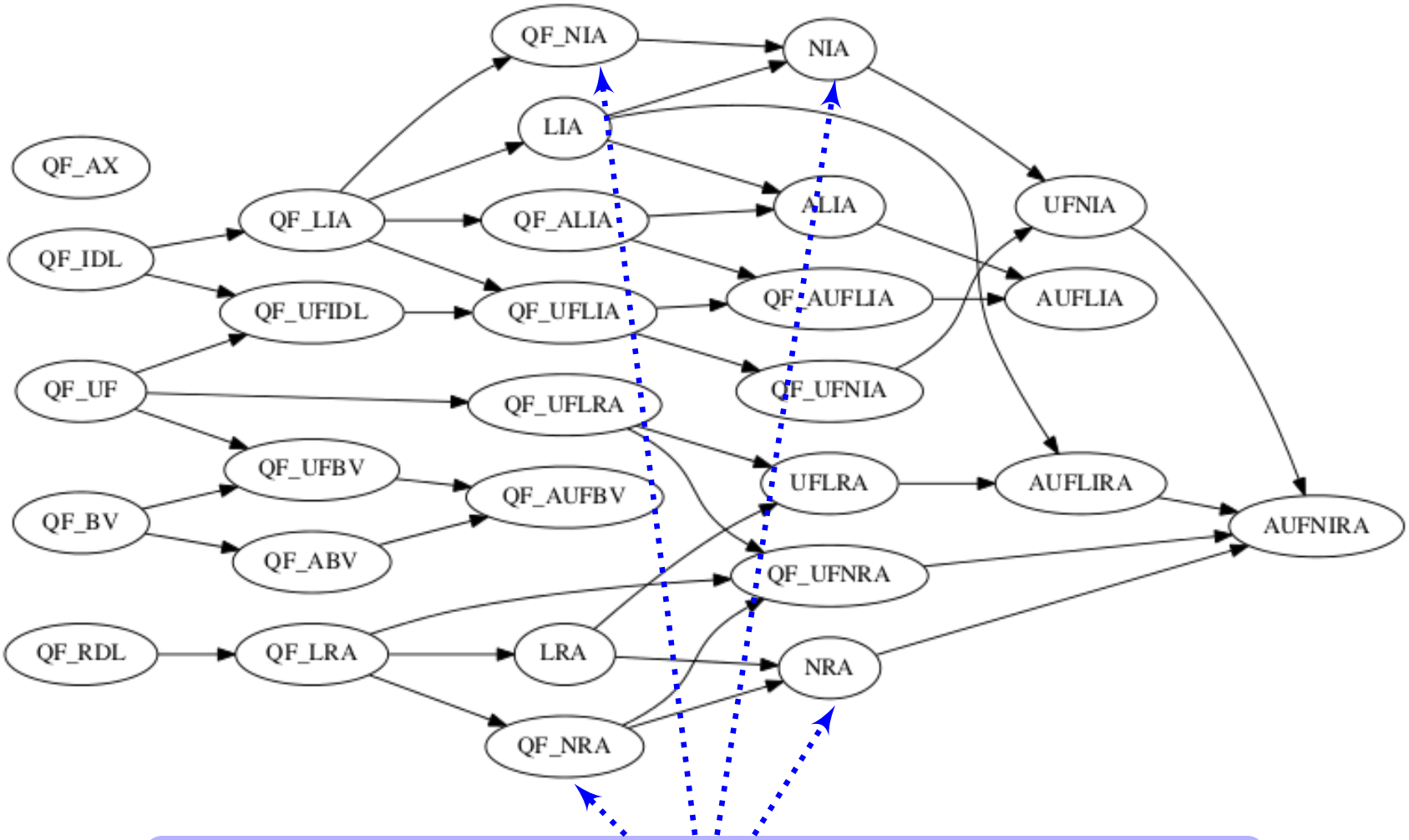


(Quantifier-free) real/integer linear arithmetic

$$3x + 7y = 8$$

Source: `http://smtlib.cs.uiowa.edu/logics.shtml`

SMT-LIB theories

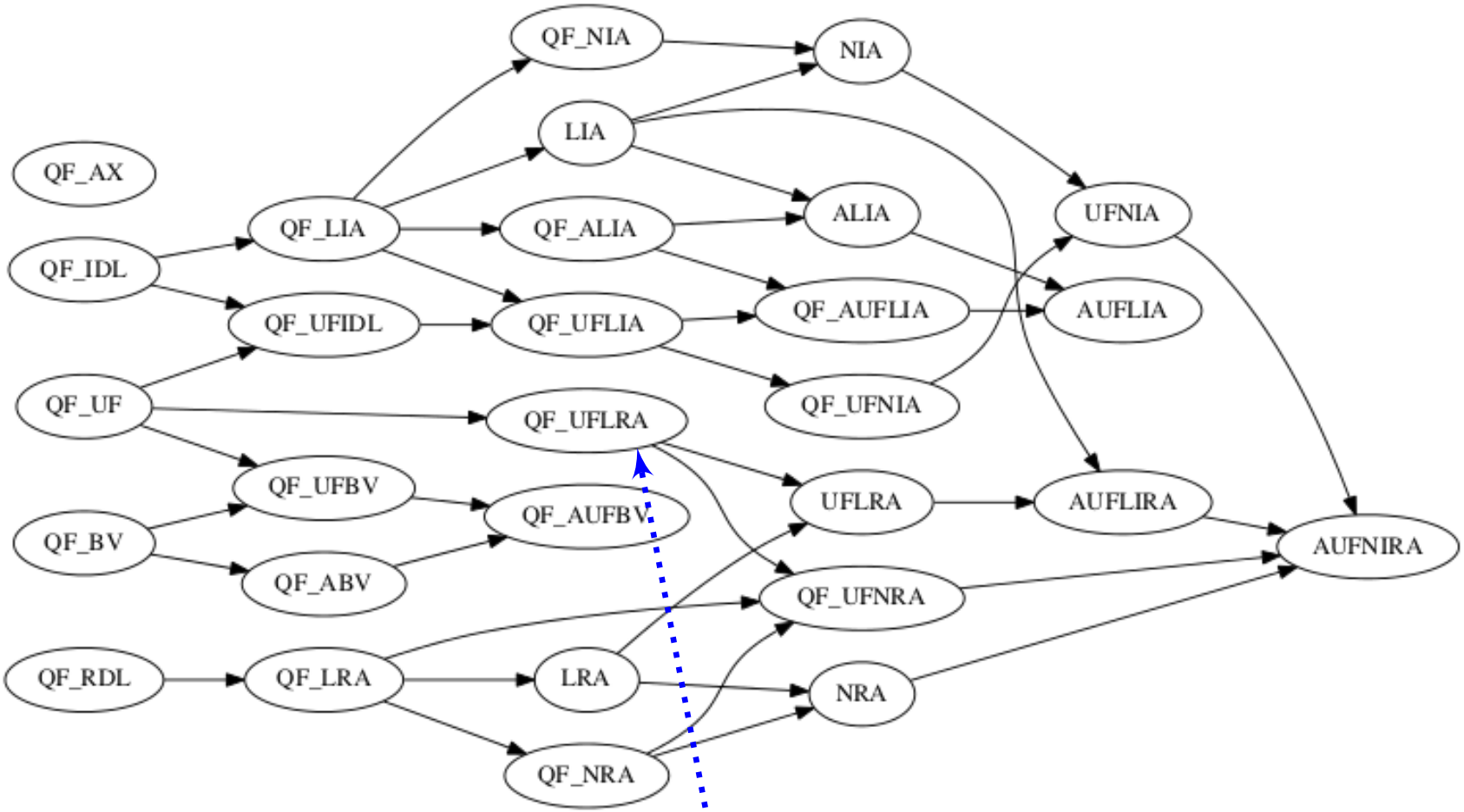


(Quantifier-free) real/integer non-linear arithmetic

$$x^2 + 2xy + y^2 \geq 0$$

Source: `http://smtlib.cs.uiowa.edu/logics.shtml`

SMT-LIB theories



Combined theories

$2f(x) + 5y > 0$

Source: `http://smtlib.cs.uiowa.edu/logics.shtml`

Eager vs. lazy SMT solving

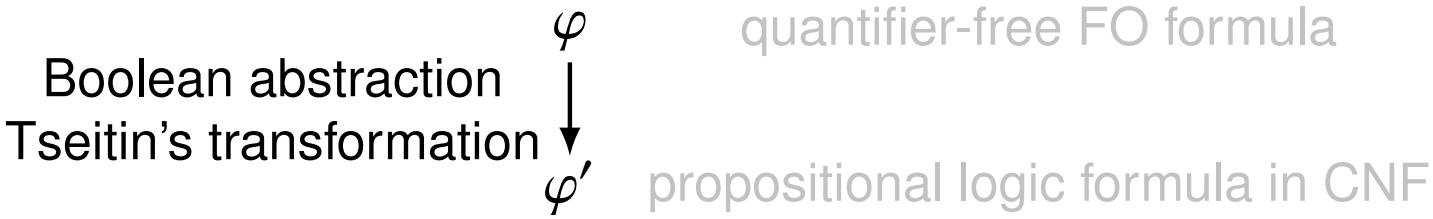
- We focus on lazy SMT solving.
- Alternative eager approach: transform problems into propositional logic and use SAT solving for satisfiability checking.
- Condition: Logic is not more expressive than propositional logic.

(Full/less) lazy SMT solving

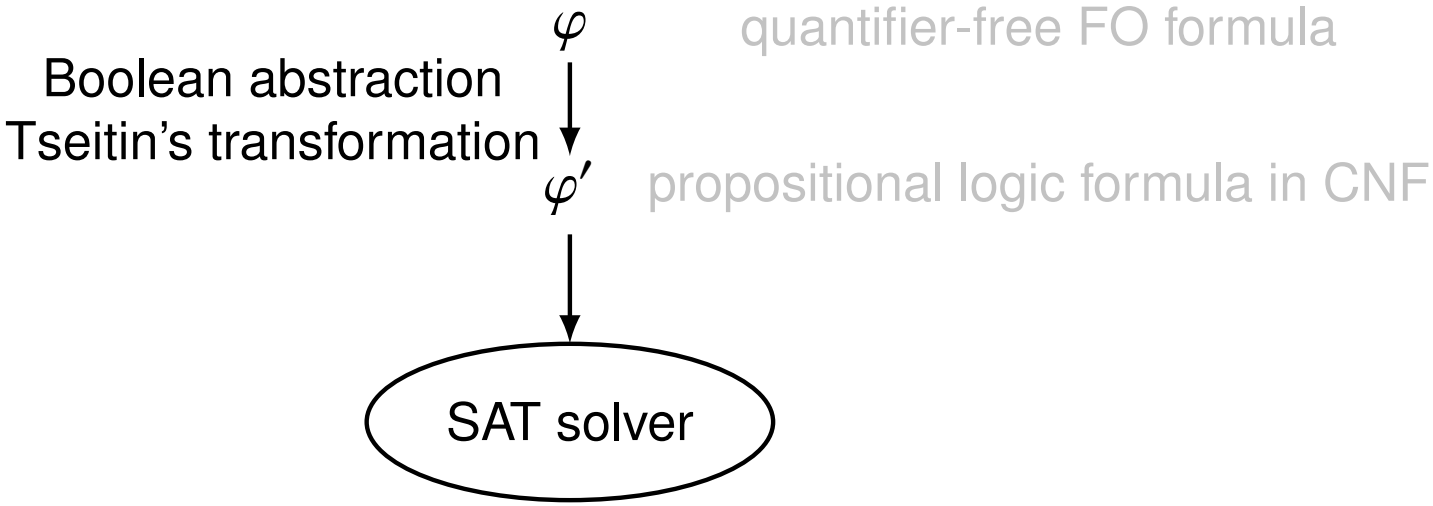
(Full/less) lazy SMT solving

φ quantifier-free FO formula

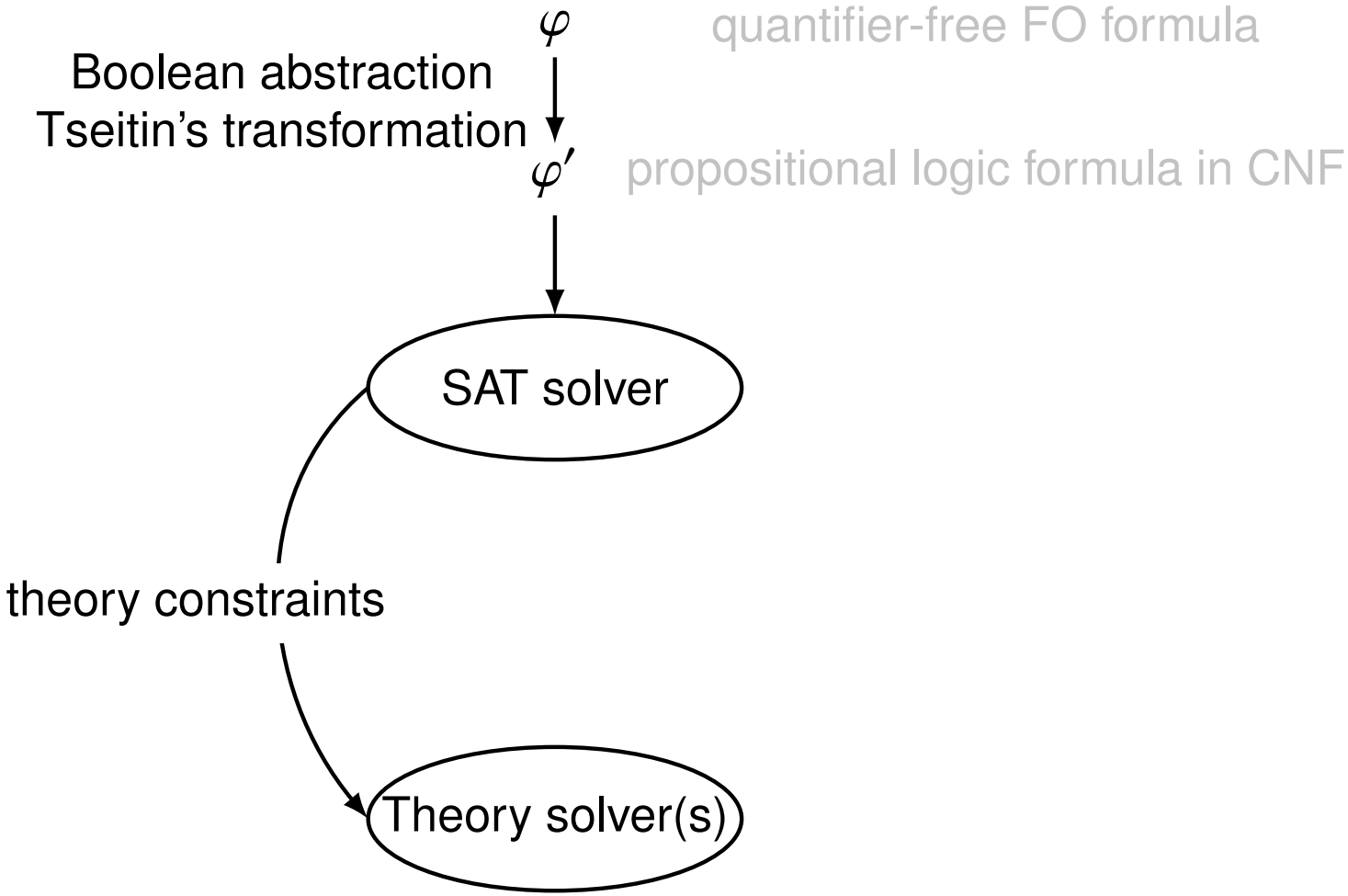
(Full/less) lazy SMT solving



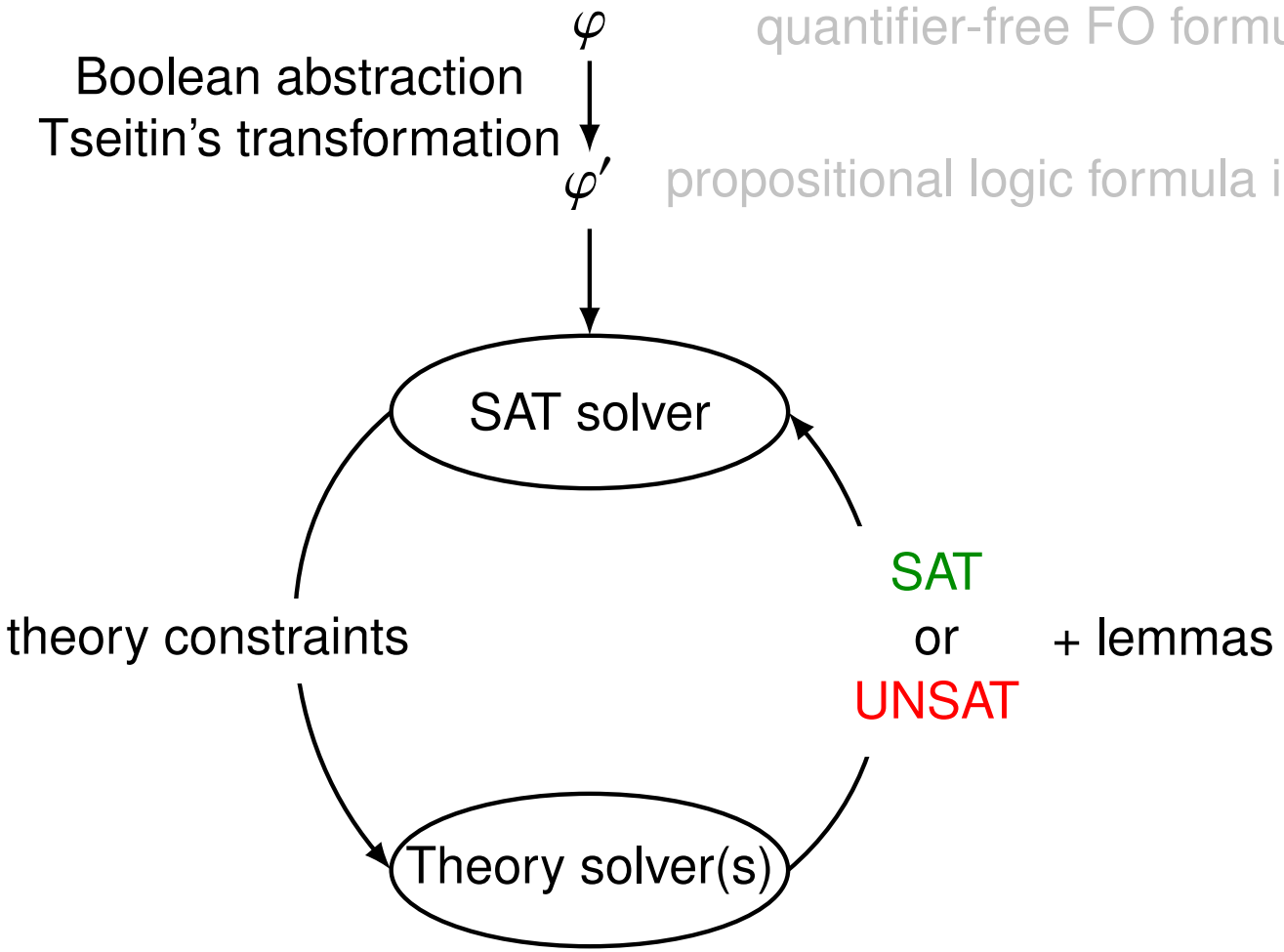
(Full/less) lazy SMT solving



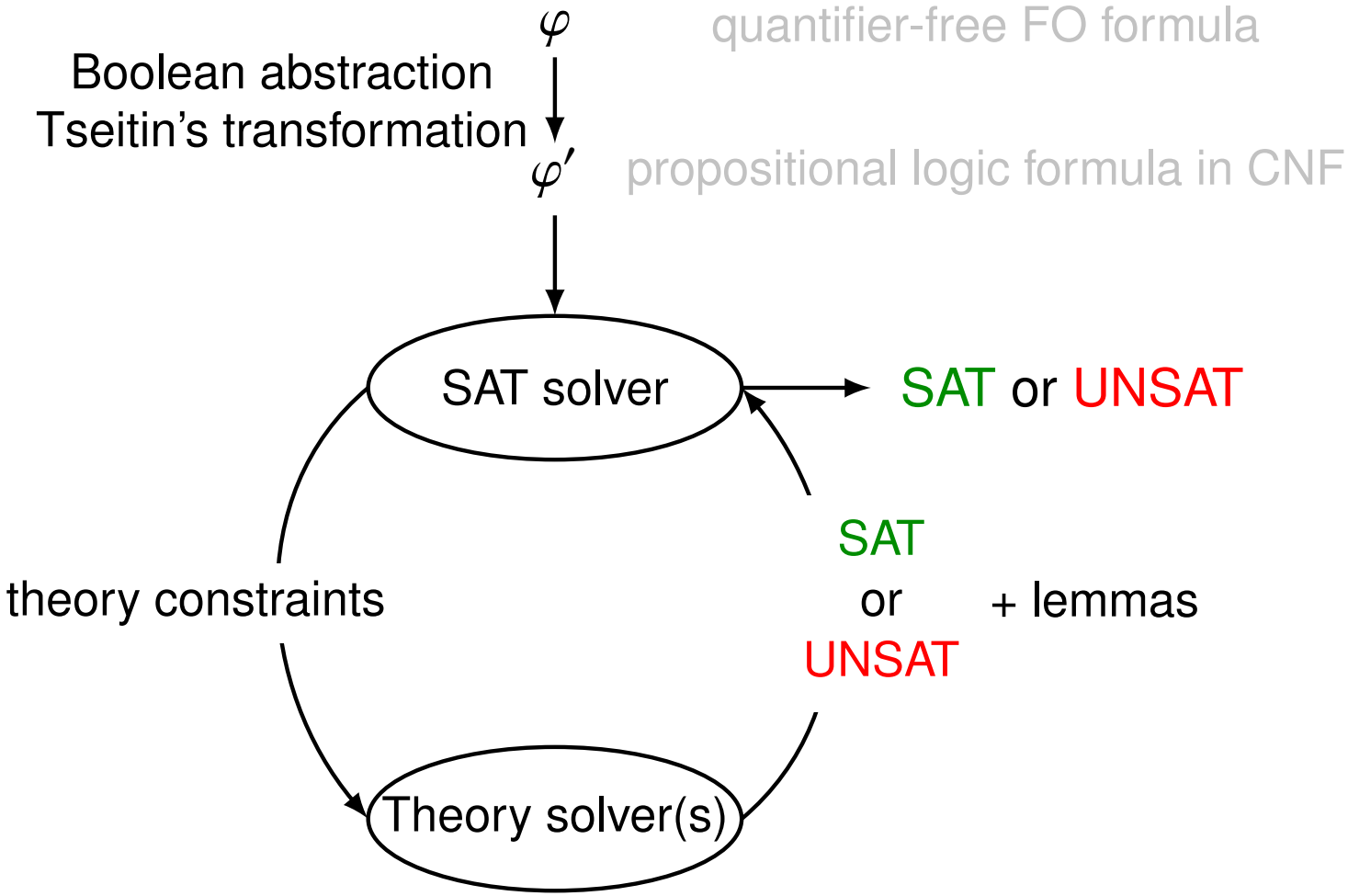
(Full/less) lazy SMT solving



(Full/less) lazy SMT solving



(Full/less) lazy SMT solving



Less lazy SMT solving

Less lazy SMT solving

$$(x < 0 \vee x > 2) \wedge (x^2 = 1 \vee x^2 < 0)$$

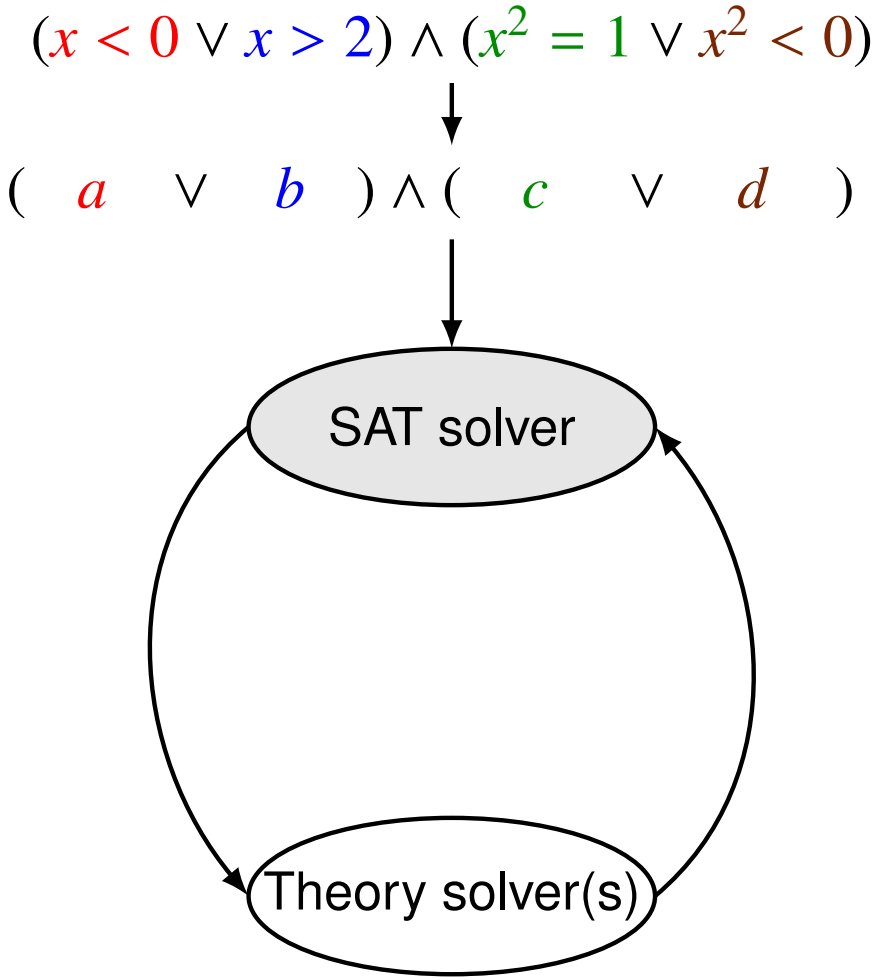
Less lazy SMT solving

$$(x < 0 \vee x > 2) \wedge (x^2 = 1 \vee x^2 < 0)$$

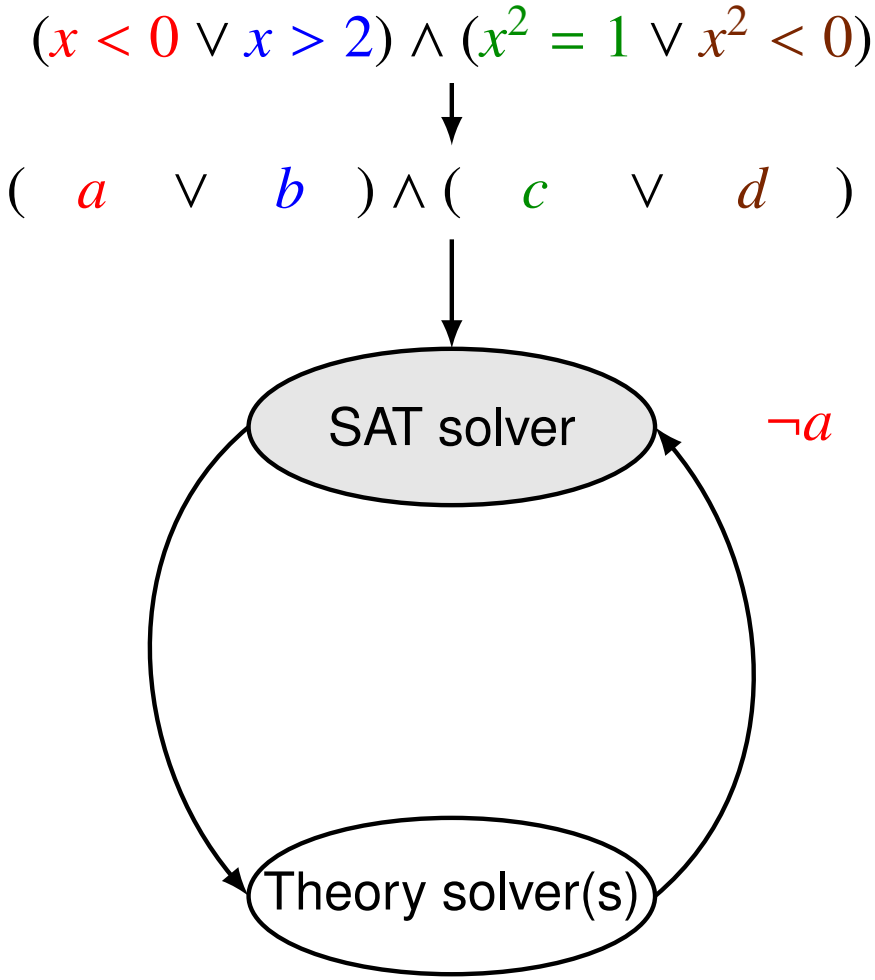
$$\downarrow$$

$$(\ a \ \vee \ b \ ) \wedge (\ c \ \vee \ d \ )$$

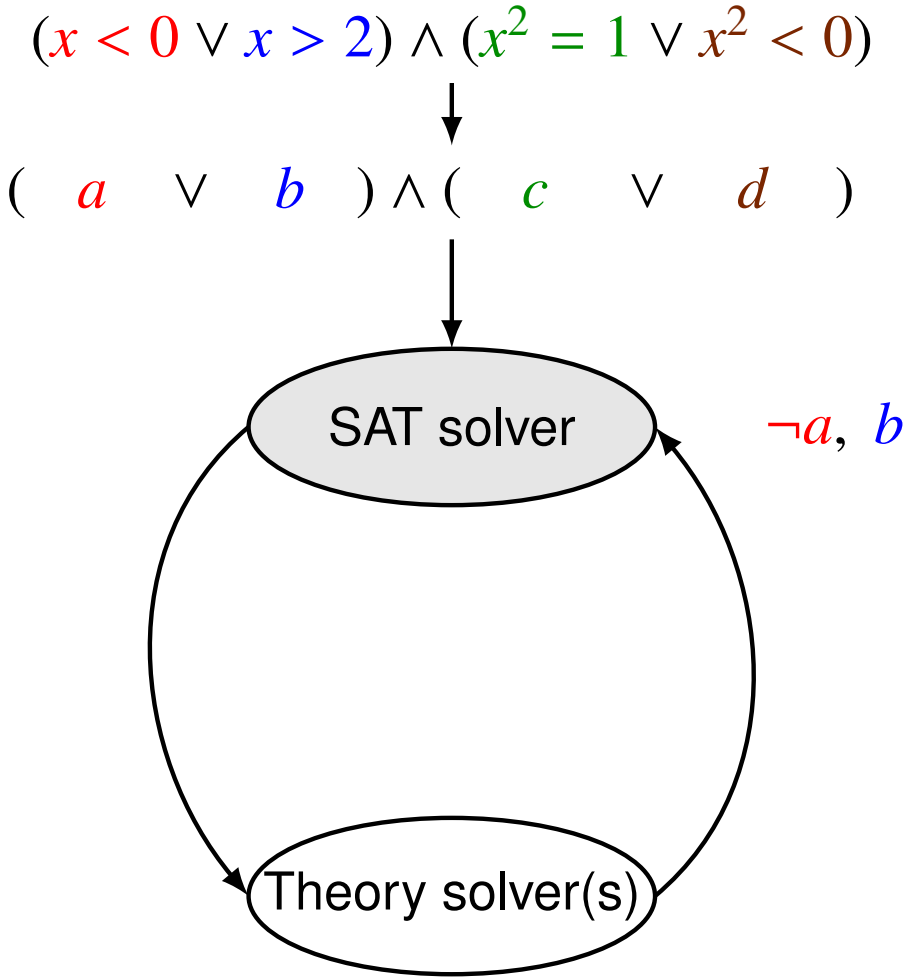
Less lazy SMT solving



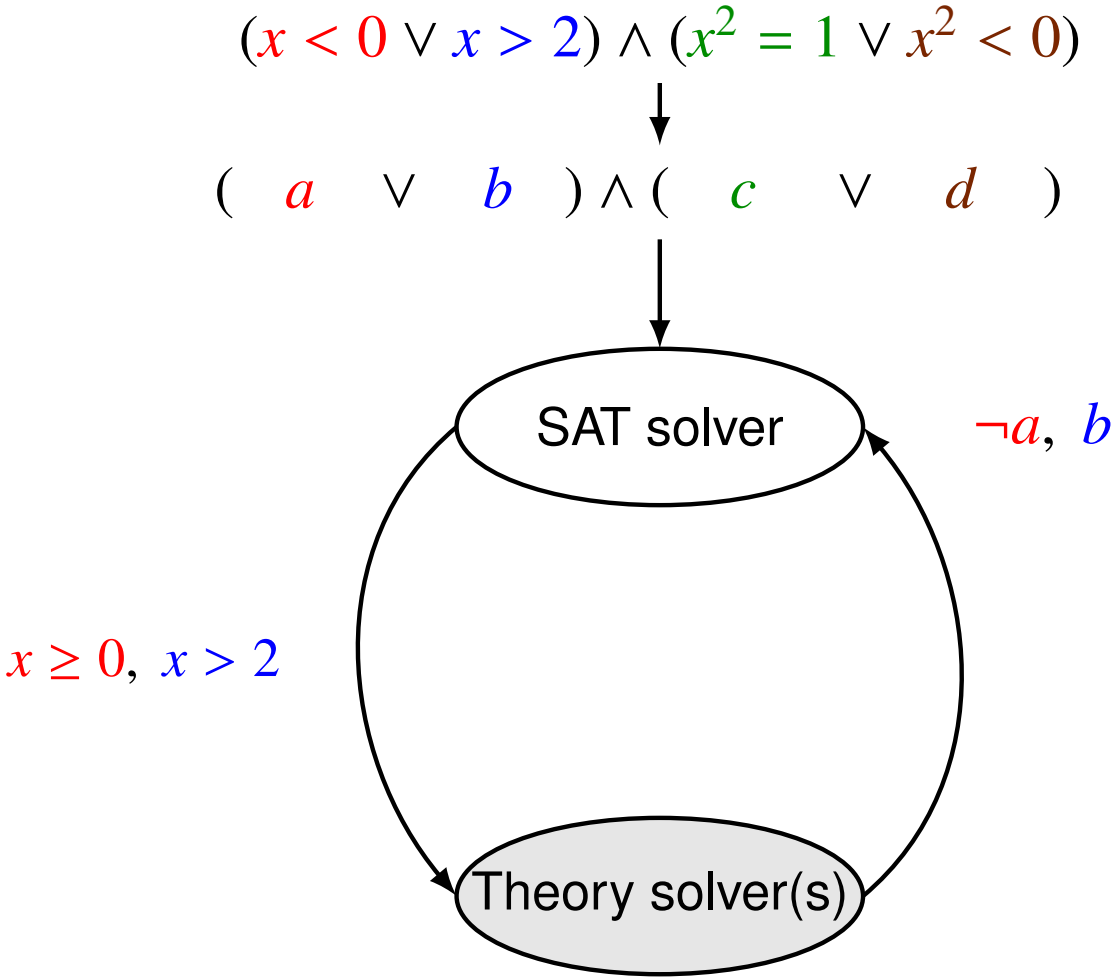
Less lazy SMT solving



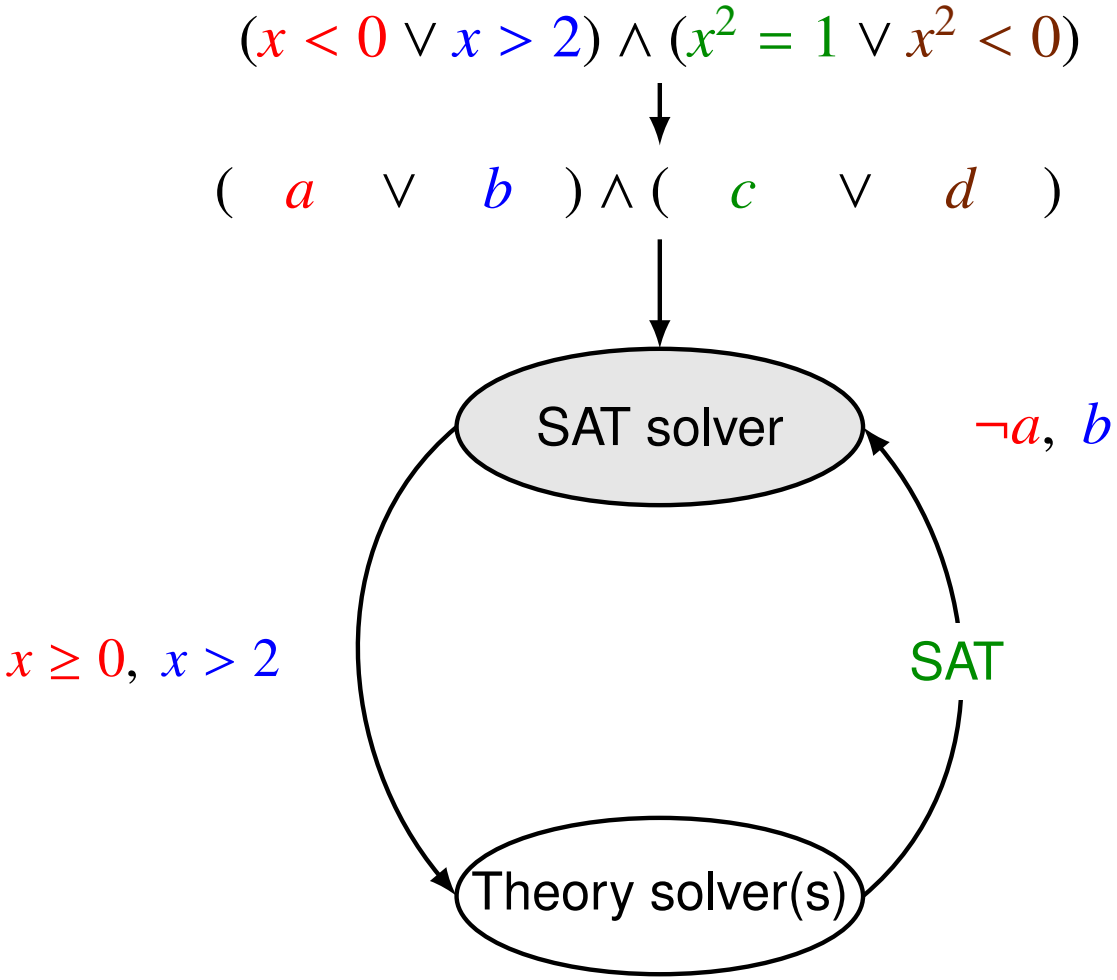
Less lazy SMT solving



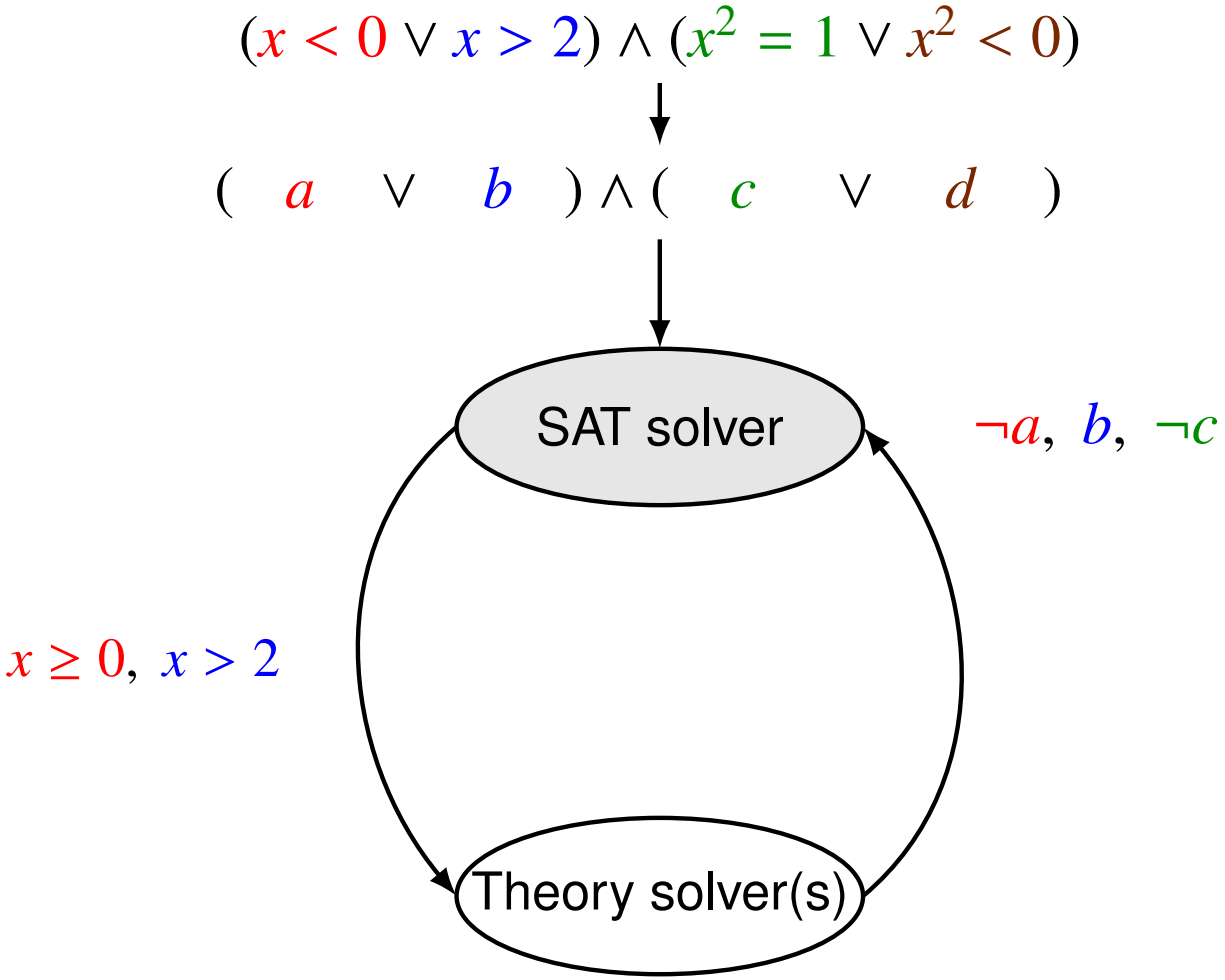
Less lazy SMT solving



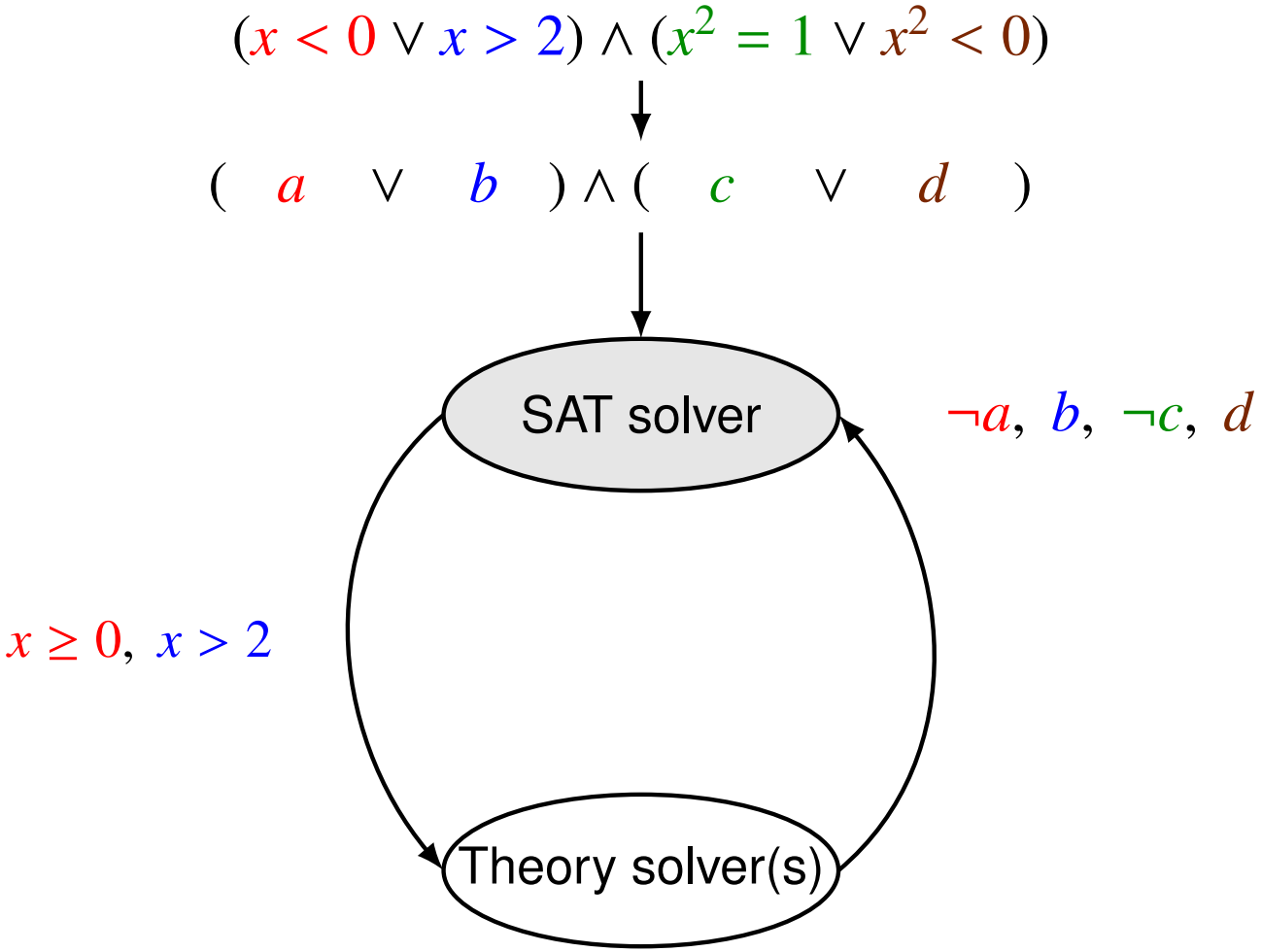
Less lazy SMT solving



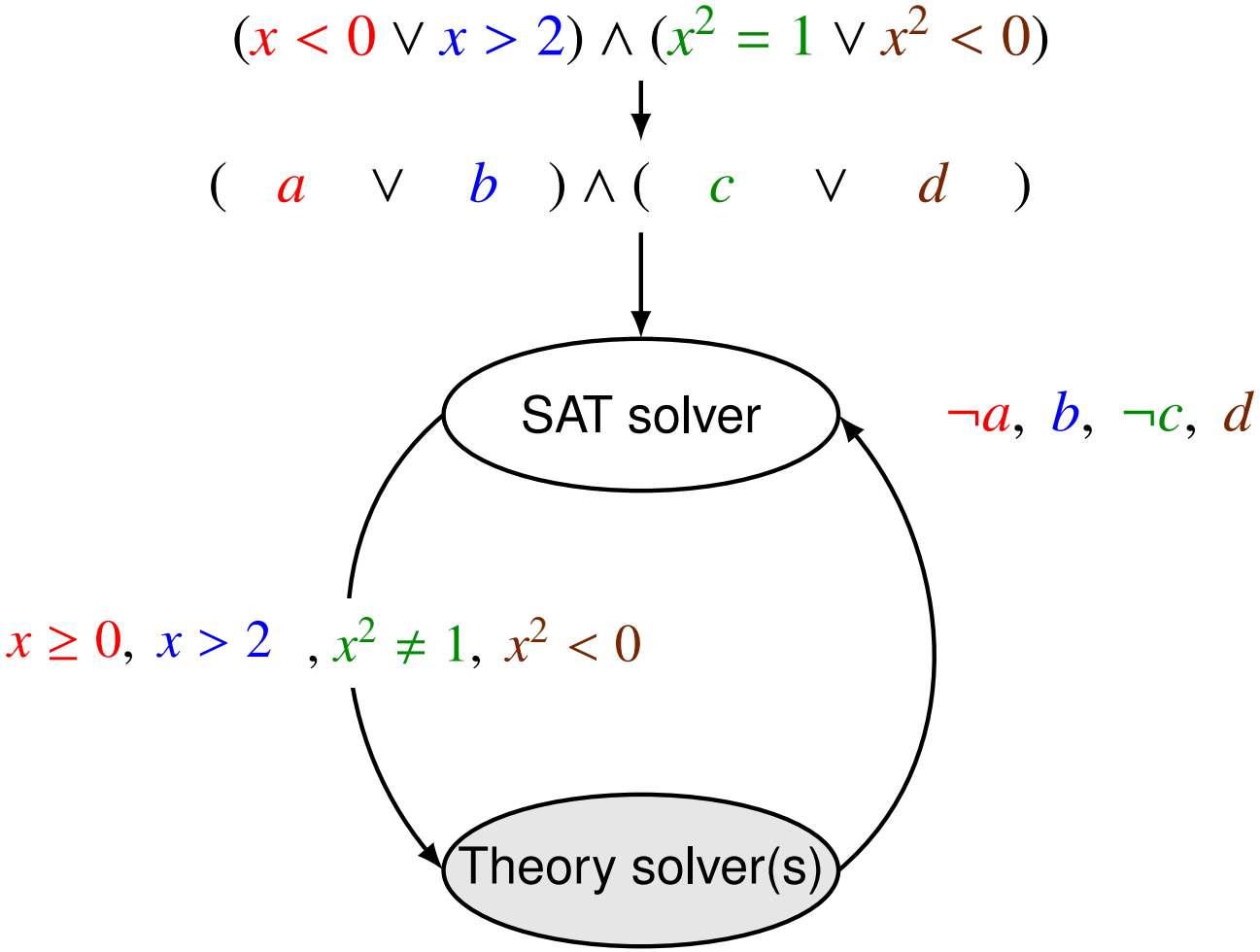
Less lazy SMT solving



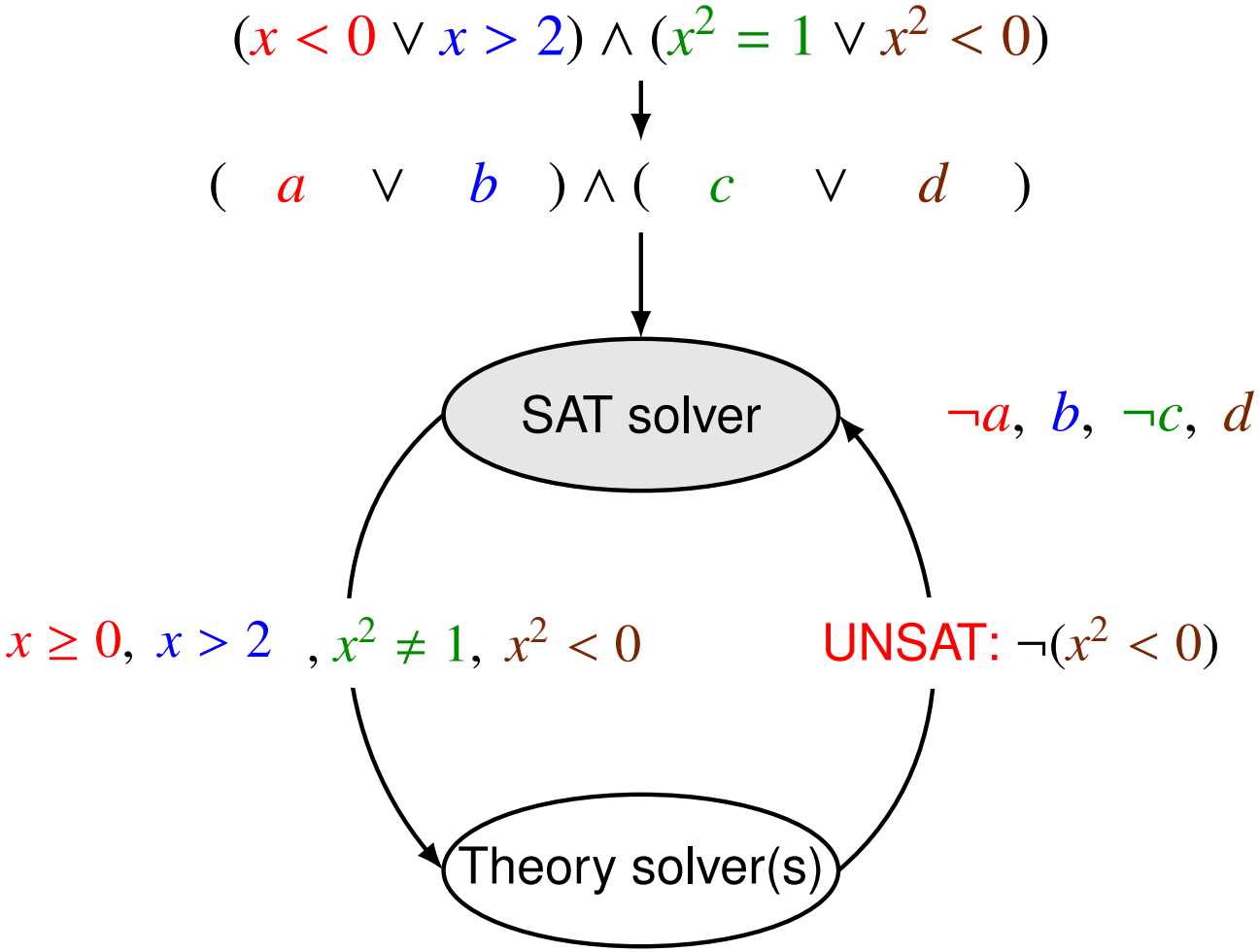
Less lazy SMT solving



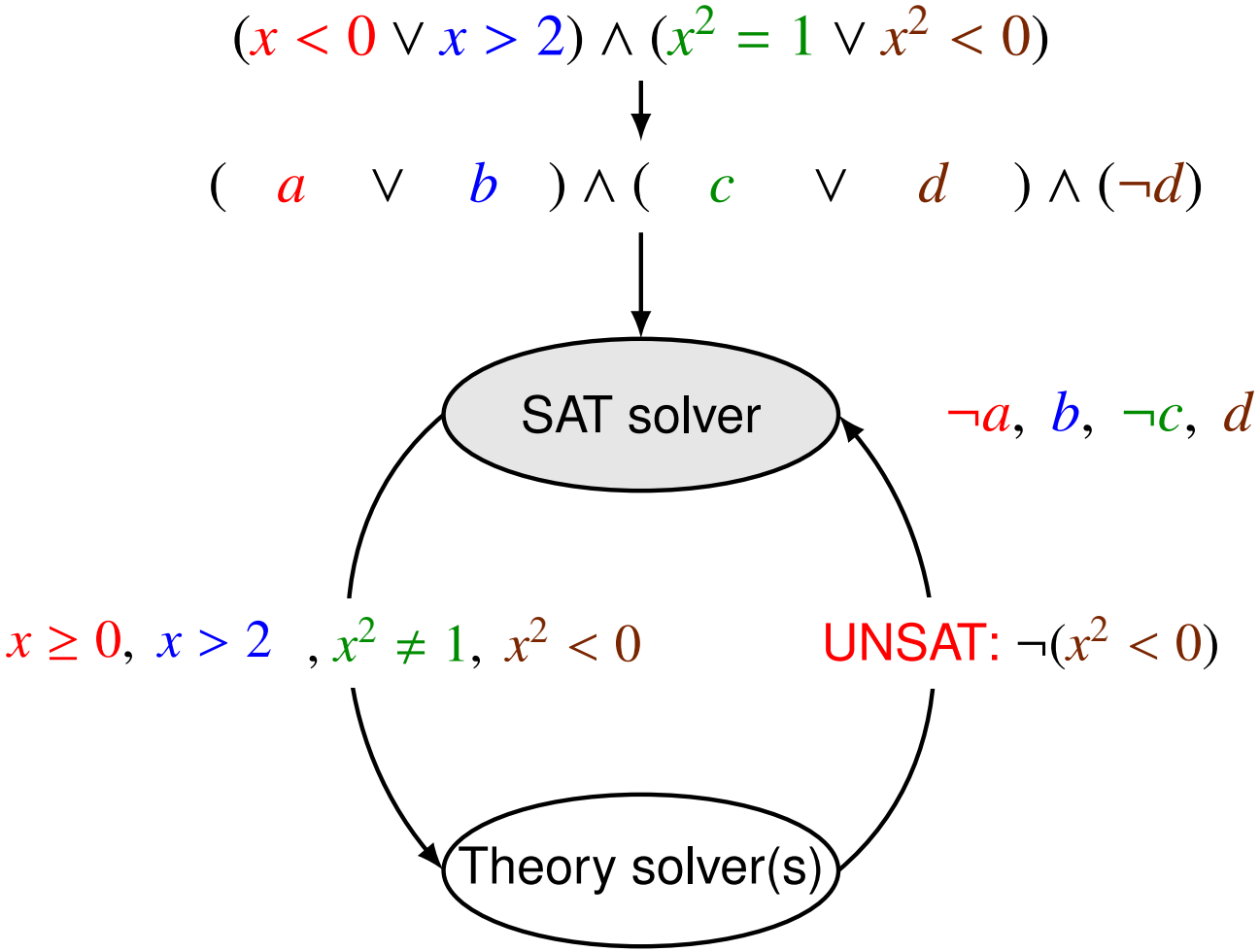
Less lazy SMT solving



Less lazy SMT solving



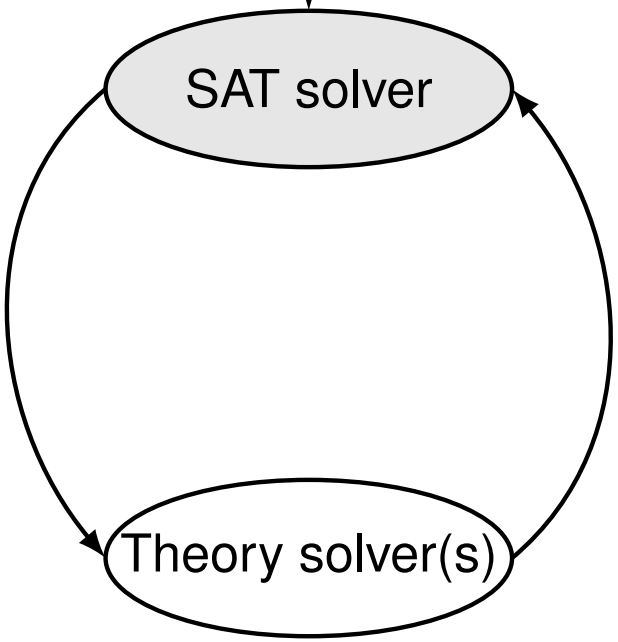
Less lazy SMT solving



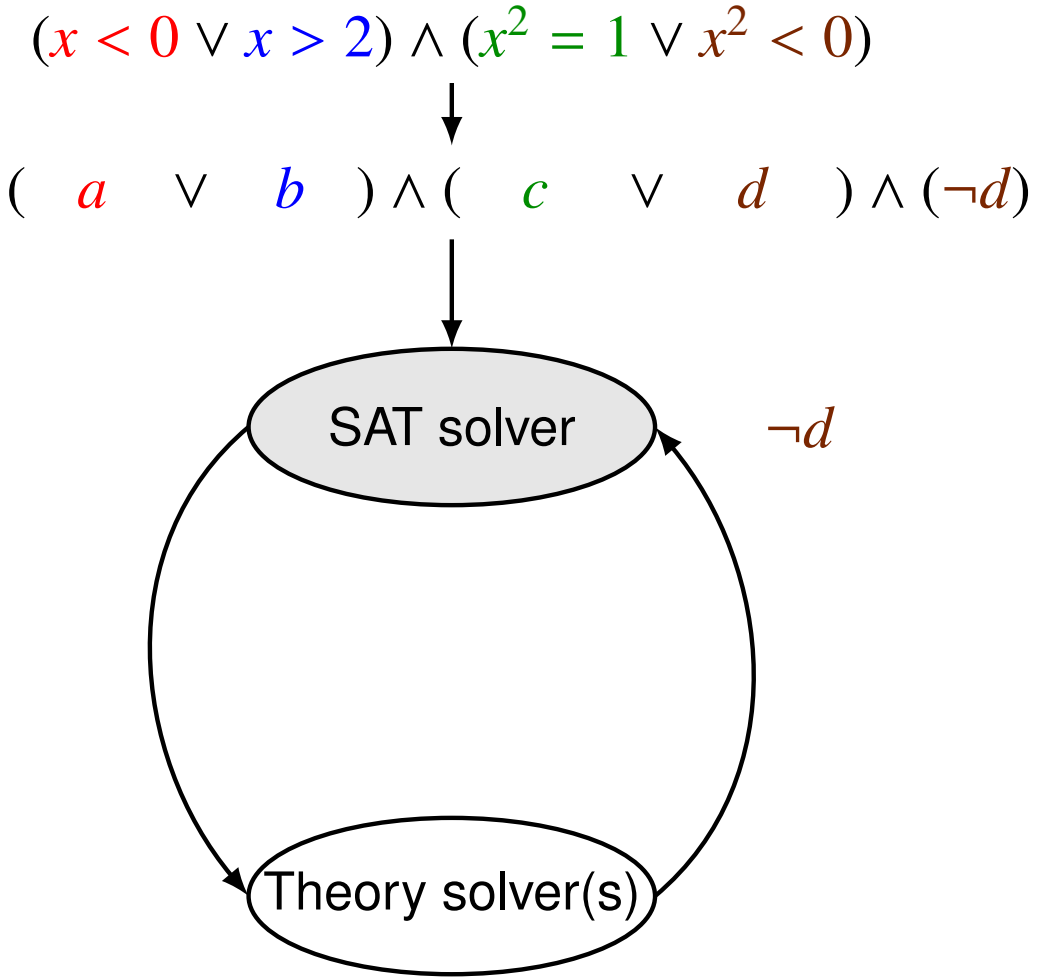
Less lazy SMT solving

$$(x < 0 \vee x > 2) \wedge (x^2 = 1 \vee x^2 < 0)$$

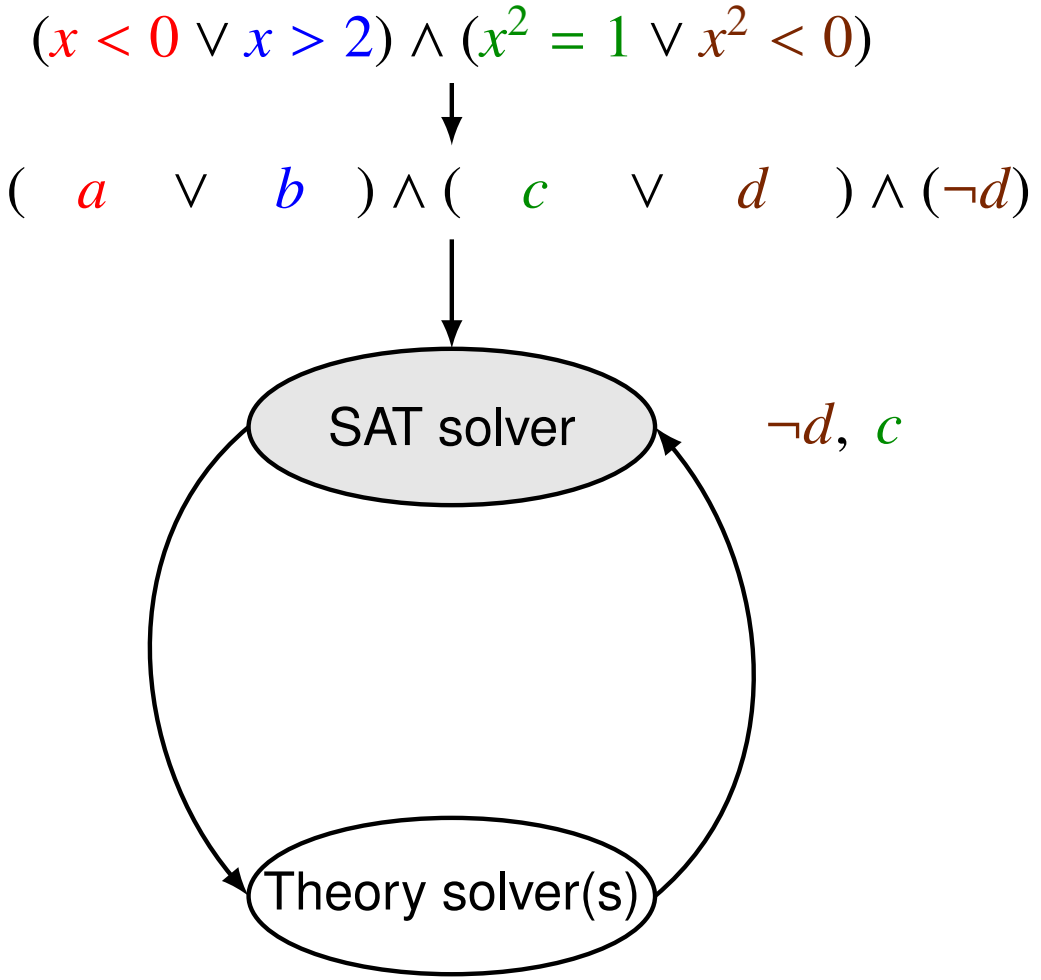
$$(\ a \ \vee \ b \ ) \wedge (\ c \ \vee \ d \ ) \wedge (\neg d)$$



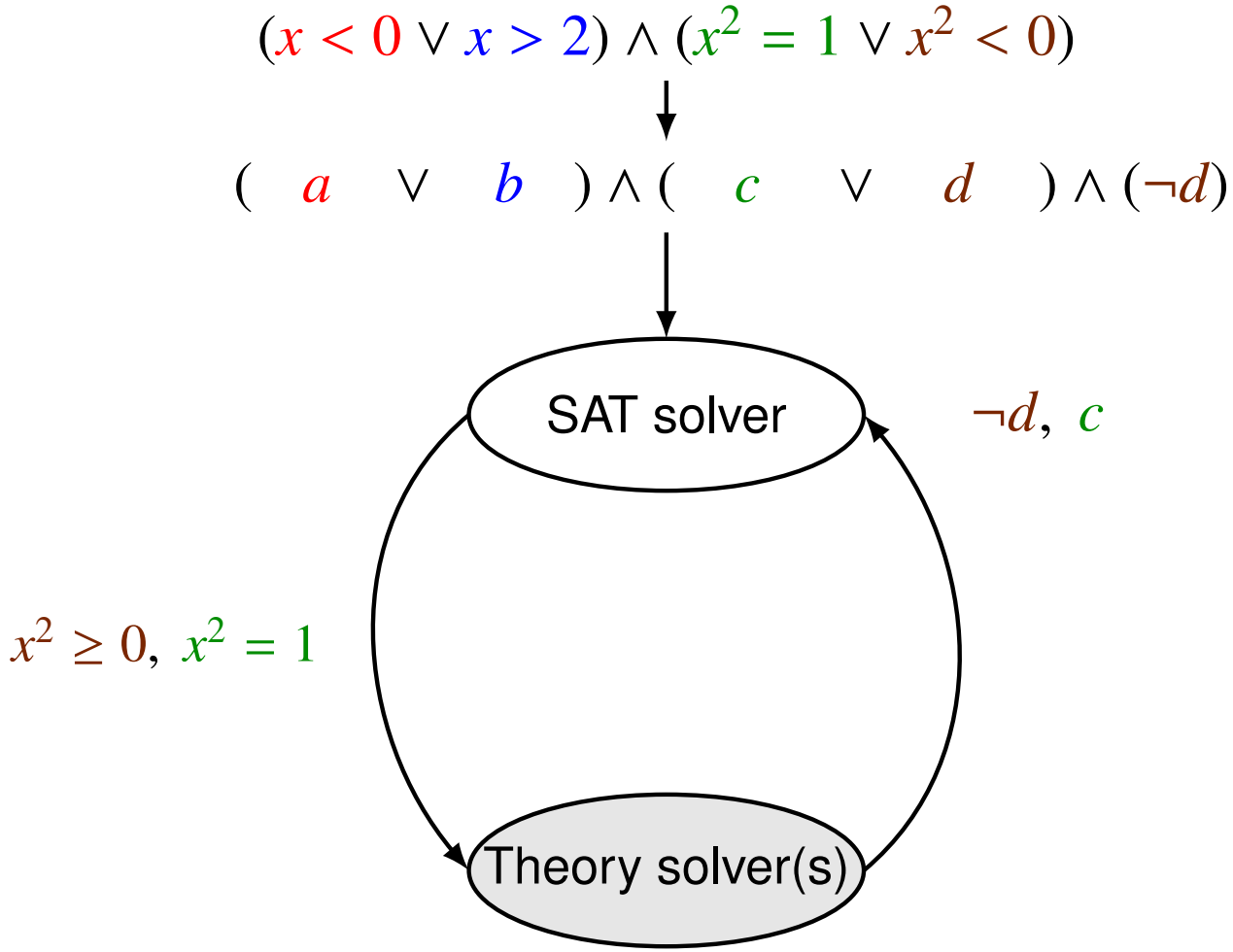
Less lazy SMT solving



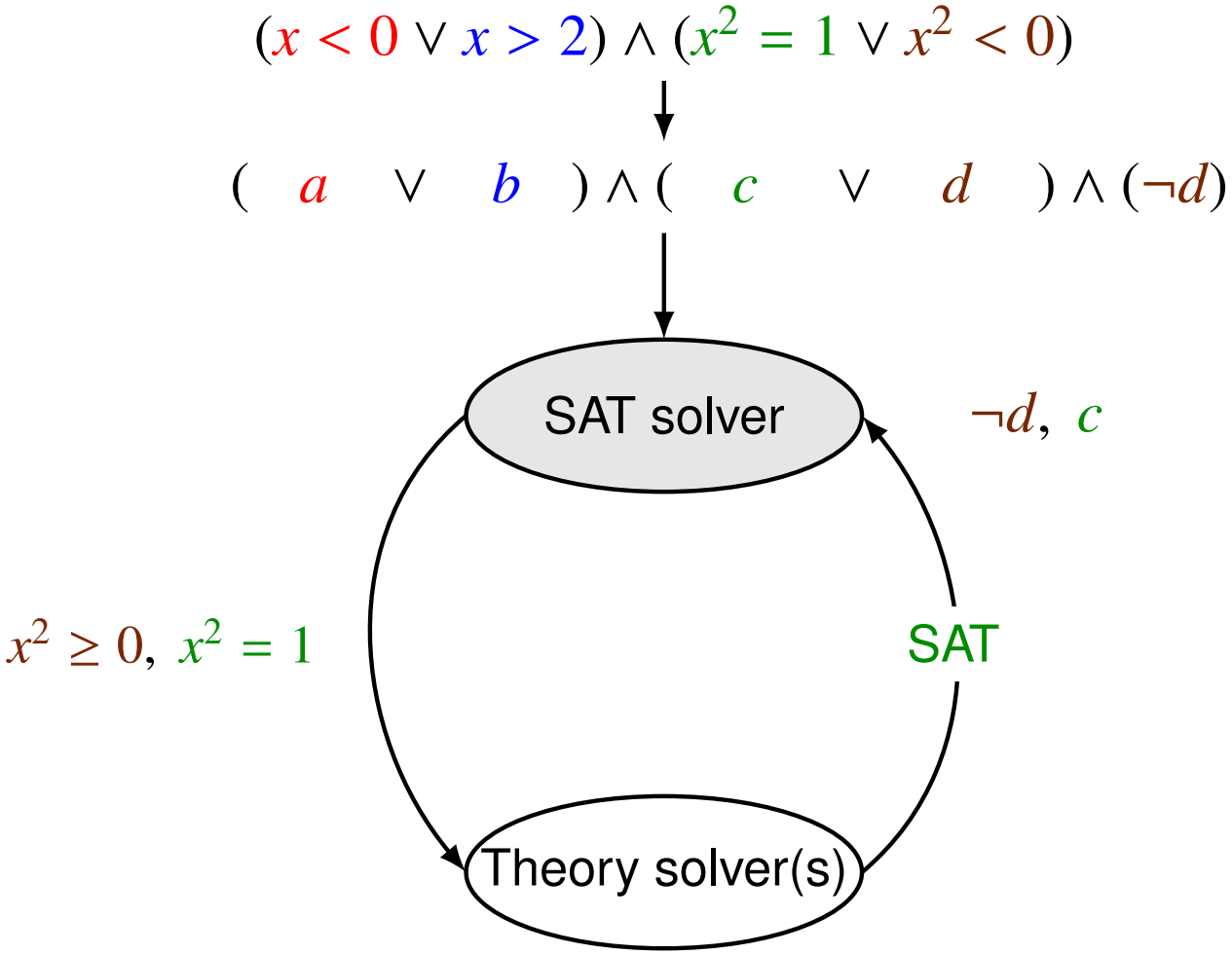
Less lazy SMT solving



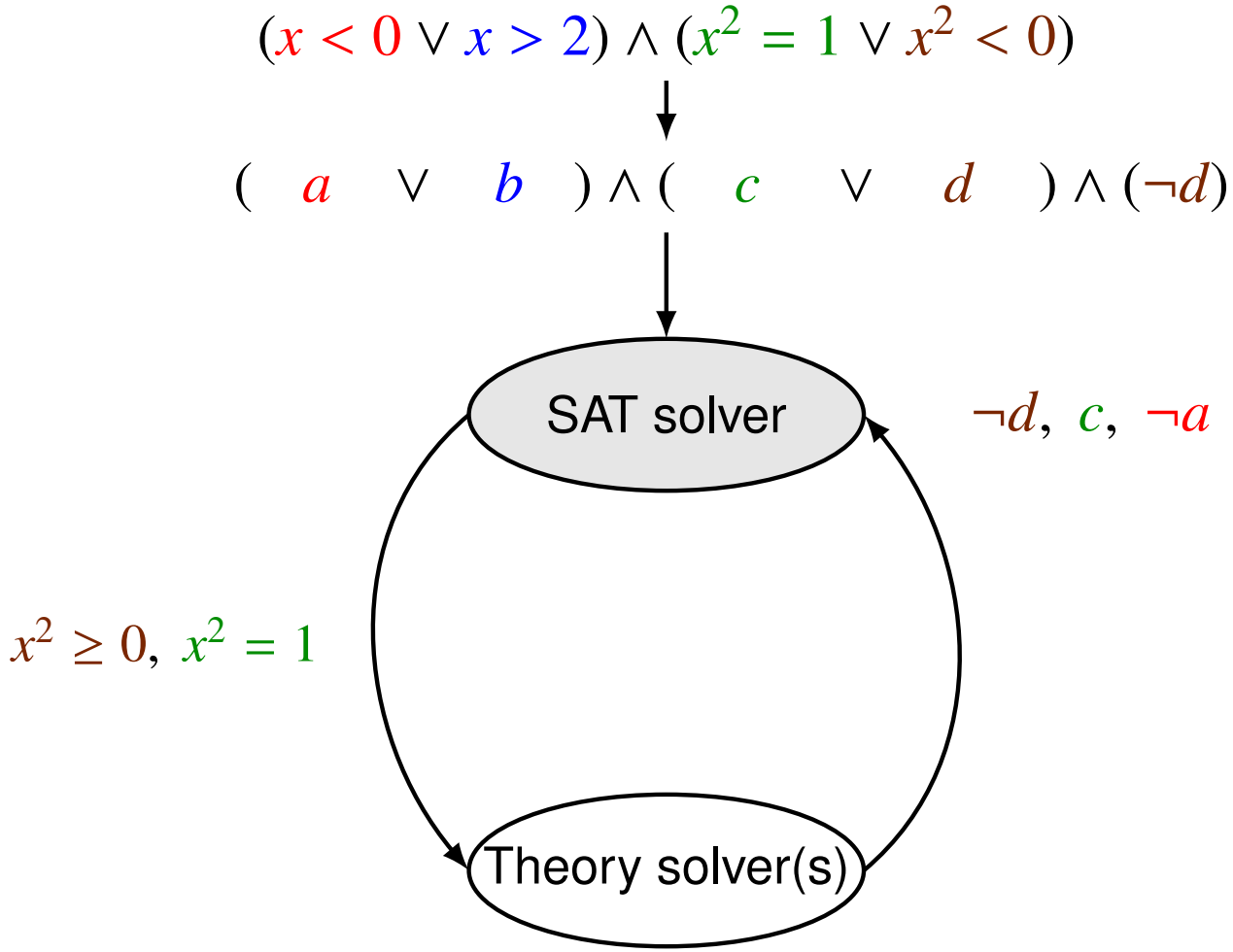
Less lazy SMT solving



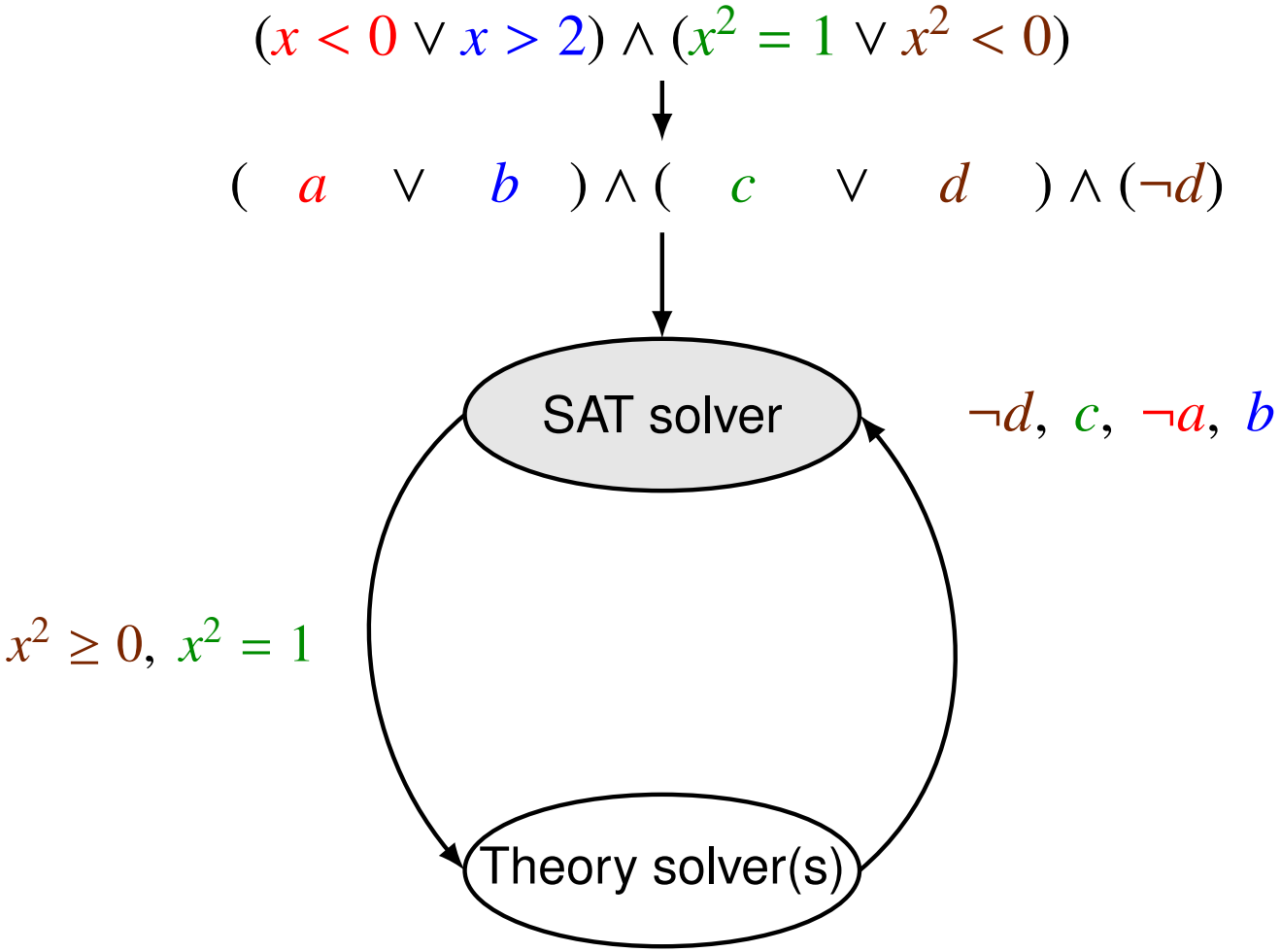
Less lazy SMT solving



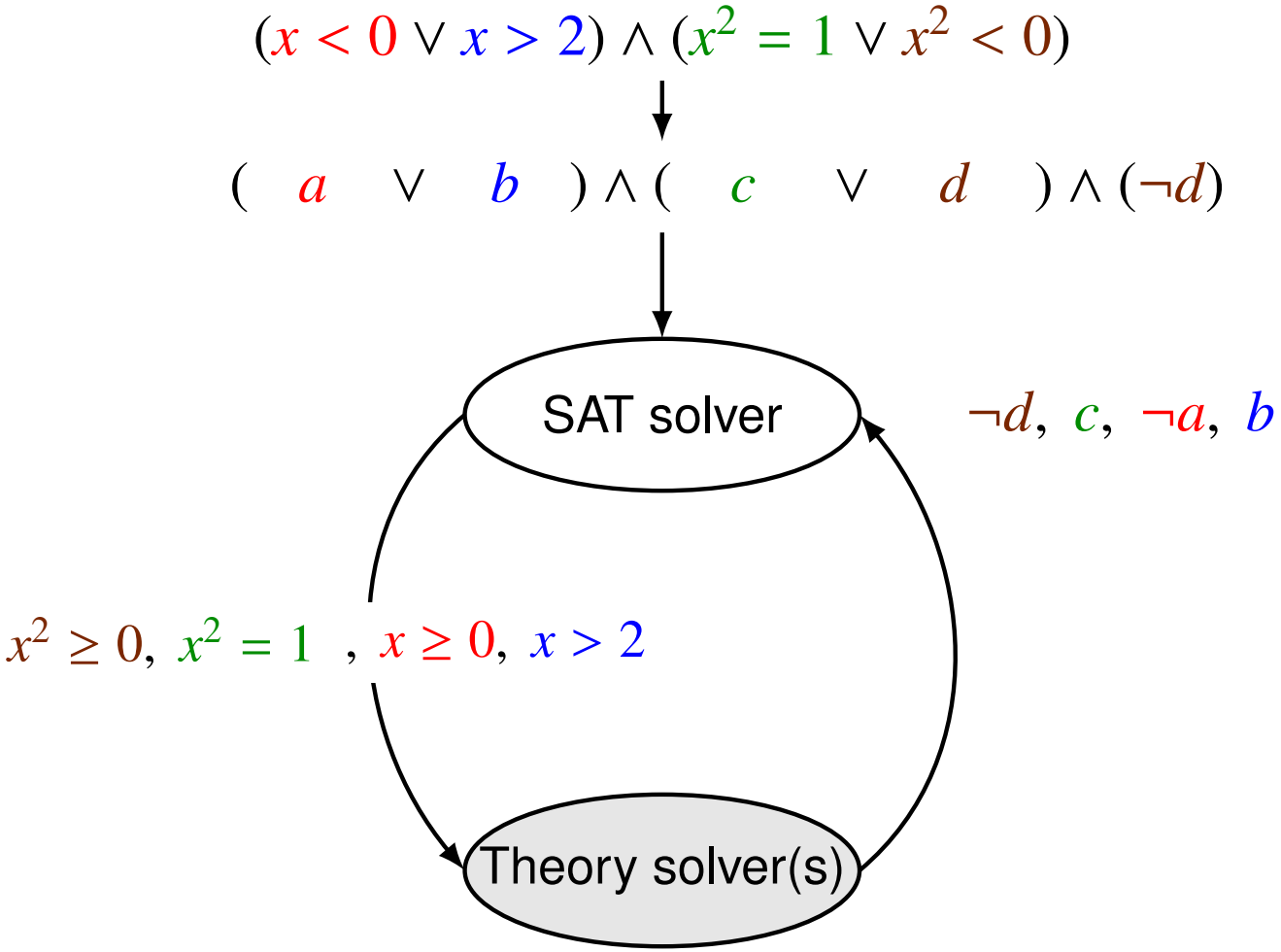
Less lazy SMT solving



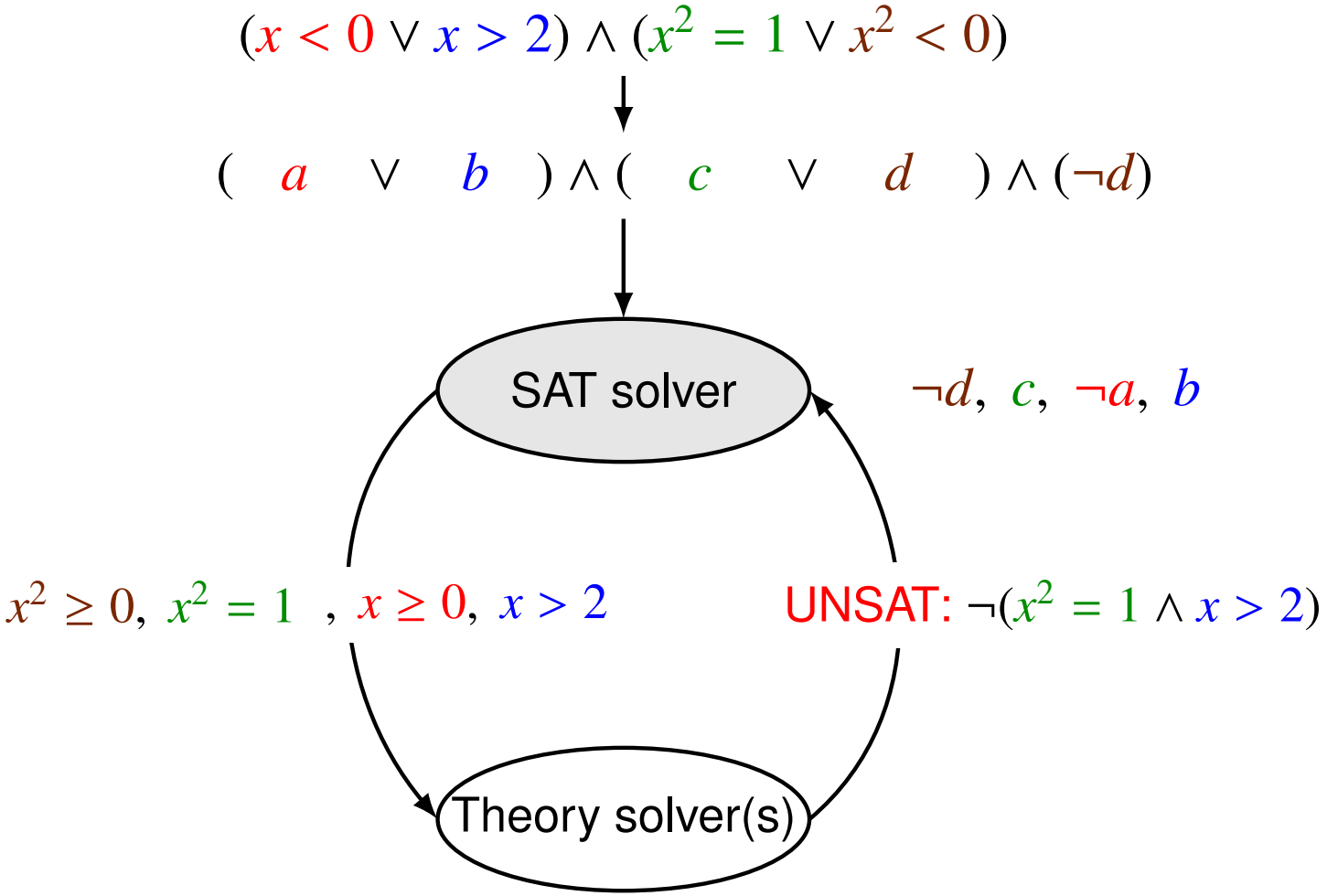
Less lazy SMT solving



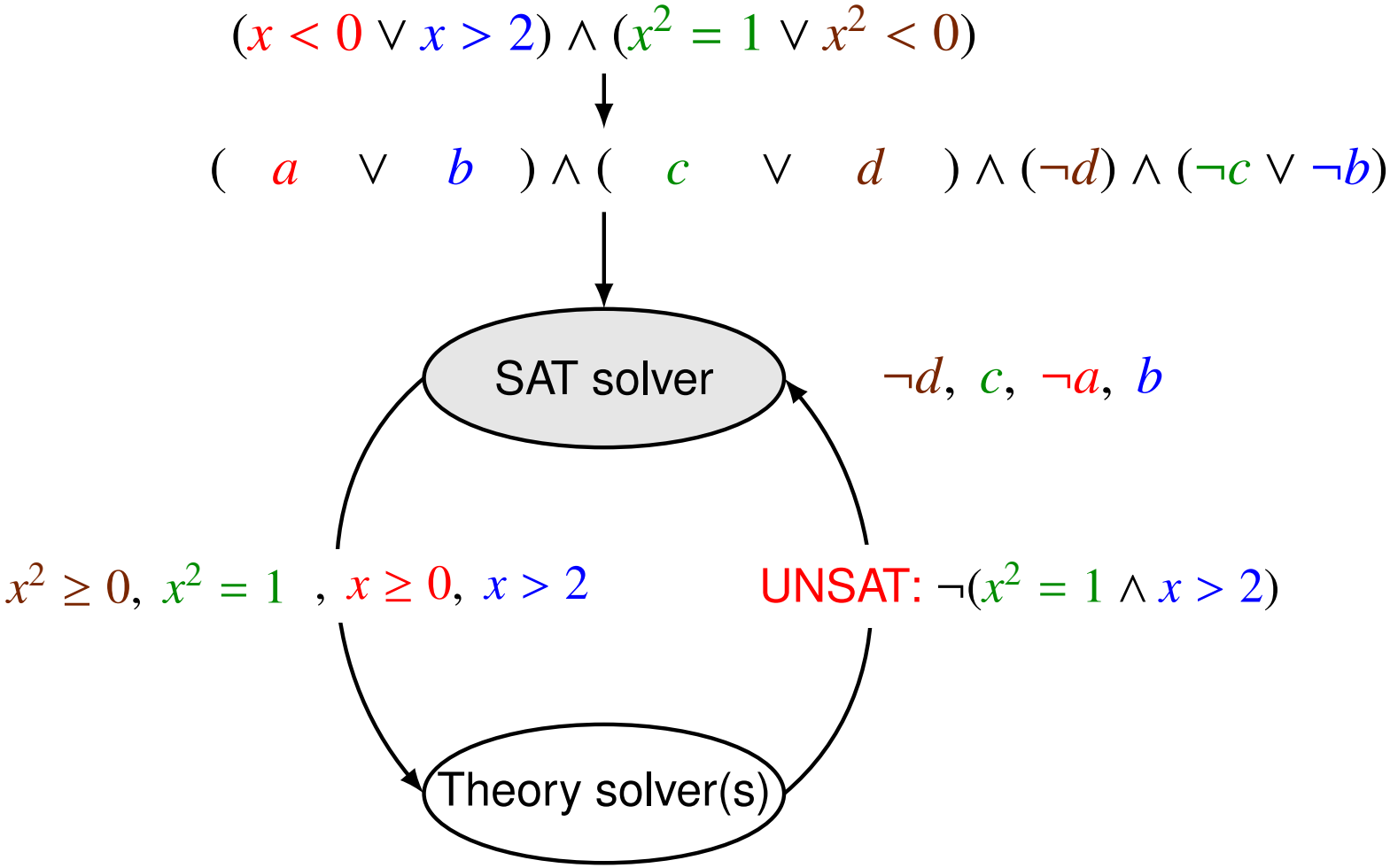
Less lazy SMT solving



Less lazy SMT solving



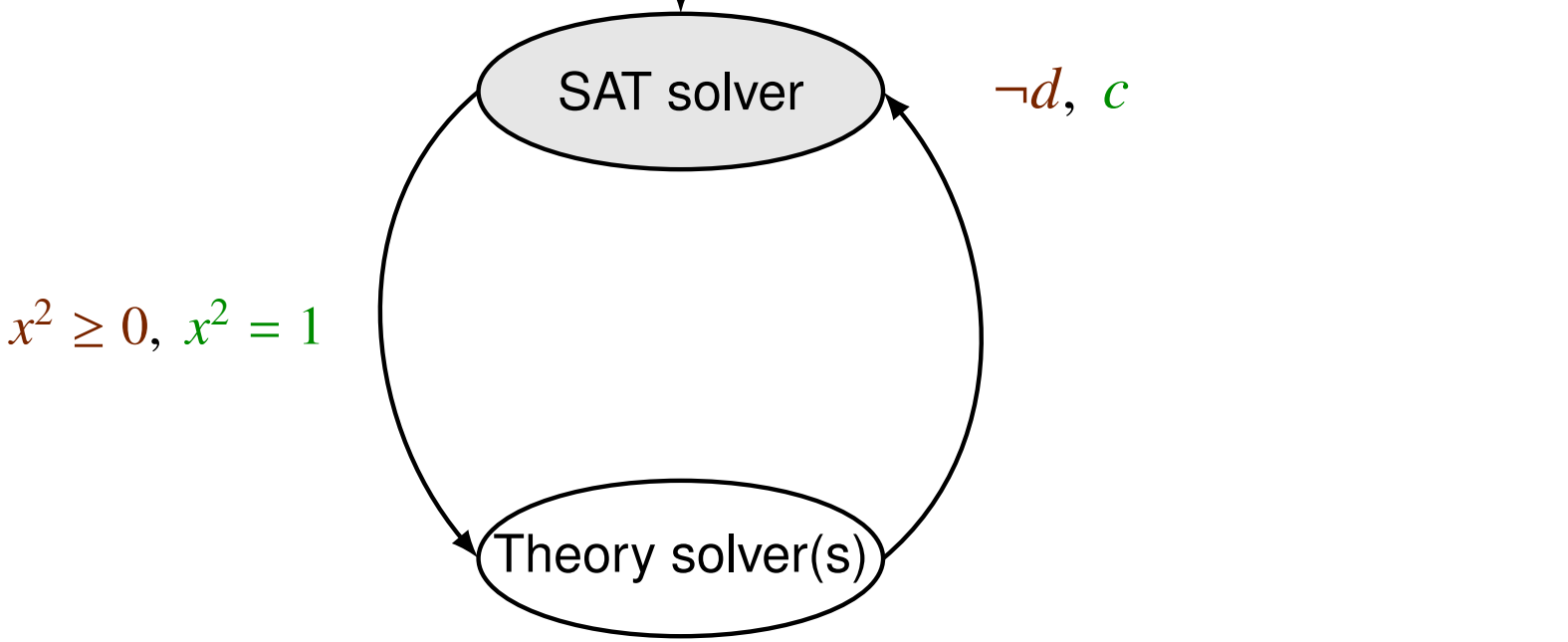
Less lazy SMT solving



Less lazy SMT solving

$(x < 0 \vee x > 2) \wedge (x^2 = 1 \vee x^2 < 0)$

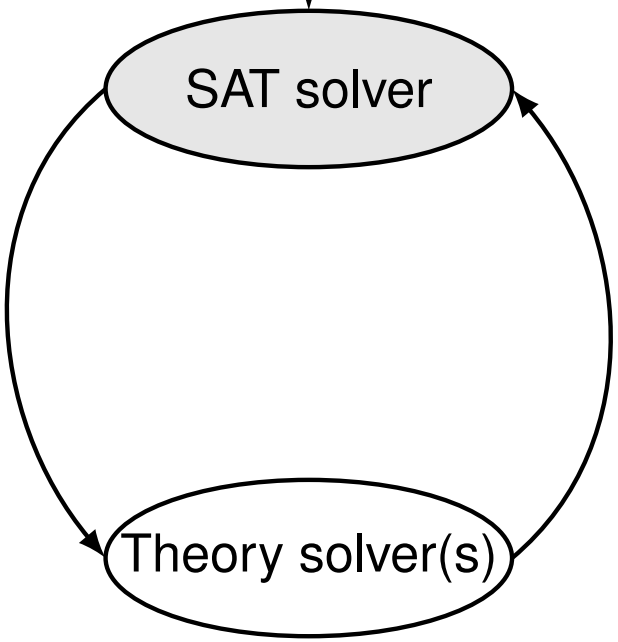
$(\ a\ \vee\ b\ ) \wedge (\ c\ \vee\ d\ ) \wedge (\neg d) \wedge (\neg c \vee \neg b)$



Less lazy SMT solving

$(x < 0 \vee x > 2) \wedge (x^2 = 1 \vee x^2 < 0)$

$(\ a\ \vee\ b\ ) \wedge (\ c\ \vee\ d\ ) \wedge (\neg d) \wedge (\neg c \vee \neg b)$



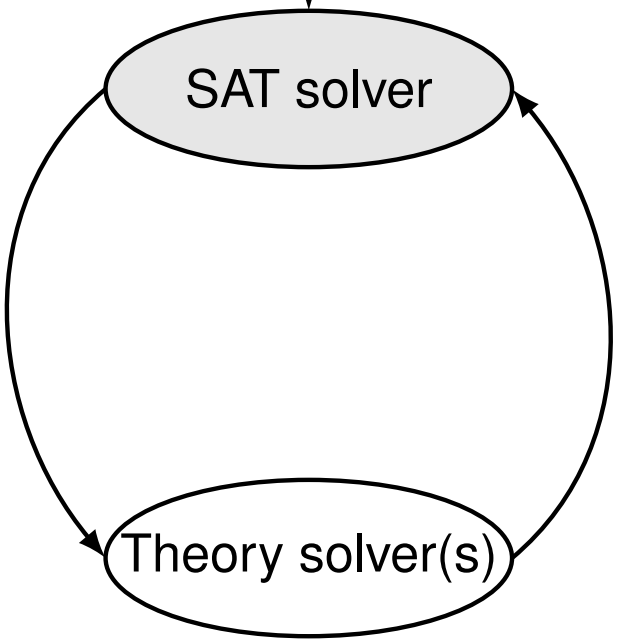
$\neg d,\ c,\ \neg b$

$x^2 \geq 0,\ x^2 = 1$

Less lazy SMT solving

$(x < 0 \vee x > 2) \wedge (x^2 = 1 \vee x^2 < 0)$

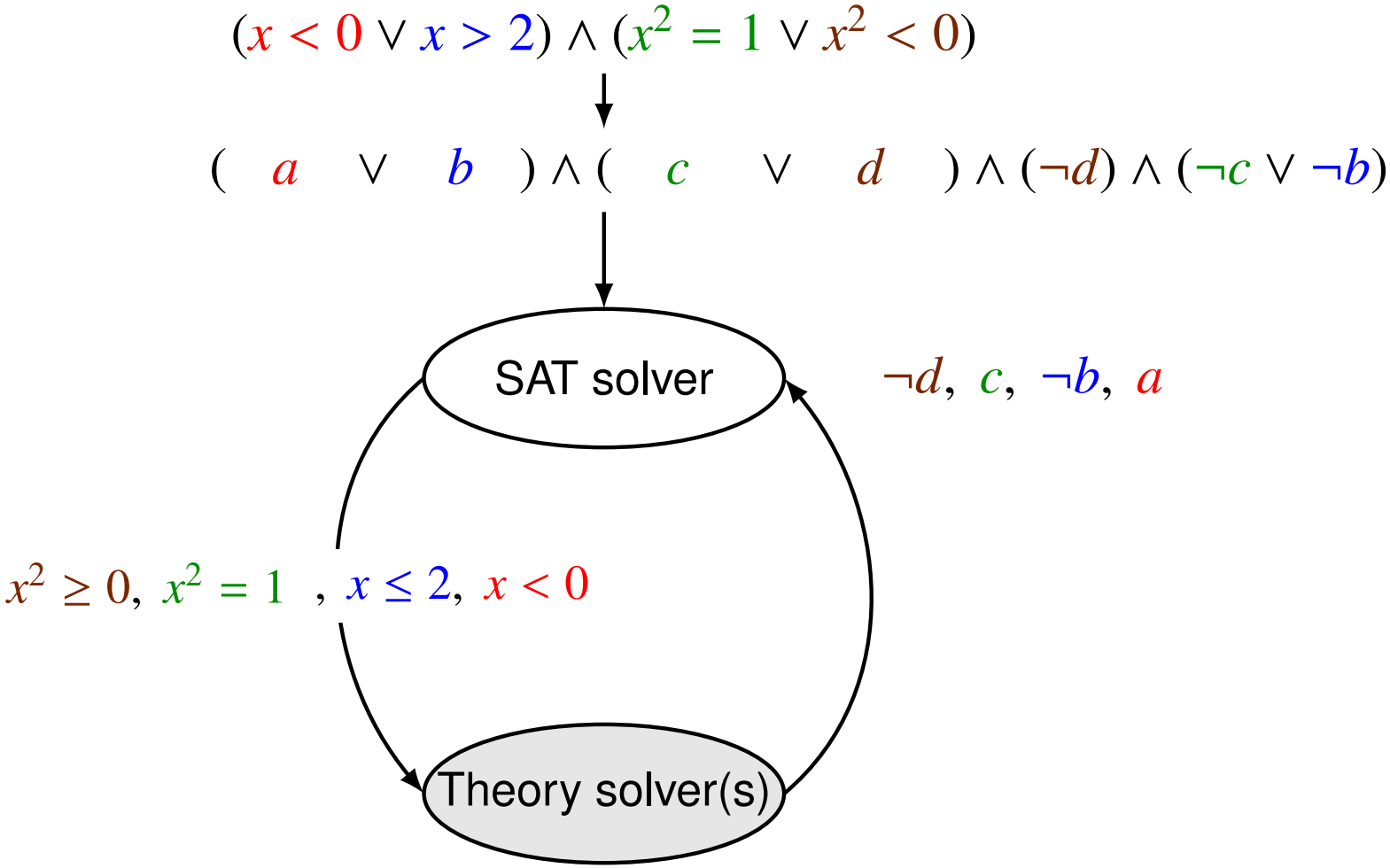
$(\ a\ \vee\ b\ ) \wedge (\ c\ \vee\ d\ ) \wedge (\neg d) \wedge (\neg c \vee \neg b)$



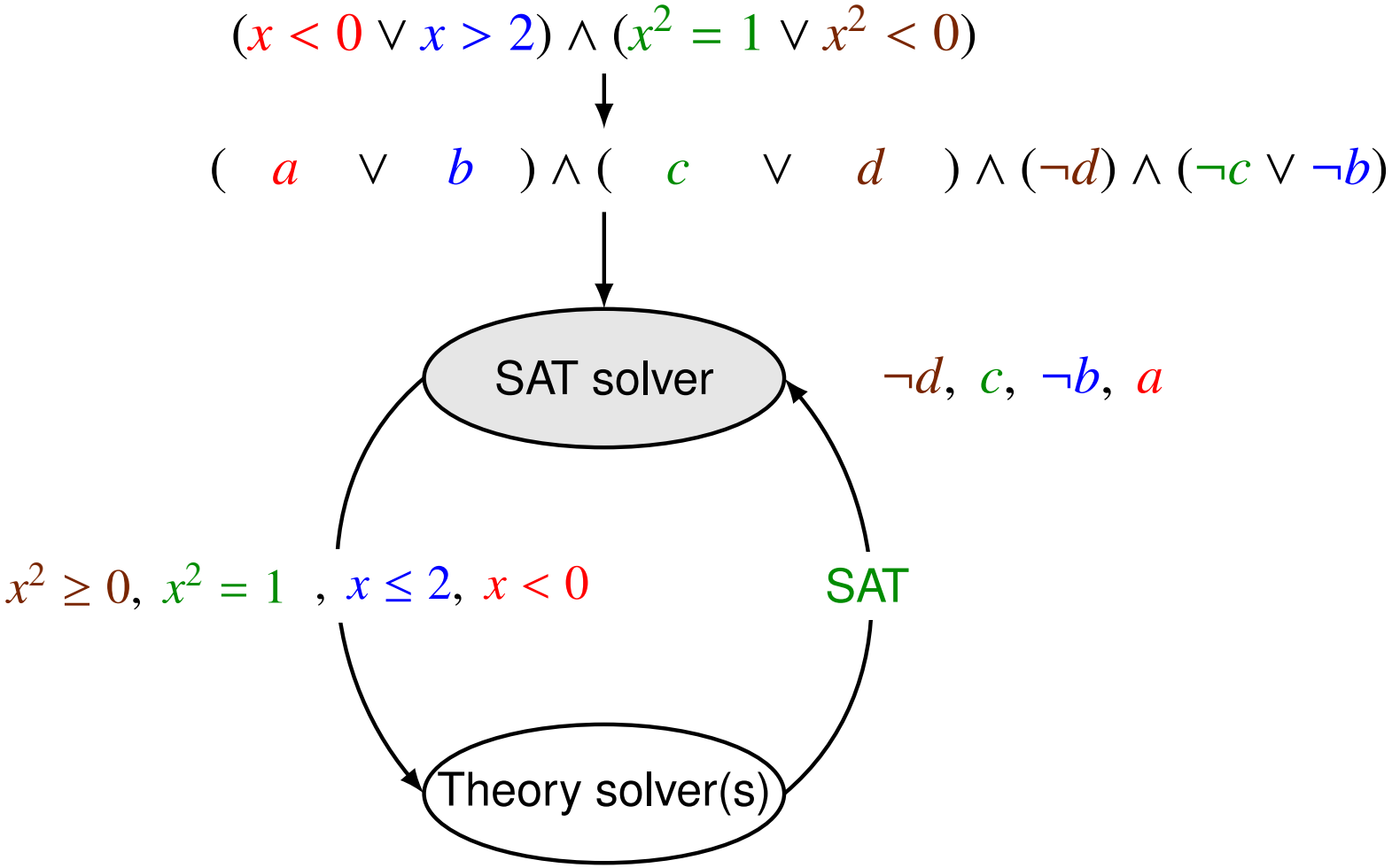
$\neg d,\ c,\ \neg b,\ a$

$x^2 \geq 0,\ x^2 = 1$

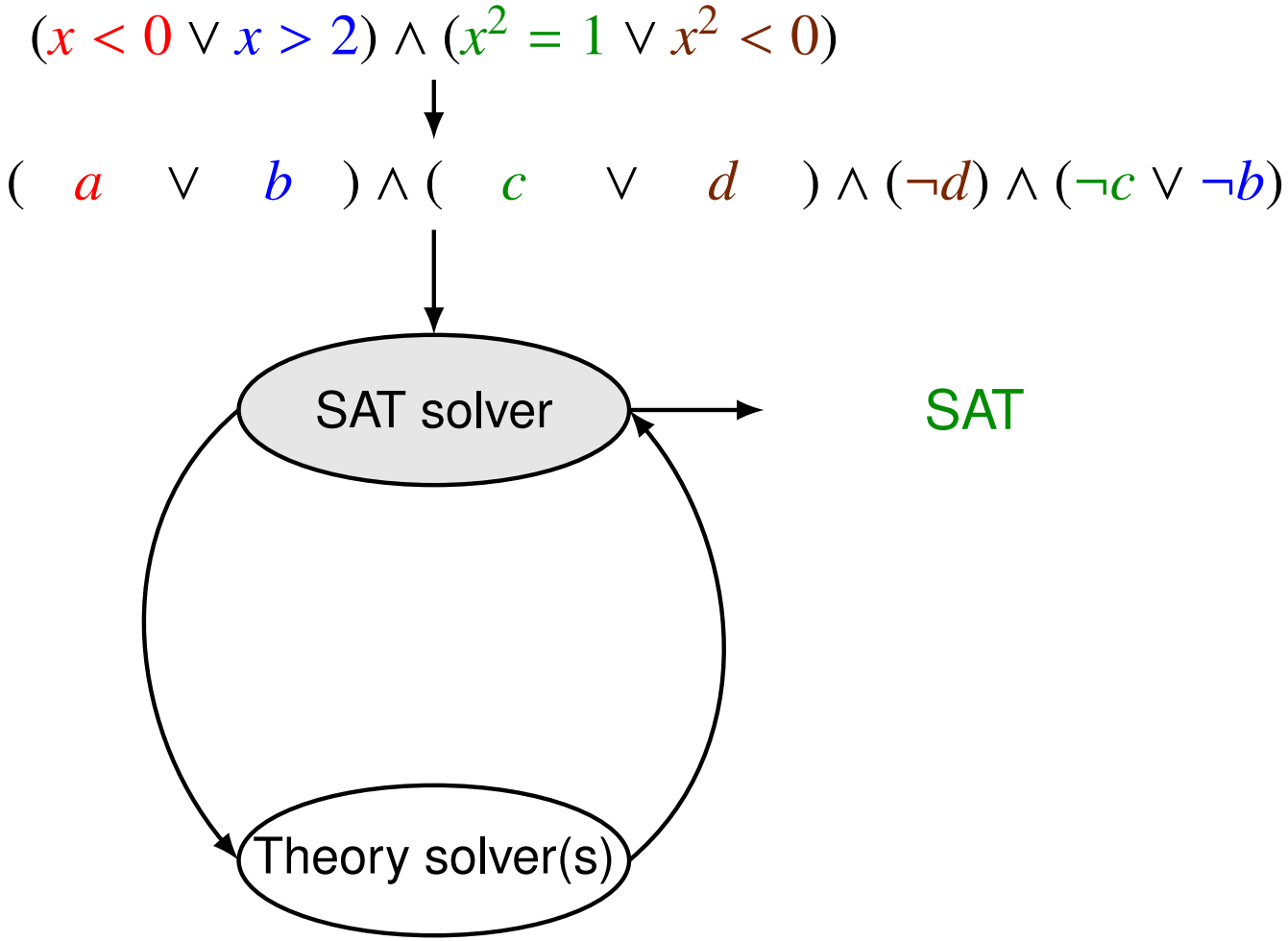
Less lazy SMT solving



Less lazy SMT solving



Less lazy SMT solving



Some theory solver candidates for arithmetic theories

Linear real arithmetic:

- Simplex
- Ellipsoid method
- Fourier-Motzkin variable elimination (mostly preprocessing)
- Interval constraint propagation (incomplete)

Linear integer arithmetic:

- Cutting planes, Gomory cuts
- Branch-and-bound (incomplete)
- Bit-blasting (eager)
- Interval constraint propagation (incomplete)

SMT solvers: `Alt-Ergo, CVC4, iSAT3, MathSAT5, OpenSMT2, SMT-RAT, veriT, Yices2, Z3`

Non-linear real arithmetic:

- Cylindrical algebraic decomposition
- Gröbner bases (mostly preprocessing/simplification)
- Virtual substitution (focus on low degrees)
- Interval constraint propagation (incomplete)

Non-linear integer arithmetic:

- Generalised branch-and-bound (incomplete)
- Bit-blasting (eager, incomplete)

SMT solvers: `Alt-Ergo, AProVE, iSAT3, MiniSmt, SMT-RAT, Z3`

Some corresponding implementations in CAS

Gröbner bases

- CoCoA, F4, Maple, Mathematica, Maxima, Singular, Reduce, ...

Cylindrical algebraic decomposition (CAD)

- Mathematica, QEPCAD, Reduce, ...

Virtual substitution (VS)

- Reduce, ...

Strength: Efficient for conjunctions of real constraints.

Some corresponding implementations in CAS

Gröbner bases

- CoCoA, F4, Maple, Mathematica, Maxima, Singular, Reduce, ...

Cylindrical algebraic decomposition (CAD)

- Mathematica, QEPCAD, Reduce, ...

Virtual substitution (VS)

- Reduce, ...

Strength: Efficient for conjunctions of real constraints.

So why don't we just plug in an algebraic decision procedure as theory solver into an SMT solver?

Why not use CAS out of the box?

- Theory solvers should be SMT-compliant, i.e., they should
  work incrementally,
  generate lemmas explaining inconsistencies, and
  be able to backtrack.

Why not use CAS out of the box?

- Theory solvers should be SMT-compliant, i.e., they should
  work incrementally,
  generate lemmas explaining inconsistencies, and
  be able to backtrack.

- Originally, the mentioned methods
  are not SMT-compliant,
  they are seldomly available as libraries, and
  are usually not thread-safe.

Why not use CAS out of the box?

- Theory solvers should be SMT-compliant, i.e., they should
  work incrementally,
  generate lemmas explaining inconsistencies, and
  be able to backtrack.

- Originally, the mentioned methods
  are not SMT-compliant,
  they are seldomly available as libraries, and
  are usually not thread-safe.

- Usually, SMT-adaptations are tricky.

  For illustration, let us have a high-level look at some solutions.

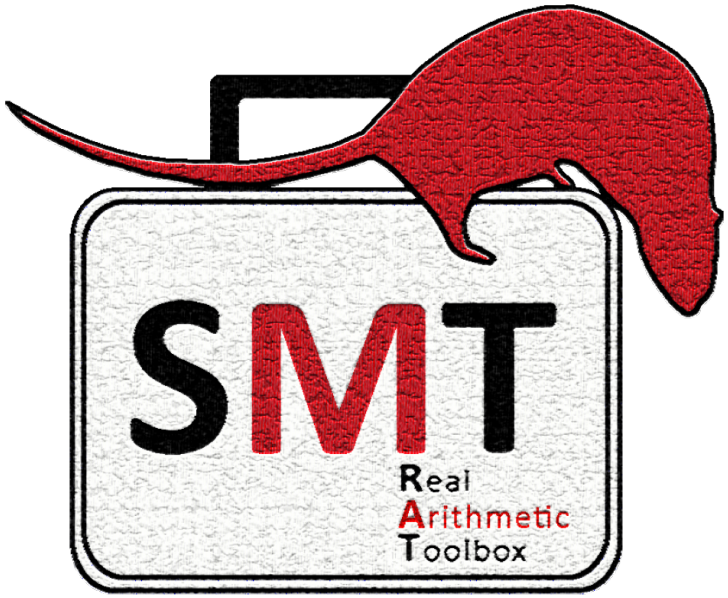
Our SMT-RAT library

We have developed the SMT-RAT library of theory modules.

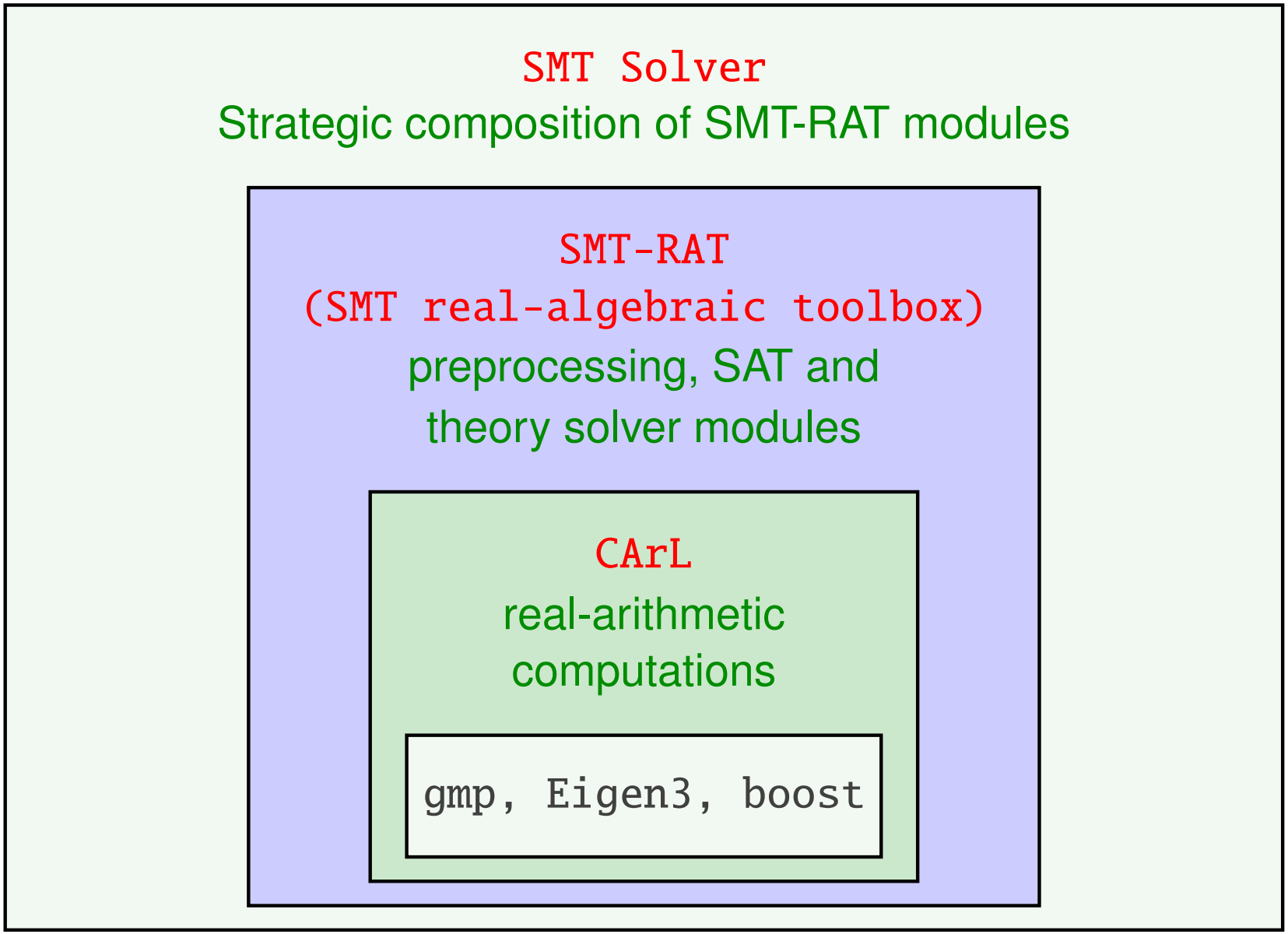
[SAT'12, SAT'15]

A new release came out in June 2015.

`https://github.com/smtrat/smtrat/wiki`



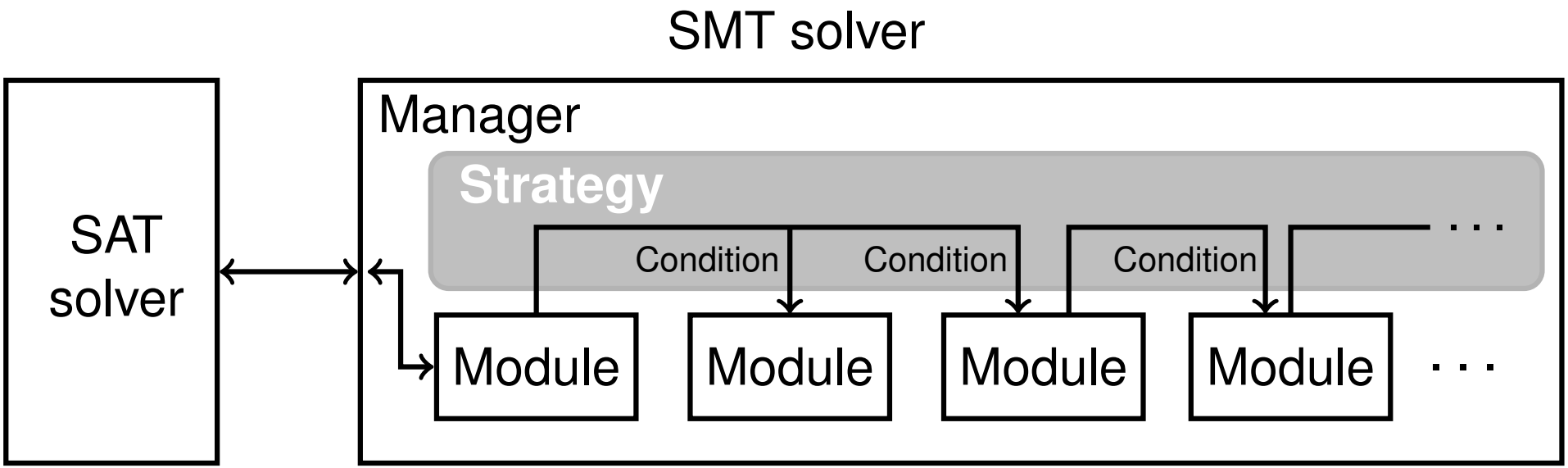
Our SMT-RAT library



Solver modules in SMT-RAT

- Libraries for basic computations [NFM'11, CAI'11]
- SAT solver
- CNF converter
- Preprocessing/simplifying modules
- Interval constraint propagation
- Simplex
- Virtual substitution [FCT'11, PhD Corzilius]
- CAD [CADE-24, PhD Loup, PhD Kremer]
- Gröbner bases [CAI'13]
- Generalised branch-and-bound
- Under construction:
  equality, uninterpreted functions, bit-vector arithmetic

Strategic composition of solver modules in SMT-RAT



The key idea of the virtual substitution

Eliminate one existential quantifier in favour of a finite disjunction over parametric test points.

$$\exists x, y: \ (y = 0 \ \vee \ y^2 + 1 < 0) \ \wedge \ x - 3 \leq 0 \ \wedge \ xy + 1 < 0$$

Test candidates

eliminate x

$\Leftrightarrow$

$$\begin{aligned}
\exists y: \quad & ((y = 0 \ \vee \ y^2 + 1 < 0) \ \wedge \ x - 3 \leq 0 \ \wedge \ xy + 1 < 0)[-\infty/\mathbf{x}] \\
\vee \quad & ((y = 0 \ \vee \ y^2 + 1 < 0) \ \wedge \ x - 3 \leq 0 \ \wedge \ xy + 1 < 0)[\mathbf{3}/\mathbf{x}] \quad) \\
\vee (\ \mathbf{y \neq 0} \quad \wedge \quad & ((y = 0 \ \vee \ y^2 + 1 < 0) \ \wedge \ x - 3 \leq 0 \ \wedge \ xy + 1 < 0)[-\tfrac{\mathbf{1}}{\mathbf{y}} + \epsilon/\mathbf{x}] \quad)
\end{aligned}$$

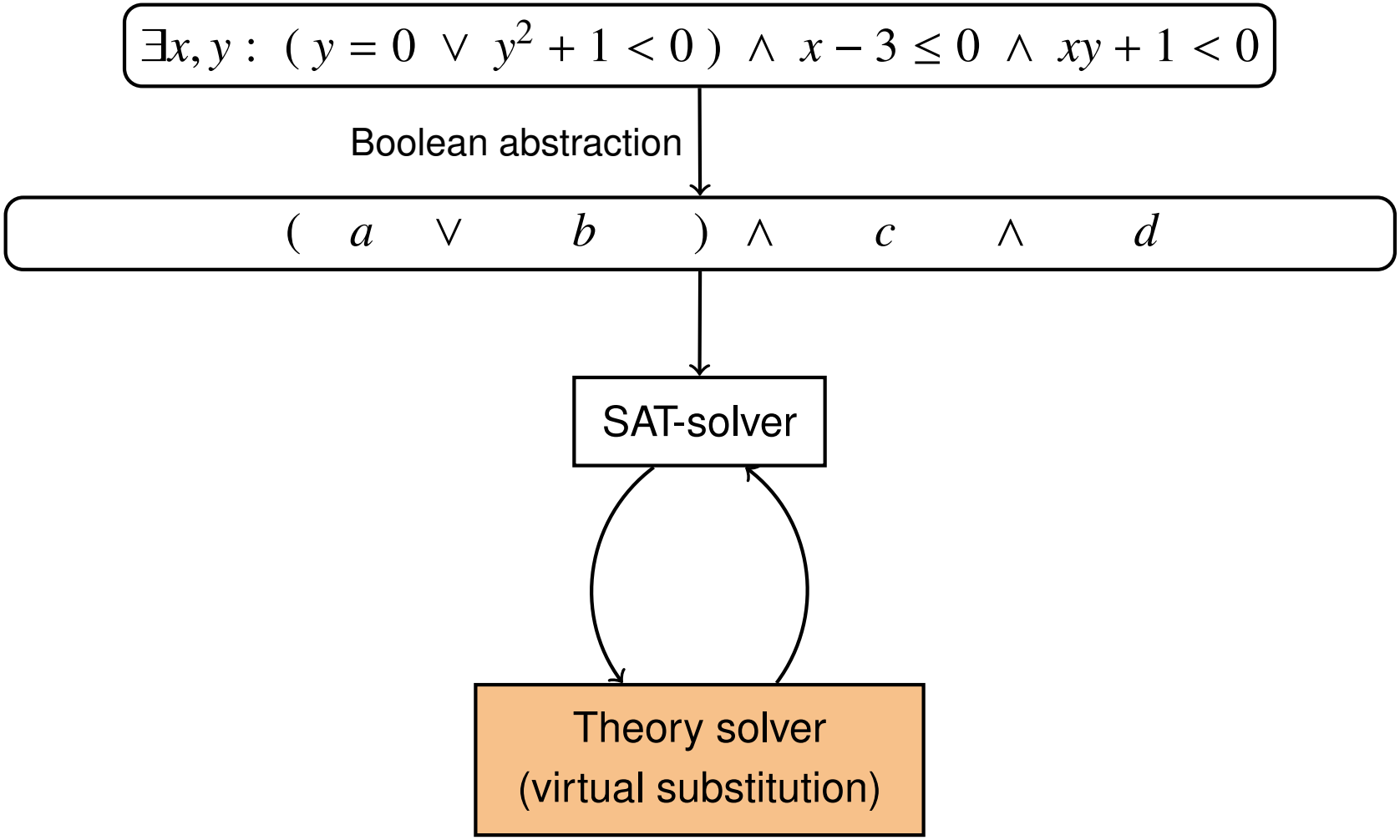
Side condition

eliminate y

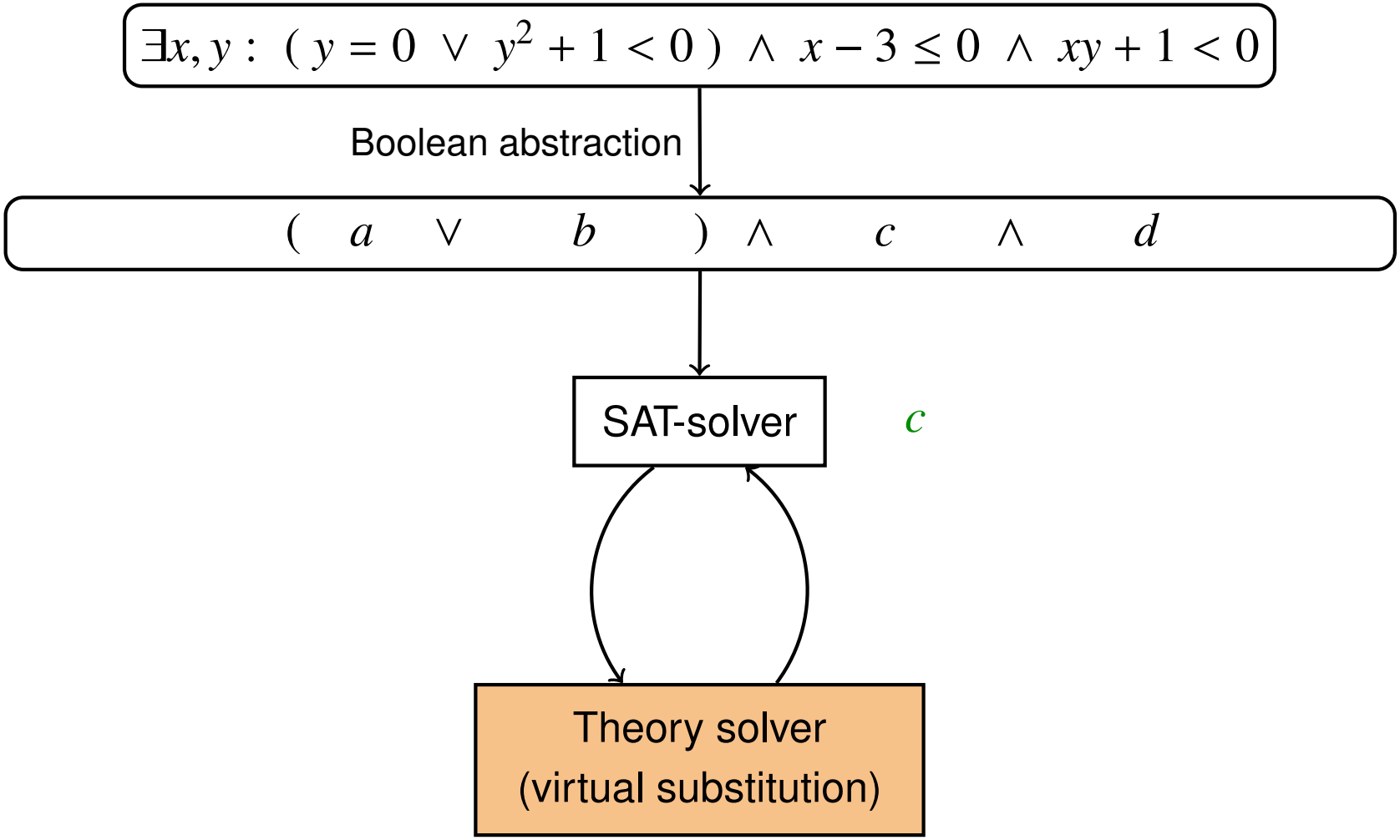
$\Leftrightarrow$

$\ldots$

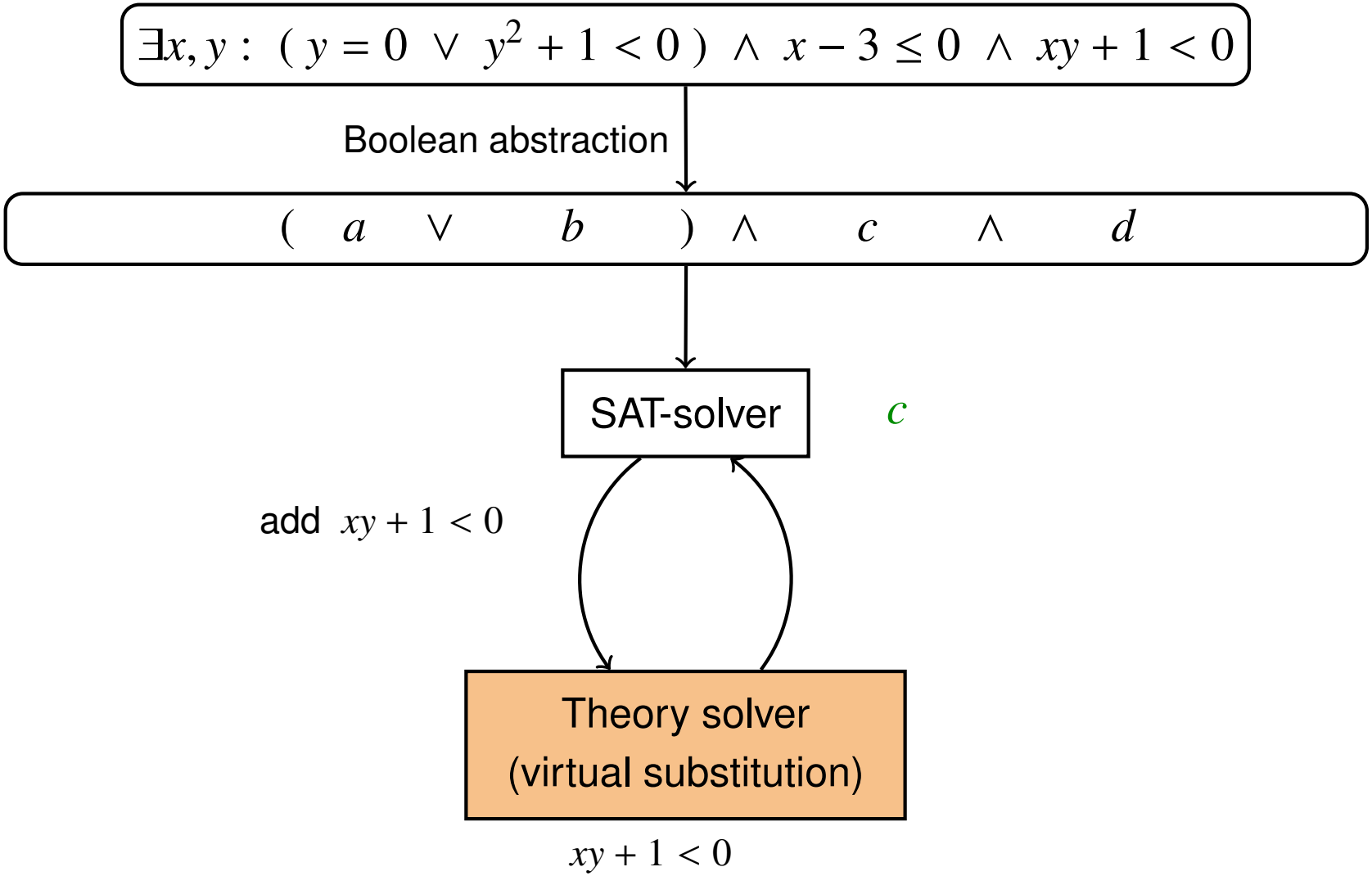
SMT solver


$\exists x, y : \ (\ y = 0 \ \vee \ y^2 + 1 < 0 \ ) \ \wedge \ x - 3 \leq 0 \ \wedge \ xy + 1 < 0$
Boolean abstraction
$(\quad a \quad \vee \quad b \quad) \ \wedge \quad c \quad \wedge \quad d$
SAT-solver
Theory solver
(virtual substitution)

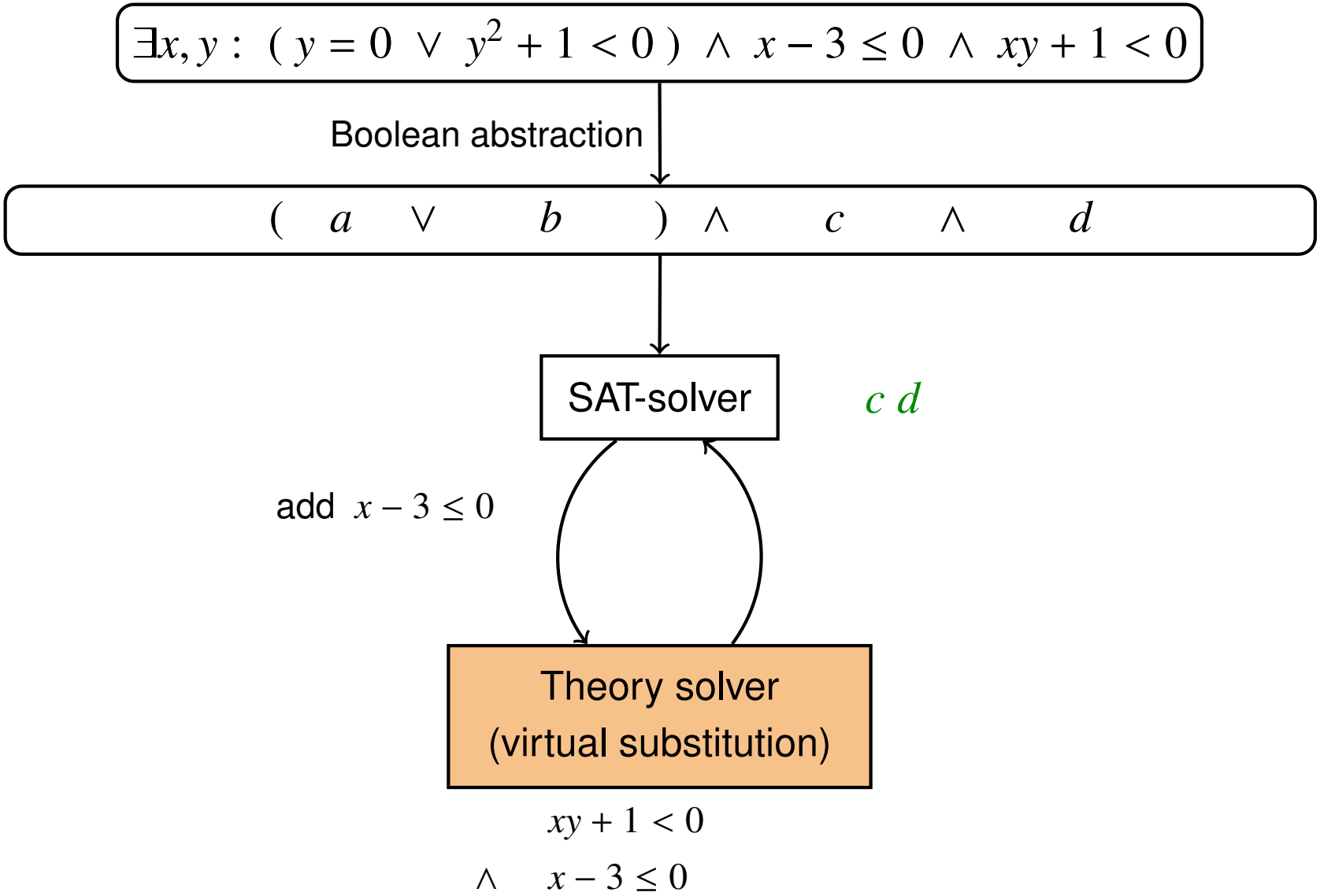
SMT solver


$\exists x, y : \ (\ y = 0 \ \vee \ y^2 + 1 < 0 \ ) \ \wedge \ x - 3 \leq 0 \ \wedge \ xy + 1 < 0$
Boolean abstraction
$(\quad a \quad \vee \quad b \quad) \ \wedge \quad c \quad \wedge \quad d$
SAT-solver
c
Theory solver
(virtual substitution)

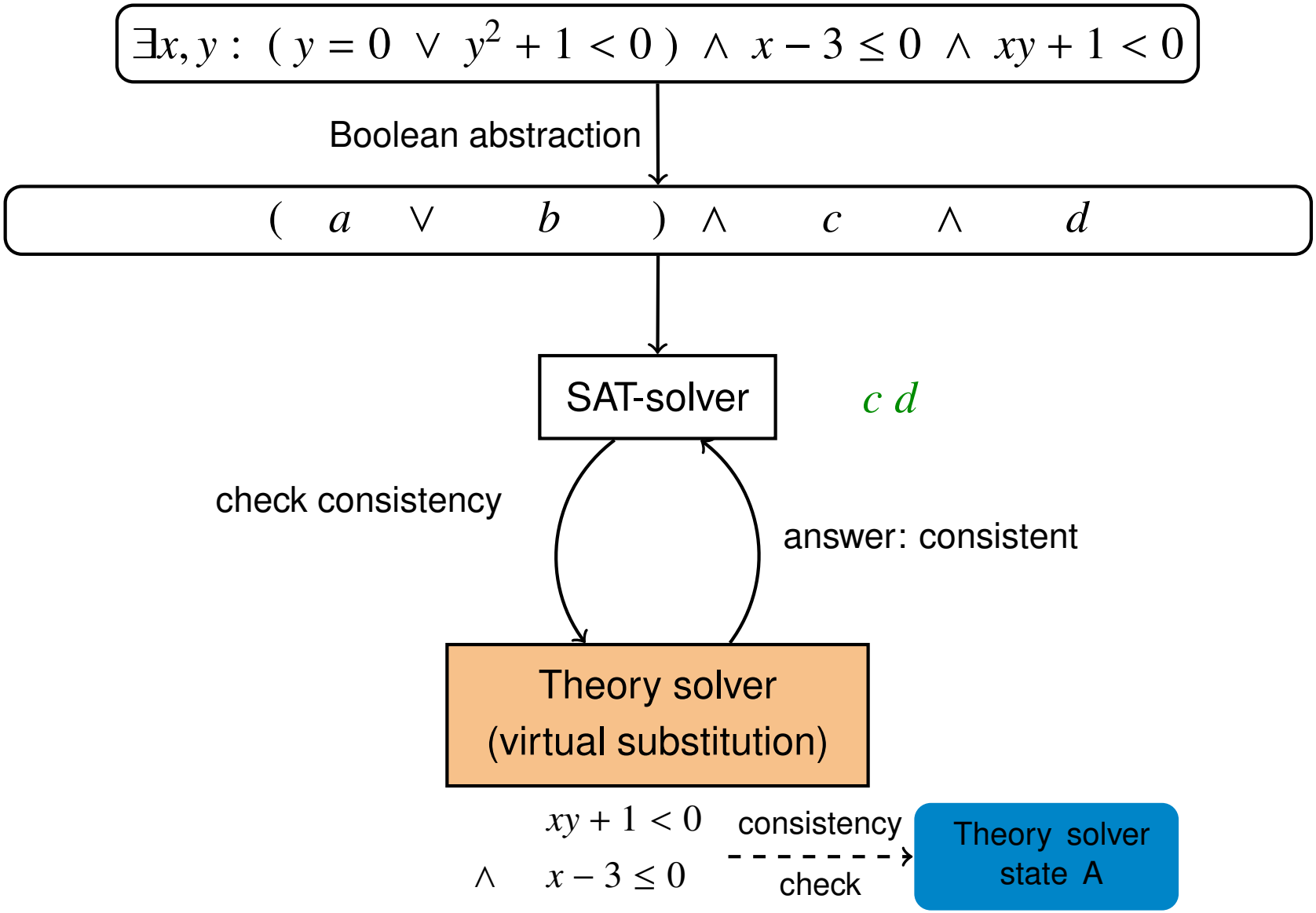
SMT solver


$\exists x, y : \ (\ y = 0 \ \vee \ y^2 + 1 < 0 \ ) \ \wedge \ x - 3 \leq 0 \ \wedge \ xy + 1 < 0$
Boolean abstraction
$(\quad a \quad \vee \quad b \quad) \ \wedge \quad c \quad \wedge \quad d$
SAT-solver
c
add $xy + 1 < 0$
Theory solver
(virtual substitution)
$xy + 1 < 0$

SMT solver


$\exists x, y : \ (\ y = 0 \ \vee \ y^2 + 1 < 0 \ ) \ \wedge \ x - 3 \leq 0 \ \wedge \ xy + 1 < 0$
Boolean abstraction
$(\quad a \quad \vee \quad b \quad) \ \wedge \quad c \quad \wedge \quad d$
SAT-solver
$c \ d$
add $x - 3 \leq 0$
Theory solver
(virtual substitution)
$xy + 1 < 0$
$\wedge \quad x - 3 \leq 0$

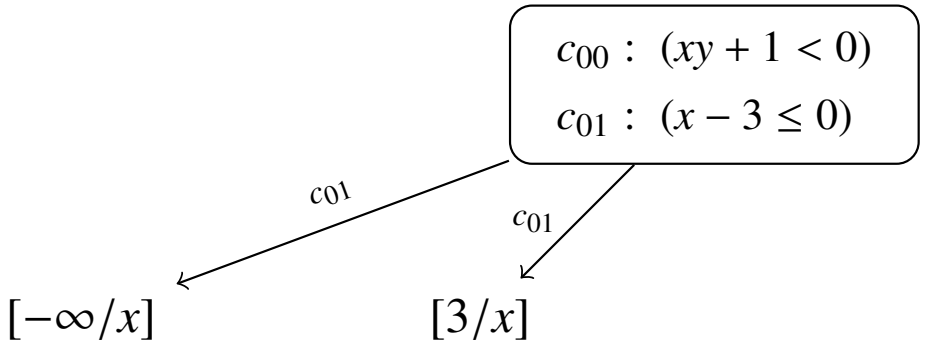
SMT solver


$\exists x, y : \ (\ y = 0 \ \vee \ y^2 + 1 < 0 \ ) \ \wedge \ x - 3 \leq 0 \ \wedge \ xy + 1 < 0$
Boolean abstraction
$(\quad a \quad \vee \quad b \quad) \ \wedge \quad c \quad \wedge \quad d$
SAT-solver
$c \ d$
check consistency
answer: consistent
Theory solver
(virtual substitution)
$xy + 1 < 0$
$\wedge \quad x - 3 \leq 0$
consistency
check
Theory solver
state A

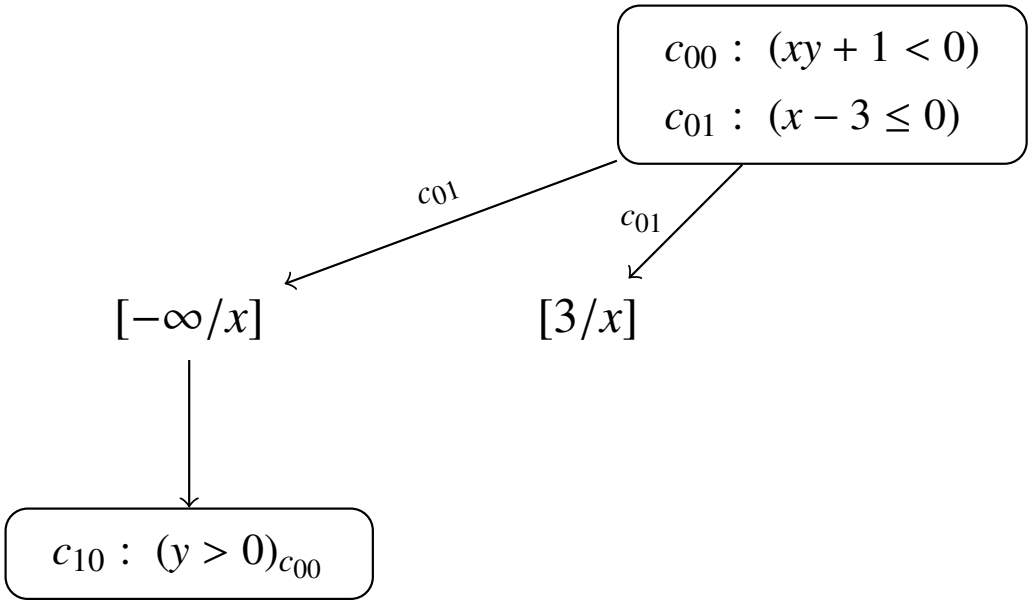
Theory solver: add $xy + 1 < 0$, add $x - 3 \leq 0$, check consistency

$$c_{00} : (xy + 1 < 0)$$
$$c_{01} : (x - 3 \leq 0)$$

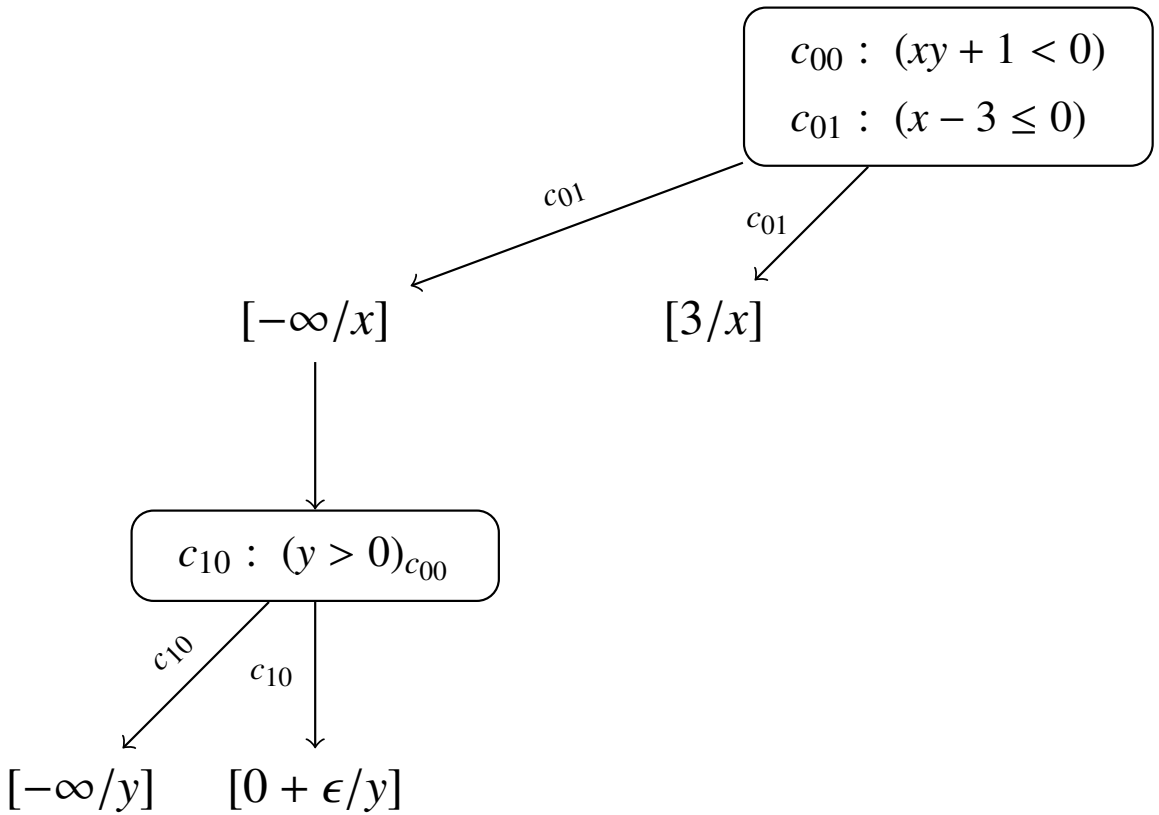
Theory solver: add $xy + 1 < 0$, add $x - 3 \leq 0$, check consistency



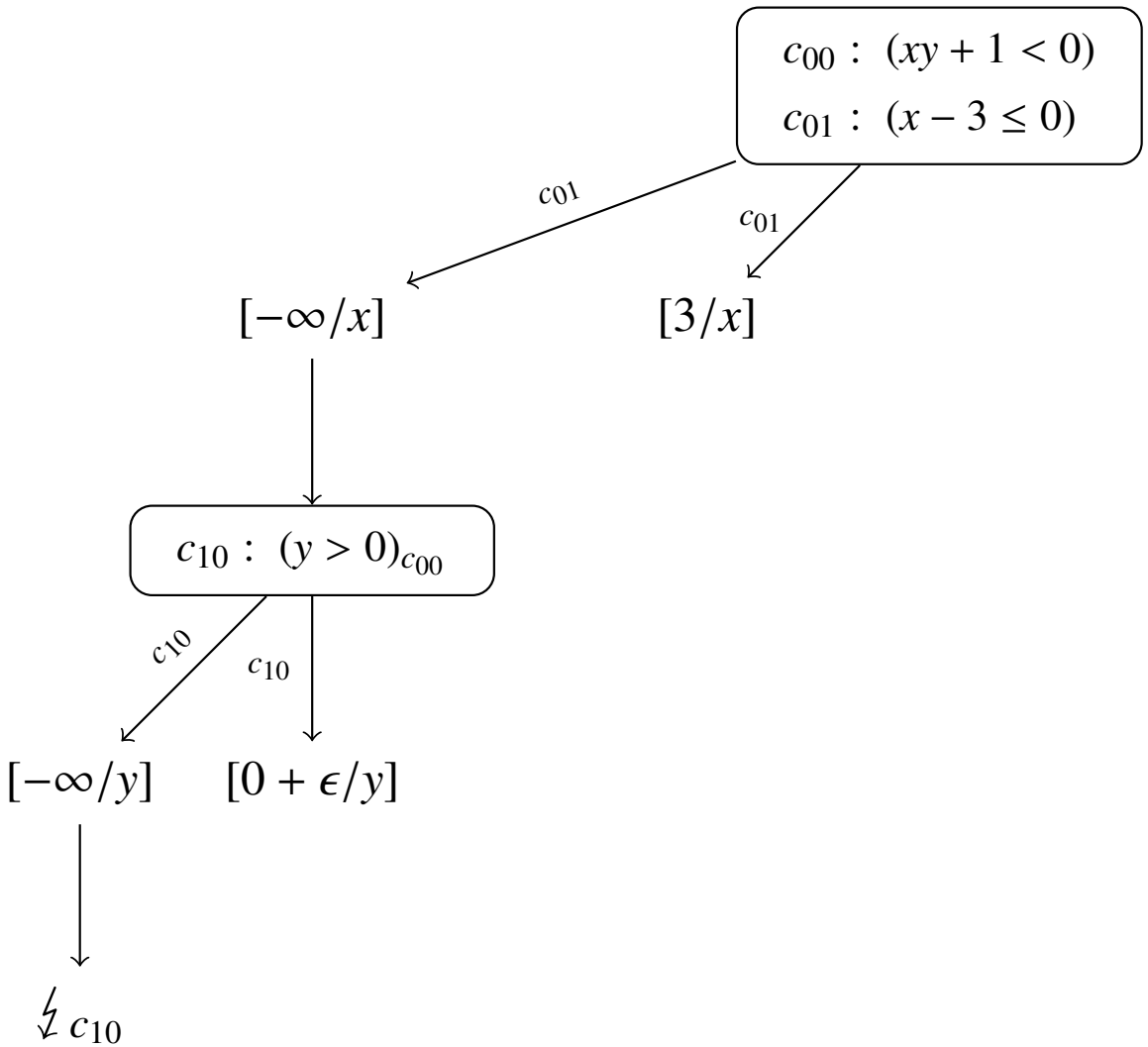
Theory solver: add $xy + 1 < 0$, add $x - 3 \leq 0$, check consistency



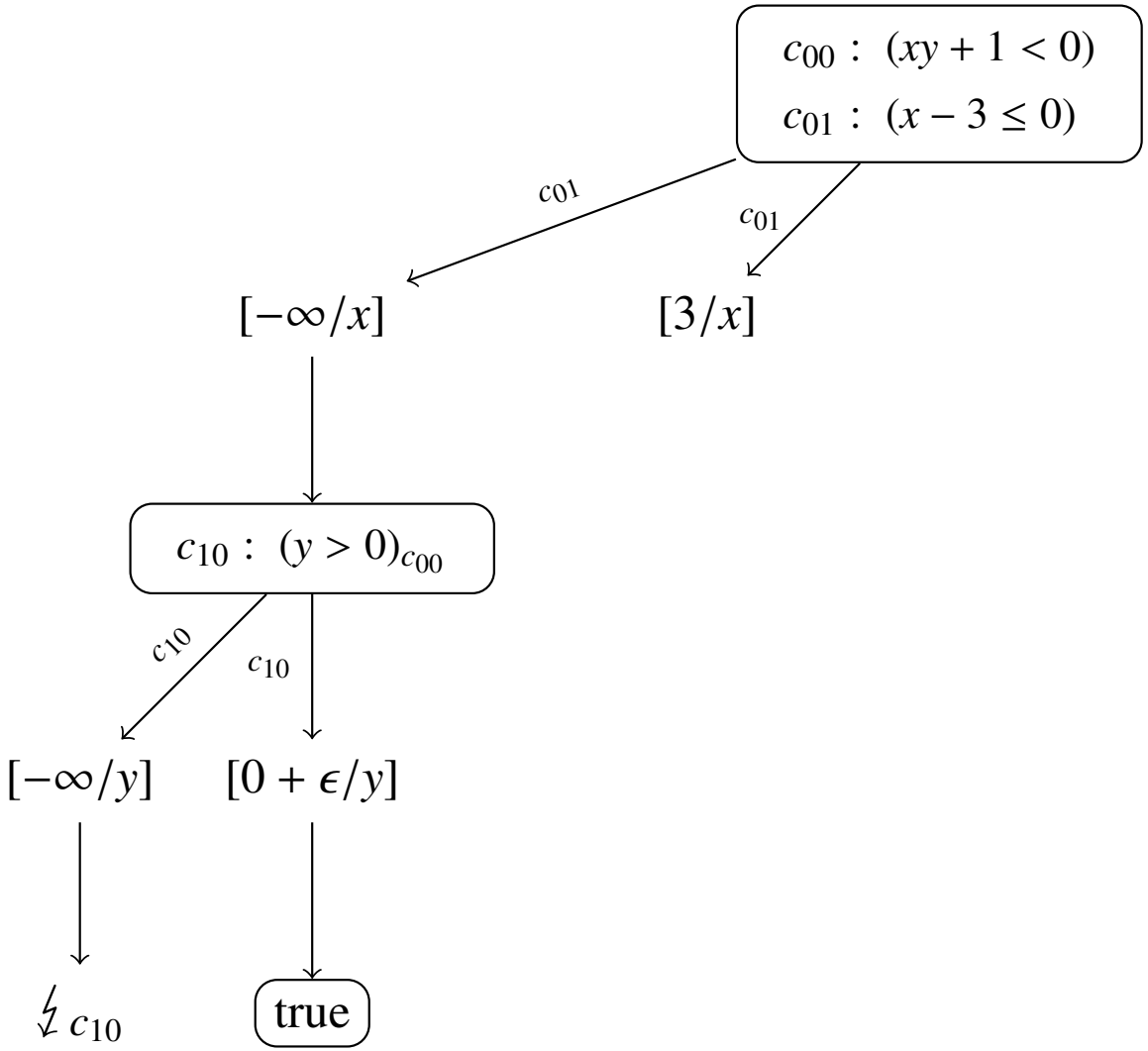
Theory solver: add $xy + 1 < 0$, add $x - 3 \leq 0$, check consistency



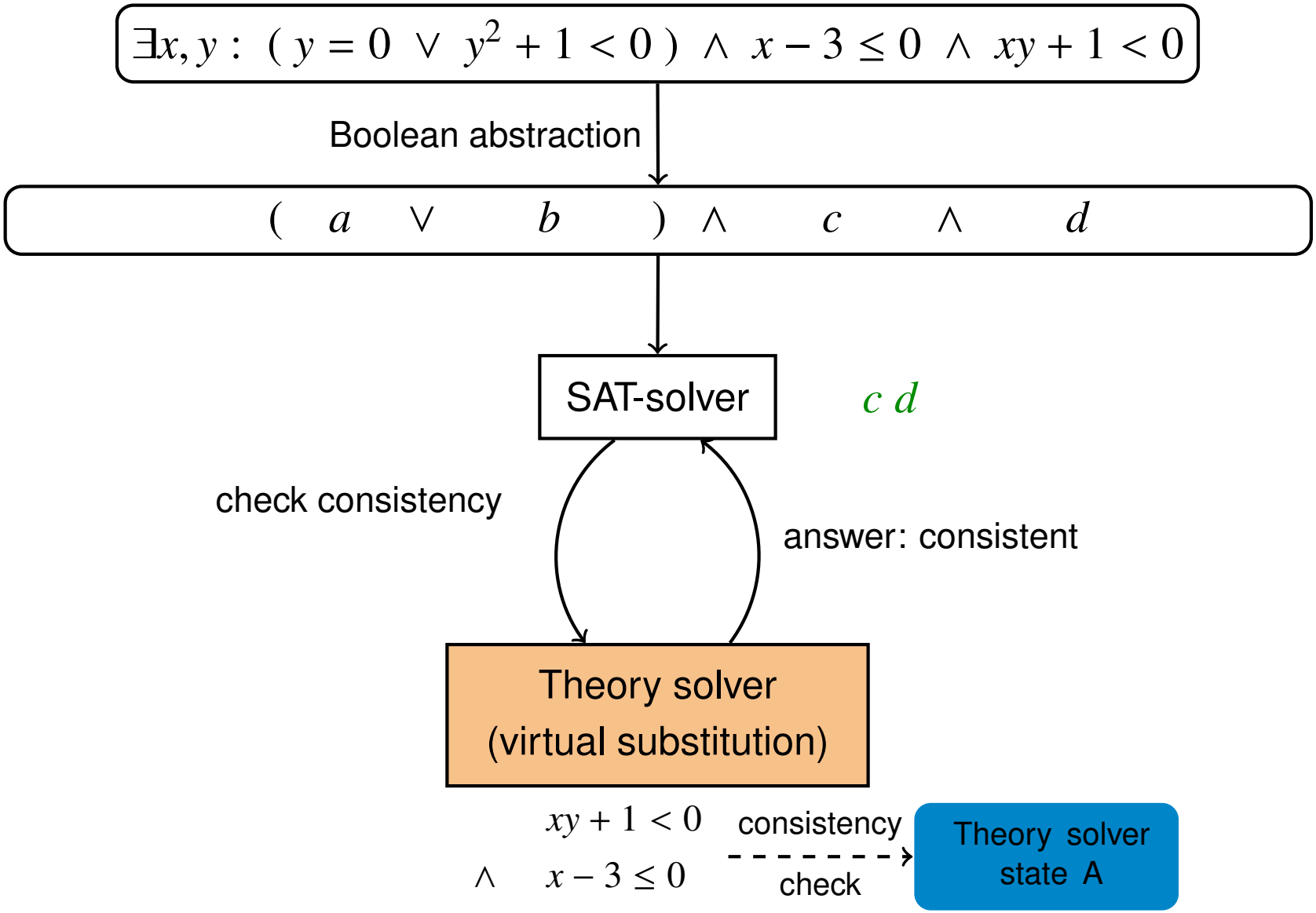
Theory solver: add $xy + 1 < 0$, add $x - 3 \leq 0$, check consistency



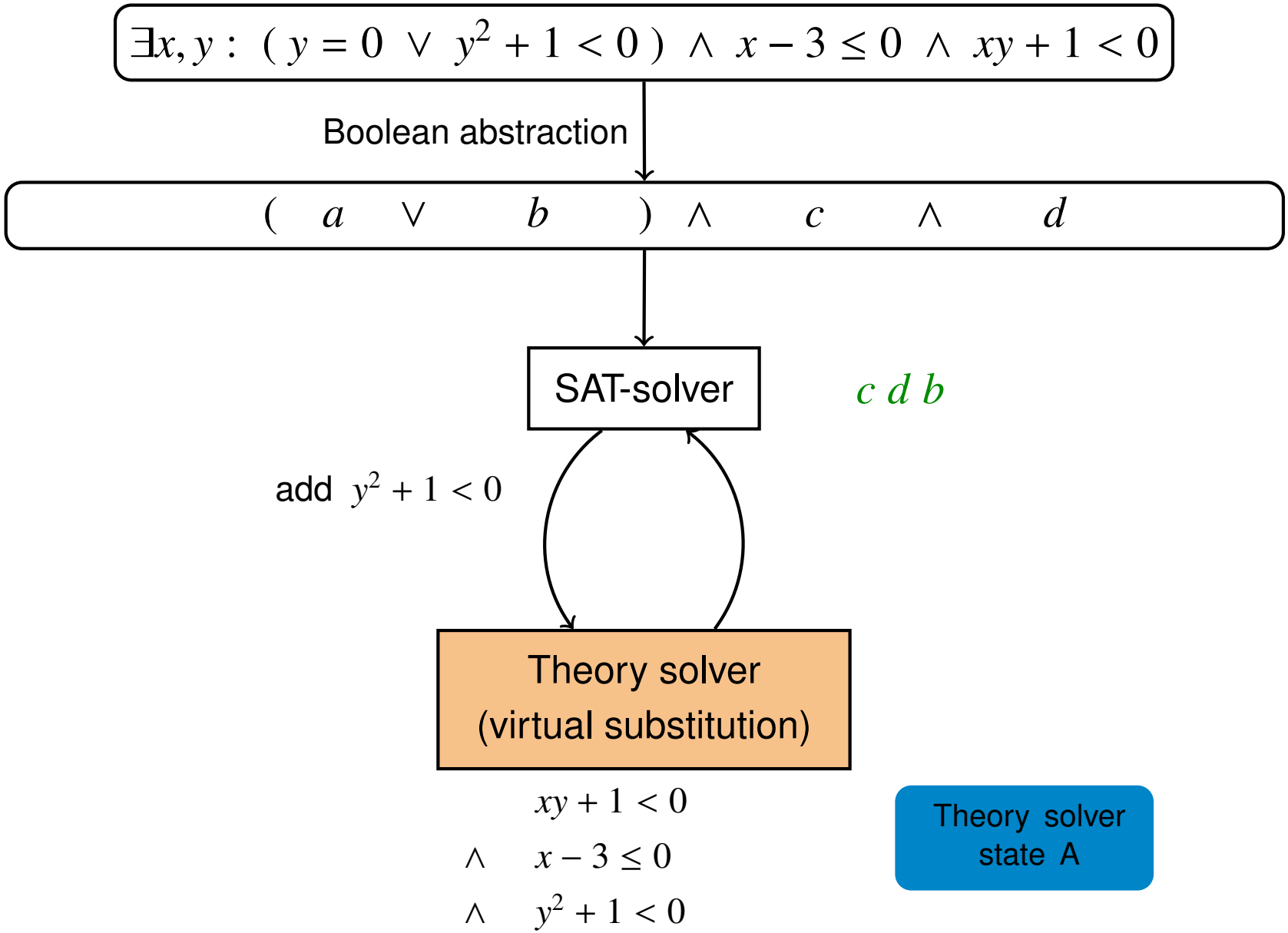
Theory solver: add $xy + 1 < 0$, add $x - 3 \leq 0$, check consistency



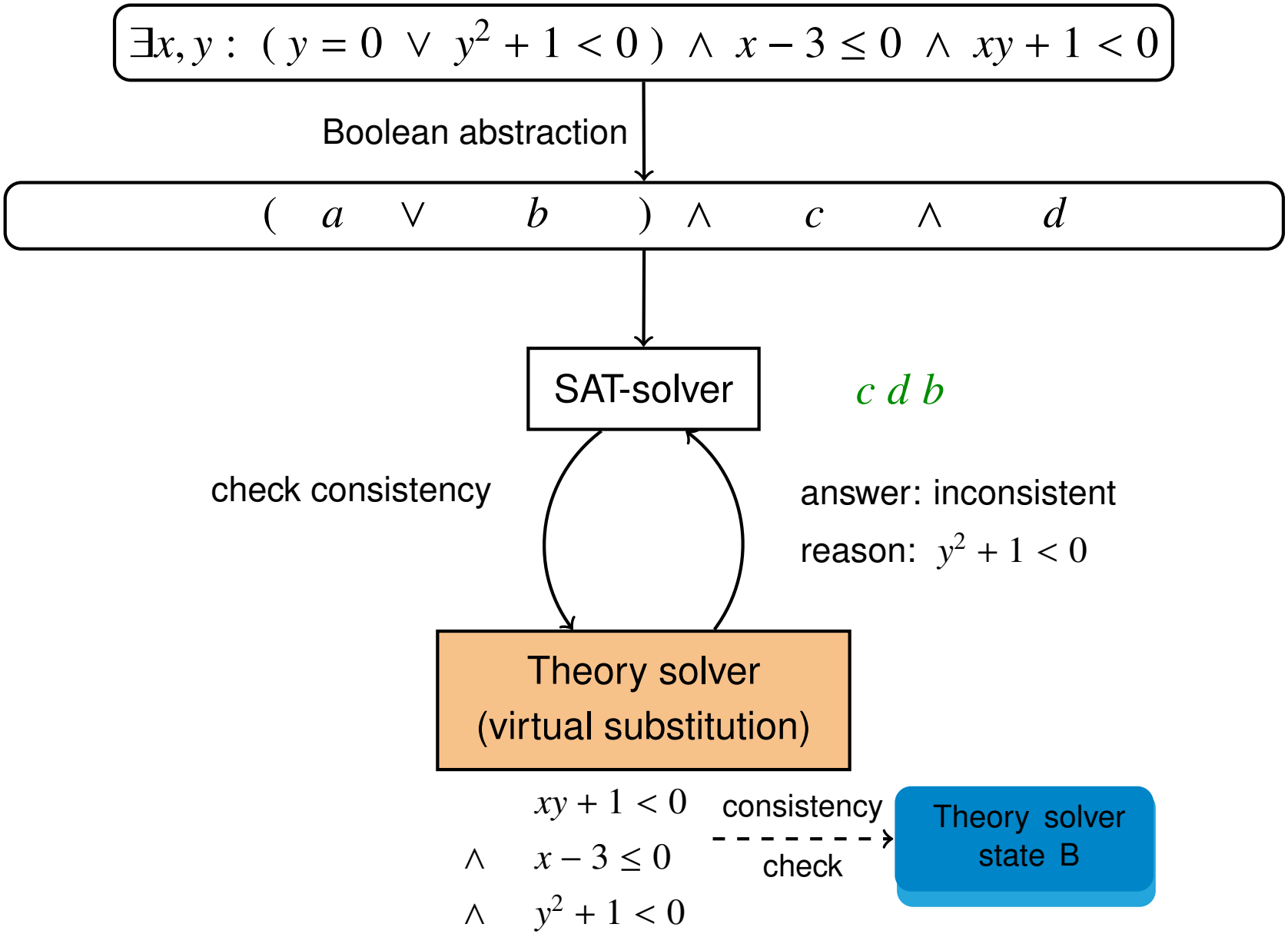
SMT solver


$\exists x, y : \ (\ y = 0 \ \vee \ y^2 + 1 < 0 \ ) \ \wedge \ x - 3 \leq 0 \ \wedge \ xy + 1 < 0$
Boolean abstraction
$(\quad a \quad \vee \quad b \quad) \ \wedge \quad c \quad \wedge \quad d$
SAT-solver
$c \ d$
check consistency
answer: consistent
Theory solver
(virtual substitution)
$xy + 1 < 0$
$\wedge \quad x - 3 \leq 0$
consistency
check
Theory solver
state A

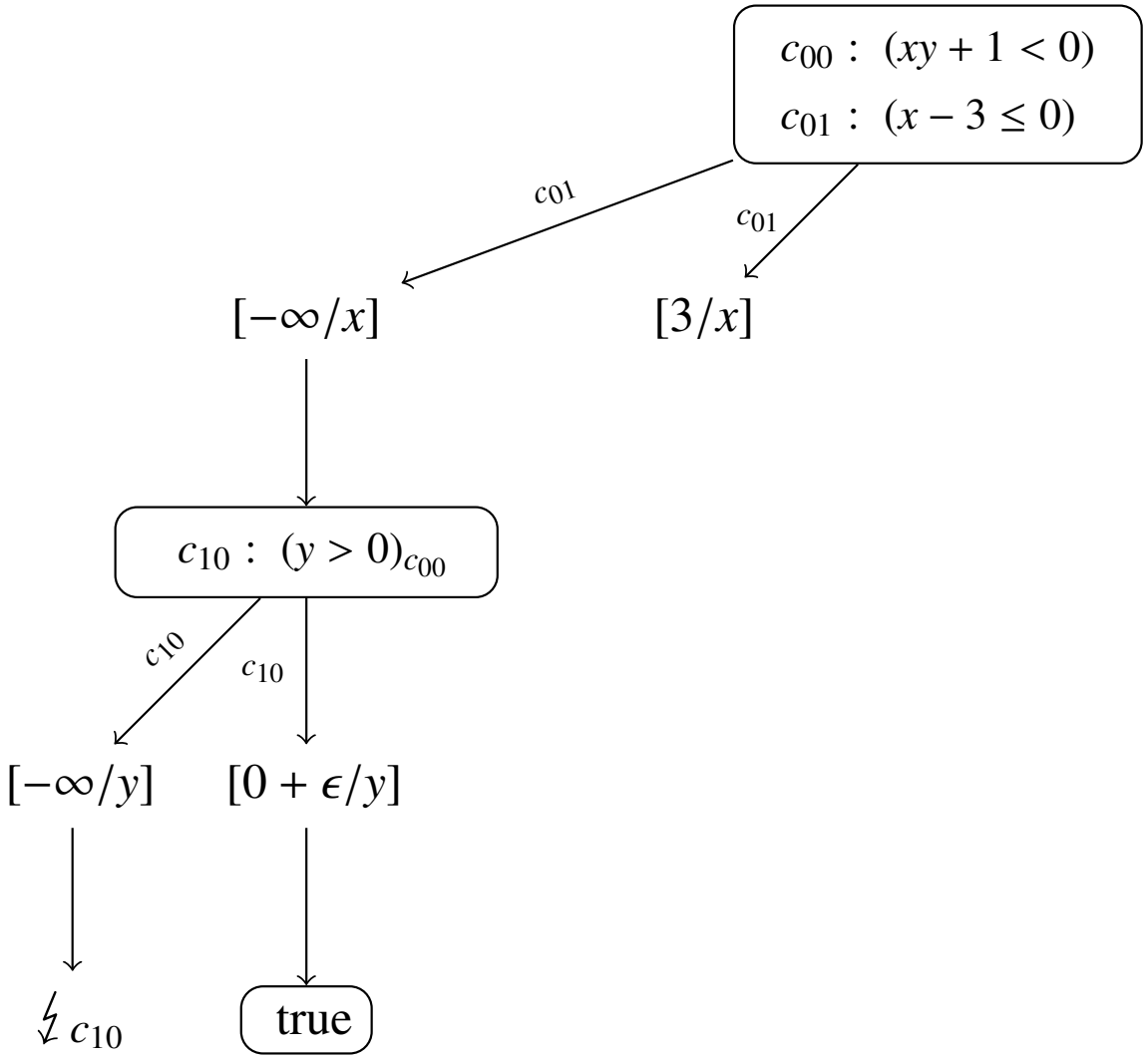
SMT solver


$\exists x, y : (y = 0 \vee y^2 + 1 < 0) \wedge x - 3 \leq 0 \wedge xy + 1 < 0$
Boolean abstraction
$(a \vee b) \wedge c \wedge d$
SAT-solver
$c\ d\ b$
add $y^2 + 1 < 0$
Theory solver
(virtual substitution)
$xy + 1 < 0$
$\wedge\ x - 3 \leq 0$
$\wedge\ y^2 + 1 < 0$
Theory solver
state A

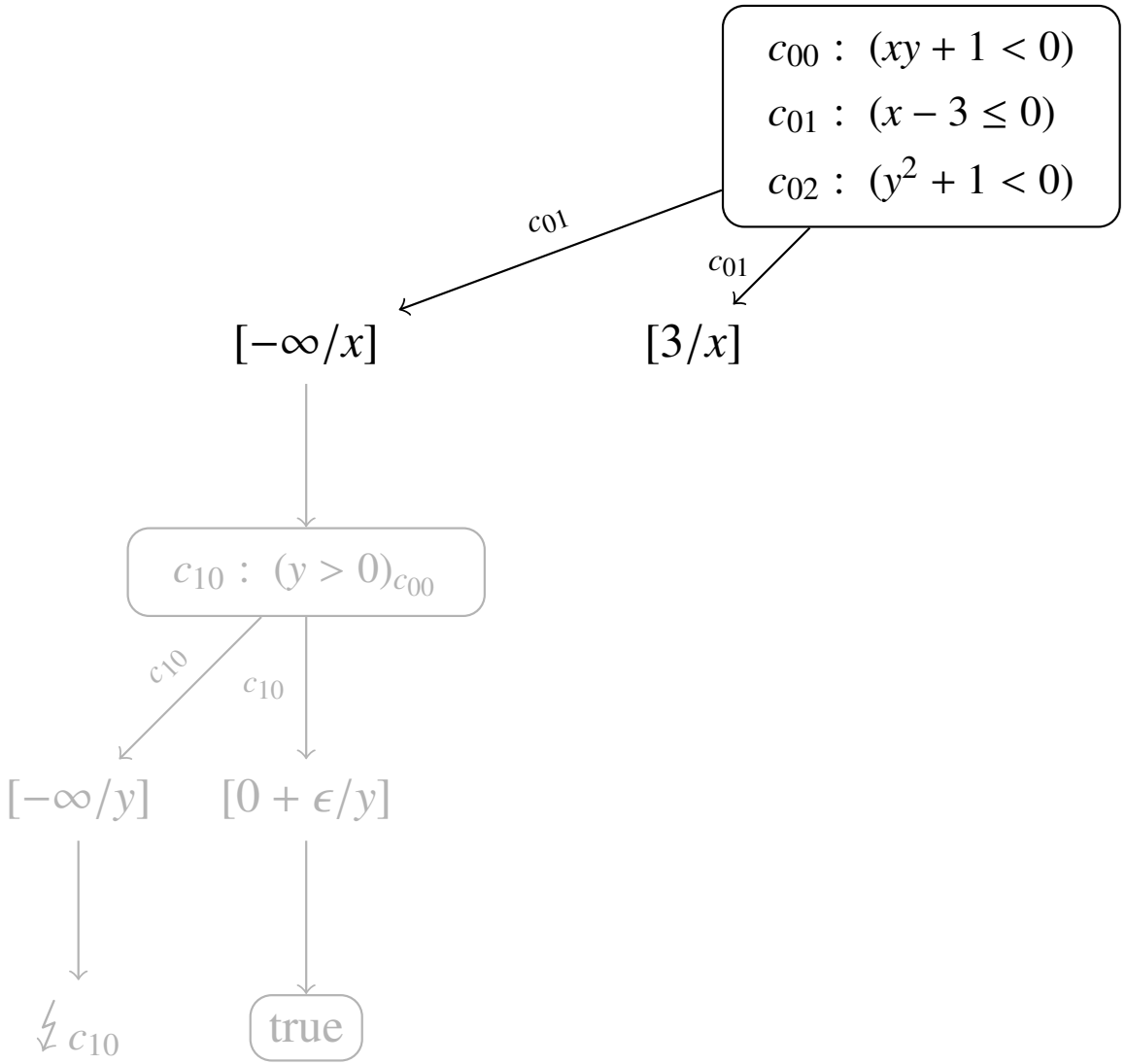
SMT solver


$\exists x, y: \ (\ y = 0 \ \vee \ y^2 + 1 < 0\ ) \ \wedge \ x - 3 \leq 0 \ \wedge \ xy + 1 < 0$
Boolean abstraction
$(\ a \ \vee \ b \ ) \ \wedge \ c \ \wedge \ d$
SAT-solver
$c\ d\ b$
check consistency
answer: inconsistent
reason: $y^2 + 1 < 0$
Theory solver
(virtual substitution)
$xy + 1 < 0$
$\wedge \quad x - 3 \leq 0$
$\wedge \quad y^2 + 1 < 0$
consistency
check
Theory solver
state B

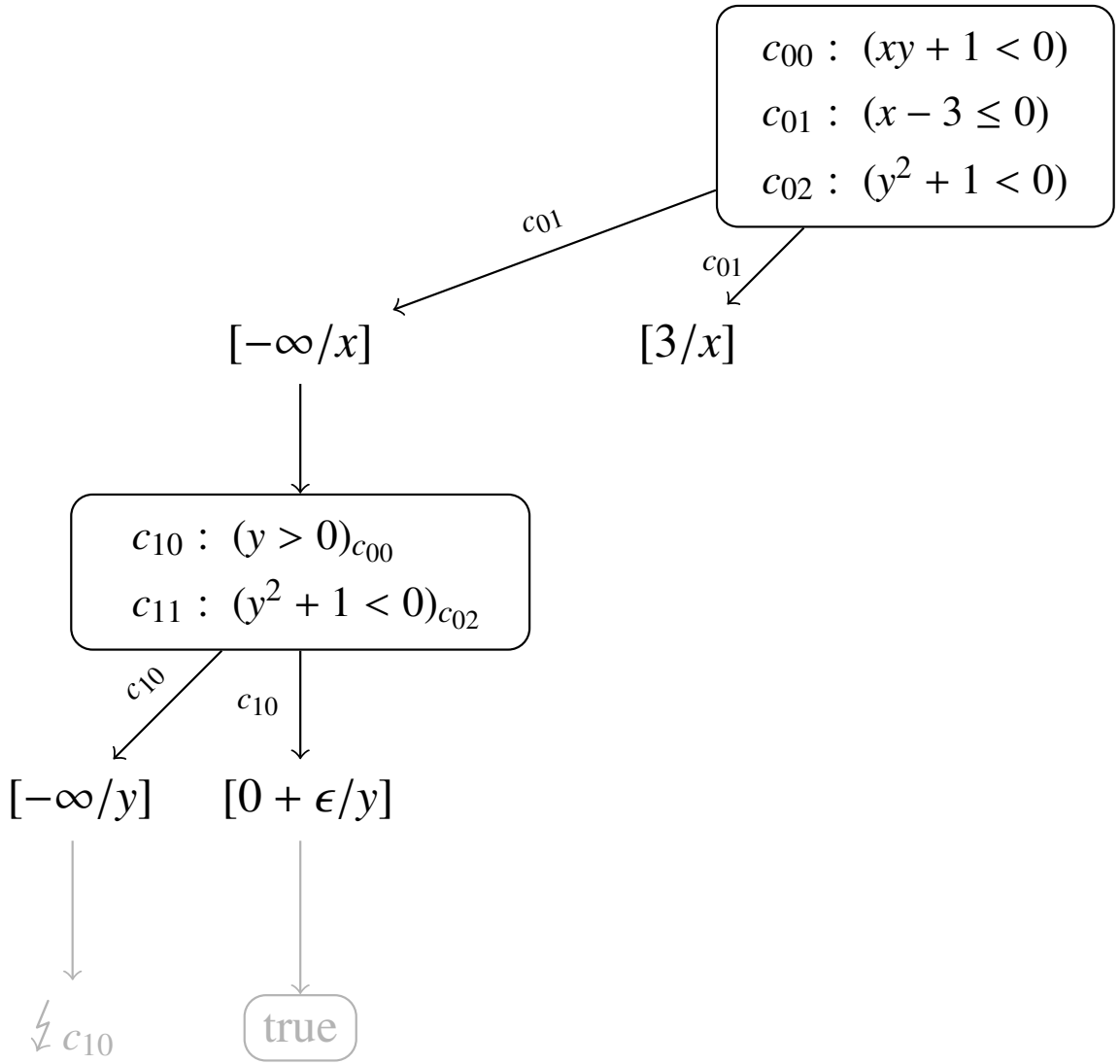
Theory solver: add $y^2 + 1 < 0$, check consistency



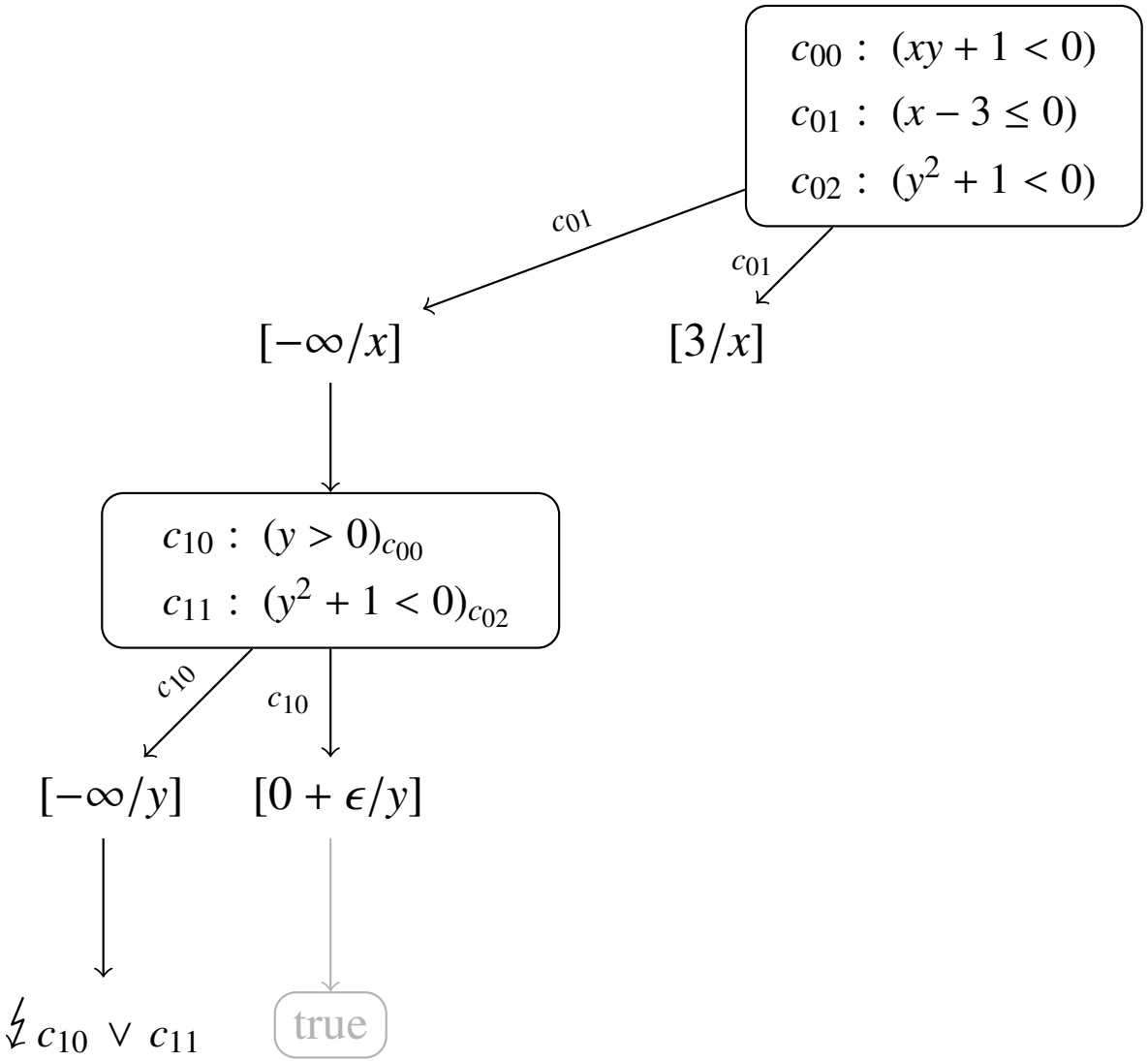
Theory solver: add $y^2 + 1 < 0$, check consistency



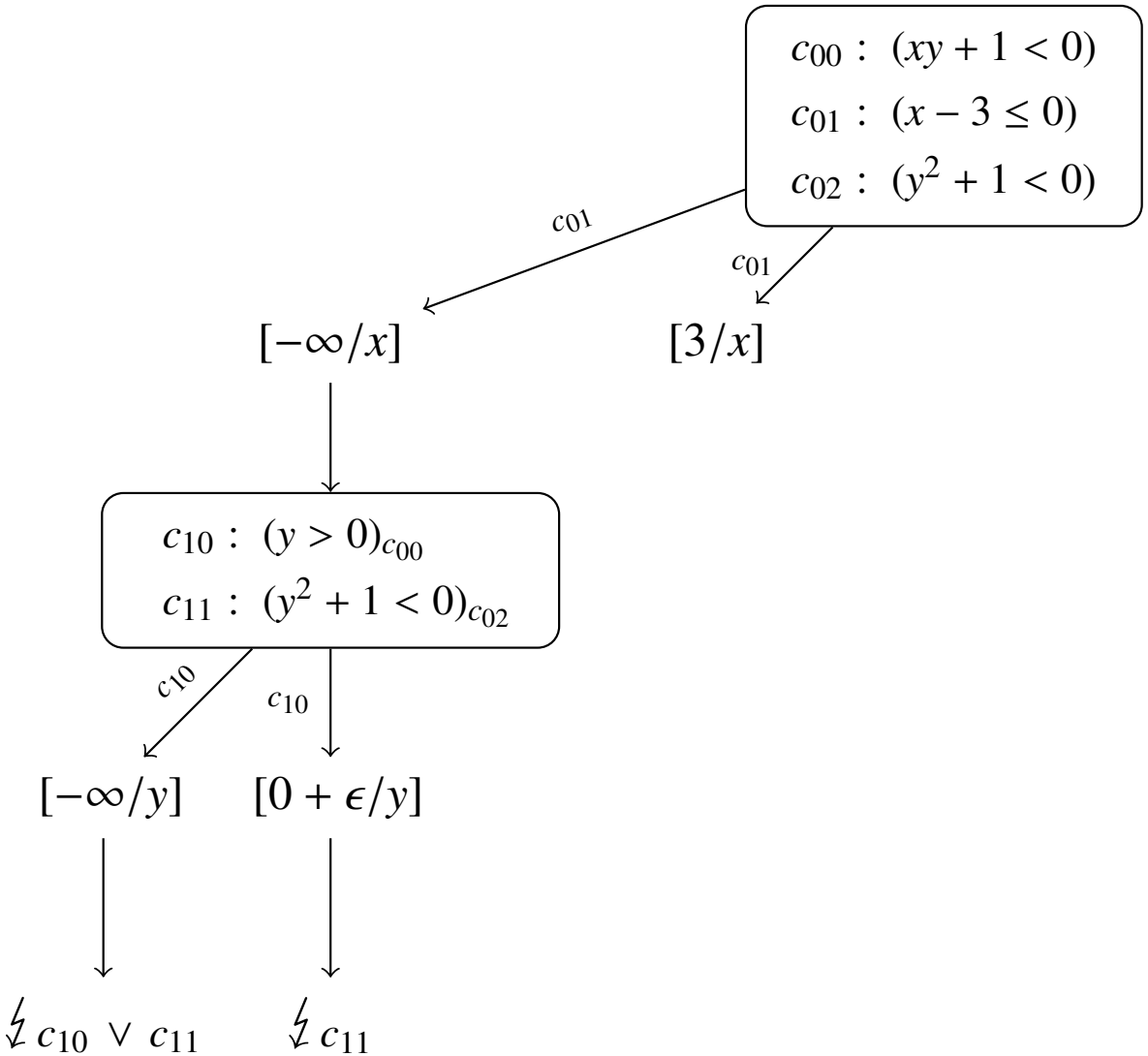
Theory solver: add $y^2 + 1 < 0$, check consistency



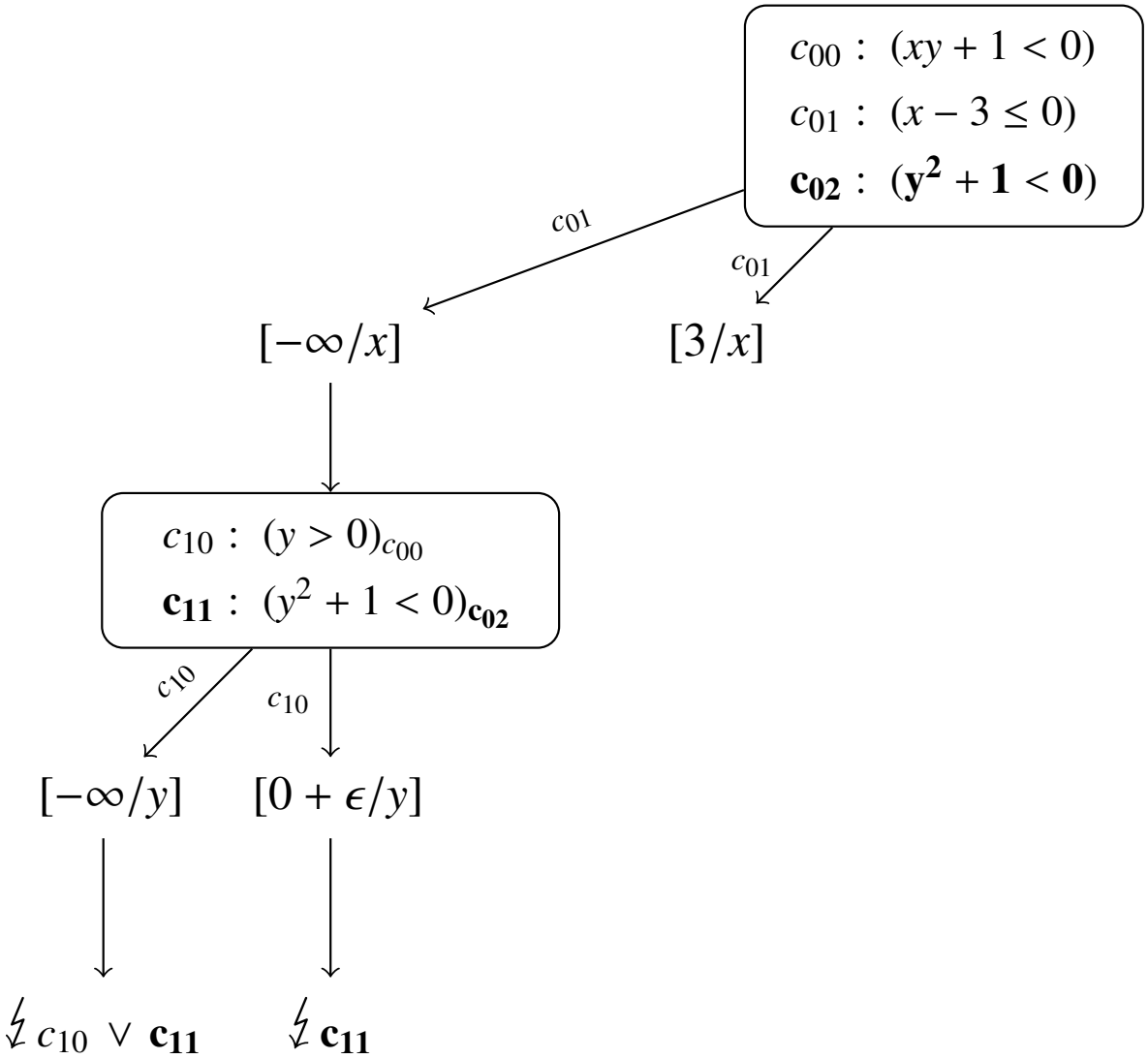
Theory solver: add $y^2 + 1 < 0$, check consistency



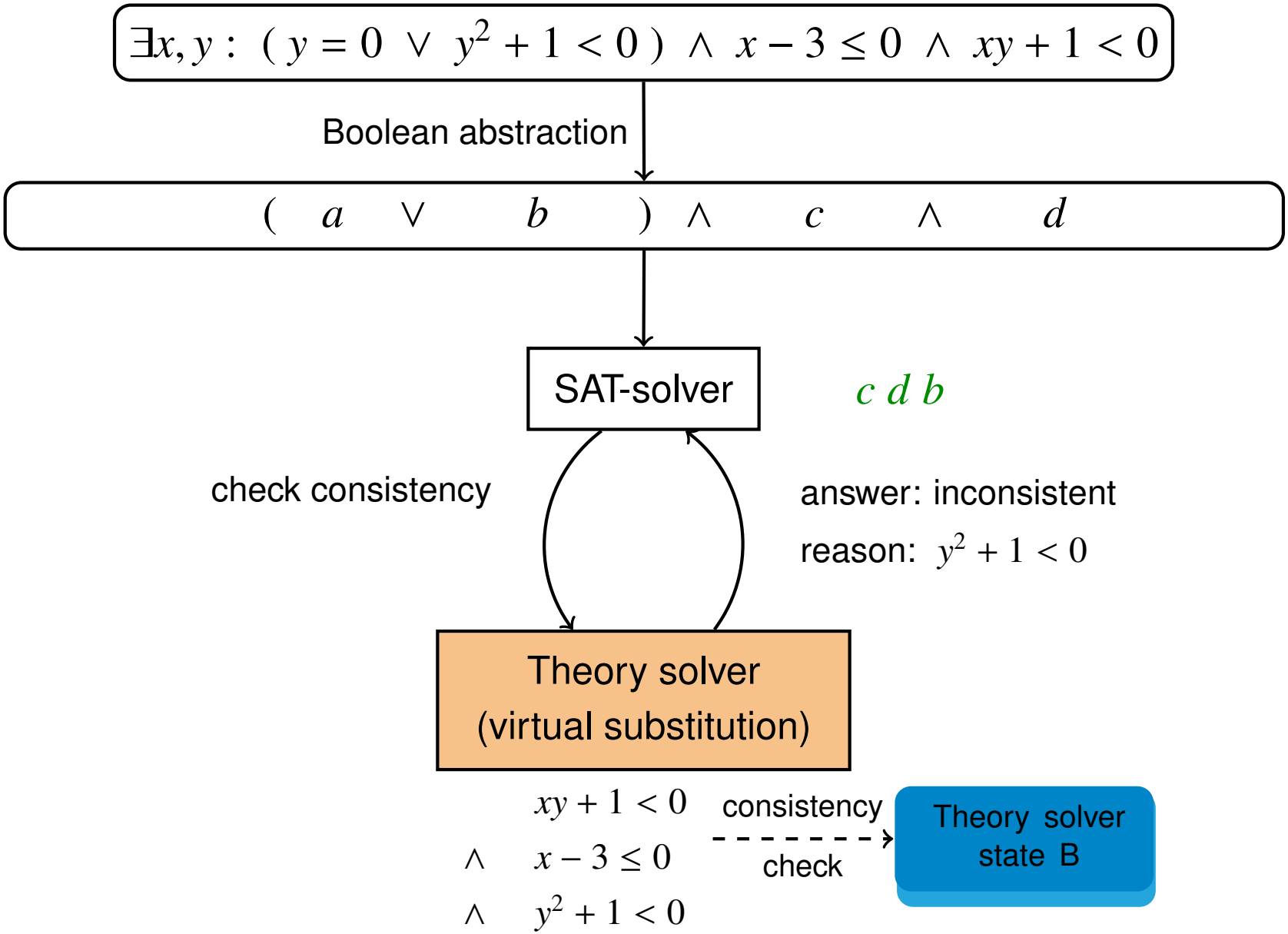
Theory solver: add $y^2 + 1 < 0$, check consistency



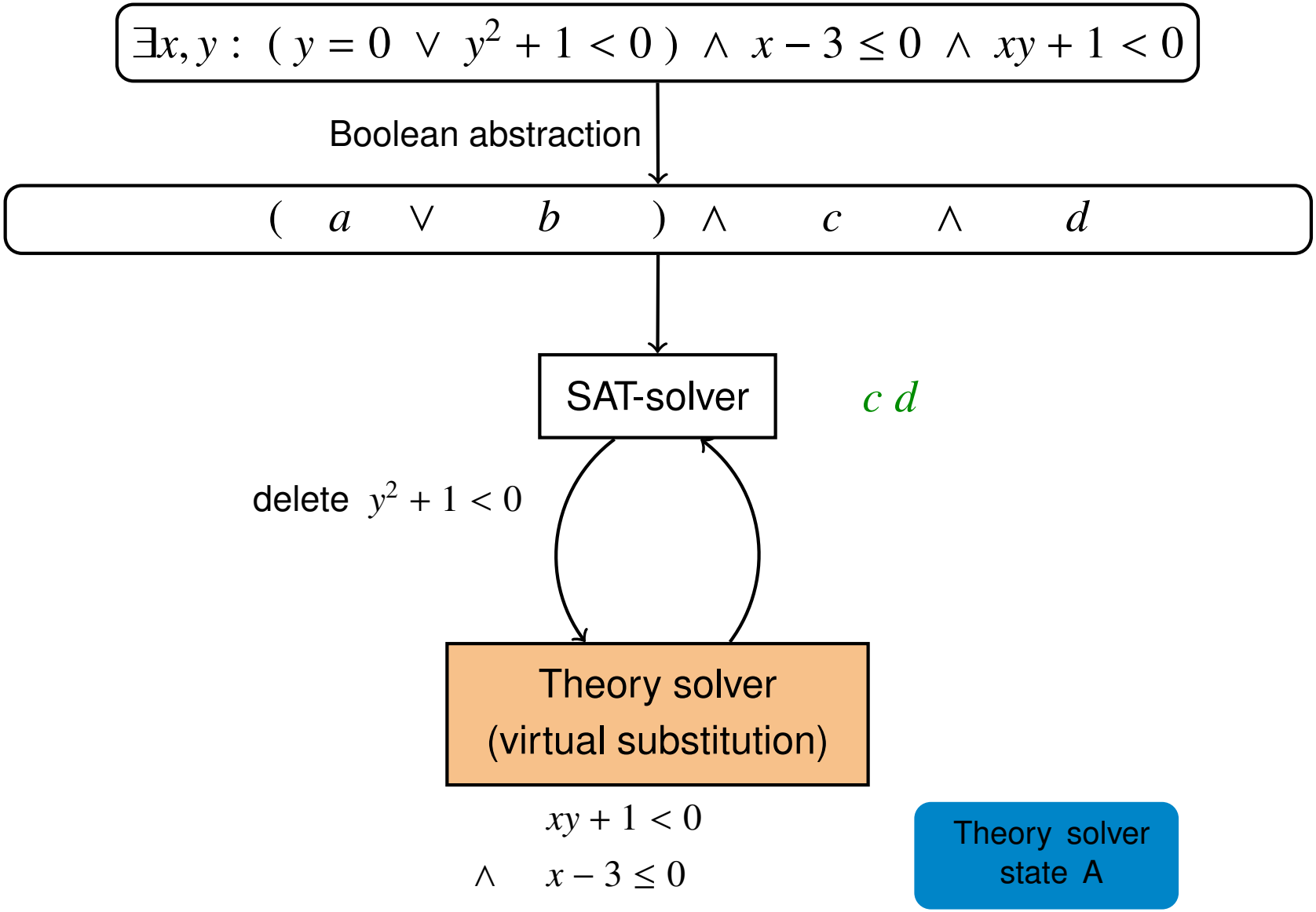
Theory solver: add $y^2 + 1 < 0$, check consistency



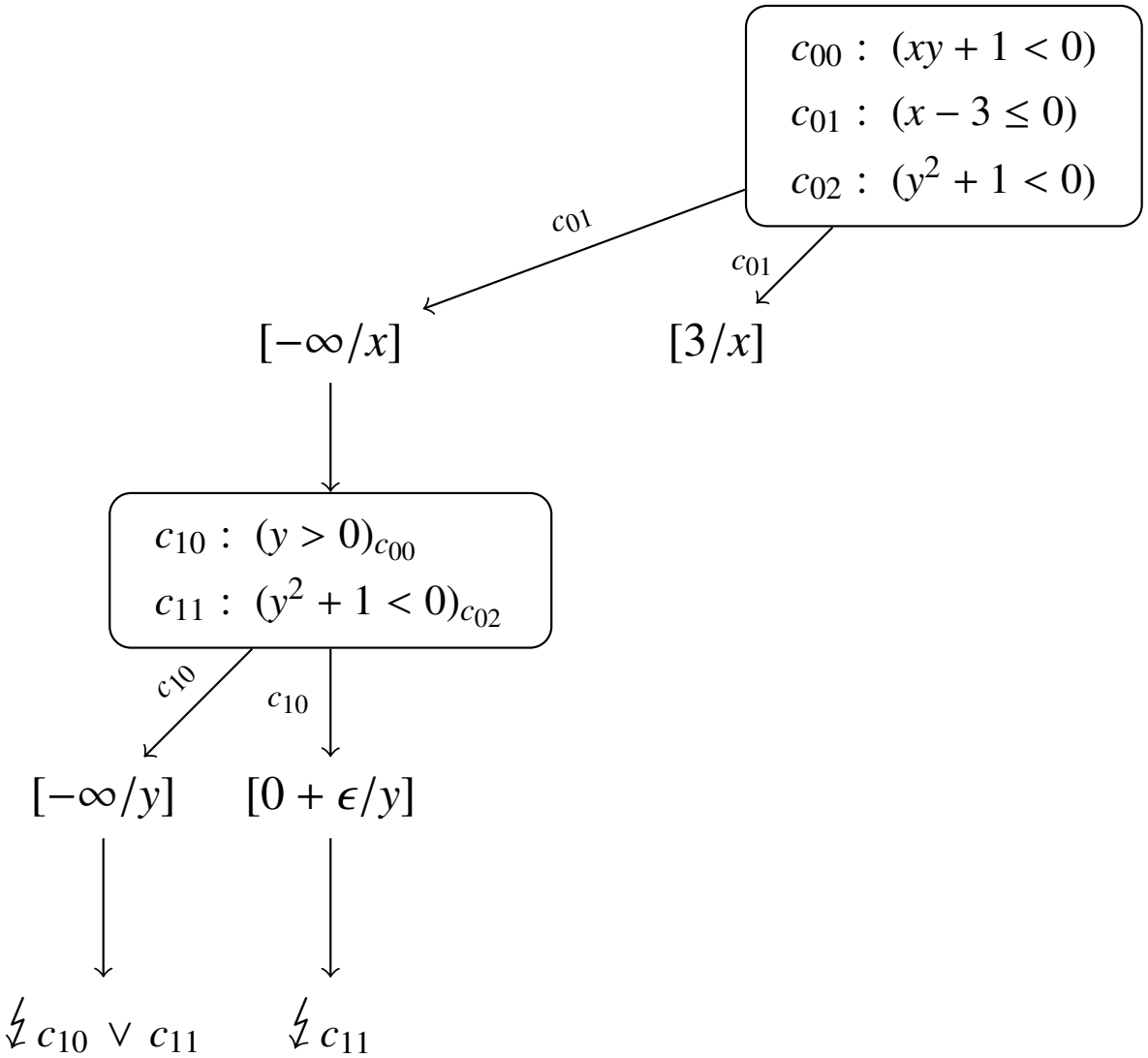
SMT solver


$\exists x, y: \ (\ y = 0 \ \vee \ y^2 + 1 < 0\ ) \ \wedge \ x - 3 \leq 0 \ \wedge \ xy + 1 < 0$
Boolean abstraction
$(\ a \ \vee \ b \ ) \ \wedge \ c \ \wedge \ d$
SAT-solver
$c\ d\ b$
check consistency
answer: inconsistent
reason: $y^2 + 1 < 0$
Theory solver
(virtual substitution)
$xy + 1 < 0$
$\wedge \quad x - 3 \leq 0$
$\wedge \quad y^2 + 1 < 0$
consistency
check
Theory solver
state B

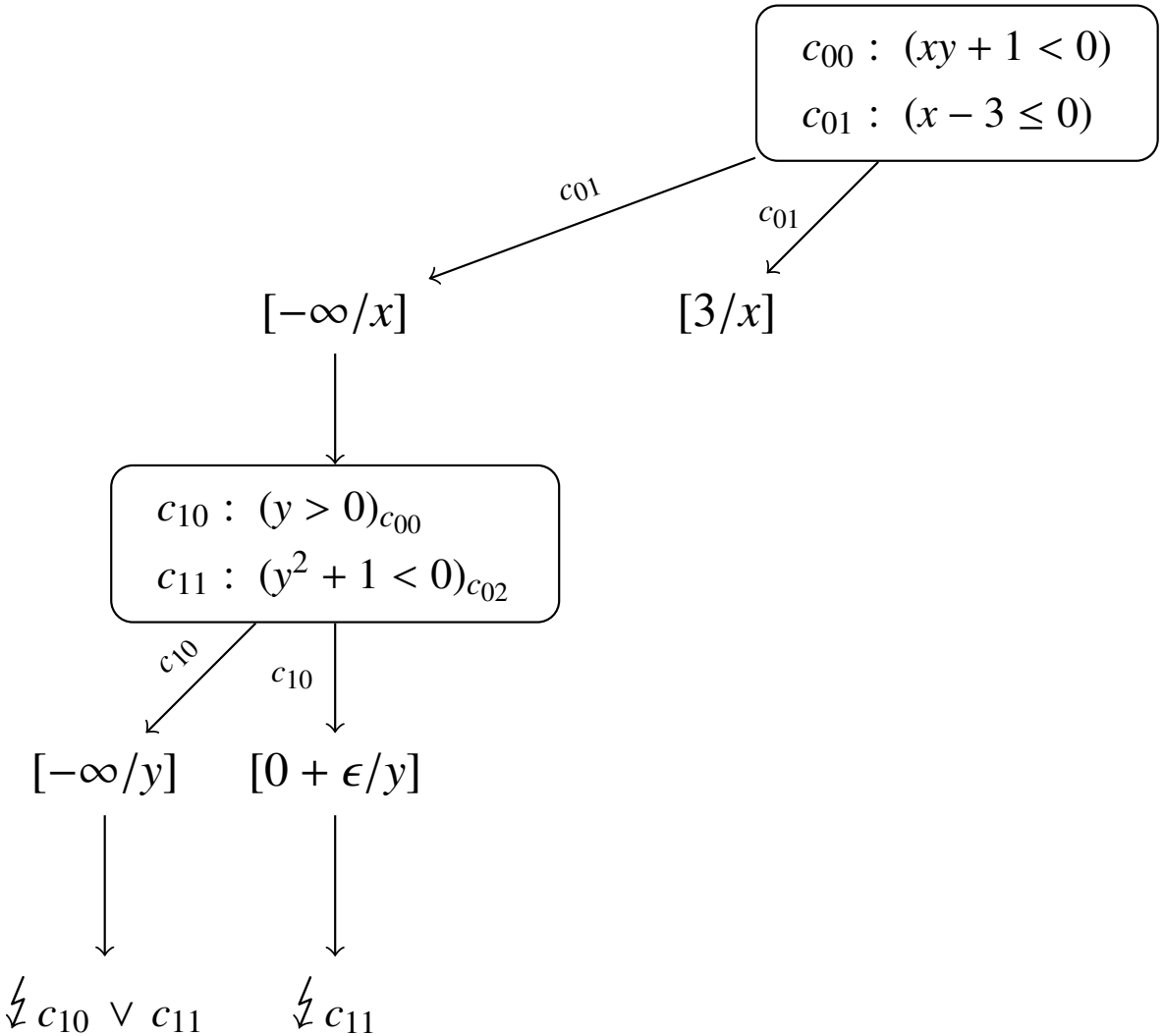
SMT solver


$\exists x, y : \ (y = 0 \ \vee \ y^2 + 1 < 0) \ \wedge \ x - 3 \leq 0 \ \wedge \ xy + 1 < 0$
Boolean abstraction
$(\ a \ \vee \ b \ ) \ \wedge \ c \ \wedge \ d$
SAT-solver
$c \ d$
delete $y^2 + 1 < 0$
Theory solver
(virtual substitution)
$xy + 1 < 0$
$\wedge \quad x - 3 \leq 0$
Theory solver
state A

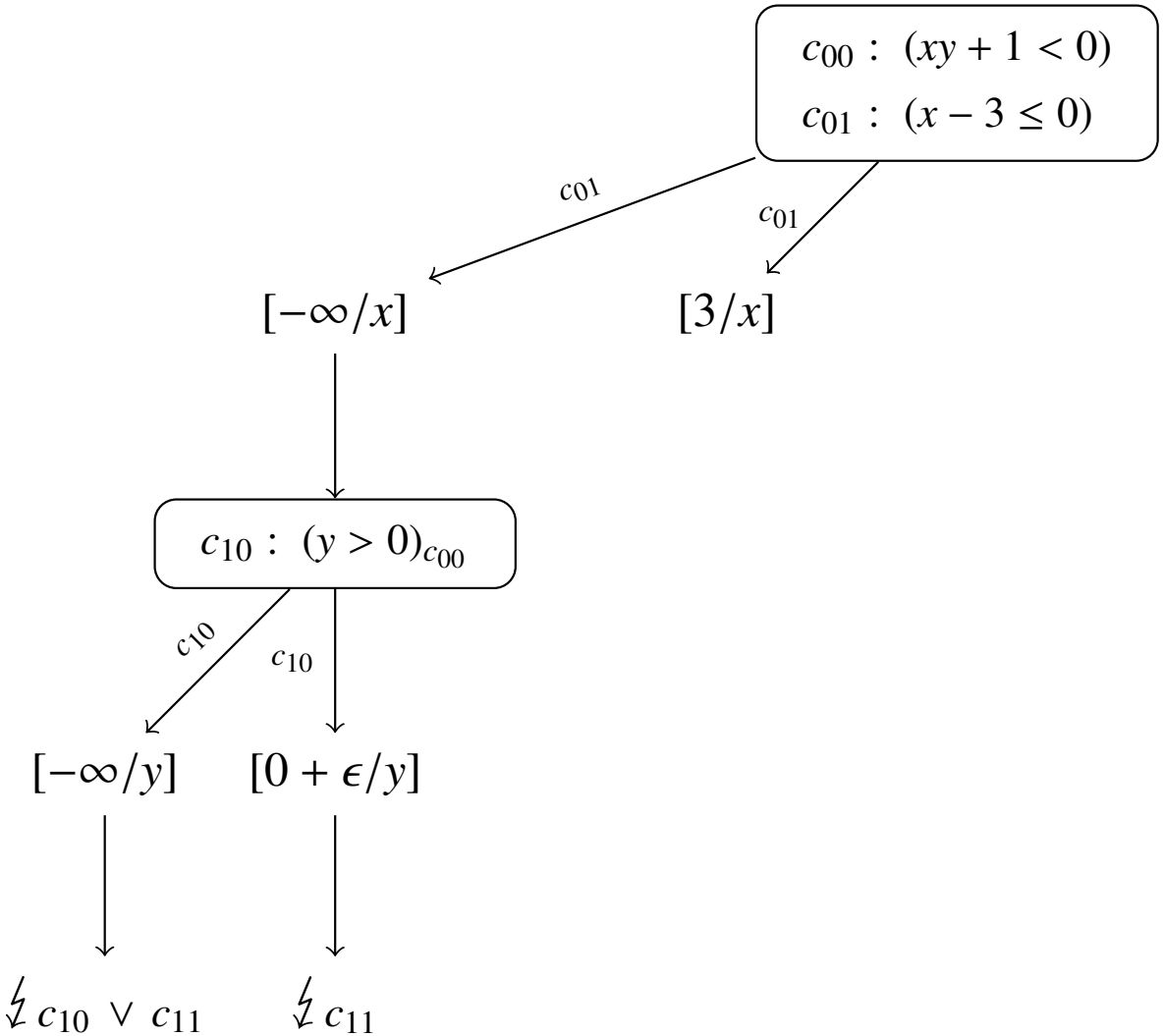
Theory solver: delete $y^2 + 1 < 0$



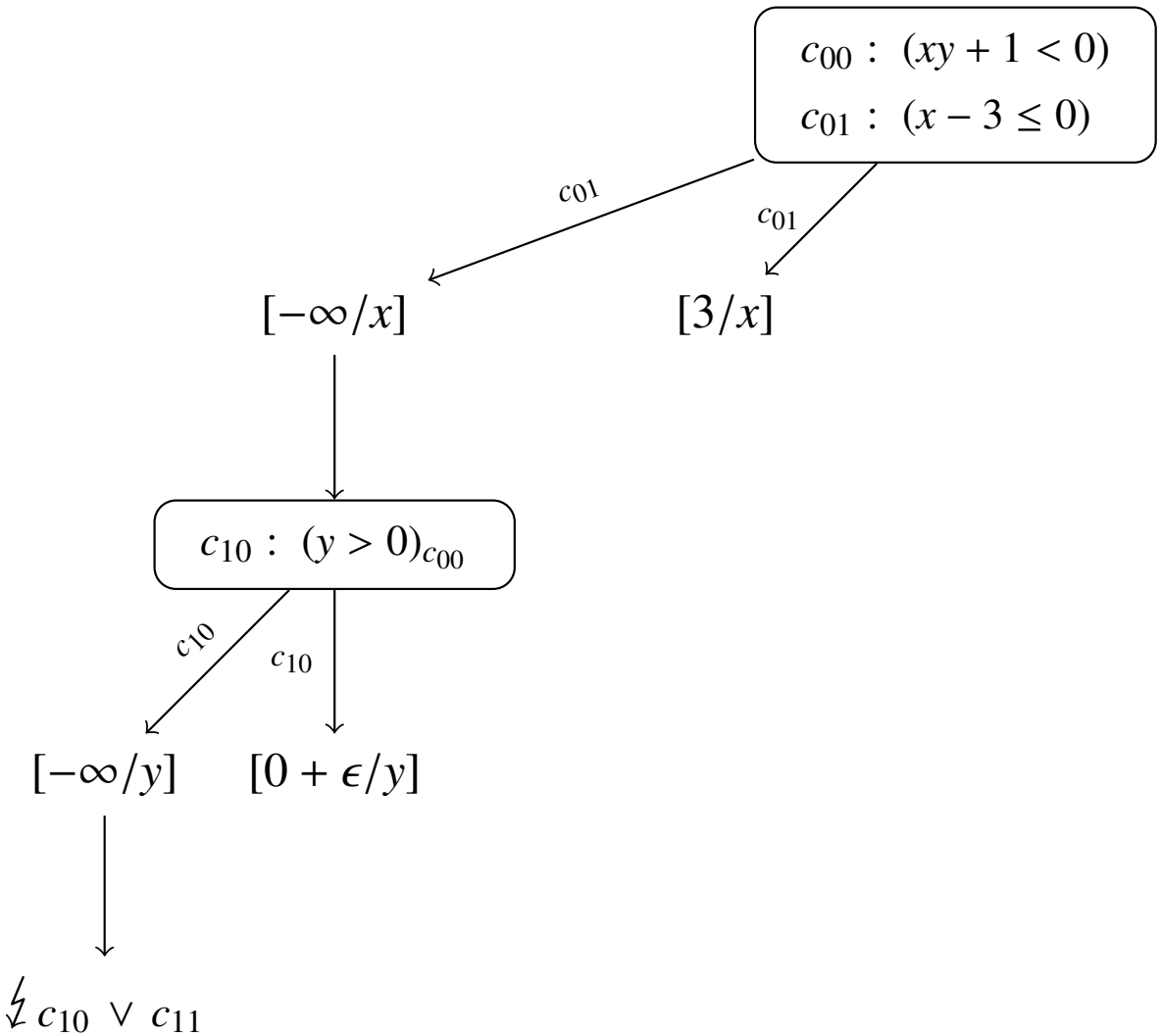
Theory solver: delete $y^2 + 1 < 0$



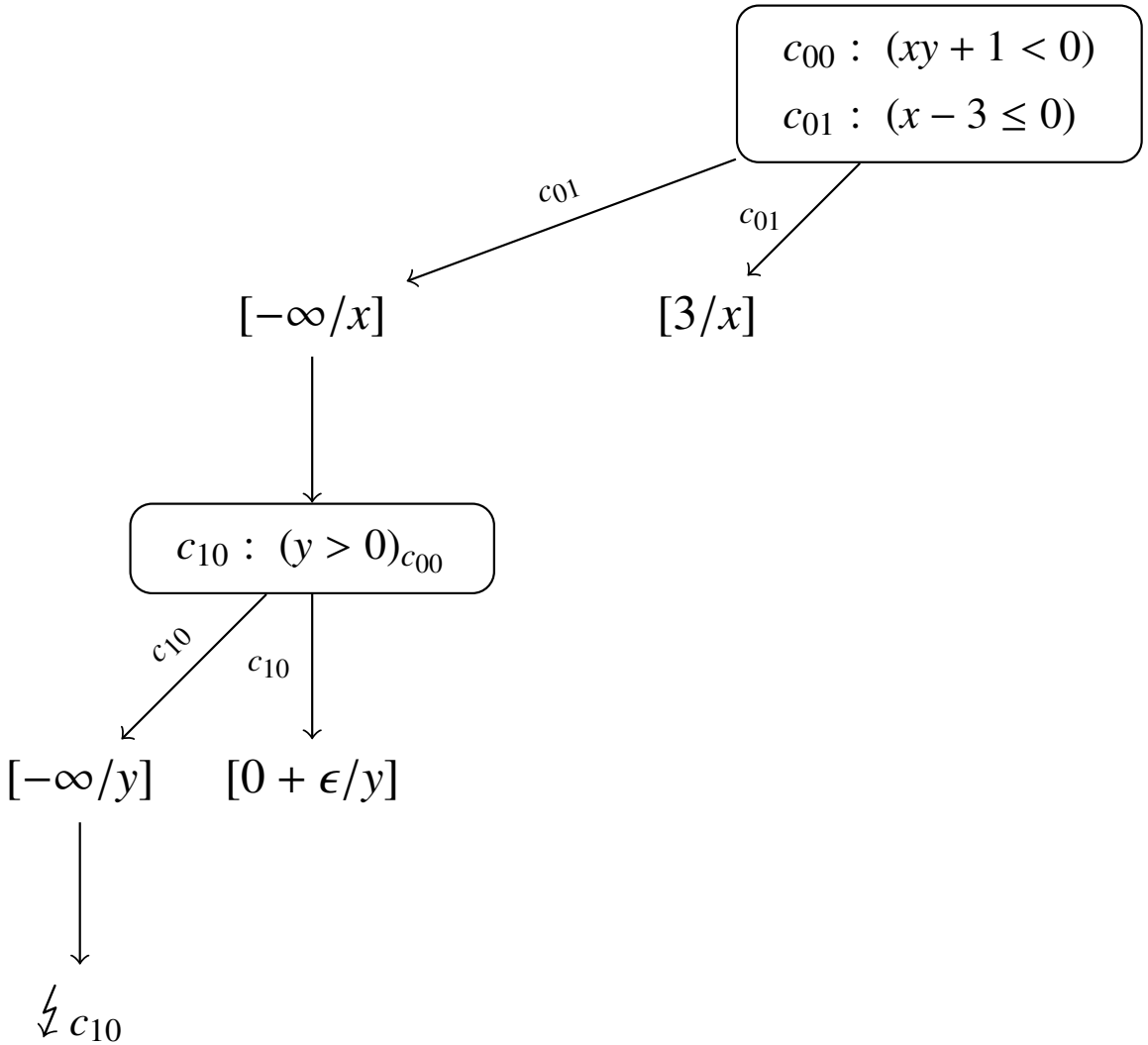
Theory solver: delete $y^2 + 1 < 0$



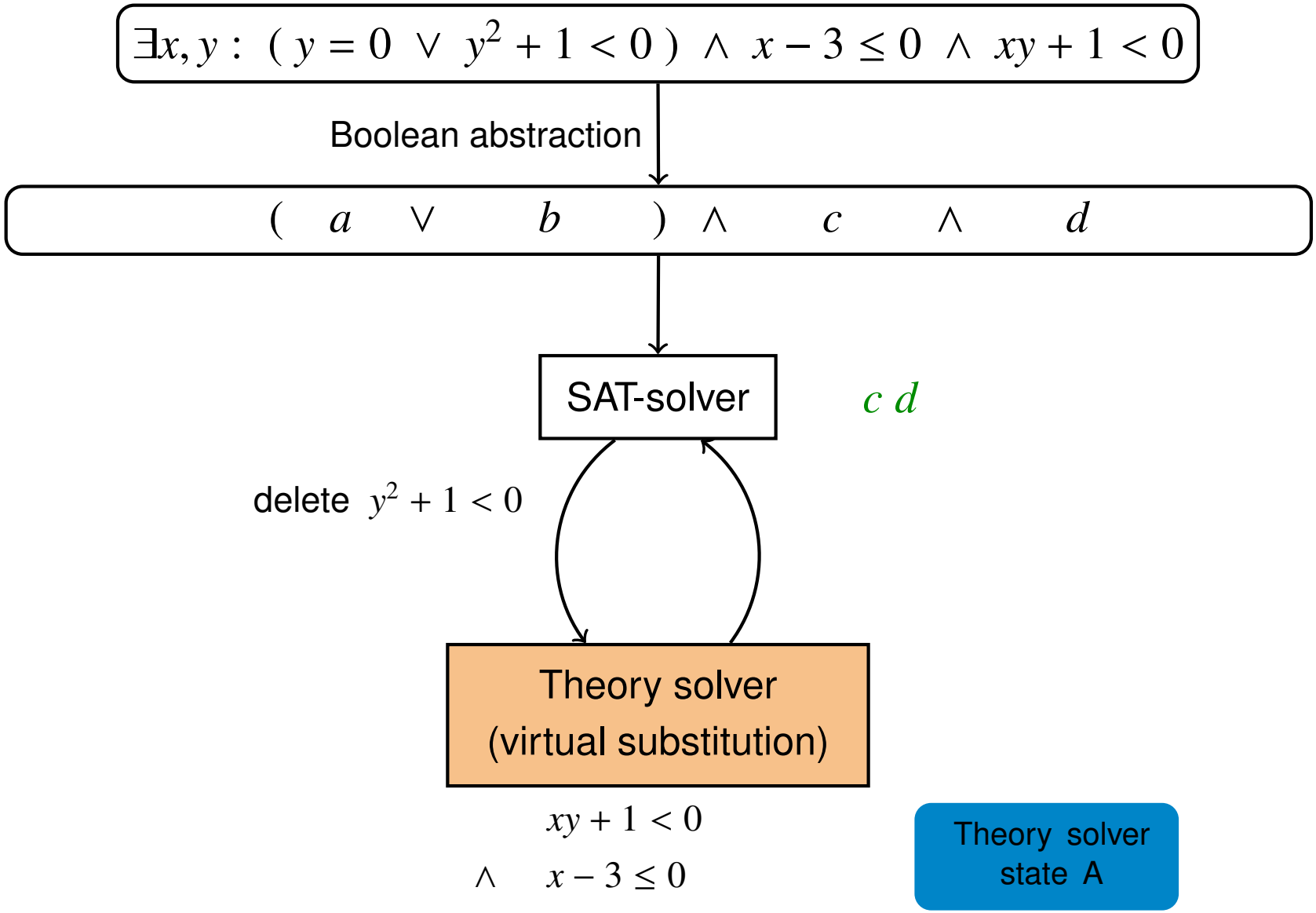
Theory solver: delete $y^2 + 1 < 0$



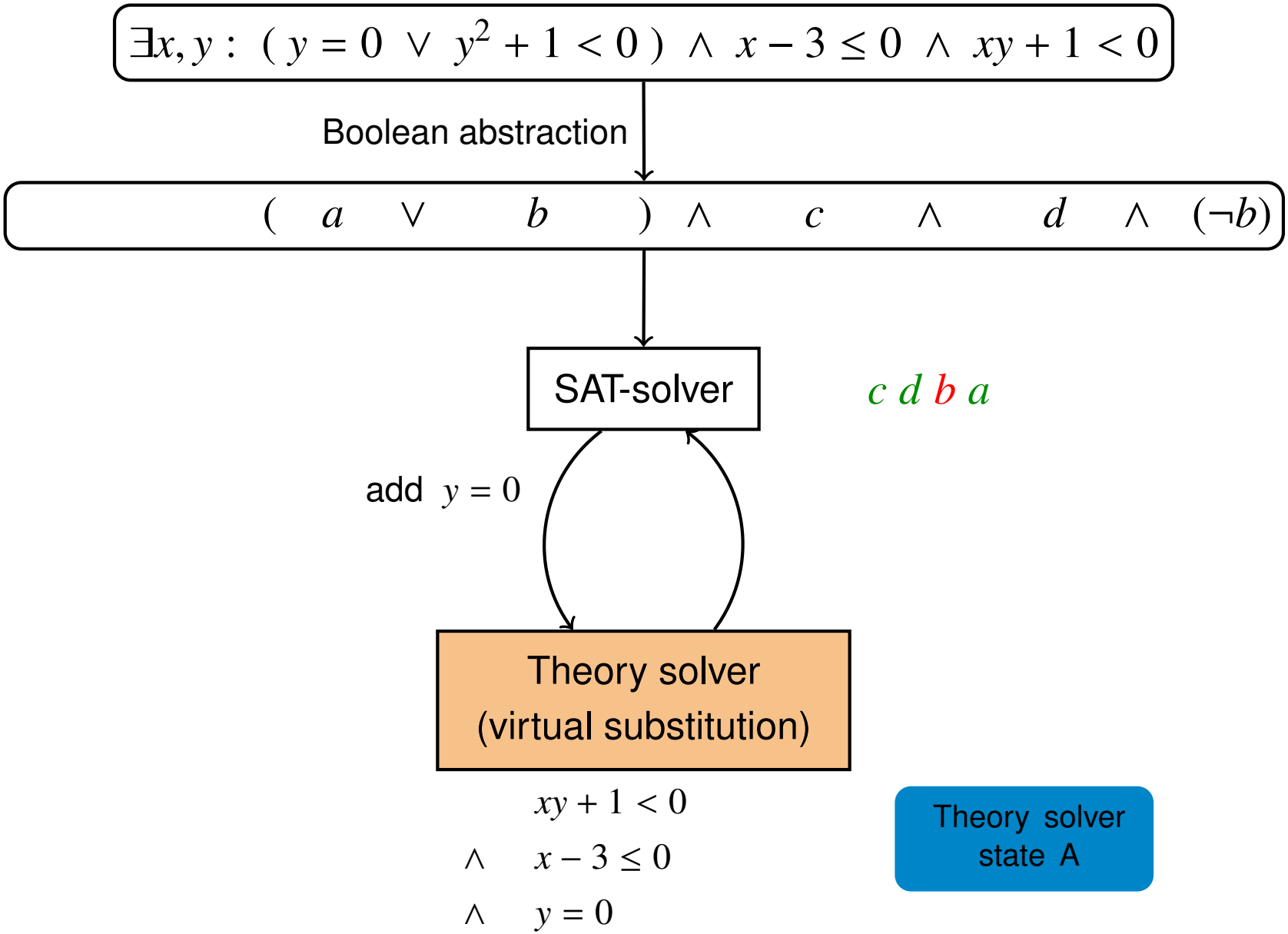
Theory solver: delete $y^2 + 1 < 0$



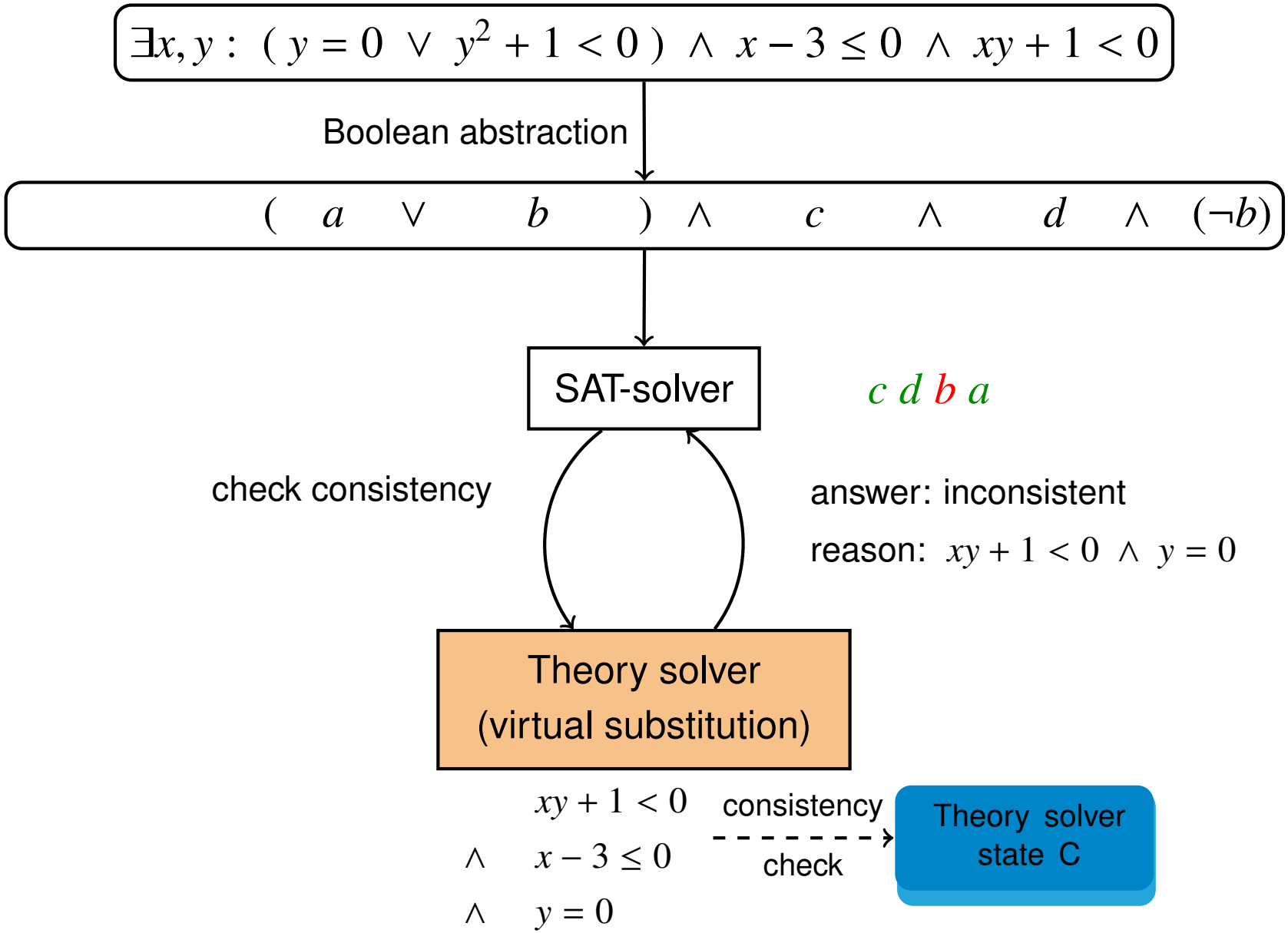
SMT solver


$\exists x, y : \ (\ y = 0 \ \vee \ y^2 + 1 < 0 \ ) \ \wedge \ x - 3 \leq 0 \ \wedge \ xy + 1 < 0$
Boolean abstraction
$(\quad a \quad \vee \quad b \quad) \ \wedge \quad c \quad \wedge \quad d$
SAT-solver
$c \ d$
delete $y^2 + 1 < 0$
Theory solver
(virtual substitution)
$xy + 1 < 0$
$\wedge \quad x - 3 \leq 0$
Theory solver
state A

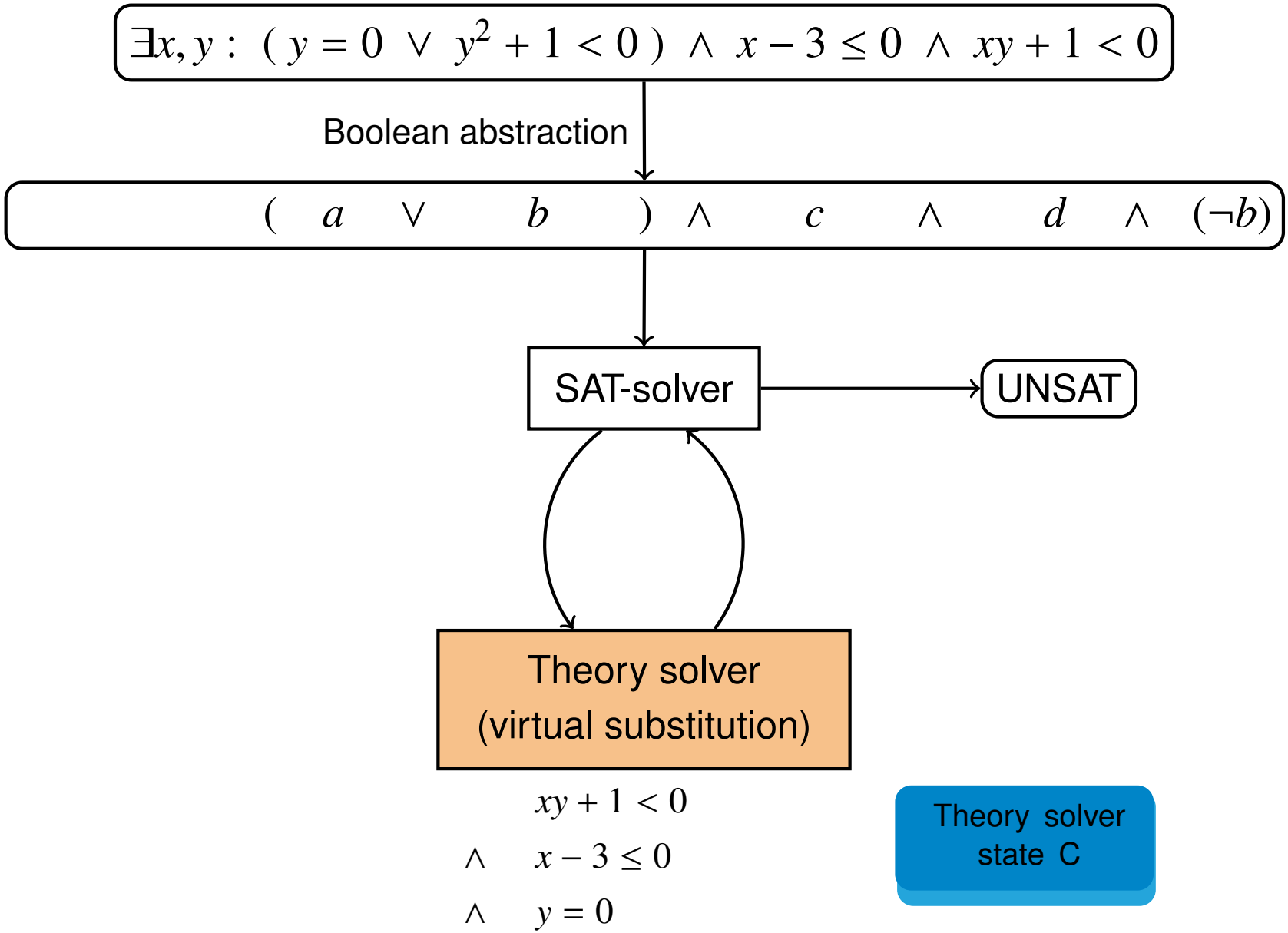
SMT solver


$\exists x, y : (y = 0 \vee y^2 + 1 < 0) \wedge x - 3 \leq 0 \wedge xy + 1 < 0$
Boolean abstraction
$(a \vee b) \wedge c \wedge d \wedge (\neg b)$
SAT-solver
$c\ d\ b\ a$
add $y = 0$
Theory solver
(virtual substitution)
$xy + 1 < 0$
$\wedge\ x - 3 \leq 0$
$\wedge\ y = 0$
Theory solver
state A

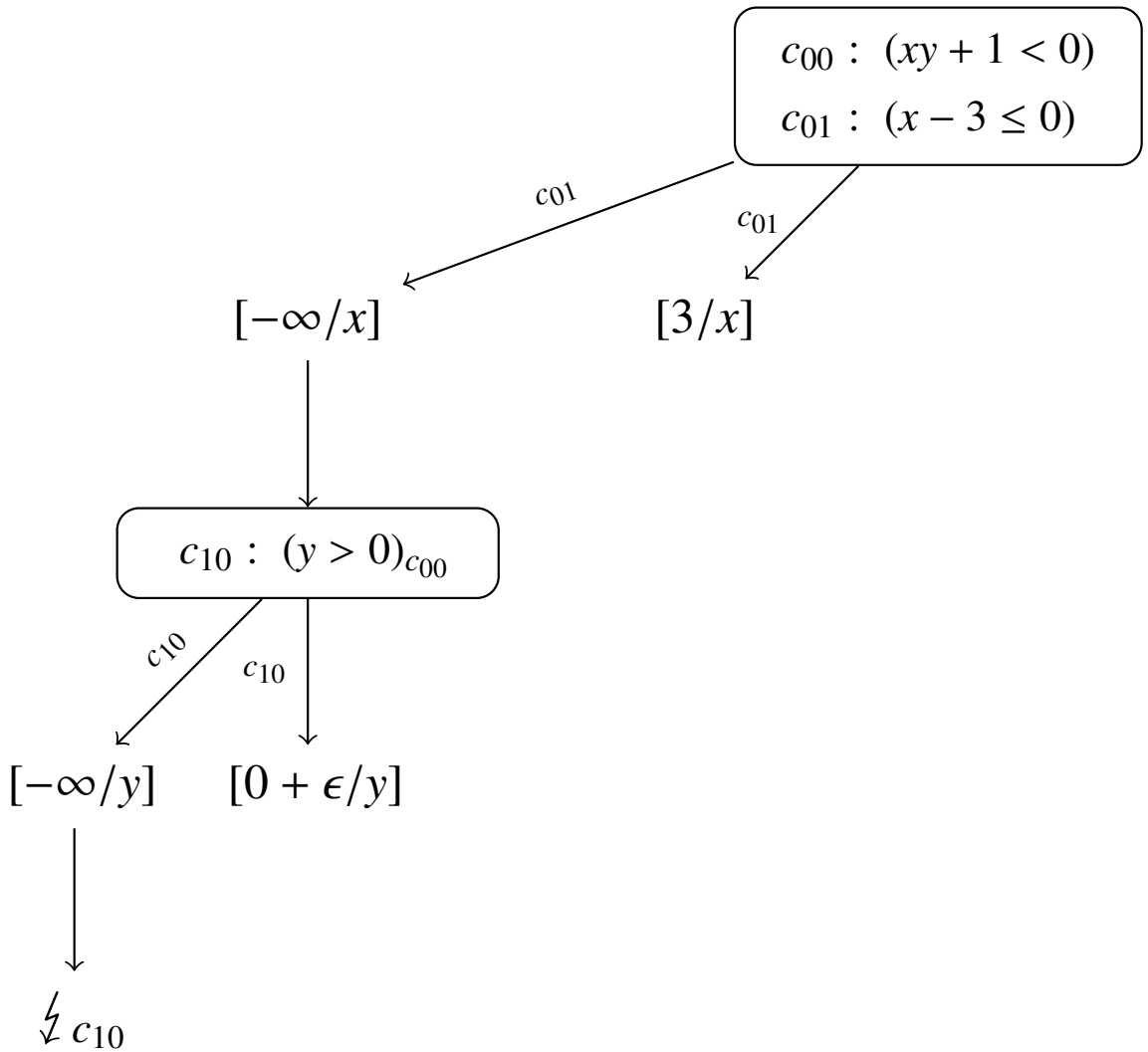
SMT solver


$\exists x, y: \ (\ y = 0 \ \vee \ y^2 + 1 < 0\ ) \ \wedge \ x - 3 \leq 0 \ \wedge \ xy + 1 < 0$
Boolean abstraction
$(\ a \ \vee \ b \ ) \ \wedge \ c \ \wedge \ d \ \wedge \ (\neg b)$
SAT-solver
$c\ d\ b\ a$
check consistency
answer: inconsistent
reason: $xy + 1 < 0 \ \wedge \ y = 0$
Theory solver
(virtual substitution)
$xy + 1 < 0$
$\wedge \quad x - 3 \leq 0$
$\wedge \quad y = 0$
consistency
check
Theory solver
state C

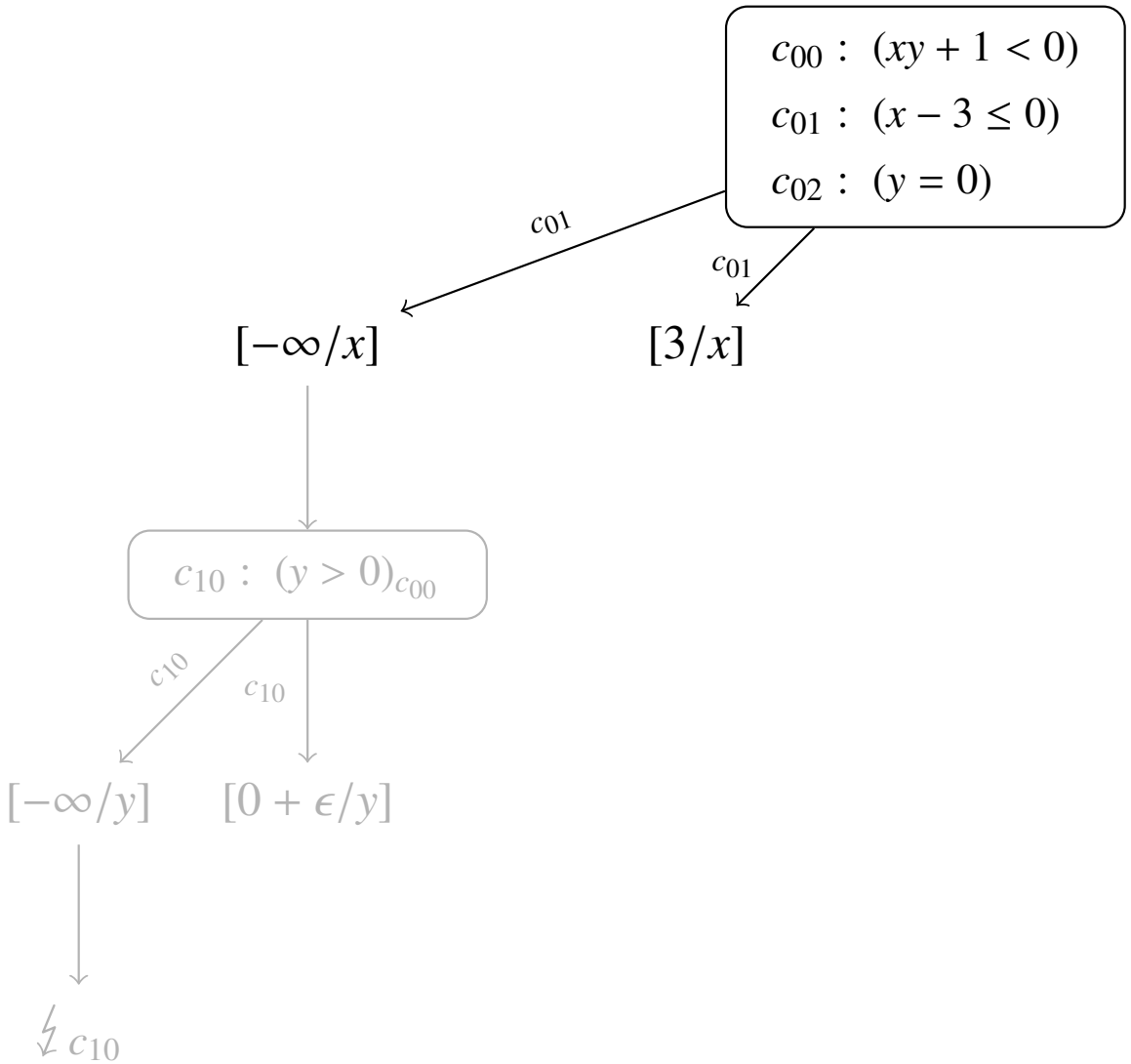
SMT solver


$\exists x, y : \ (\ y = 0 \ \vee \ y^2 + 1 < 0 \ ) \ \wedge \ x - 3 \leq 0 \ \wedge \ xy + 1 < 0$
Boolean abstraction
$(\quad a \quad \vee \quad b \quad) \ \wedge \quad c \quad \wedge \quad d \quad \wedge \quad (\neg b)$
SAT-solver
UNSAT
Theory solver
(virtual substitution)
$xy + 1 < 0$
$\wedge \quad x - 3 \leq 0$
$\wedge \quad y = 0$
Theory solver
state C

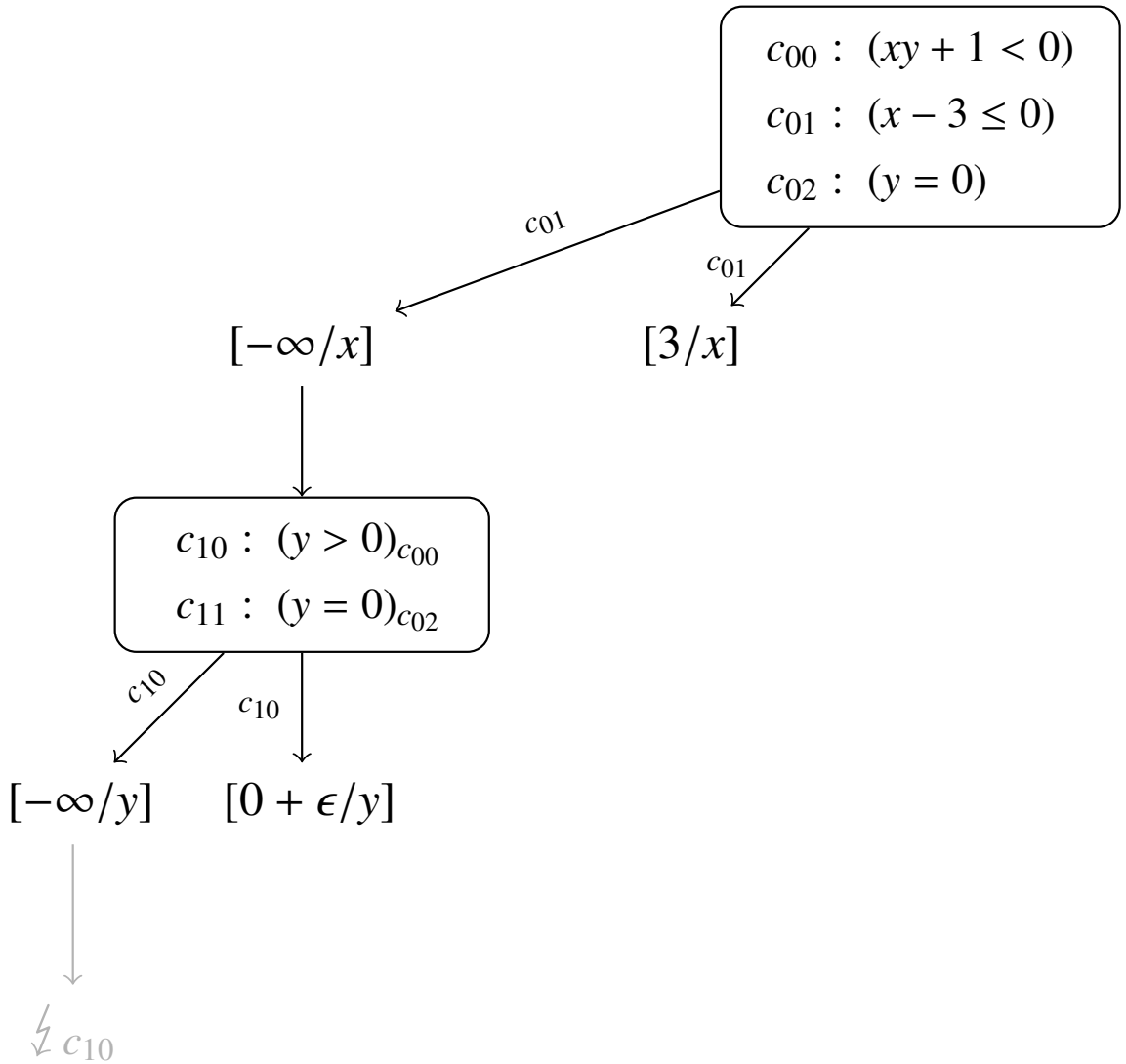
Theory Solver: add $y = 0$, check consistency



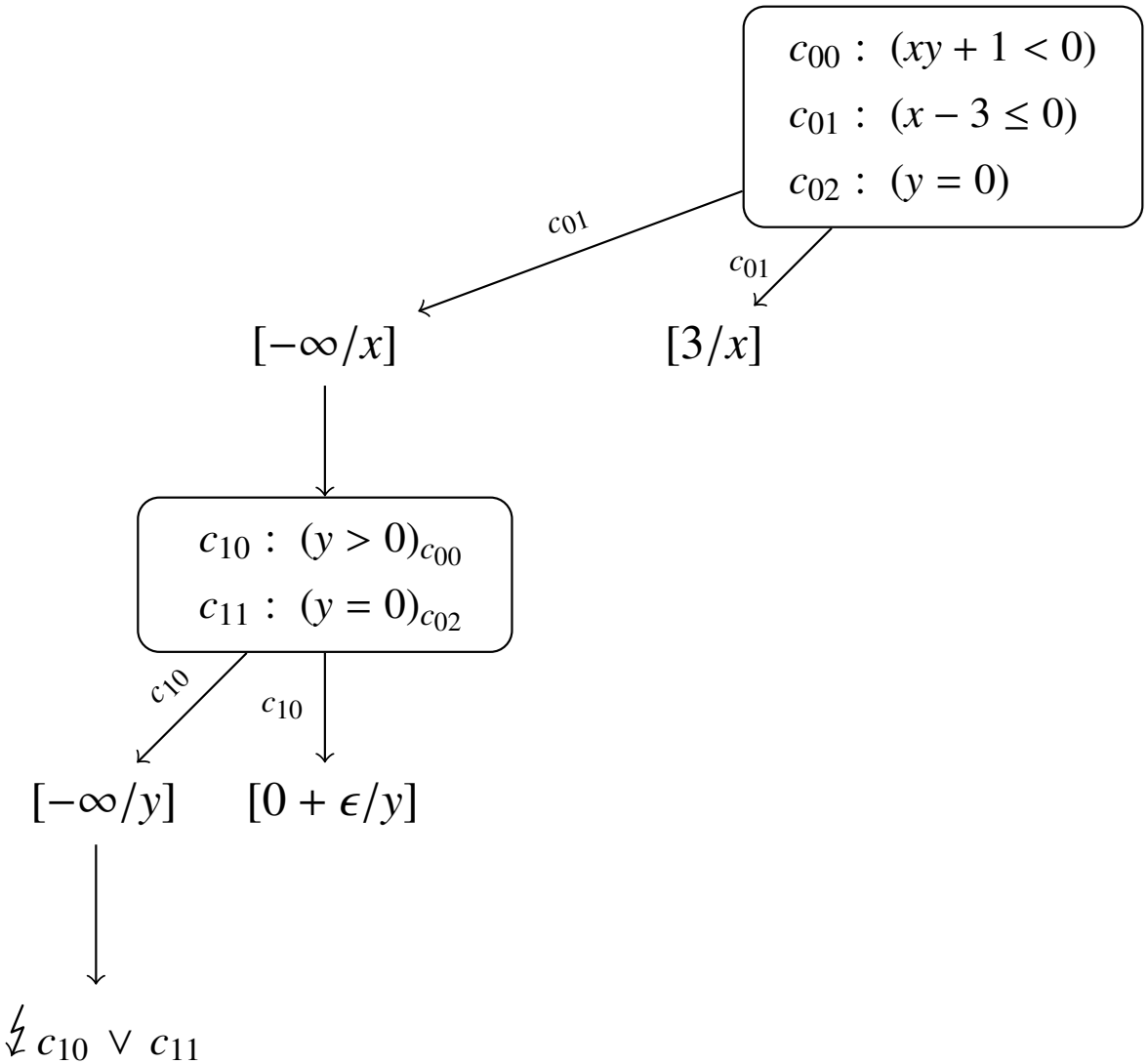
Theory Solver: add $y = 0$, check consistency



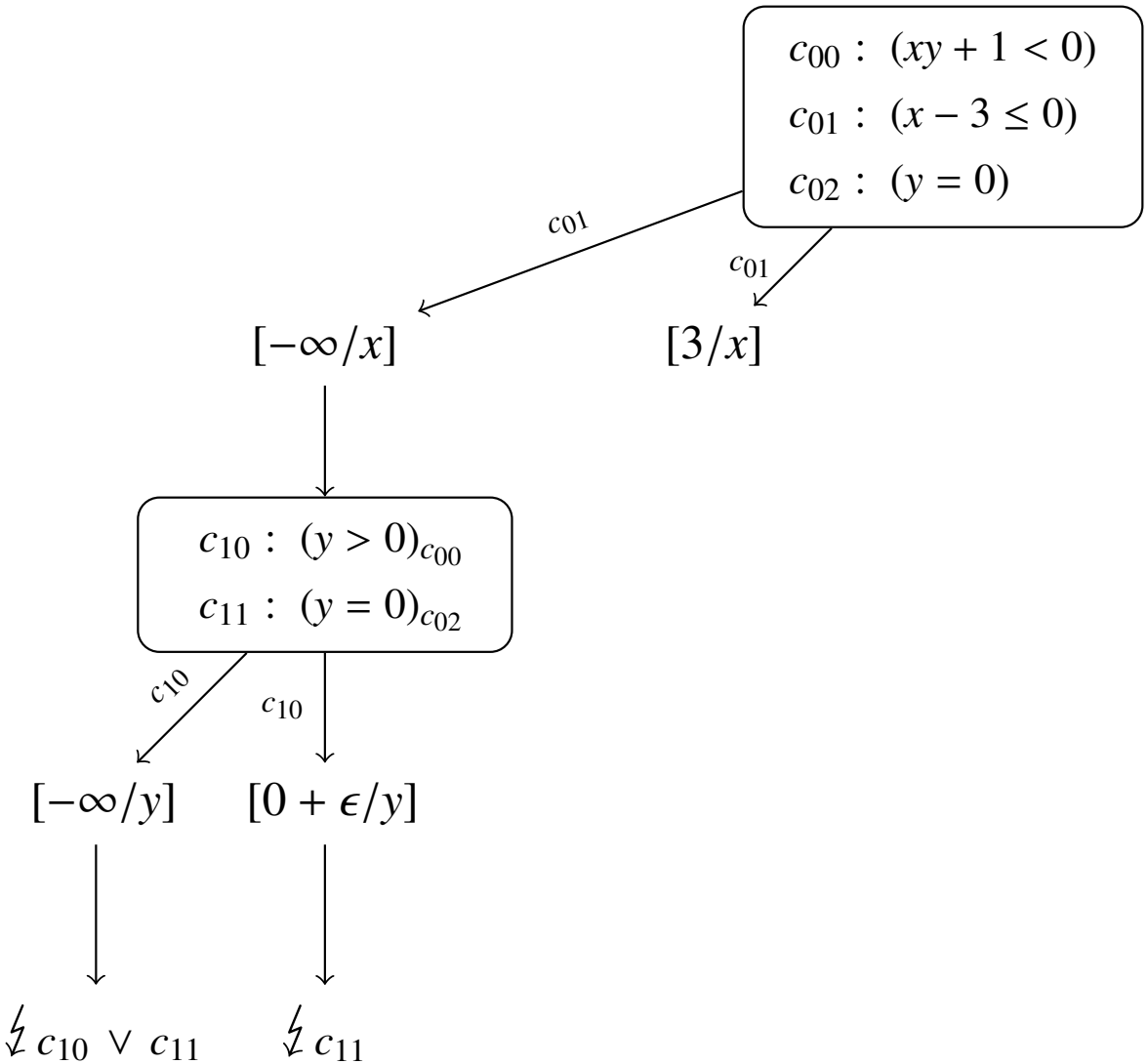
Theory Solver: add $y = 0$, check consistency



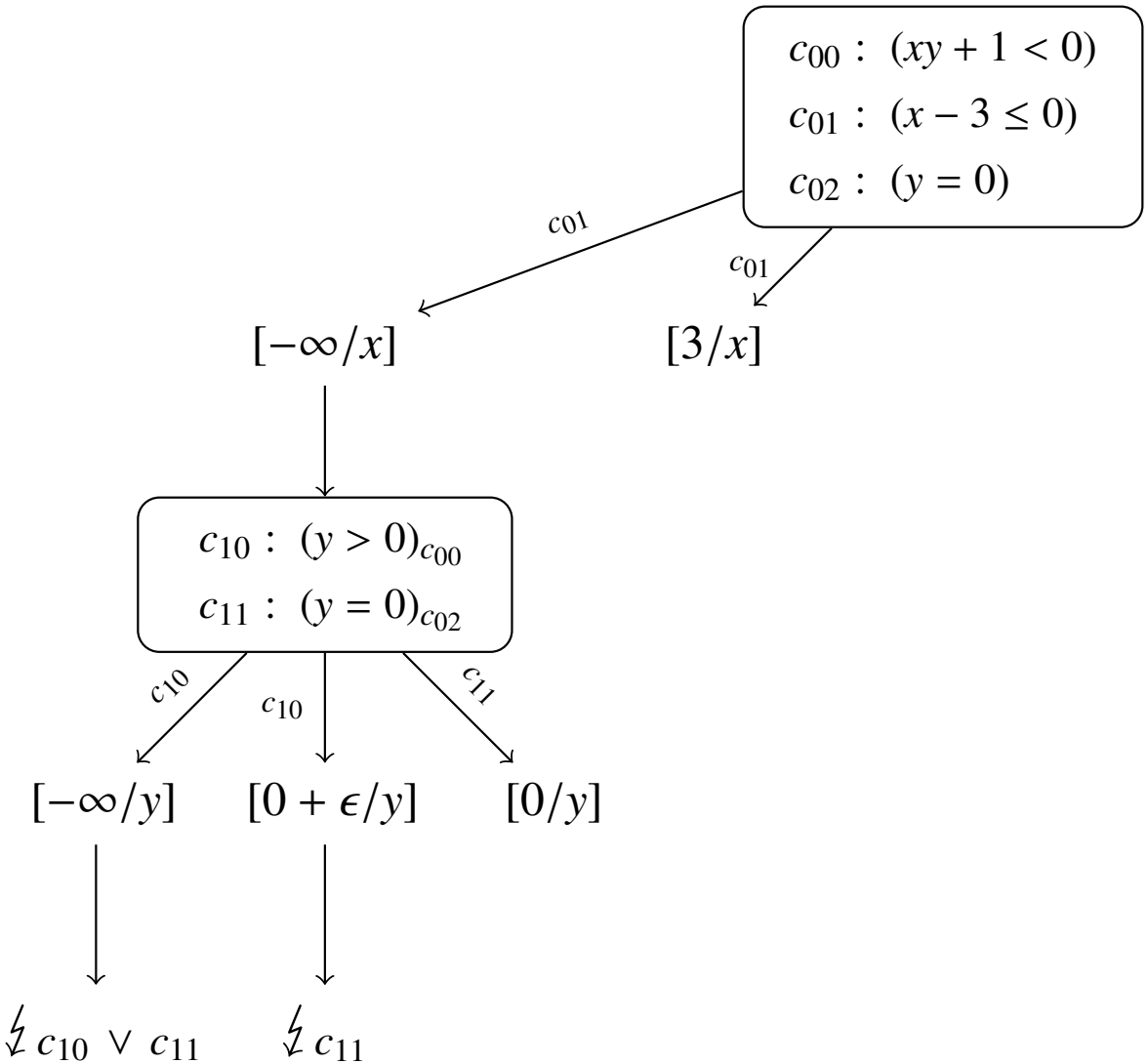
Theory Solver: add $y = 0$, check consistency



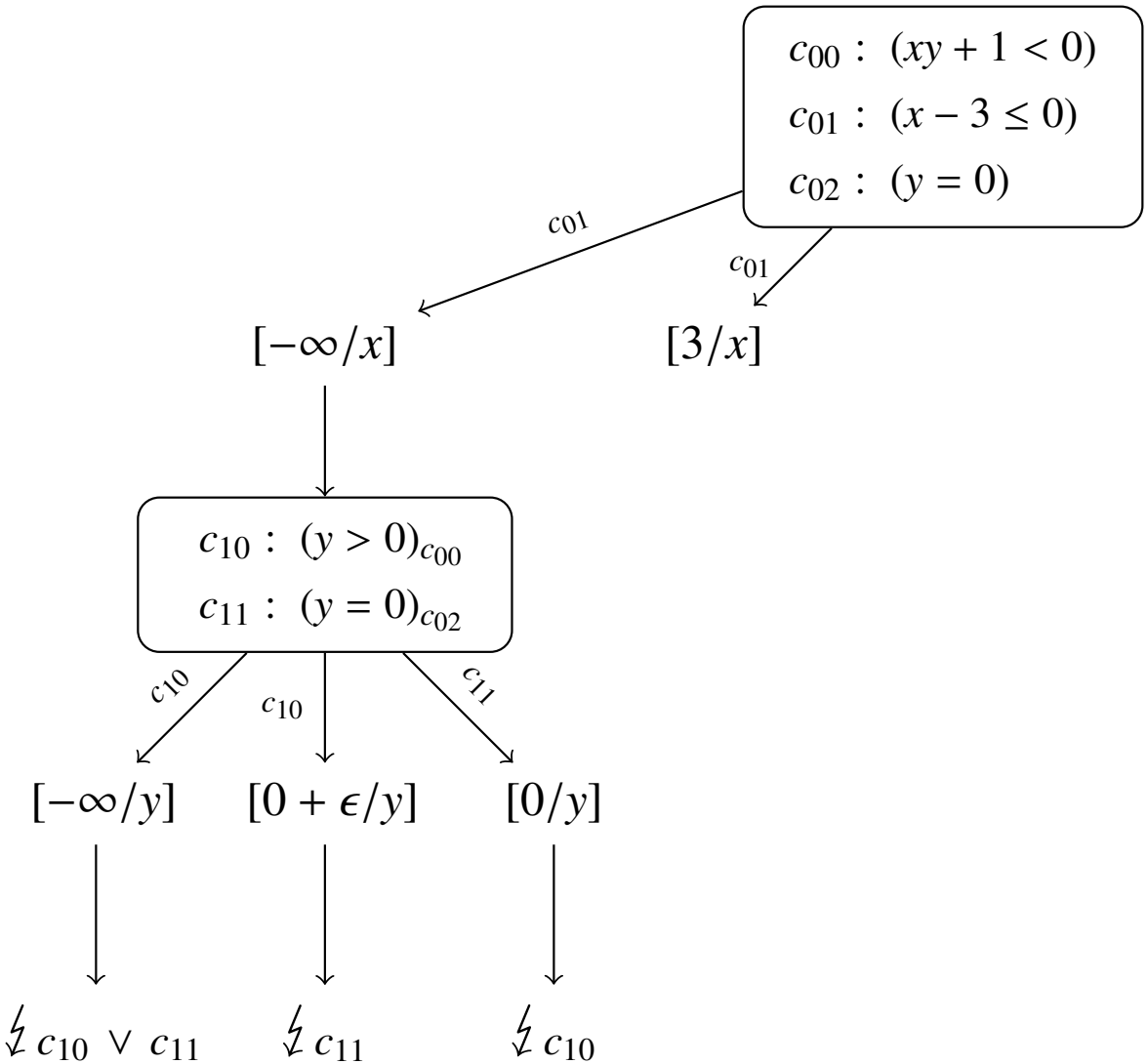
Theory Solver: add $y = 0$, check consistency



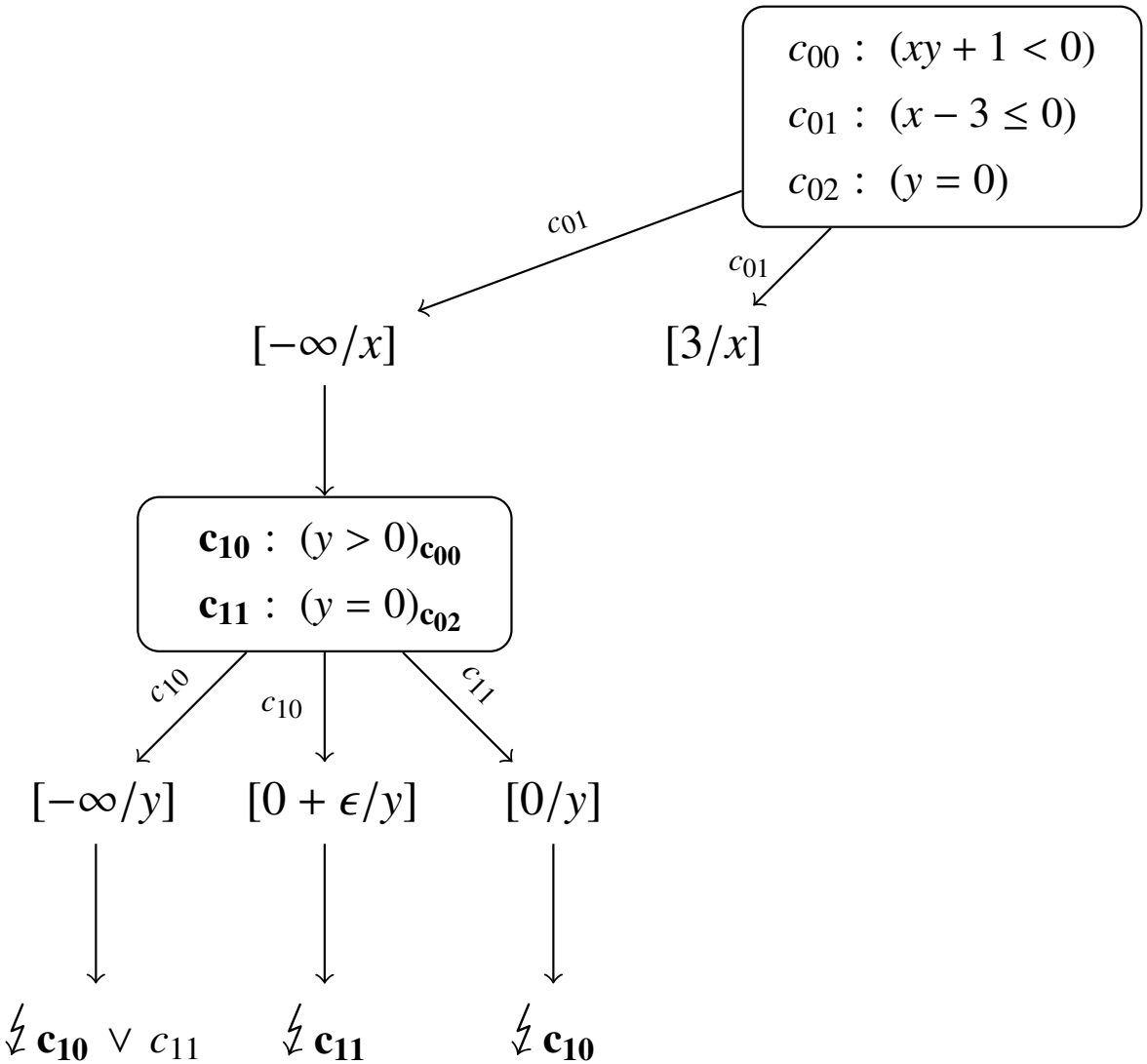
Theory Solver: add $y = 0$, check consistency



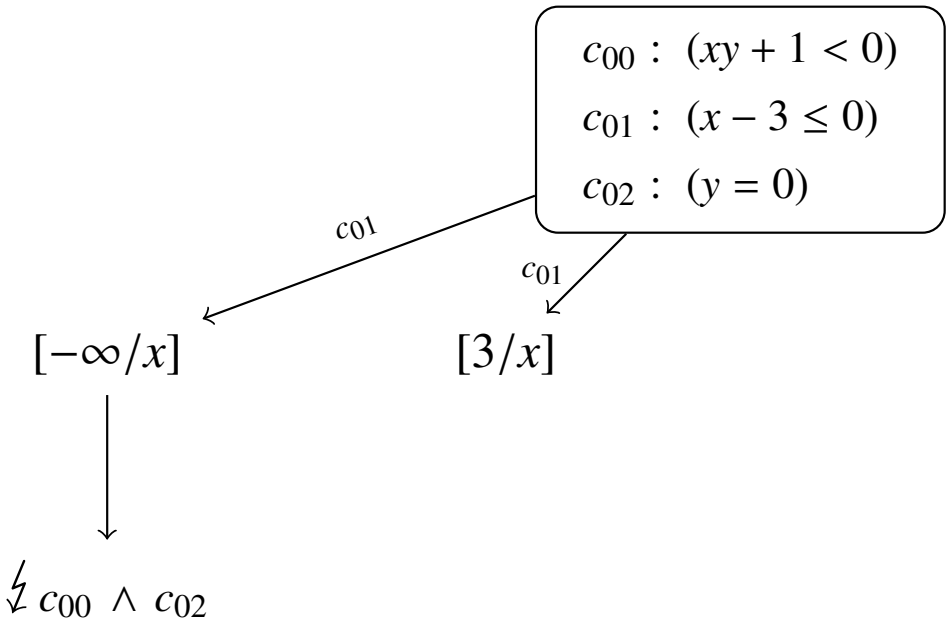
Theory Solver: add $y = 0$, check consistency



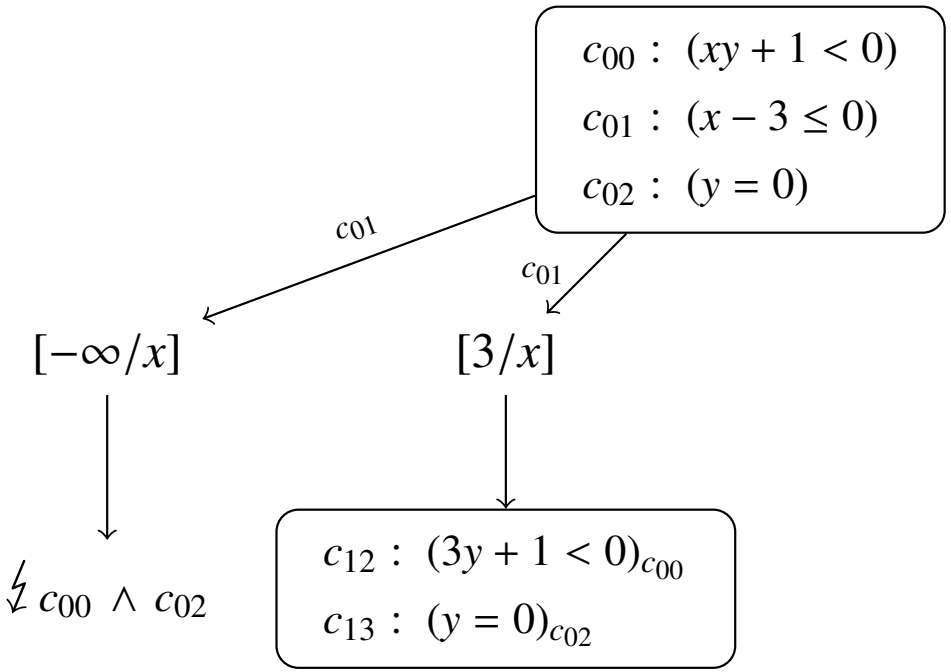
Theory Solver: add $y = 0$, check consistency



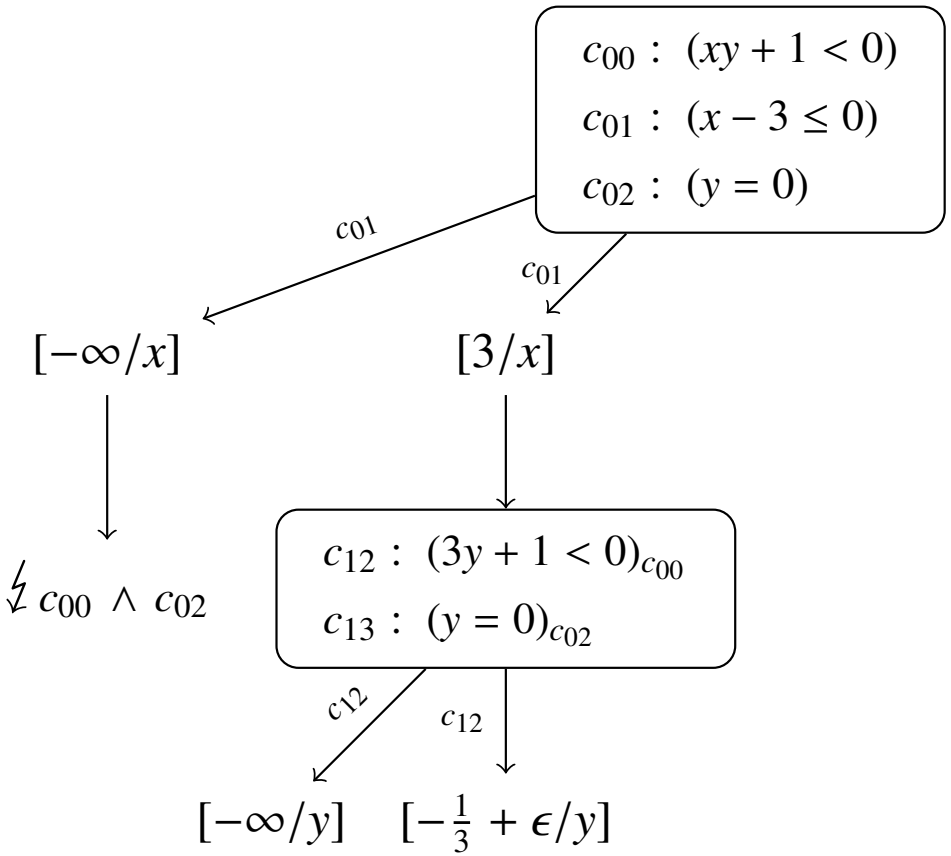
Theory Solver: add $y = 0$, check consistency



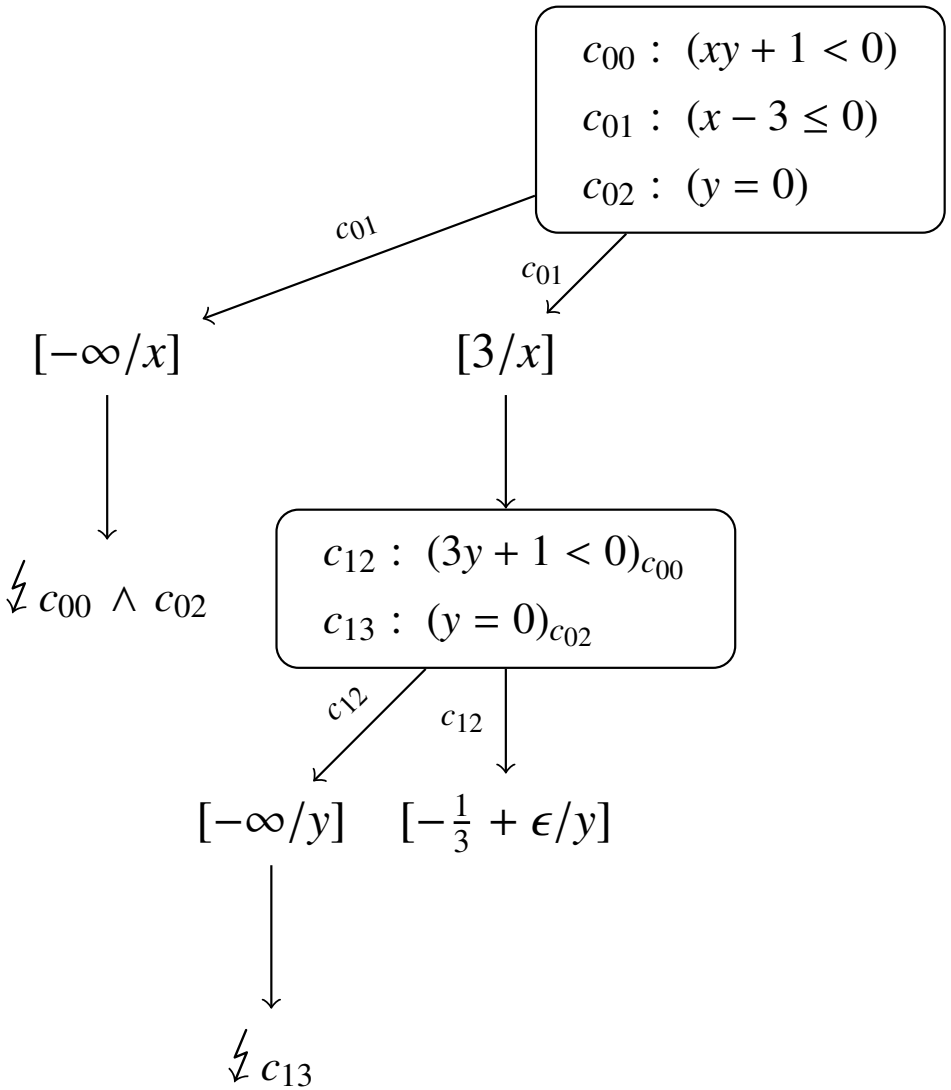
Theory Solver: add $y = 0$, check consistency



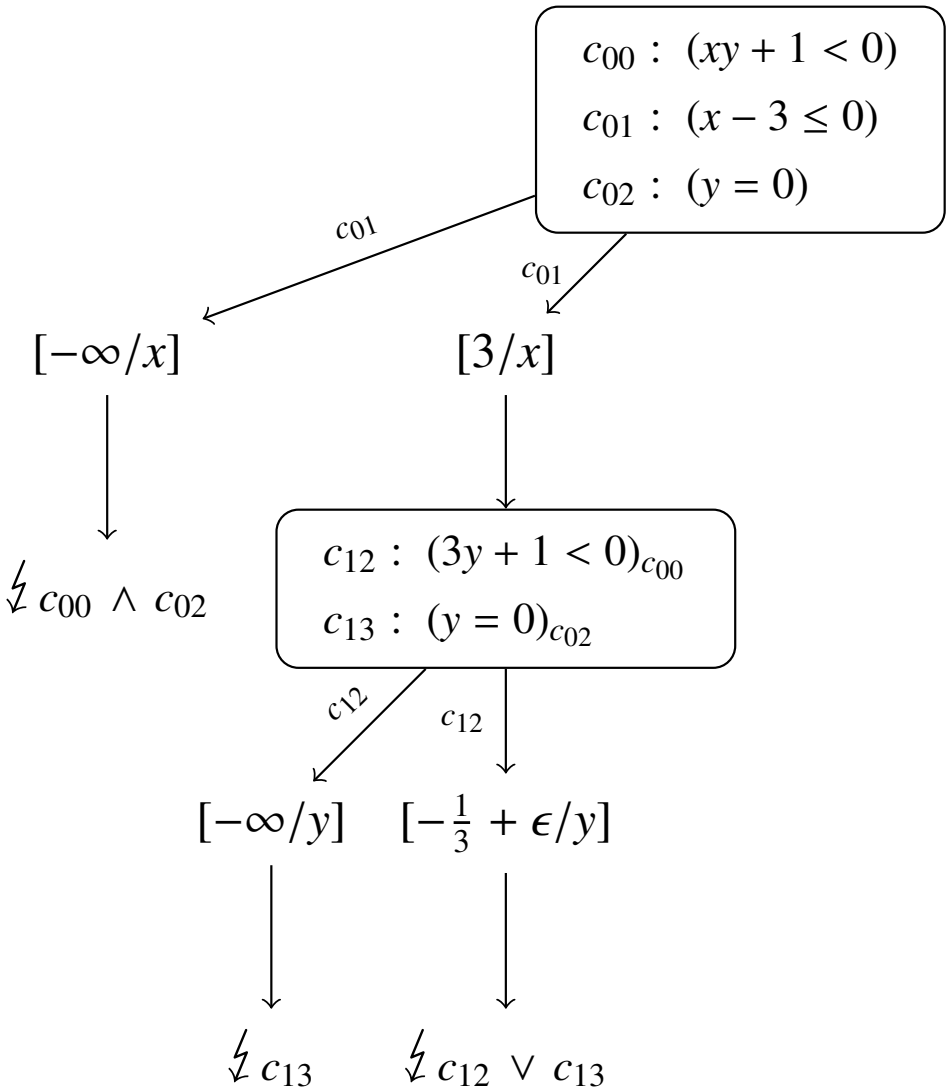
Theory Solver: add $y = 0$, check consistency



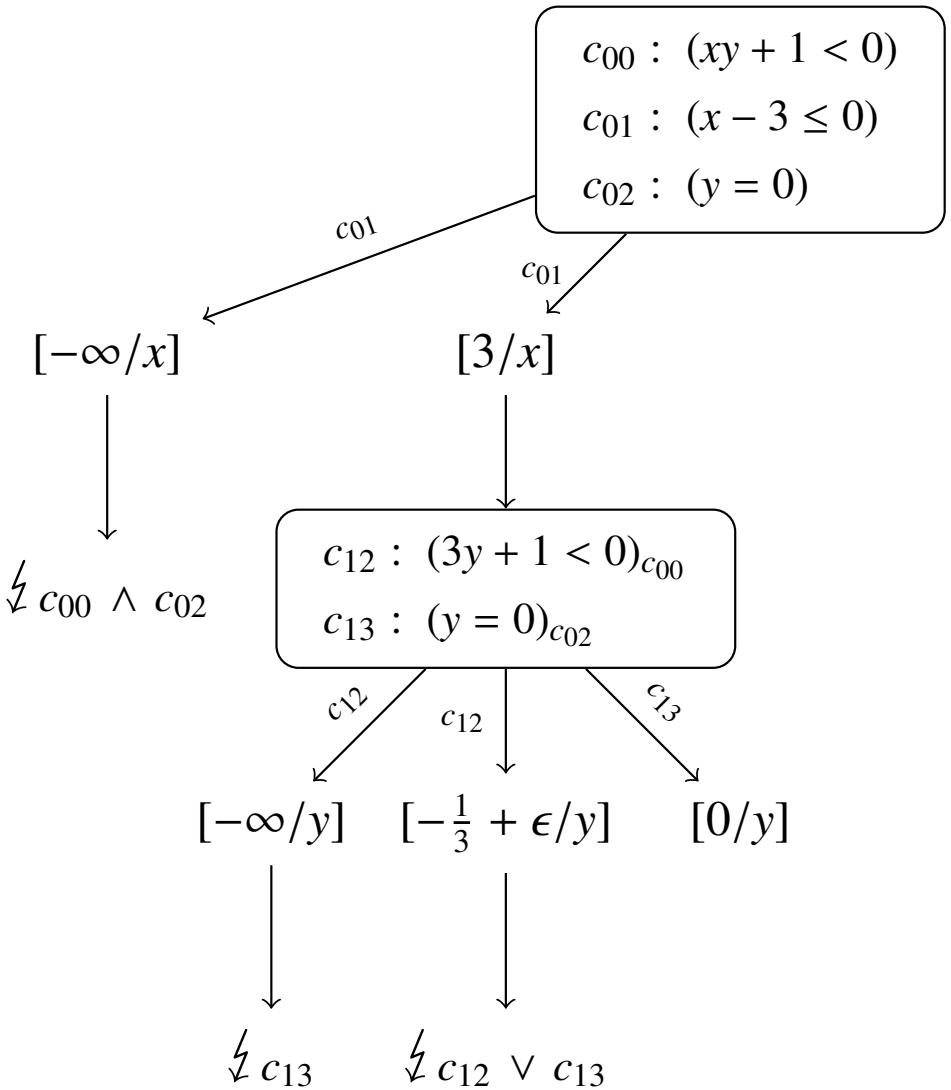
Theory Solver: add $y = 0$, check consistency



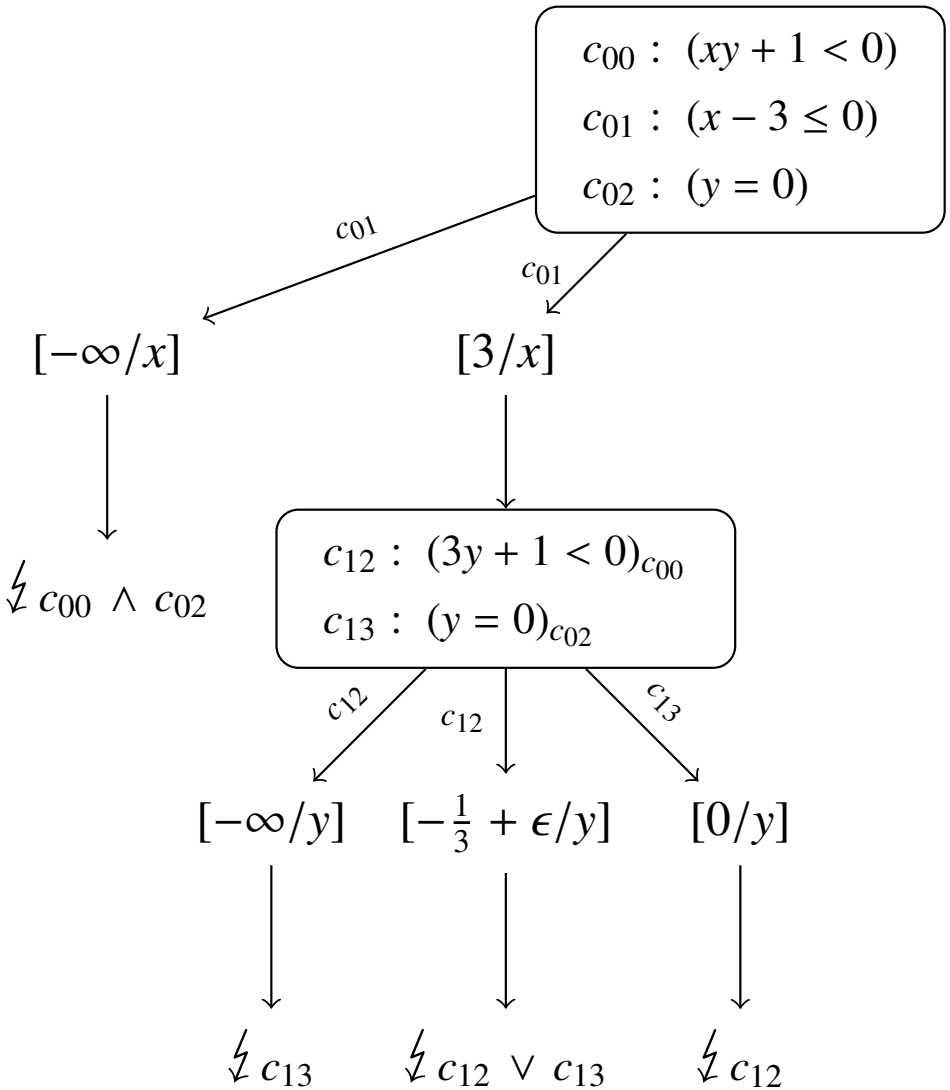
Theory Solver: add $y = 0$, check consistency



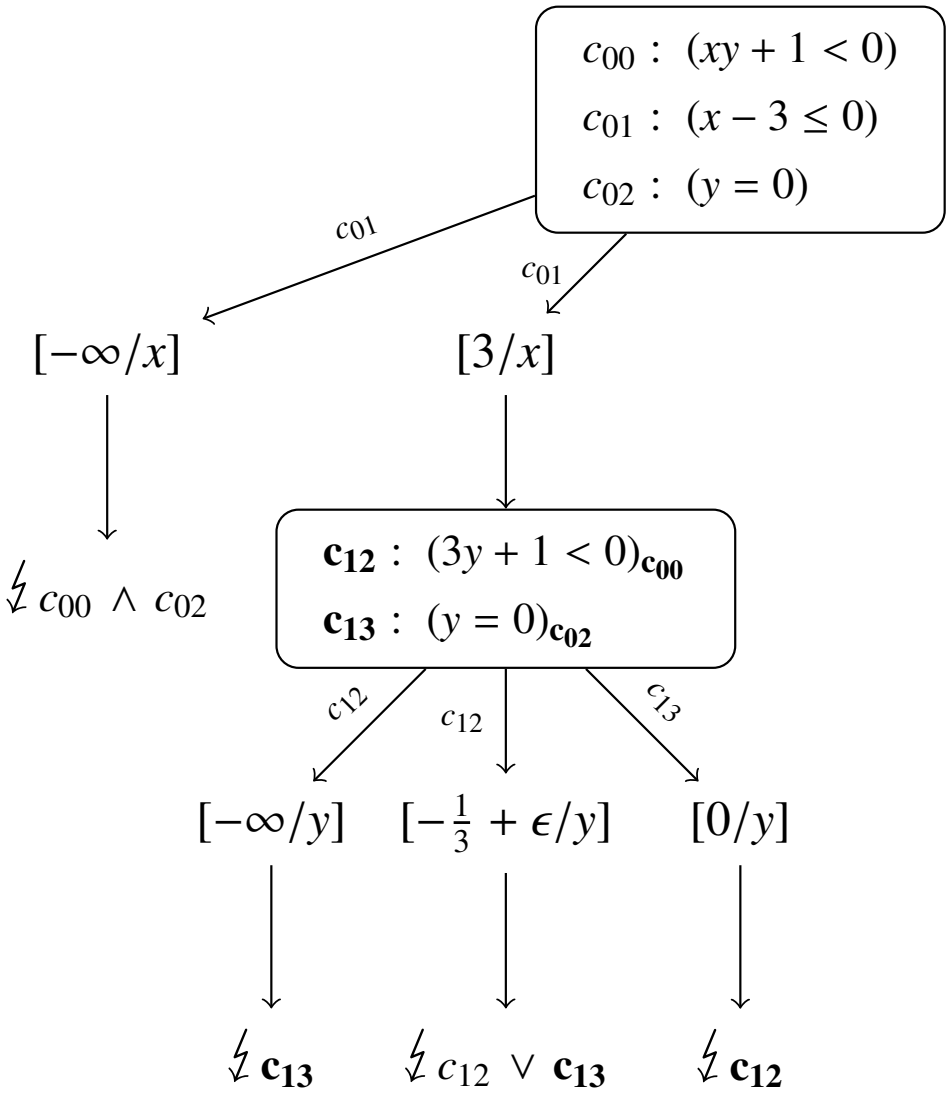
Theory Solver: add $y = 0$, check consistency



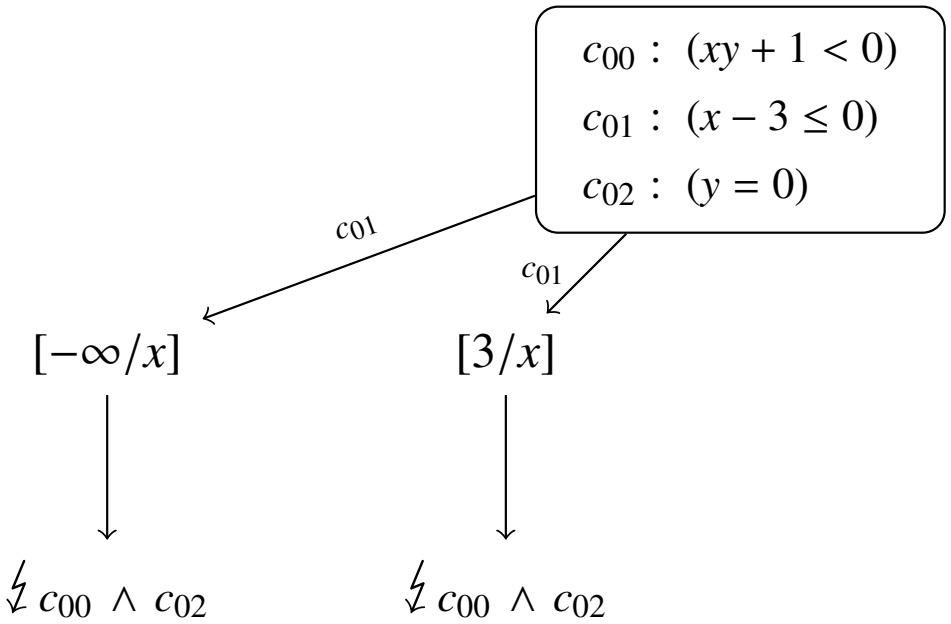
Theory Solver: add $y = 0$, check consistency



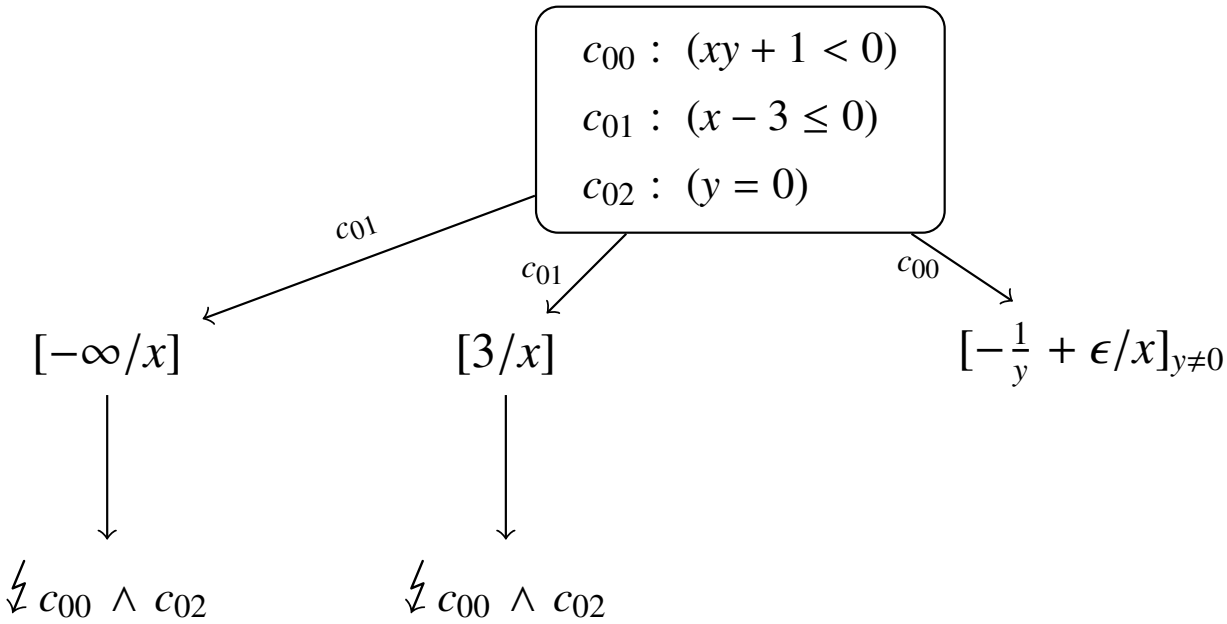
Theory Solver: add $y = 0$, check consistency



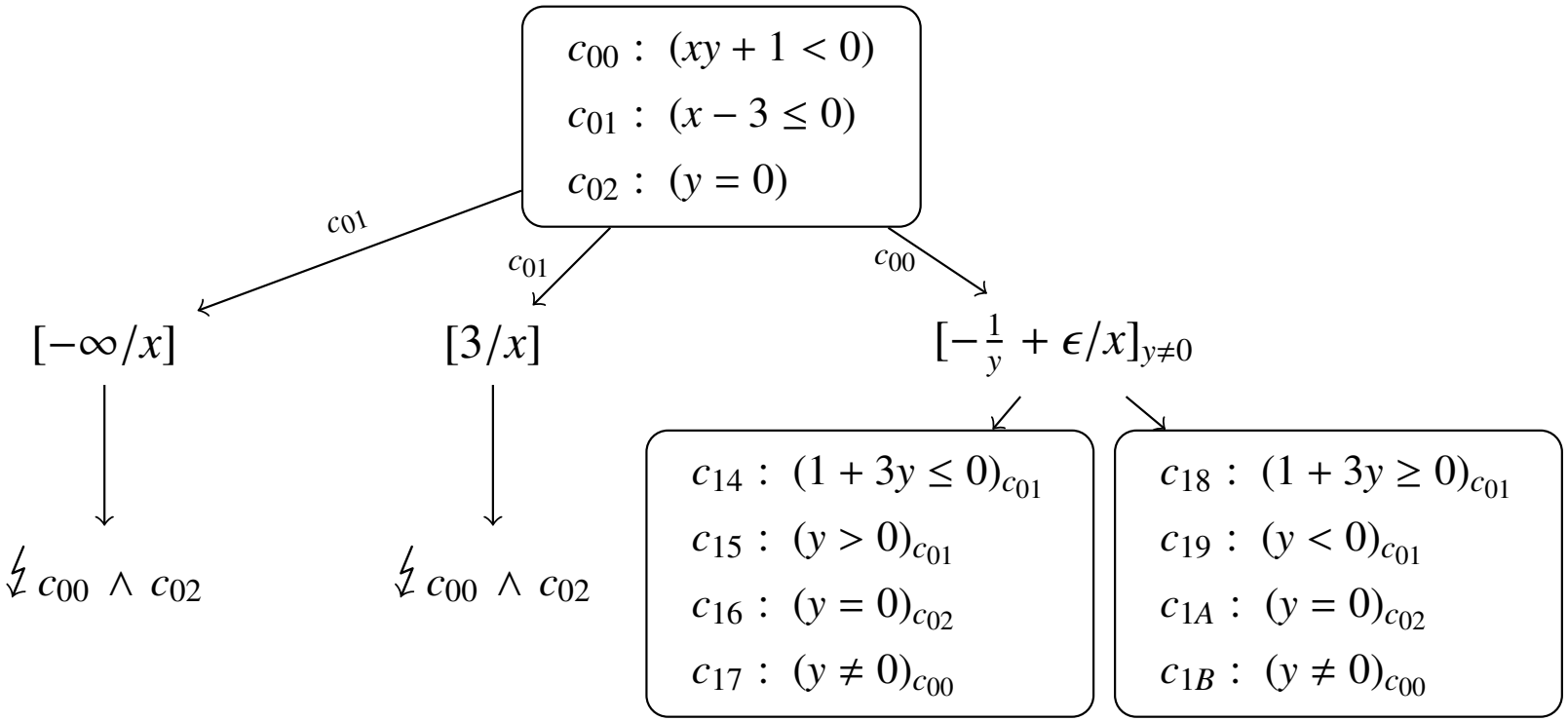
Theory Solver: add $y = 0$, check consistency



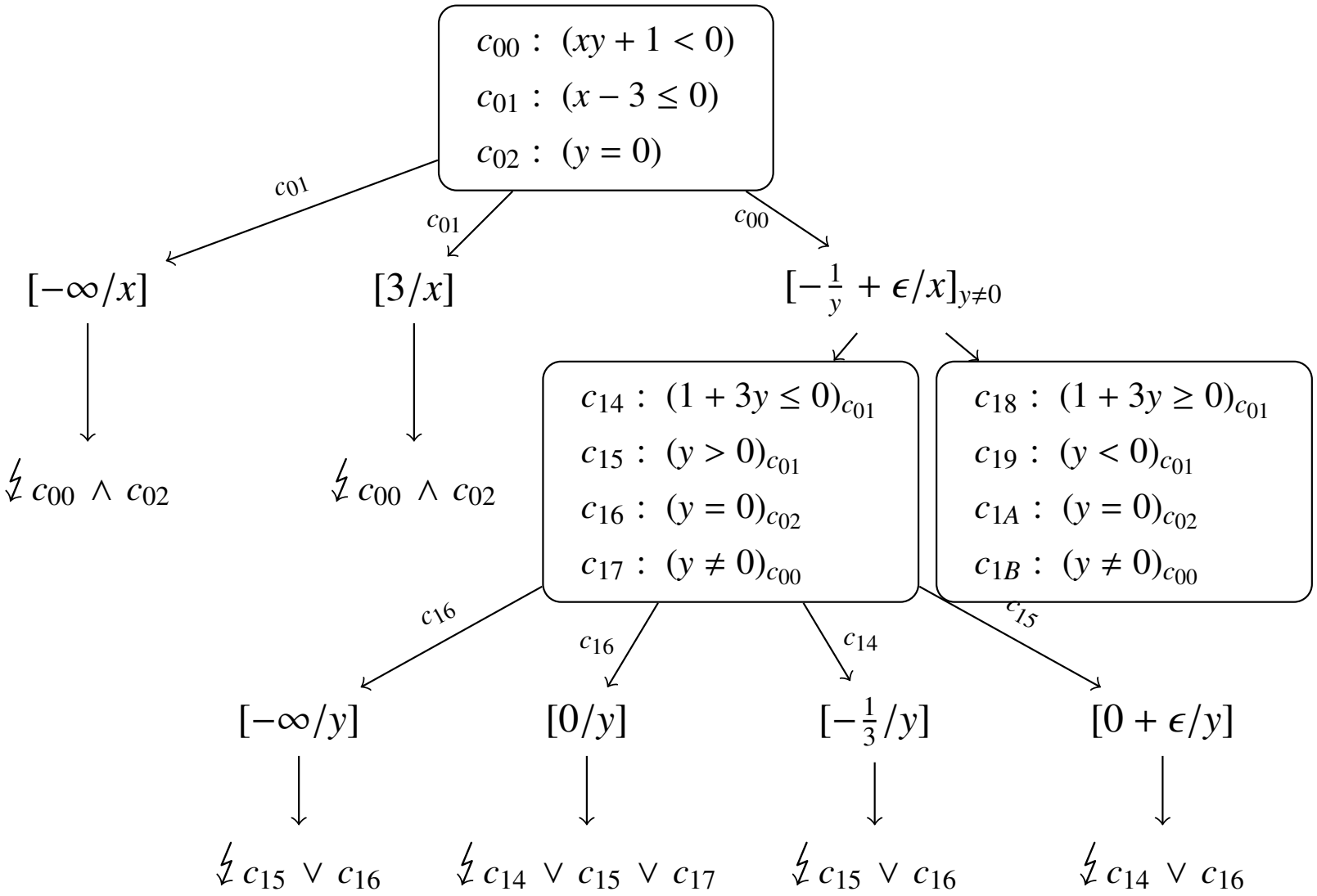
Theory Solver: add $y = 0$, check consistency



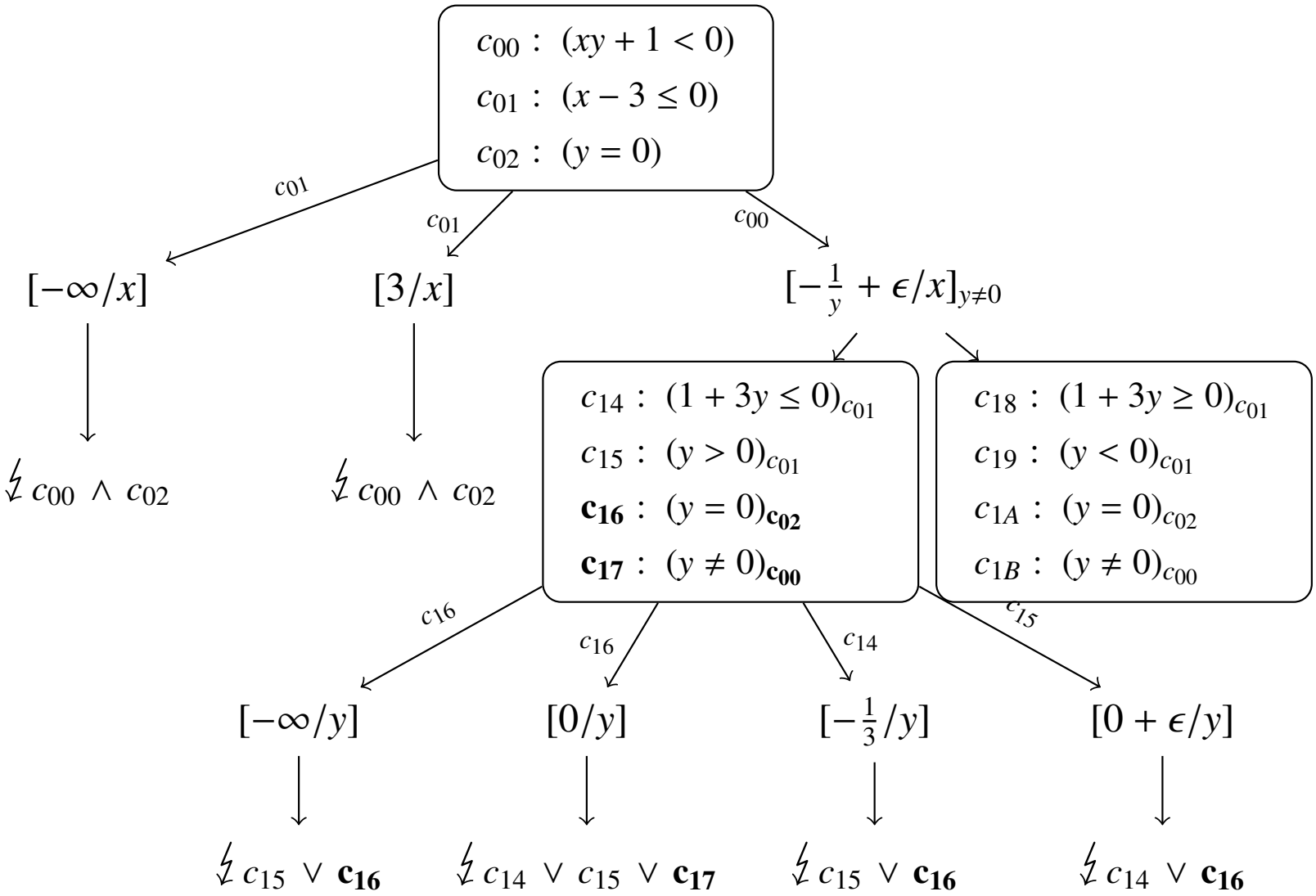
Theory Solver: add $y = 0$, check consistency



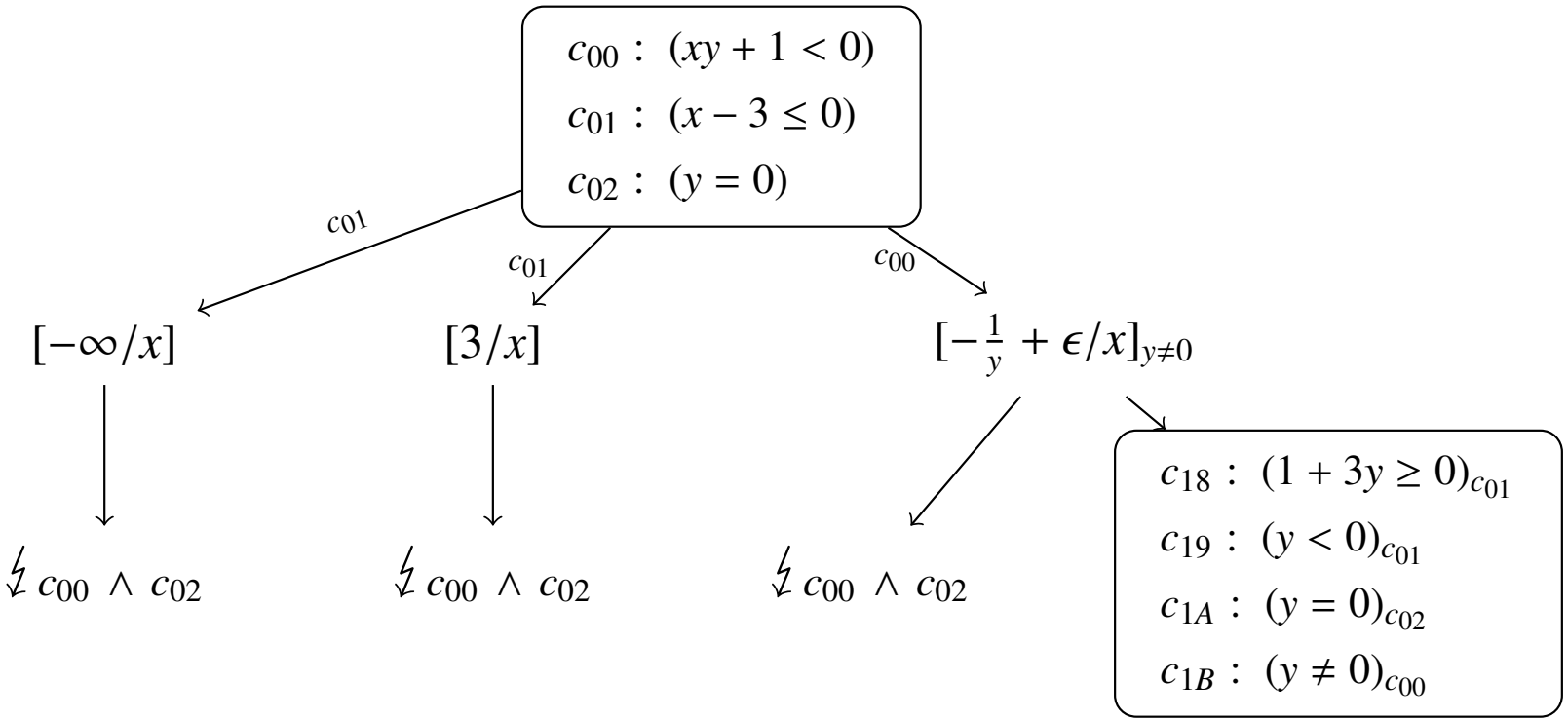
Theory Solver: add $y = 0$, check consistency



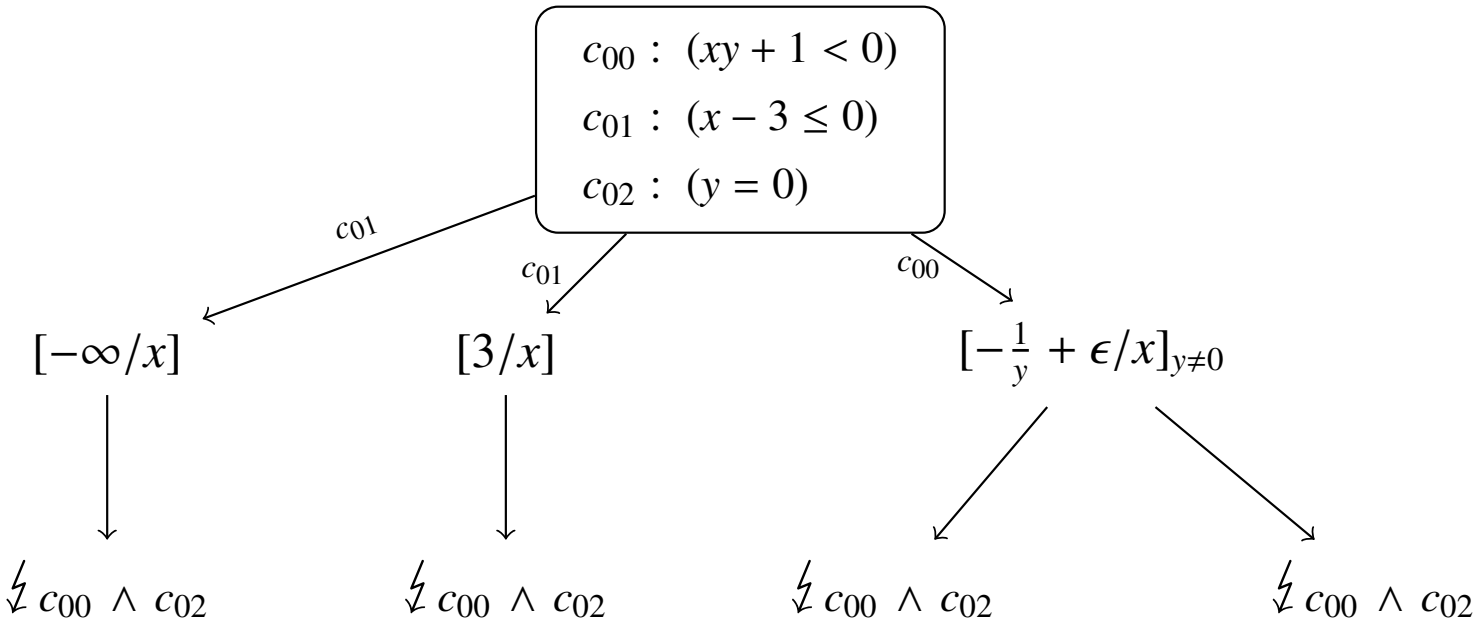
Theory Solver: add $y = 0$, check consistency



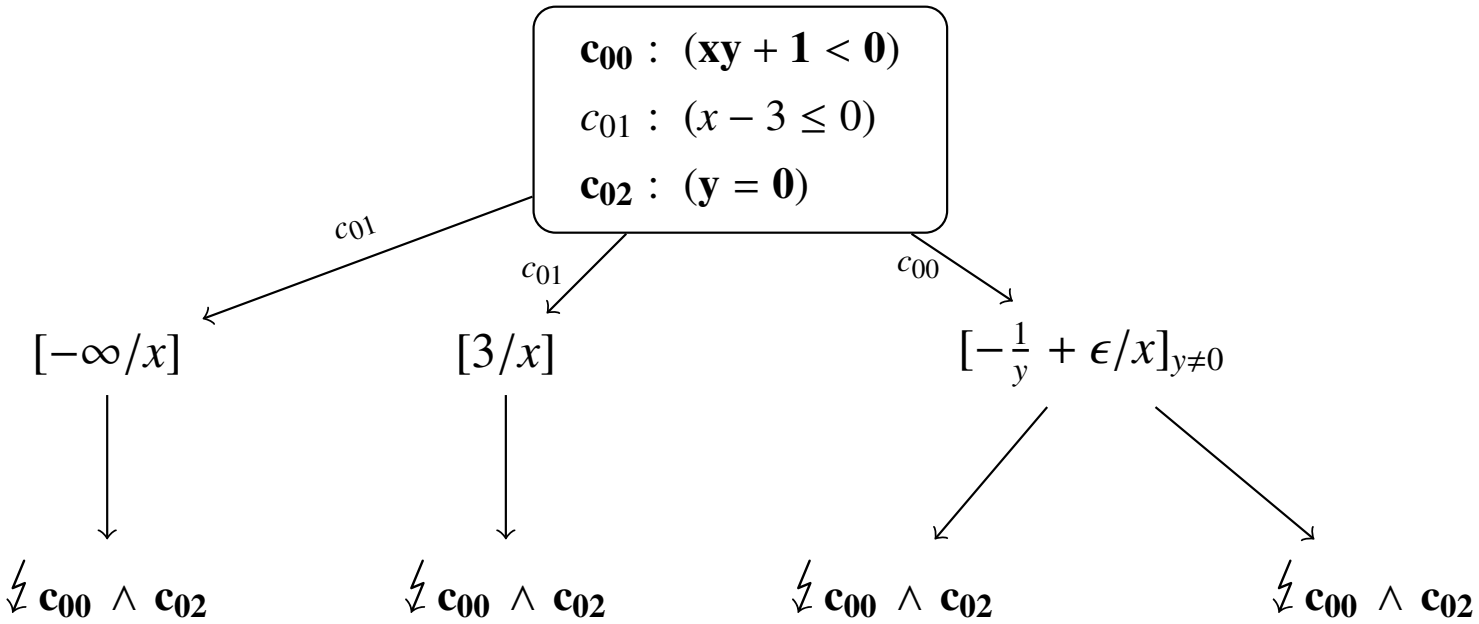
Theory Solver: add $y = 0$, check consistency



Theory Solver: add $y = 0$, check consistency

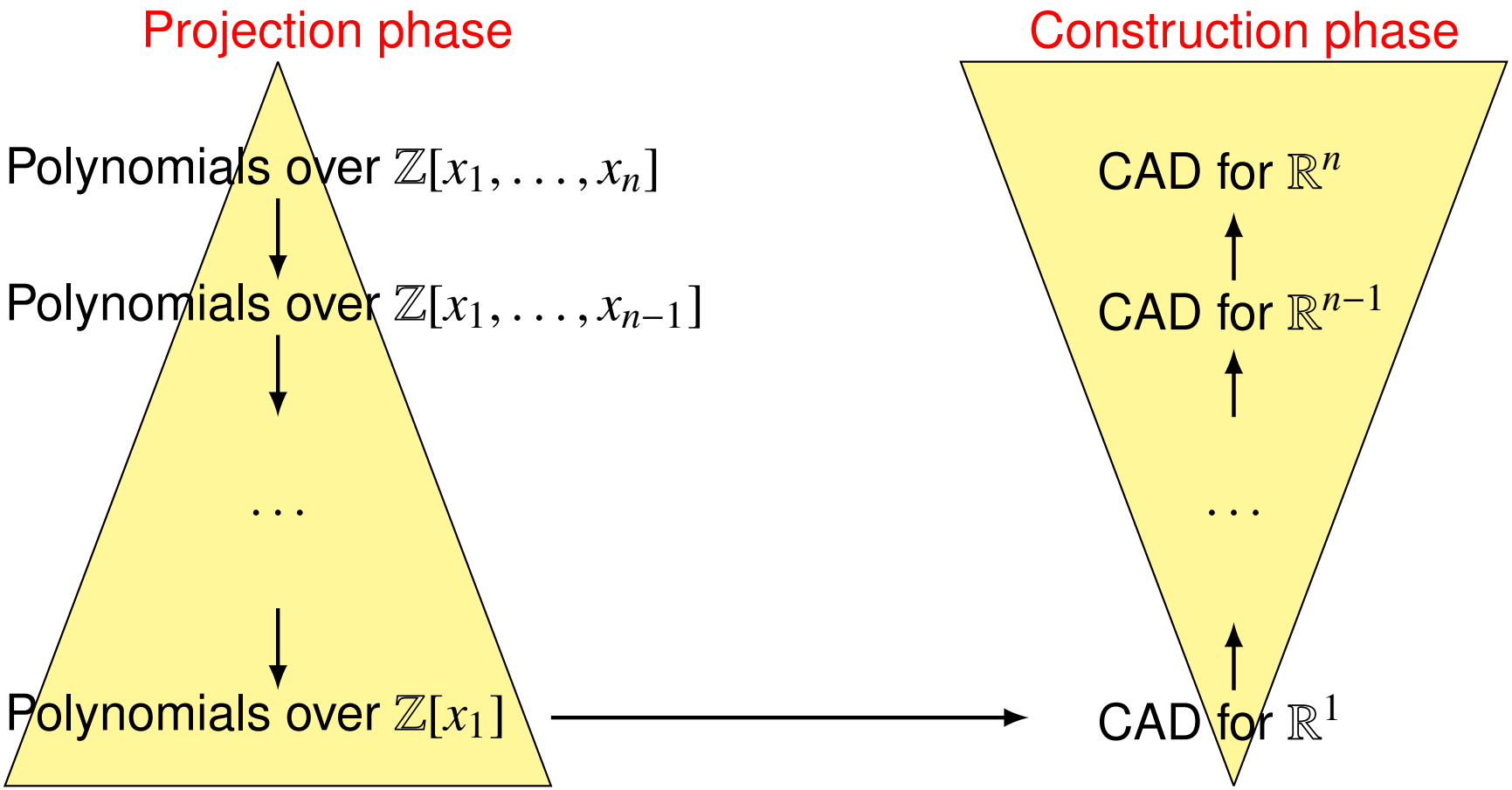


Theory Solver: add $y = 0$, check consistency



CAD for $\mathbb{R}^n$

A CAD for a set of polynomials from $\mathbb{Z}[x_1, \ldots, x_n]$ splits $\mathbb{R}^n$ into sign-invariant regions.



Some experimental results

We compare:

- `Z3` (SMT solver, Microsoft)
- `redlog` (reference implementation of virtual substitution in `Reduce`)
- SMT-RAT with two strategies.

$\texttt{rat}_1$: $\texttt{CNF} \to \texttt{Preproc} \to \texttt{SAT} \longrightarrow \texttt{ICP} \longrightarrow \texttt{VirtualSub} \to \texttt{CAD}$

$\texttt{rat}_2$: $\texttt{CNF} \to \texttt{Preproc} \overset{\to}{\underset{\to}{}} \begin{array}{l} \texttt{SAT} \longrightarrow \texttt{ICP} \longrightarrow \texttt{VirtualSub} \to \texttt{CAD} \\ \texttt{SAT} \to \texttt{Simplex} \to \texttt{VirtualSub} \to \texttt{CAD} \end{array}$

Some experimental results

Benchmark (#examples)	z3 #	z3 time	redlog #	redlog time	rat_1 #	rat_1 time	rat_2 #	rat_2 time
Hong (20)	40.0%	5.6	30.0%	3.7	100.0%	< 1	100.0%	< 1
- sat	0	0	0	0	0	0	0	0
- unsat	8	3.7	6	5.6	20	< 1	20	< 1
Kissing (45)	68.9%	1248.7	13.3%	3.3	35.6%	375.9	28.9%	54.4
- sat	31	1248.7	6	3.3	16	375.9	13	54.4
- unsat	0	0	0	0	0	0	0	0
Meti-Tarski (7713)	99.9%	405.6	96.6%	11617.9	92.8%	4658.3	95.6%	3109.4
- sat	5025	140.8	4859	7128.7	4740	2952.1	4815	2290.4
- unsat	2681	264.8	2590	4489.2	2418	1706.2	2560	819
Zankl (166)	53.0%	267.6	22.3%	178.0	25.9%	217.4	25.9%	101.3
- sat	61	266.3	27	156.0	27	216.8	26	80.4
- unsat	27	1.3	10	22.0	16	< 1	17	20.9
Keymaera (421)	99.8%	11.8	99.5%	209.3	96.9%	17	98.1%	25.3
- sat	0	0	0	0	0	0	0	0
- unsat	420	11.8	419	209.3	408	17	413	25.3
Witness (99)	21.2%	153.5	5.1%	62.1	64.6%	332.2	75.8%	937.9
- sat	4	106	5	62.1	47	331.9	58	937.6
- unsat	17	47.5	0	0	17	< 1	17	< 1

Upcoming research directions in SMT solving

Improve usability:

- User-friendly models
- Dedicated SMT solvers

Increase scalability:

- Performance optimisation (better lemmas, heuristics, cache behaviour, . . .)
- Novel combination of decision procedures
- Parallelisation

Extend functionality:

- Unsatisfiable cores, proofs, interpolants
- Quantified arithmetic formulas
- Linear and non-linear (global) optimisation

Two communities, quite disjoint

	Symbolic computation	SMT solving
Research aim	Automated computation with formal objects	Automated satisfiability checking for combined theories
Arithmetic focus	Constraint sets Exact, complete	Boolean structures Exact, efficient
Tools	Computer algebra systems Large, general-purpose	SAT and SMT solvers Small, dedicated
Conf.	ACA, CASC, ISSAC, . . .	CADE, SAT, SMT, . . .
Groups	SIGSAM	SMT-LIB, SAT Live!
Journals	J. Symb. Comput., AAECC, ACM Comm. in Computer Algebra (SIGSAM Bulletin)	J. on Satisfiability, Boolean Modeling and Computation

A forthcoming Dagstuhl Seminar

Conclusion: More interactions desirable

Future Directions for Research in Symbolic Computation

"More interactions are desirable between mathematicians and computer scientists interested in symbolic computation, between researchers in numerical and symbolic computing, and between software builders and users."

Report of a Workshop on Symbolic and Algebraic Computation, 1988

Ann Boyle and B. F. Caviness, Editors; Anthony C. Hearn, Workshop Chairperson