

SAT Solving

Dr. Nils Froleyks

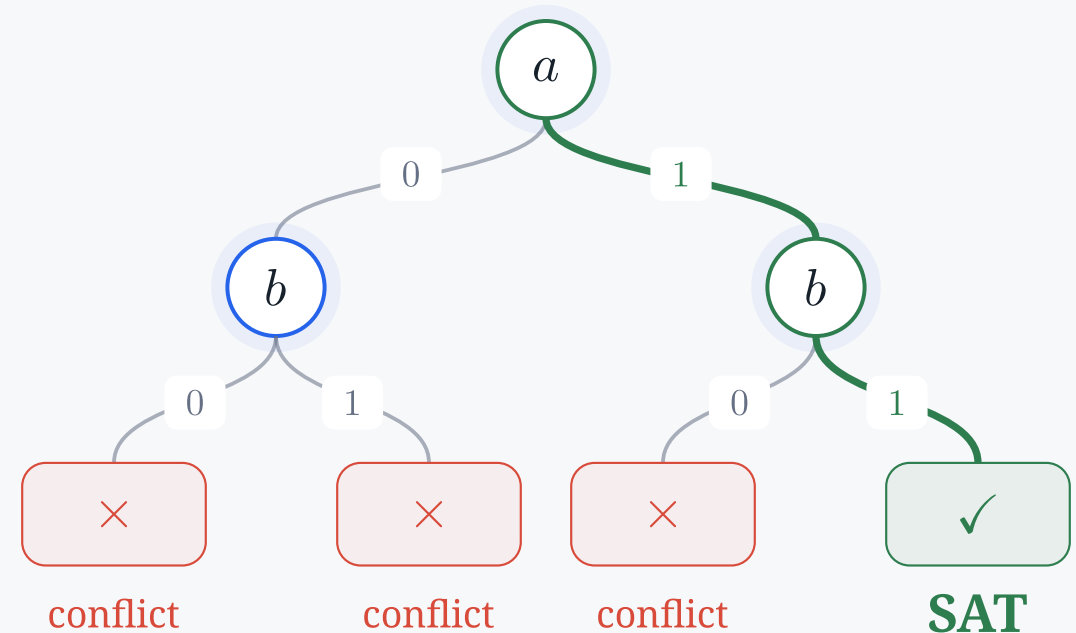
KU LEUVEN

30 September 2026

“The SAT problem is evidently a killer app, because it is key to the solution of so many other problems.”

— Donald E. Knuth

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



$$a = 1, \quad b = 1$$

Thinking, Fast and Slow



System 1

Thinking fast

Automatic and intuitive

LLMs

Generate a candidate response

Context



Candidate

Patterns learned from examples

System 2

Thinking slow

Deliberate and effortful

SAT solving

Check explicit logical constraints

Formula



SAT / UNSAT

A precise logical decision

Thinking, Fast and Slow

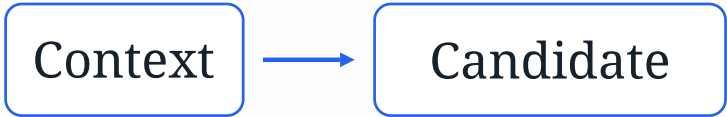
		3		2		6		
9			3		5			1
		1	8		6	4		
		8	1		2	9		
7								8
		6	7		8	2		
		2	6		9	5		
8			2		3			9
		5		1		3		

System 1

Thinking fast
Automatic and intuitive

LLMs 166.086 s

Generate a candidate response



Patterns learned from examples

System 2

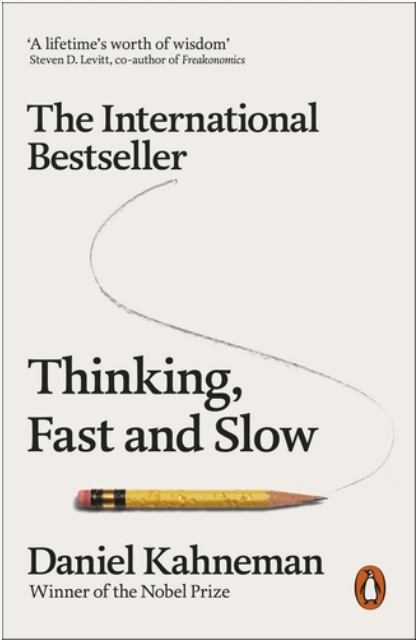
Thinking slow
Deliberate and effortful

SAT solving 1.725 ms

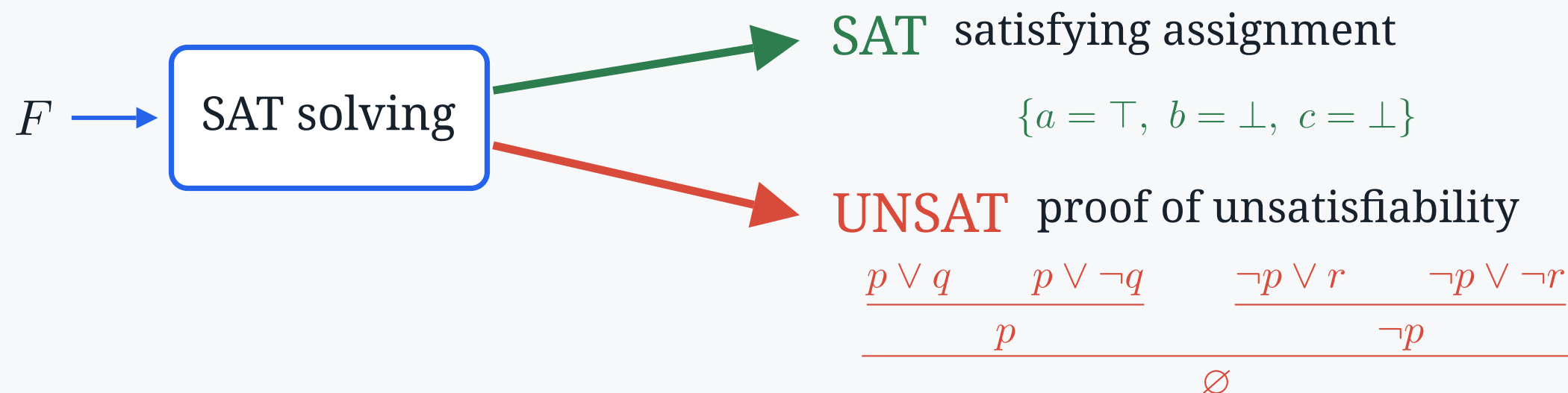
Check explicit logical constraints



A precise logical decision



Boolean Satisfiability Problem (SAT)



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

Propositional logic: $\wedge, \vee, \neg$, Boolean variables

Conjunctive Normal Form (CNF)

Clause: disjunction of literals (set of literals)

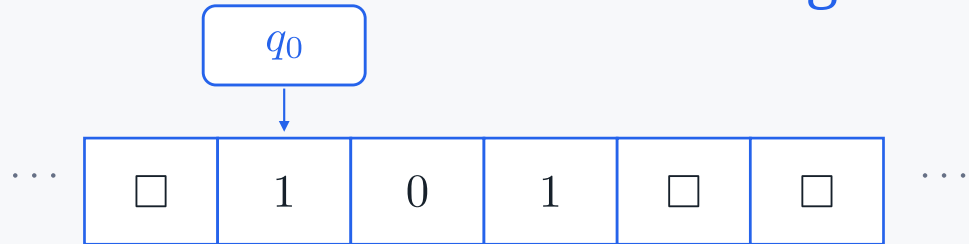
Literal: Boolean variable or its negation

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

$$a \quad \neg a$$

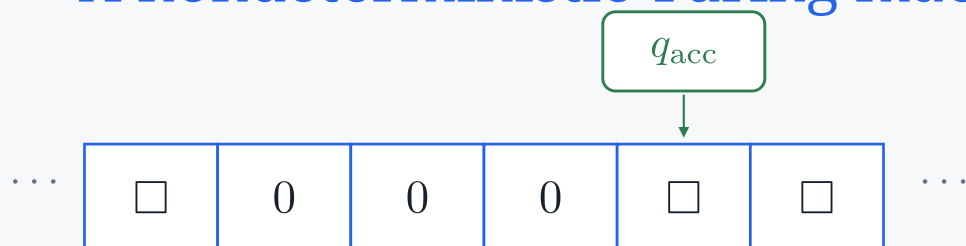
The First NP-Complete Problem

A nondeterministic Turing machine



The First NP-Complete Problem

A nondeterministic Turing machine



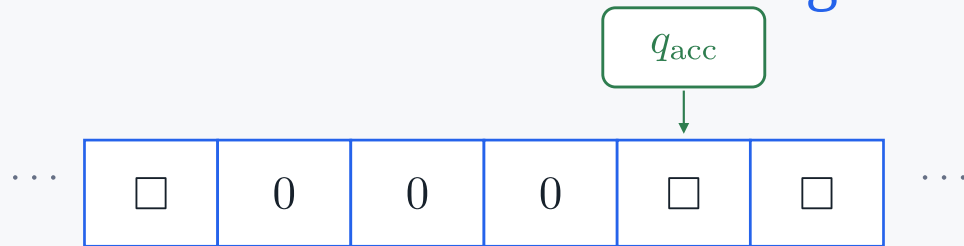
Computation history

$t = 0$	$\square$	1	0	1	$\square$	$\square$	q_0
$t = 1$	$\square$	0	0	1	$\square$	$\square$	q_0
$t = 2$	$\square$	0	0	1	$\square$	$\square$	q_0
$t = 3$	$\square$	0	0	0	$\square$	$\square$	q_{acc}

One accepting computation

The First NP-Complete Problem

A nondeterministic Turing machine



Computation history

$t = 0$	1 0 1	q_0
$t = 1$	0 0 1	q_0
$t = 2$	0 0 1	q_0
$t = 3$	0 0 0	q_{acc}

One accepting computation

SAT encoding

$$F_M = F_{\text{start}} \wedge F_{\text{move}} \wedge F_{\text{accept}}$$

Variables describe tape symbols, head positions and machine states.

F_M is satisfiable



M has an accepting computation within T steps

Polynomial-time encoding for $T = p(|x|)$

Cook–Levin theorem: SAT is NP-complete

Stephen A. Cook · May 1971

The complexity of theorem-proving procedures
STOC, pp. 151–158

Leonid A. Levin · 1973

Universal Sequential Search Problems
Problemy Peredachi Informatsii 9(3), pp. 115–116

The First NP-Complete Problem

Appears in
Proceedings Third Annual
ACM Symposium
Theory of Computing
May, 1971

The Complexity of Theorem-Proving Procedures
Stephen A. Cook
University of Toronto

Summary

It is shown that any recognition problem solved by a polynomial time-bounded nondeterministic Turing machine can be "reduced" to the problem of determining whether a given propositional formula is a tautology. Here "reduced" means, roughly speaking, that the first problem can be solved deterministically in polynomial time provided an oracle is available for solving the second. From this notion of reducible, polynomial degrees of difficulty are defined, and it is shown that the problem of determining tautologyhood has the same polynomial degree as the problem of determining whether the first of two given graphs is isomorphic to a subgraph of the second. Other examples are discussed. A method of measuring the complexity of proof procedures for the predicate calculus is introduced and discussed.

Throughout this paper, a set of strings means a set of strings on some fixed, large, finite alphabet Σ . This alphabet is large enough to include symbols for all sets described here. All Turing machines are deterministic recognition devices, unless the contrary is explicitly stated.

1. Tautologies and Polynomial Reducibility.
Let us fix a formalism for the propositional calculus in

certain recursive set of strings on this alphabet, and we are interested in the problem of finding a good lower bound on its possible recognition times. We provide no such lower bound here, but theorem 1 will give evidence that {tautologies} is a difficult set to recognize, since many apparently difficult problems can be reduced to determining tautologyhood. By reduced we mean, roughly speaking, that if tautologyhood could be decided instantly (by an "oracle") then these problems could be decided in polynomial time. In order to make this notion precise, we introduce query machines, which are like Turing machines with oracles in [1].

A query machine is a multitape Turing machine with a distinguished tape called the query tape, and three distinguished states called the query state, yes state, and no state, respectively. If M is a query machine and T is a set of strings, then a T-computation of M is a computation of M in which initially M is in the initial state and has an input string w on its input tape, and each time M assumes the query state there is a string u on the query tape, and the next state M assumes is the yes state if $u \in T$ and the no state if $u \notin T$. We think of an "oracle", which knows T , placing M in the yes state or no state.

Stephen A. Cook · May 1971

The complexity of theorem-proving procedures
STOC, pp. 151–158

ПРОБЛЕМЫ ПЕРЕДАЧИ ИНФОРМАЦИИ
Том IX 1973 Вып. 3

КРАТКИЕ СООБЩЕНИЯ

УДК 519.14

УНИВЕРСАЛЬНЫЕ ЗАДАЧИ ПЕРЕБОРА

Л. А. Левин

В статье рассматривается несколько известных массовых задач «переборного типа» и доказывается, что эти задачи можно решать лишь за такое время, за которое можно решать вообще любые задачи указанного типа.

После уточнения понятия алгоритма была доказана алгоритмическая неразрешимость ряда классических массовых проблем (например, проблем тождества элементов группы, гомеоморфности многообразий, разрешимости диофантовых уравнений и других). Тем самым был снят вопрос о нахождении практического способа их решения. Однако существование алгоритмов для решения других задач не снимает для них аналогичного вопроса из-за фантастически большого объема работы, предписываемого этими алгоритмами. Такова ситуация с так называемыми переборными задачами: минимизации булевых функций, поиска доказательств ограниченной длины, выяснения изоморфности графов и другими. Все эти задачи решаются тривиальными алгоритмами, состоящими в переборе всех возможностей. Однако эти алгоритмы требуют экспоненциального времени работы и у математиков сложилось убеждение, что более простые алгоритмы для них невозможны. Был получен ряд серьезных аргументов в пользу его справедливости (см. [1, 2]), однако доказать это утверждение не удалось никому. (Например, до сих пор не доказано, что для нахождения математических доказательств нужно больше времени, чем для их проверки.)

Однако если предположить, что вообще существует какая-нибудь (хотя бы искусственно построенная) массовая задача переборного типа, неразрешимая простыми (в смысле объема вычислений) алгоритмами, то можно показать, что этим же свойством обладают и многие «классические» переборные задачи (в том числе задача минимизации, задача поиска доказательств и др.). В этом и состоят основные результаты статьи.

Leonid A. Levin · 1973

Universal Sequential Search Problems

Problemy Peredachi Informatsii 9(3), pp. 115–116

How big is the search space for a SAT solver?

Search space for 267 variables

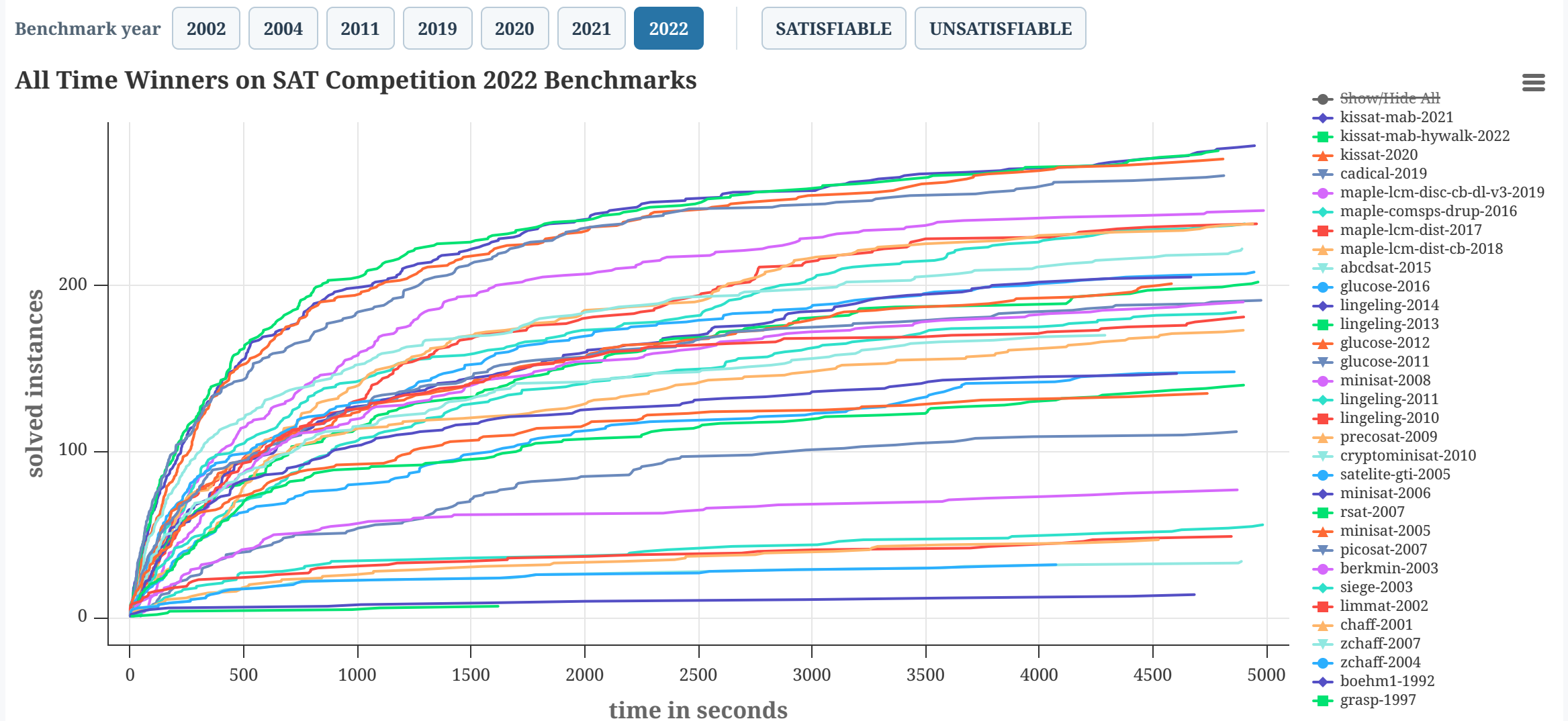
$$2^{267} \approx 1.76 \times 10^{80}$$

The number of protons in the
observable universe
(Eddington)



[gettyimage.com]

Evolution of SAT solvers



How humans think...

```
def ProblemSolving(F: Problem) -> bool:
    Decisions = {}
    while Problem not solved:
        if Decisions lead to a conflict:
            analyze the conflict:
                • try to learn something from it
                • if learned the problem is unsolvable: return UNSAT
            Decisions.drop() # Change our mind on previous decisions
        else:
            ℓ = make a decision
            Decisions.add(ℓ) # Make a decision
    return SAT
```


How humans think...

```
def ProblemSolving(F: CNF) -> bool:
    Decisions = {}
    while Problem not solved:
        if Decisions lead to a conflict:
            analyze the conflict:
                • try to learn something from it
                • if learned the problem is unsolvable: return UNSAT
            Decisions.drop() # Change our mind on previous decisions
        else:
            ℓ = make a decision
            Decisions.add(ℓ) # Make a decision
    return SAT
```


How humans think...

```
def ProblemSolving(F: CNF) -> bool:
    Decisions = {} # Partial assignment
    while Problem not solved:
        if Decisions lead to a conflict:
            analyze the conflict:
                • try to learn something from it
                • if learned the problem is unsolvable: return UNSAT
            Decisions.drop() # Change our mind on previous decisions
        else:
            ℓ = make a decision
            Decisions.add(ℓ) # Make a decision
    return SAT
```

How humans think...

```
def ProblemSolving(F: CNF) -> bool:
    Decisions = {} # Partial assignment
    while not all variables are assigned in Decisions:
        if Decisions lead to a conflict:
            analyze the conflict:
                • try to learn something from it
                • if learned the problem is unsolvable: return UNSAT
            Decisions.drop() # Change our mind on previous decisions
        else:
            ℓ = make a decision
            Decisions.add(ℓ) # Make a decision
    return SAT
```

How humans think...

```
def ProblemSolving(F: CNF) -> bool:
    Decisions = {} # Partial assignment
    while not all variables are assigned in Decisions:
        if conflict := UnitPropagation(F, Decisions):
            analyze the conflict:
                • try to learn something from it
                • if learned the problem is unsolvable: return UNSAT
            Decisions.drop() # Change our mind on previous decisions
        else:
            ℓ = make a decision
            Decisions.add(ℓ) # Make a decision
    return SAT
```

How humans think...

```
def ProblemSolving(F: CNF) -> bool:
    Decisions = {} # Partial assignment
    while not all variables are assigned in Decisions:
        if conflict := UnitPropagation(F, Decisions):
            analyze the conflict:
                • try to learn something from it
                • if learned the problem is unsolvable: return UNSAT
            Decisions.drop() # Change our mind on previous decisions
        else:
            ℓ = make a decision
            Decisions.add(ℓ) # Make a decision
    return SAT
```

How to identify a conflict?

Unit Propagation

UnitPropagation($F, \{a\}$) :

Remove clauses containing a (immediately satisfied).

Remove occurrences of $\neg a$. Extend decisions with unit clauses and repeat.

$$F = (a \vee b) \wedge (\neg a \vee b \vee \neg c) \wedge (\neg a \vee \neg b) \wedge \dots$$

A unit clause forces its only unassigned literal to True.

Conflict: removing the last literal leaves the empty clause $\perp$

Unit Propagation

UnitPropagation($F, \{a\}$) :

Remove clauses containing a (immediately satisfied).

Remove occurrences of $\neg a$. Extend decisions with unit clauses and repeat.

$$F = \boxed{\cancel{(a \vee b)} \wedge (\neg a \vee b \vee \neg c) \wedge (\neg a \vee \neg b) \wedge \dots} \quad \{a\}$$

A unit clause forces its only unassigned literal to True.

Conflict: removing the last literal leaves the empty clause $\perp$

Unit Propagation

UnitPropagation($F, \{a\}$) :

Remove clauses containing a (immediately satisfied).

Remove occurrences of $\neg a$. Extend decisions with unit clauses and repeat.

$$F = \boxed{(\cancel{a} \vee \cancel{b}) \wedge (\cancel{\neg a} \vee b \vee \neg c) \wedge (\cancel{\neg a} \vee \neg b) \wedge \dots} \quad \{a\}$$

A unit clause forces its only unassigned literal to True.

Conflict: removing the last literal leaves the empty clause $\perp$

Unit Propagation

UnitPropagation($F, \{a\}$) :

Remove clauses containing a (immediately satisfied).

Remove occurrences of $\neg a$. Extend decisions with unit clauses and repeat.

$$F = \boxed{\cancel{(a \vee b)} \wedge (\cancel{\neg a} \vee b \vee \neg c) \wedge (\cancel{\neg a} \vee \neg b) \wedge \dots} \quad \{a\}$$

$$\boxed{(b \vee \neg c) \wedge (\neg b) \wedge \dots}$$

A unit clause forces its only unassigned literal to True.

Conflict: removing the last literal leaves the empty clause $\perp$

Unit Propagation

UnitPropagation($F, \{a\}$) :

Remove clauses containing a (immediately satisfied).

Remove occurrences of $\neg a$. Extend decisions with unit clauses and repeat.

$$F = \boxed{\cancel{(a \vee b)} \wedge (\cancel{\neg a} \vee b \vee \neg c) \wedge (\cancel{\neg a} \vee \neg b) \wedge \dots} \quad \{a\}$$
$$\boxed{(b \vee \neg c) \wedge (\neg b) \wedge \dots} \quad \{a, \neg b\}$$

A unit clause forces its only unassigned literal to True.

Conflict: removing the last literal leaves the empty clause $\perp$

Unit Propagation

UnitPropagation($F, \{a\}$) :

Remove clauses containing a (immediately satisfied).

Remove occurrences of $\neg a$. Extend decisions with unit clauses and repeat.

$$F = \boxed{\cancel{(a \vee b)} \wedge (\cancel{\neg a} \vee b \vee \neg c) \wedge (\cancel{\neg a} \vee \neg b) \wedge \dots} \quad \{a\}$$
$$\boxed{(b \vee \neg c) \wedge \cancel{(\neg b)} \wedge \dots} \quad \{a, \neg b\}$$

A unit clause forces its only unassigned literal to True.

Conflict: removing the last literal leaves the empty clause $\perp$

Unit Propagation

UnitPropagation($F, \{a\}$) :

Remove clauses containing a (immediately satisfied).

Remove occurrences of $\neg a$. Extend decisions with unit clauses and repeat.

$$F = (\cancel{a} \vee \cancel{b}) \wedge (\cancel{\neg a} \vee b \vee \neg c) \wedge (\cancel{\neg a} \vee \neg b) \wedge \dots \quad \{a\}$$

$$(\cancel{b} \vee \neg c) \wedge (\cancel{\neg b}) \wedge \dots \quad \{a, \neg b\}$$

A unit clause forces its only unassigned literal to True.

Conflict: removing the last literal leaves the empty clause $\perp$

Unit Propagation

UnitPropagation($F, \{a\}$) :

Remove clauses containing a (immediately satisfied).

Remove occurrences of $\neg a$. Extend decisions with unit clauses and repeat.

$$F = (\cancel{a} \vee \cancel{b}) \wedge (\cancel{\neg a} \vee b \vee \neg c) \wedge (\cancel{\neg a} \vee \neg b) \wedge \dots \quad \{a\}$$

$$(\cancel{b} \vee \neg c) \wedge (\cancel{\neg b}) \wedge \dots \quad \{a, \neg b\}$$

$$(\neg c) \wedge \dots$$

A unit clause forces its only unassigned literal to True.

Conflict: removing the last literal leaves the empty clause $\perp$

Unit Propagation

UnitPropagation($F, \{a\}$) :

Remove clauses containing a (immediately satisfied).

Remove occurrences of $\neg a$. Extend decisions with unit clauses and repeat.

$$\begin{array}{ll} F = (\cancel{a} \vee \cancel{b}) \wedge (\cancel{\neg a} \vee b \vee \neg c) \wedge (\cancel{\neg a} \vee \neg b) \wedge \dots & \{a\} \\ (\cancel{b} \vee \neg c) \wedge (\cancel{\neg b}) \wedge \dots & \{a, \neg b\} \\ (\neg c) \wedge \dots & \{a, \neg b, \neg c\} \end{array}$$

A unit clause forces its only unassigned literal to True.

Conflict: removing the last literal leaves the empty clause $\perp$

Unit Propagation

UnitPropagation($F, \{a\}$) :

Remove clauses containing a (immediately satisfied).

Remove occurrences of $\neg a$. Extend decisions with unit clauses and repeat.

$$F = (\cancel{a} \vee \cancel{b}) \wedge (\cancel{\neg a} \vee b \vee \neg c) \wedge (\cancel{\neg a} \vee \neg b) \wedge \dots \quad \{a\}$$

$$(\cancel{b} \vee \neg c) \wedge (\cancel{\neg b}) \wedge \dots \quad \{a, \neg b\}$$

$$(\cancel{\neg c}) \wedge \dots \quad \{a, \neg b, \neg c\}$$

A unit clause forces its only unassigned literal to True.

Conflict: removing the last literal leaves the empty clause $\perp$

How humans think...

```
def ProblemSolving(F: CNF) -> bool:
    Decisions = {} # Partial assignment
    while not all variables are assigned in Decisions:
        if conflict := UnitPropagation(F, Decisions):
            analyze the conflict:
                • try to learn something from it
                • if learned the problem is unsolvable: return UNSAT
            Decisions.drop() # Change our mind on previous decisions
        else:
            ℓ = make a decision
            Decisions.add(ℓ) # Make a decision
    return SAT
```

How humans think...

```
def ProblemSolving(F: CNF) -> bool:
    Decisions = {} # Partial assignment
    while not all variables are assigned in Decisions:
        if conflict := UnitPropagation(F, Decisions):
            C,  $\ell$  = ConflictAnalysis(conflict)
            if C ==  $\perp$ : return UNSAT
            F.add(C)
            Decisions.drop_after( $\ell$ ) # Remove decisions after  $\ell$ 
        else:
             $\ell$  = make a decision
            Decisions.add( $\ell$ ) # Make a decision
    return SAT
```


Conflict Analysis: Implication Graph

$$(A \vee B)$$

$$(B \vee C)$$

$$(\neg A \vee \neg X \vee Y)$$

$$(\neg A \vee X \vee Z)$$

$$(\neg A \vee \neg Y \vee Z)$$

$$(\neg A \vee X \vee \neg Z)$$

$$(\neg A \vee \neg Y \vee \neg Z)$$

Conflict Analysis: Implication Graph

$$(A \vee B)$$

$$(B \vee C)$$

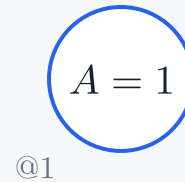
$$(\neg A \vee \neg X \vee Y)$$

$$(\neg A \vee X \vee Z)$$

$$(\neg A \vee \neg Y \vee Z)$$

$$(\neg A \vee X \vee \neg Z)$$

$$(\neg A \vee \neg Y \vee \neg Z)$$



Decide $A = 1$ at level 1.

Conflict Analysis: Implication Graph

$$(A \vee B)$$

$$(B \vee C)$$

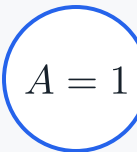
$$(\neg A \vee \neg X \vee Y)$$

$$(\neg A \vee X \vee Z)$$

$$(\neg A \vee \neg Y \vee Z)$$

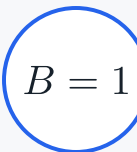
$$(\neg A \vee X \vee \neg Z)$$

$$(\neg A \vee \neg Y \vee \neg Z)$$



@1

$$A = 1$$



@2

$$B = 1$$

Decide $B = 1$ at level 2.

Conflict Analysis: Implication Graph

$$(A \vee B)$$

$$(B \vee C)$$

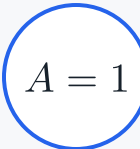
$$(\neg A \vee \neg X \vee Y)$$

$$(\neg A \vee X \vee Z)$$

$$(\neg A \vee \neg Y \vee Z)$$

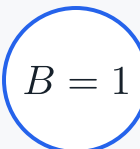
$$(\neg A \vee X \vee \neg Z)$$

$$(\neg A \vee \neg Y \vee \neg Z)$$



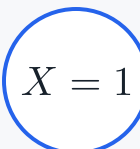
@1

$$A = 1$$



@2

$$B = 1$$



@3

$$X = 1$$

Decide $X = 1$ at level 3.

Conflict Analysis: Implication Graph

$$(A \vee B)$$

$$(B \vee C)$$

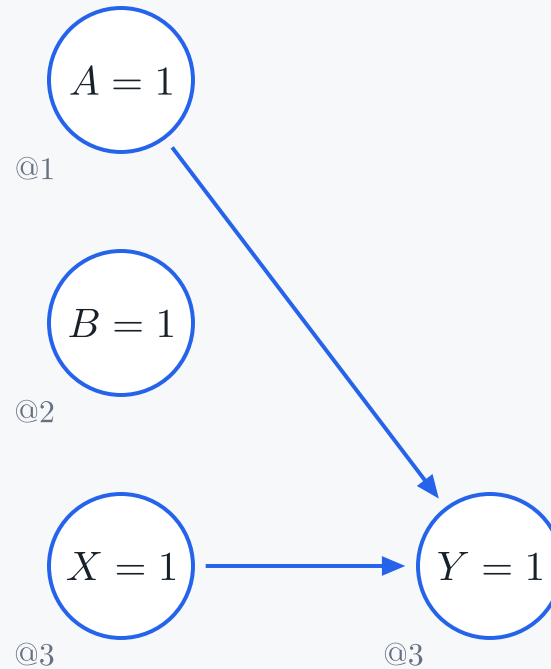
$$(\neg A \vee \neg X \vee Y)$$

$$(\neg A \vee X \vee Z)$$

$$(\neg A \vee \neg Y \vee Z)$$

$$(\neg A \vee X \vee \neg Z)$$

$$(\neg A \vee \neg Y \vee \neg Z)$$



Propagate $Y = 1$ at level 3.

Conflict Analysis: Implication Graph

$(A \vee B)$

$(B \vee C)$

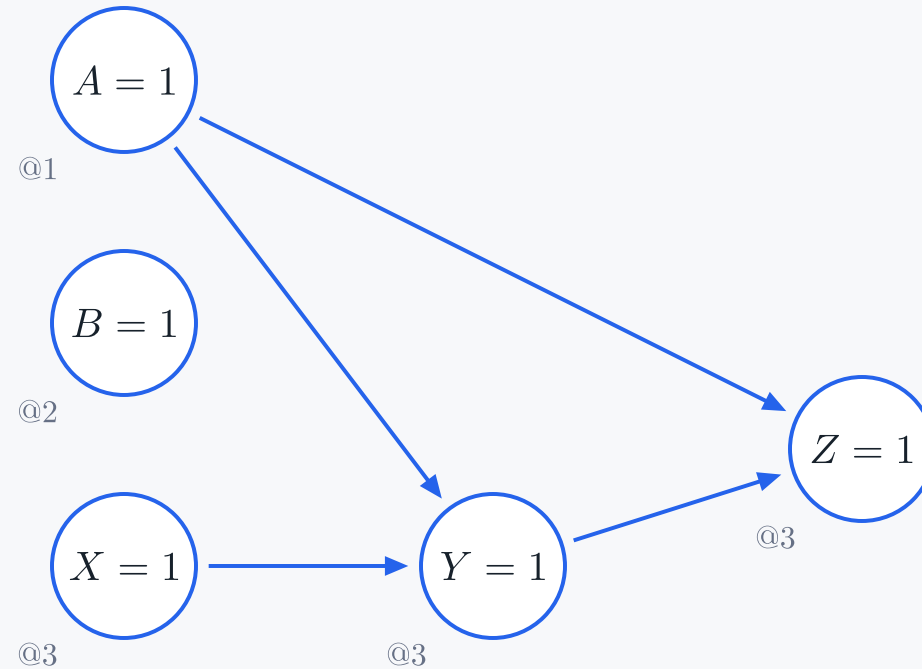
$(\neg A \vee \neg X \vee Y)$

$(\neg A \vee X \vee Z)$

$(\neg A \vee \neg Y \vee Z)$

$(\neg A \vee X \vee \neg Z)$

$(\neg A \vee \neg Y \vee \neg Z)$



Propagate $Z = 1$ at level 3.

Conflict Analysis: Implication Graph

$$(A \vee B)$$

$$(B \vee C)$$

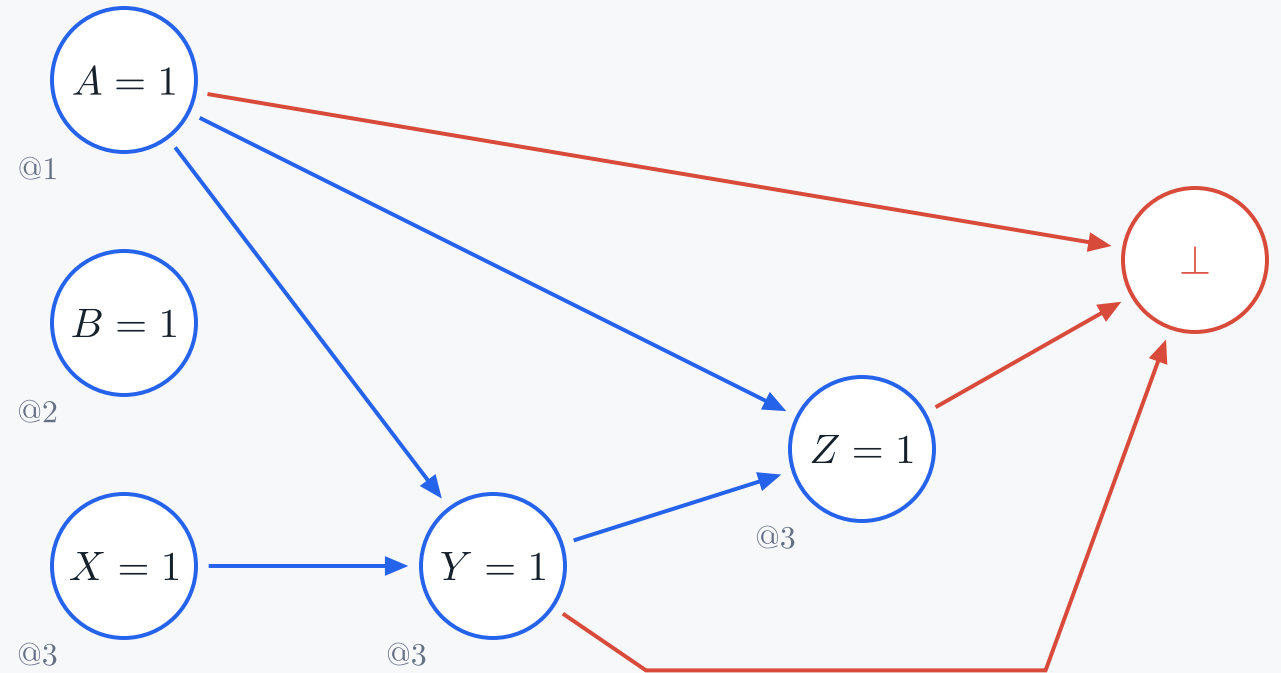
$$(\neg A \vee \neg X \vee Y)$$

$$(\neg A \vee X \vee Z)$$

$$(\neg A \vee \neg Y \vee Z)$$

$$(\neg A \vee X \vee \neg Z)$$

$$(\neg A \vee \neg Y \vee \neg Z) \Rightarrow \perp$$



Conflict at level 3.

Conflict Analysis: Implication Graph

$(A \vee B)$

$(B \vee C)$

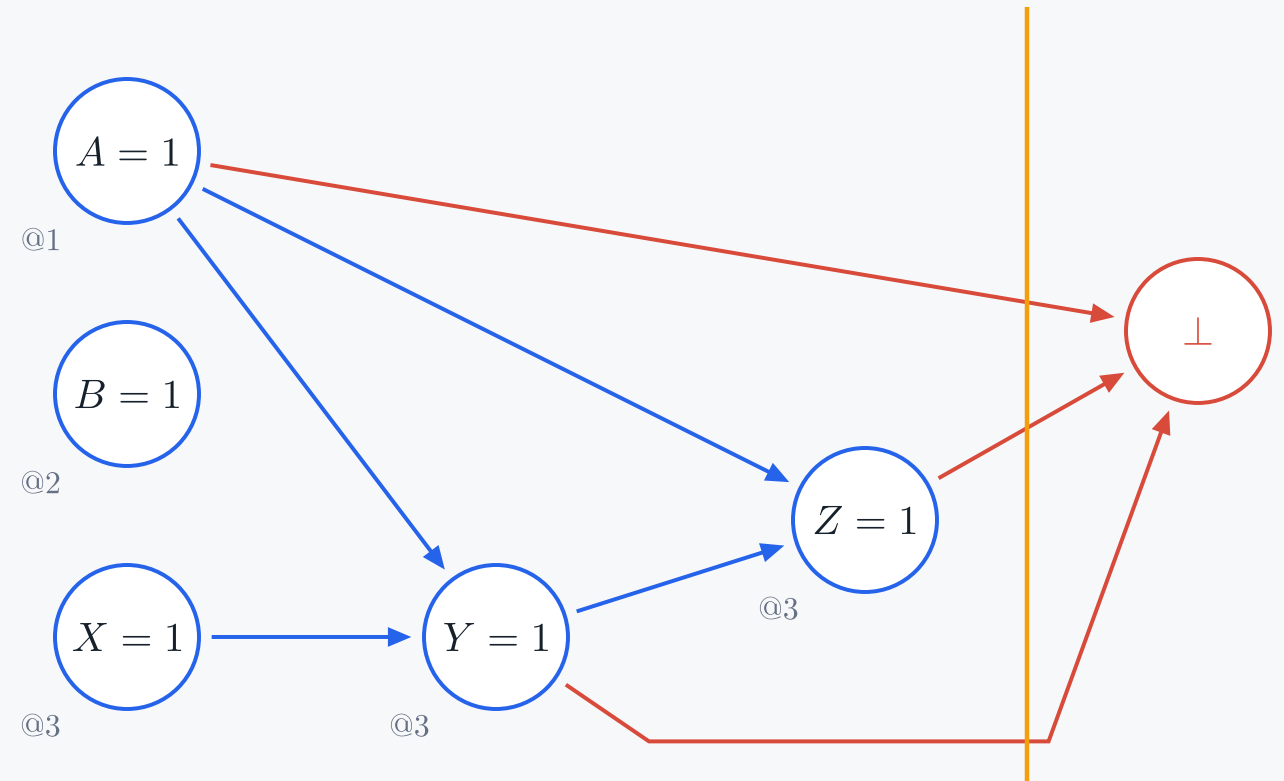
$(\neg A \vee \neg X \vee Y)$

$(\neg A \vee X \vee Z)$

$(\neg A \vee \neg Y \vee Z)$

$(\neg A \vee X \vee \neg Z)$

$(\neg A \vee \neg Y \vee \neg Z) \Rightarrow \perp$



$\neg A \vee \neg Y \vee \neg Z$

Conflict Analysis: Implication Graph

$(A \vee B)$

$(B \vee C)$

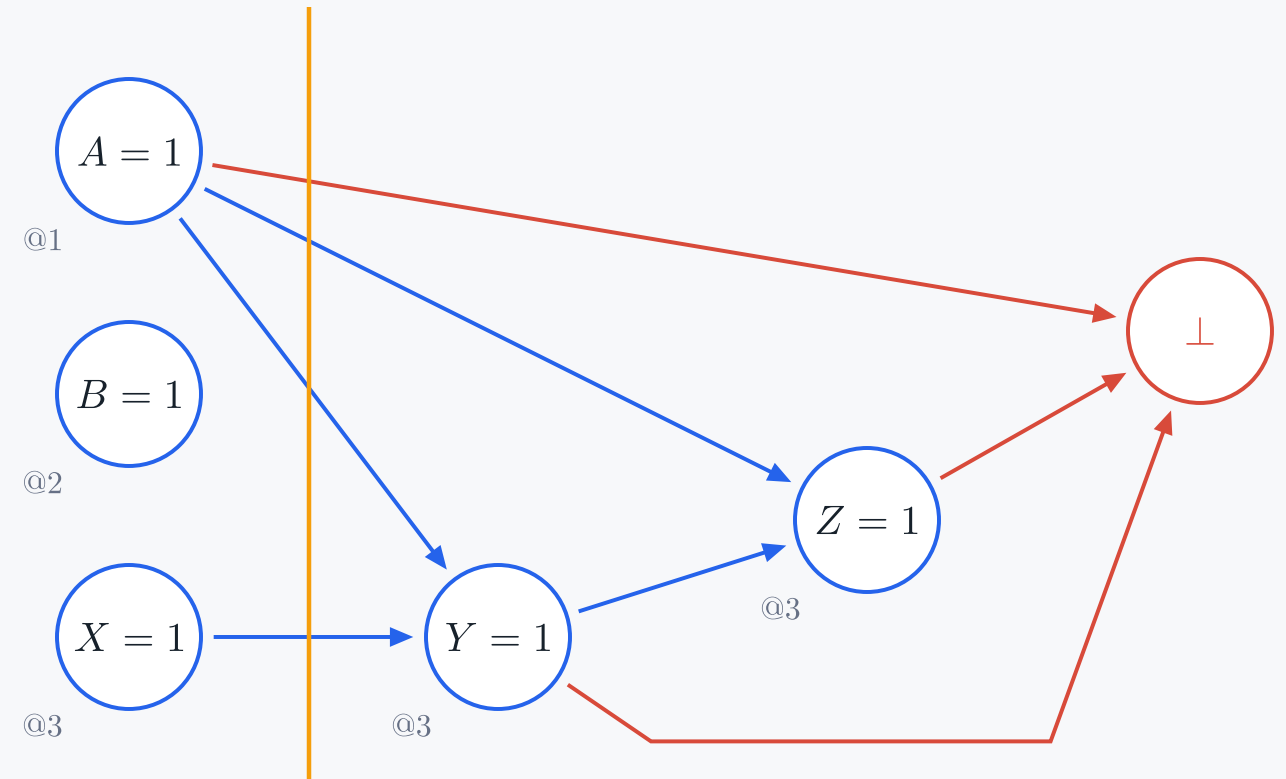
$(\neg A \vee \neg X \vee Y)$

$(\neg A \vee X \vee Z)$

$(\neg A \vee \neg Y \vee Z)$

$(\neg A \vee X \vee \neg Z)$

$(\neg A \vee \neg Y \vee \neg Z) \Rightarrow \perp$



$\neg A \vee \neg X$

Conflict Analysis: Implication Graph

$(A \vee B)$

$(B \vee C)$

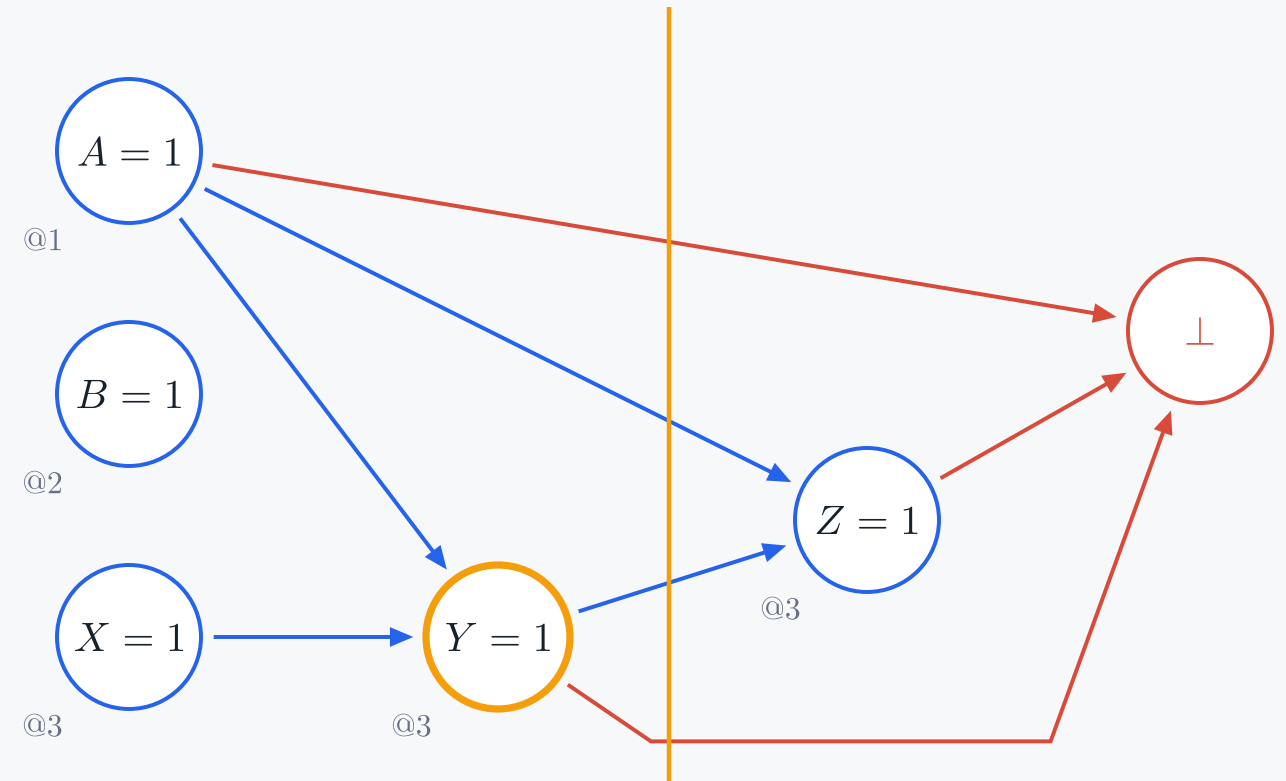
$(\neg A \vee \neg X \vee Y)$

$(\neg A \vee X \vee Z)$

$(\neg A \vee \neg Y \vee Z)$

$(\neg A \vee X \vee \neg Z)$

$(\neg A \vee \neg Y \vee \neg Z) \Rightarrow \perp$



$\neg A \vee \neg Y$

Resolution

$$(I \vee X_1 \vee \cdots \vee X_n) \wedge (\neg I \vee Y_1 \vee \cdots \vee Y_n)$$

$$(X_1 \vee \cdots \vee X_n \vee Y_1 \vee \cdots \vee Y_n)$$

Resolution

$$(I \vee X_1 \vee \cdots \vee X_n) \wedge (\neg I \vee Y_1 \vee \cdots \vee Y_n)$$

$$(X_1 \vee \cdots \vee X_n \vee Y_1 \vee \cdots \vee Y_n)$$

Examples

$$(X_1 \vee X_3 \vee \neg X_7) \wedge (\neg X_1 \vee X_2) \vdash (X_3 \vee \neg X_7 \vee X_2)$$

$$(X_4 \vee X_5) \wedge (\neg X_5) \vdash (X_4)$$

$$(X_1) \wedge (\neg X_1) \vdash \perp$$

Conflict Analysis: Implication Graph

$$(A \vee B)$$

$$(B \vee C)$$

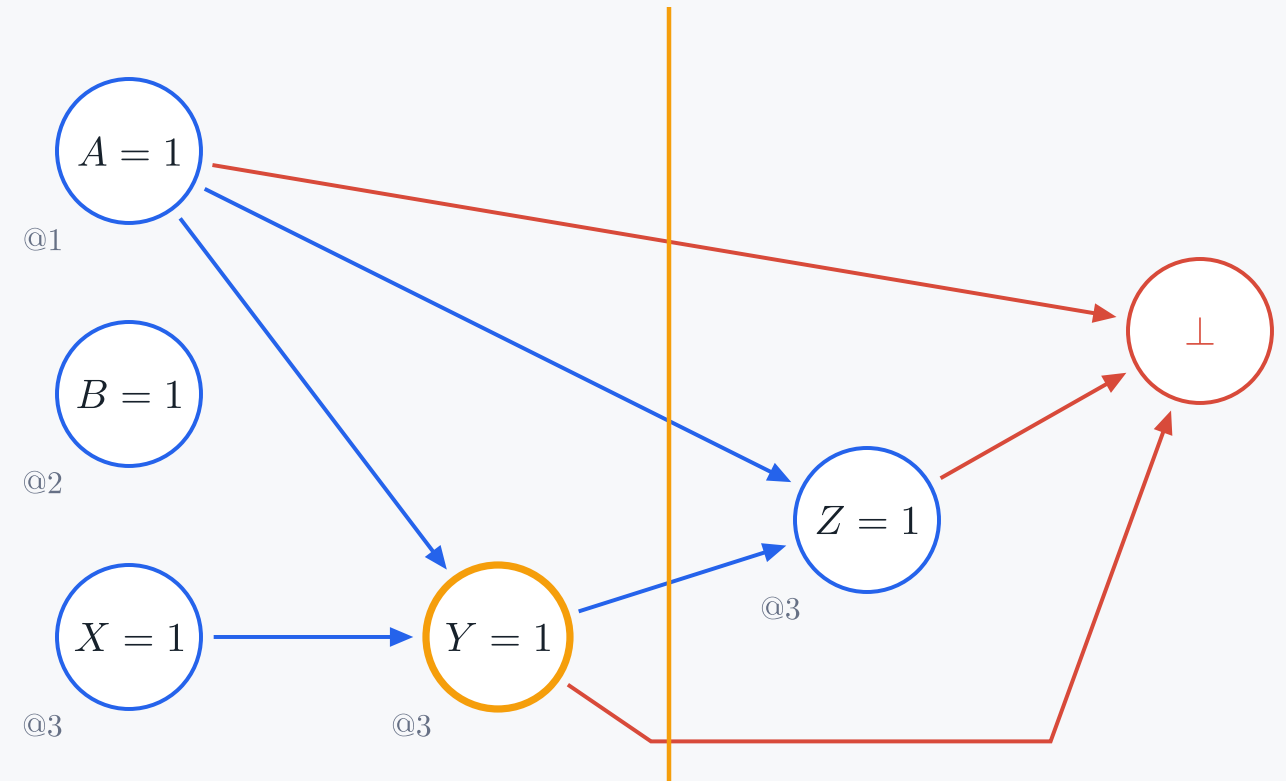
$$(\neg A \vee \neg X \vee Y)$$

$$(\neg A \vee X \vee Z)$$

$$(\neg A \vee \neg Y \vee Z)$$

$$(\neg A \vee X \vee \neg Z)$$

$$(\neg A \vee \neg Y \vee \neg Z) \Rightarrow \perp$$



$$\neg A \vee \neg Y$$

Conflict Analysis: Implication Graph

$(A \vee B)$

$(B \vee C)$

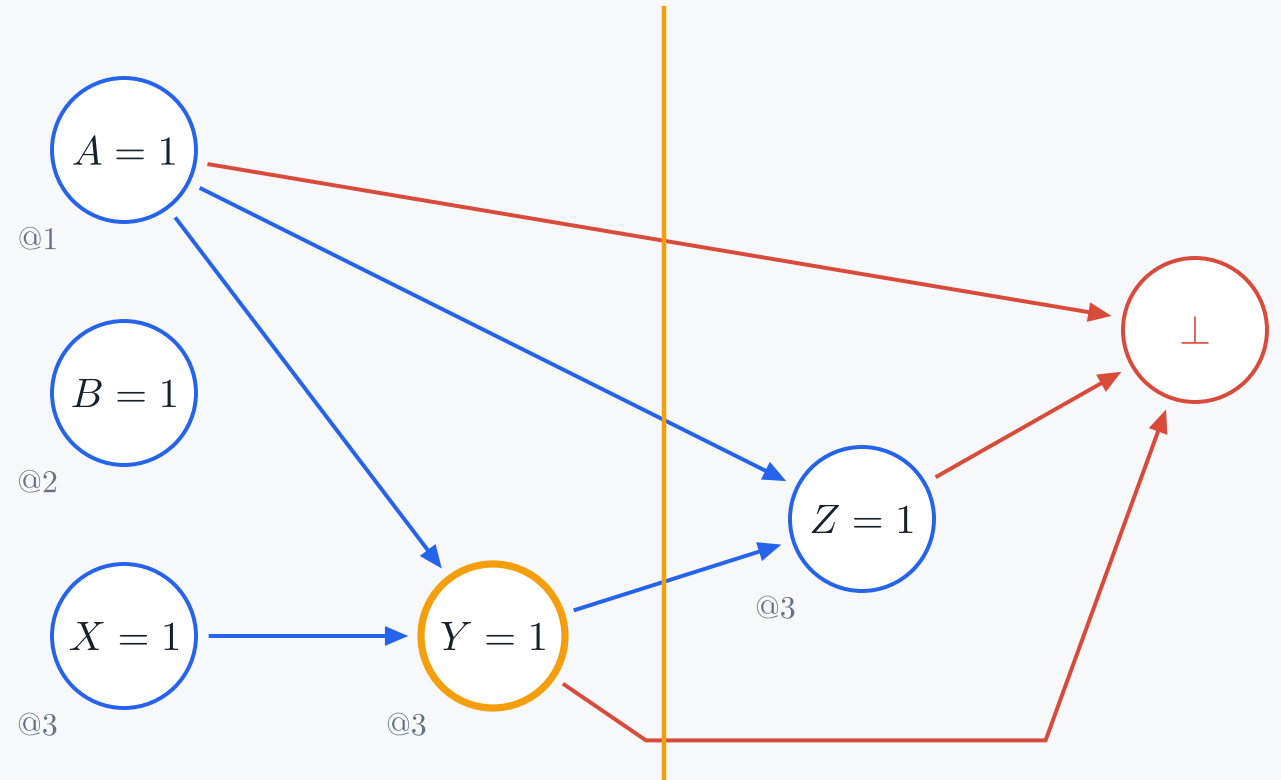
$(\neg A \vee \neg X \vee Y)$

$(\neg A \vee X \vee Z)$

$(\neg A \vee \neg Y \vee Z)$

$(\neg A \vee X \vee \neg Z)$

$(\neg A \vee \neg Y \vee \neg Z) \Rightarrow \perp$



$\neg A \vee \neg Y$ First unique implication point: Y

Y is the closest node to the conflict on every path from X.
Only $\neg Y$ is at the current decision level in this clause.

Conflict Analysis: Implication Graph

$(A \vee B)$

$(B \vee C)$

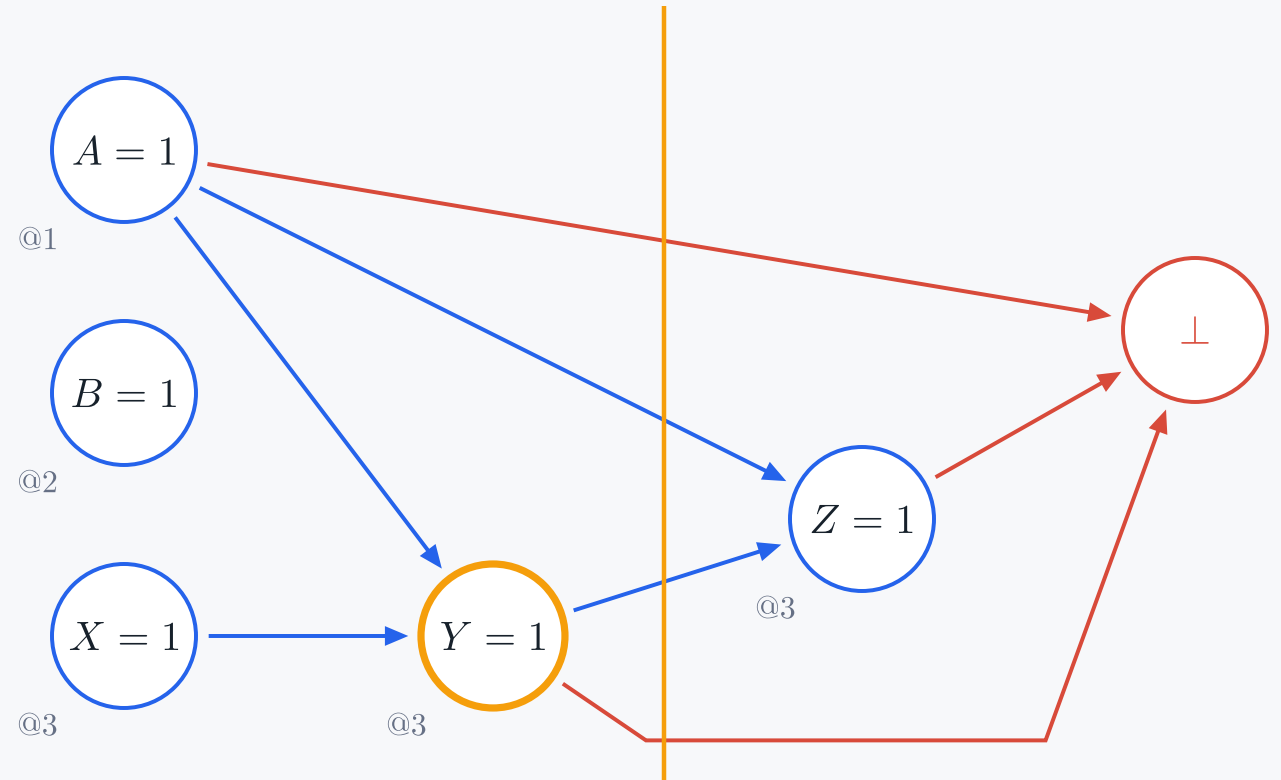
$(\neg A \vee \neg X \vee Y)$

$(\neg A \vee X \vee Z)$

$(\neg A \vee \neg Y \vee Z)$

$(\neg A \vee X \vee \neg Z)$

$(\neg A \vee \neg Y \vee \neg Z) \Rightarrow \perp$



$\neg A \vee \neg Y$ Backjump from level 3 to level 1

Level 1 is the second-highest level in the learned clause. After backjumping, the clause is unit and forces $Y = 0$.

Conflict Analysis: Implication Graph

$(A \vee B)$

$(B \vee C)$

$(\neg A \vee \neg X \vee Y)$

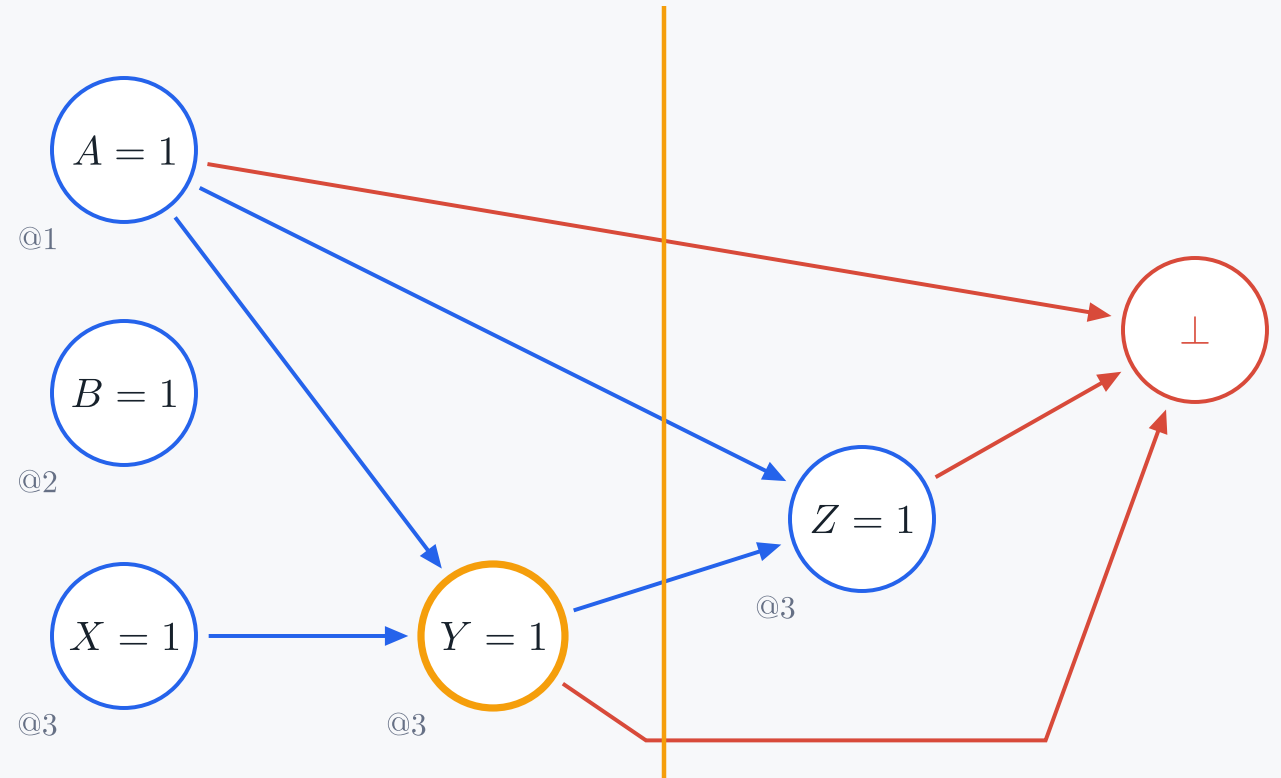
$(\neg A \vee X \vee Z)$

$(\neg A \vee \neg Y \vee Z)$

$(\neg A \vee X \vee \neg Z)$

$(\neg A \vee \neg Y \vee \neg Z) \Rightarrow \perp$

$(\neg A \vee \neg Y)$



$\neg A \vee \neg Y$ Backjump from level 3 to level 1

Level 1 is the second-highest level in the learned clause. After backjumping, the clause is unit and forces $Y = 0$.

Conflict Analysis: Implication Graph

$$(A \vee B)$$

$$(B \vee C)$$

$$(\neg A \vee \neg X \vee Y)$$

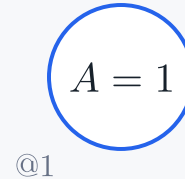
$$(\neg A \vee X \vee Z)$$

$$(\neg A \vee \neg Y \vee Z)$$

$$(\neg A \vee X \vee \neg Z)$$

$$(\neg A \vee \neg Y \vee \neg Z)$$

$$(\neg A \vee \neg Y)$$



Backjump to level 1.
Keep $A = 1$; undo B, X, Y and Z .

Conflict Analysis: Implication Graph

$$(A \vee B)$$

$$(B \vee C)$$

$$(\neg A \vee \neg X \vee Y)$$

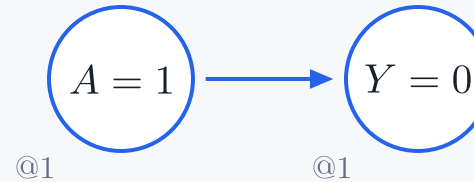
$$(\neg A \vee X \vee Z)$$

$$(\neg A \vee \neg Y \vee Z)$$

$$(\neg A \vee X \vee \neg Z)$$

$$(\neg A \vee \neg Y \vee \neg Z)$$

$$(\neg A \vee \neg Y)$$



Propagate $Y = 0$ at level 1.

The newly learned clause is always propagating.

Conflict Analysis: Implication Graph

$$(A \vee B)$$

$$(B \vee C)$$

$$(\neg A \vee \neg X \vee Y)$$

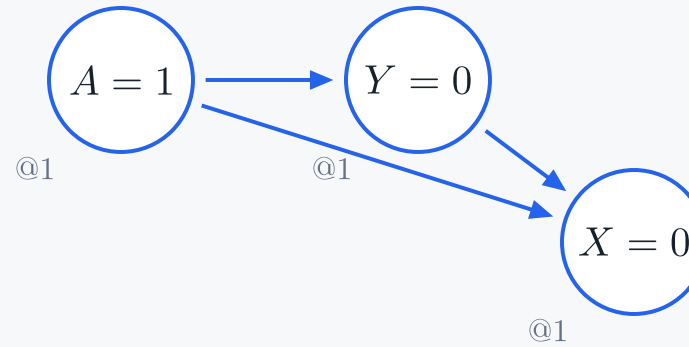
$$(\neg A \vee X \vee Z)$$

$$(\neg A \vee \neg Y \vee Z)$$

$$(\neg A \vee X \vee \neg Z)$$

$$(\neg A \vee \neg Y \vee \neg Z)$$

$$(\neg A \vee \neg Y)$$



Propagate $X = 0$ at level 1.

Conflict Analysis: Implication Graph

$$(A \vee B)$$

$$(B \vee C)$$

$$(\neg A \vee \neg X \vee Y)$$

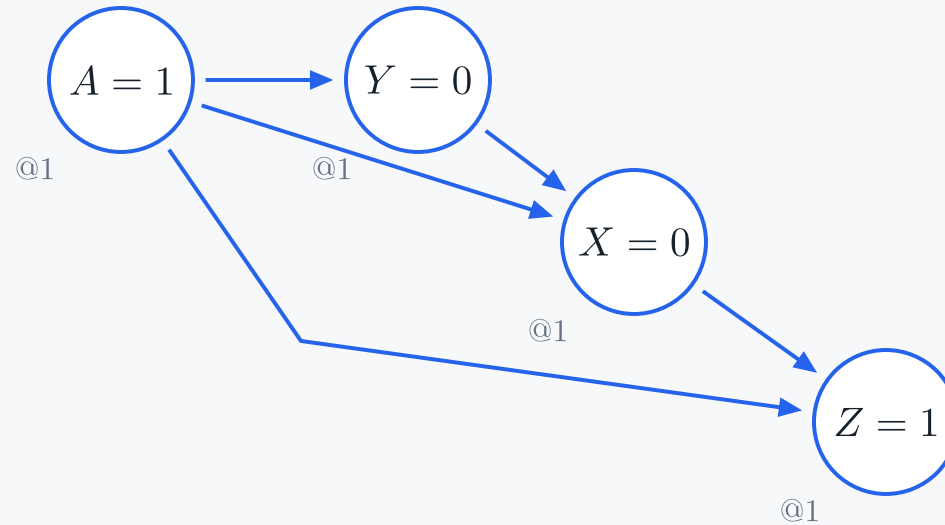
$$(\neg A \vee X \vee Z)$$

$$(\neg A \vee \neg Y \vee Z)$$

$$(\neg A \vee X \vee \neg Z)$$

$$(\neg A \vee \neg Y \vee \neg Z)$$

$$(\neg A \vee \neg Y)$$



Propagate $Z = 1$ at level 1.

Conflict Analysis: Implication Graph

$$(A \vee B)$$

$$(B \vee C)$$

$$(\neg A \vee \neg X \vee Y)$$

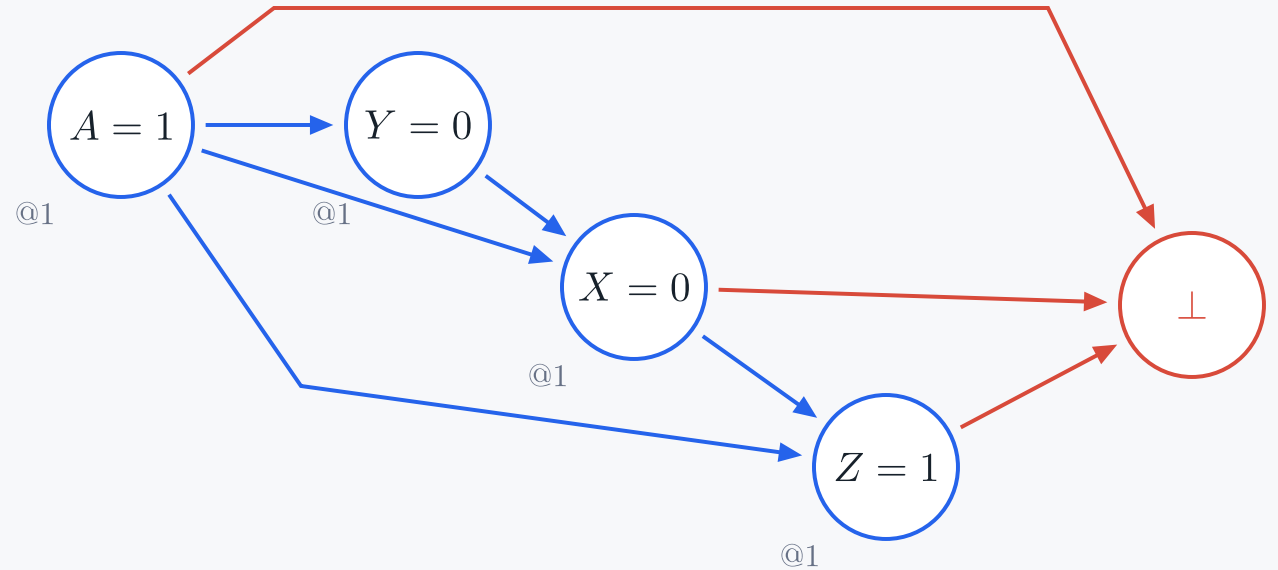
$$(\neg A \vee X \vee Z)$$

$$(\neg A \vee \neg Y \vee Z)$$

$$(\neg A \vee X \vee \neg Z) \Rightarrow \perp$$

$$(\neg A \vee \neg Y \vee \neg Z)$$

$$(\neg A \vee \neg Y)$$



Conflict at level 1.

Conflict Analysis: Implication Graph

$$(A \vee B)$$

$$(B \vee C)$$

$$(\neg A \vee \neg X \vee Y)$$

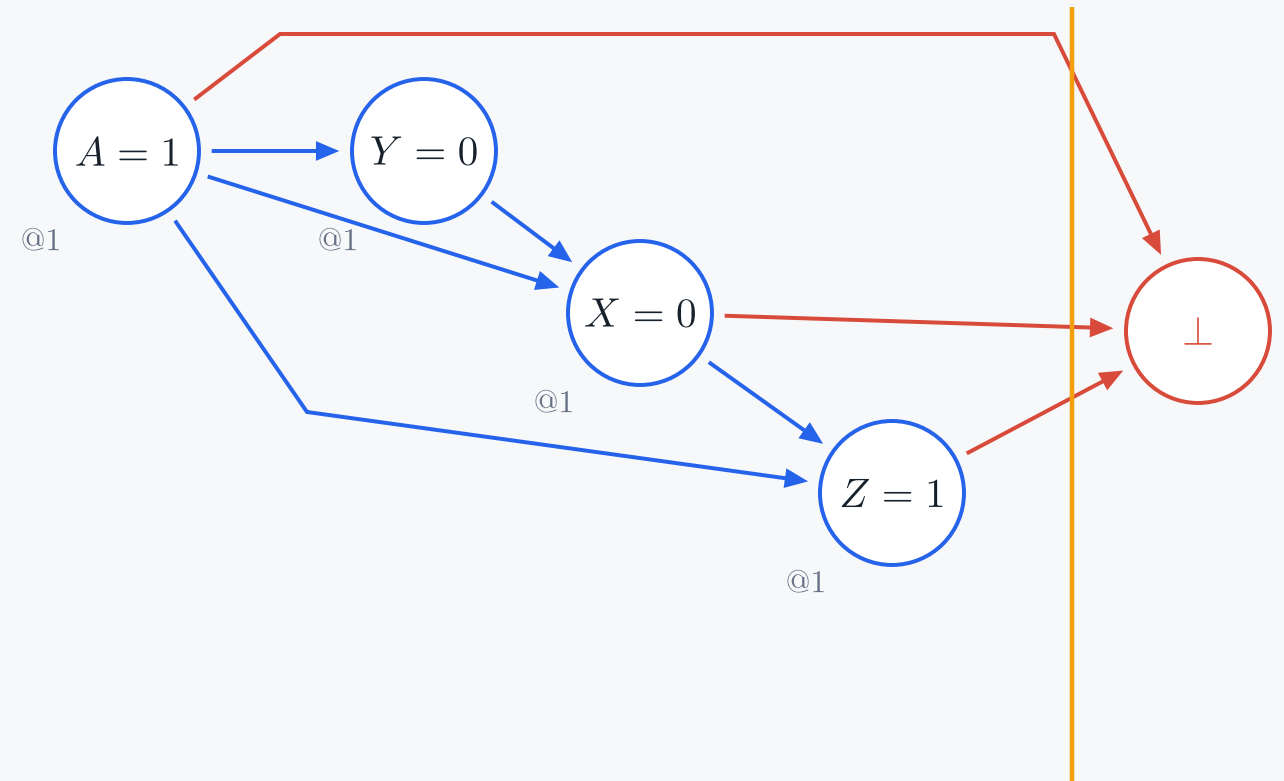
$$(\neg A \vee X \vee Z)$$

$$(\neg A \vee \neg Y \vee Z)$$

$$\boxed{(\neg A \vee X \vee \neg Z)} \Rightarrow \perp$$

$$(\neg A \vee \neg Y \vee \neg Z)$$

$$(\neg A \vee \neg Y)$$



$$\neg A \vee X \vee \neg Z$$

Conflict Analysis: Implication Graph

$$(A \vee B)$$

$$(B \vee C)$$

$$(\neg A \vee \neg X \vee Y)$$

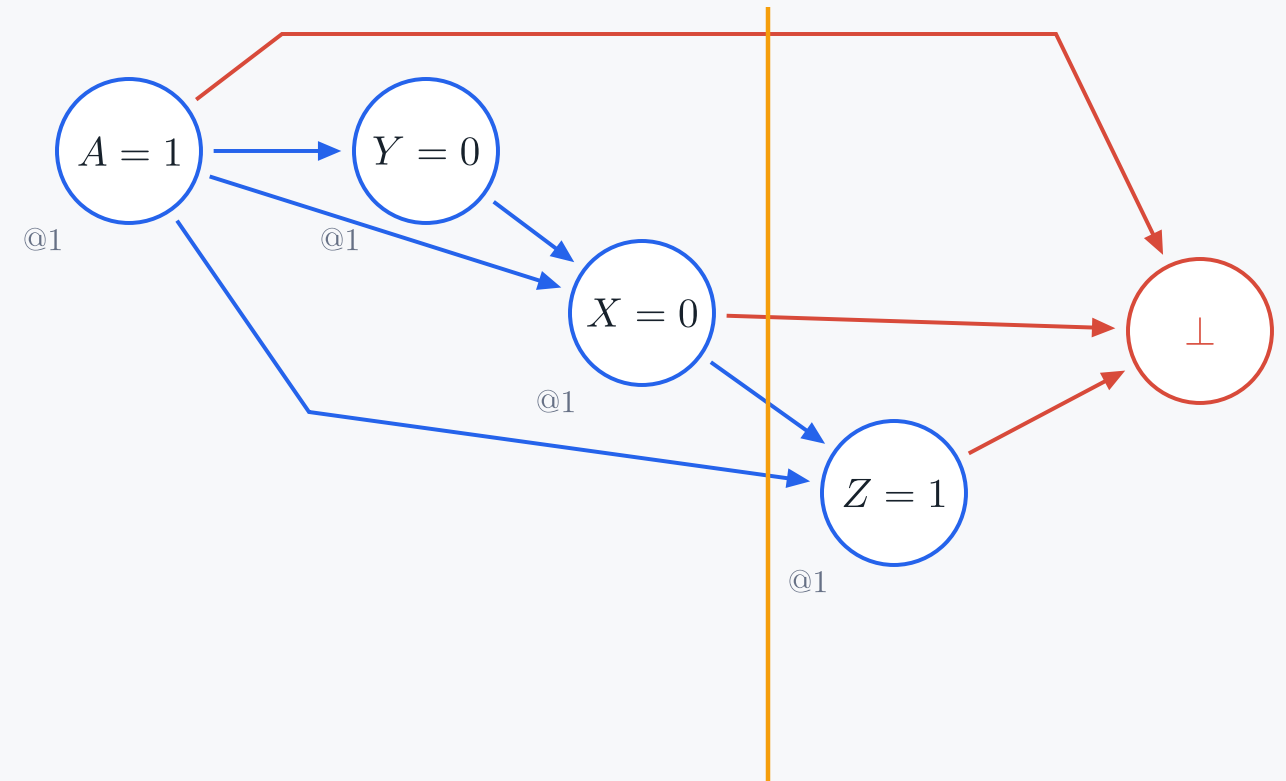
$$(\neg A \vee X \vee Z)$$

$$(\neg A \vee \neg Y \vee Z)$$

$$\boxed{(\neg A \vee X \vee \neg Z)} \Rightarrow \perp$$

$$(\neg A \vee \neg Y \vee \neg Z)$$

$$(\neg A \vee \neg Y)$$



$$\neg A \vee X$$

Conflict Analysis: Implication Graph

$$(A \vee B)$$

$$(B \vee C)$$

$$(\neg A \vee \neg X \vee Y)$$

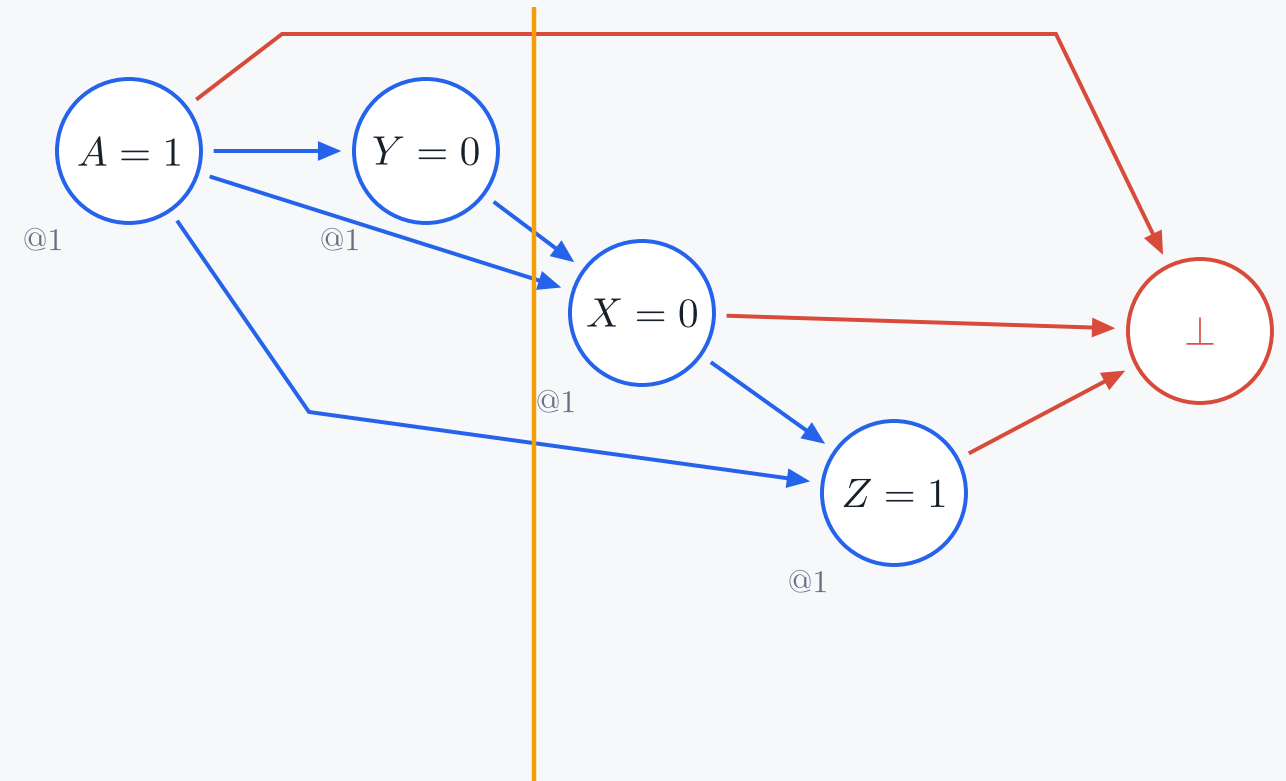
$$(\neg A \vee X \vee Z)$$

$$(\neg A \vee \neg Y \vee Z)$$

$$(\neg A \vee X \vee \neg Z) \Rightarrow \perp$$

$$(\neg A \vee \neg Y \vee \neg Z)$$

$$(\neg A \vee \neg Y)$$



$$\neg A \vee Y$$

Conflict Analysis: Implication Graph

$$(A \vee B)$$

$$(B \vee C)$$

$$(\neg A \vee \neg X \vee Y)$$

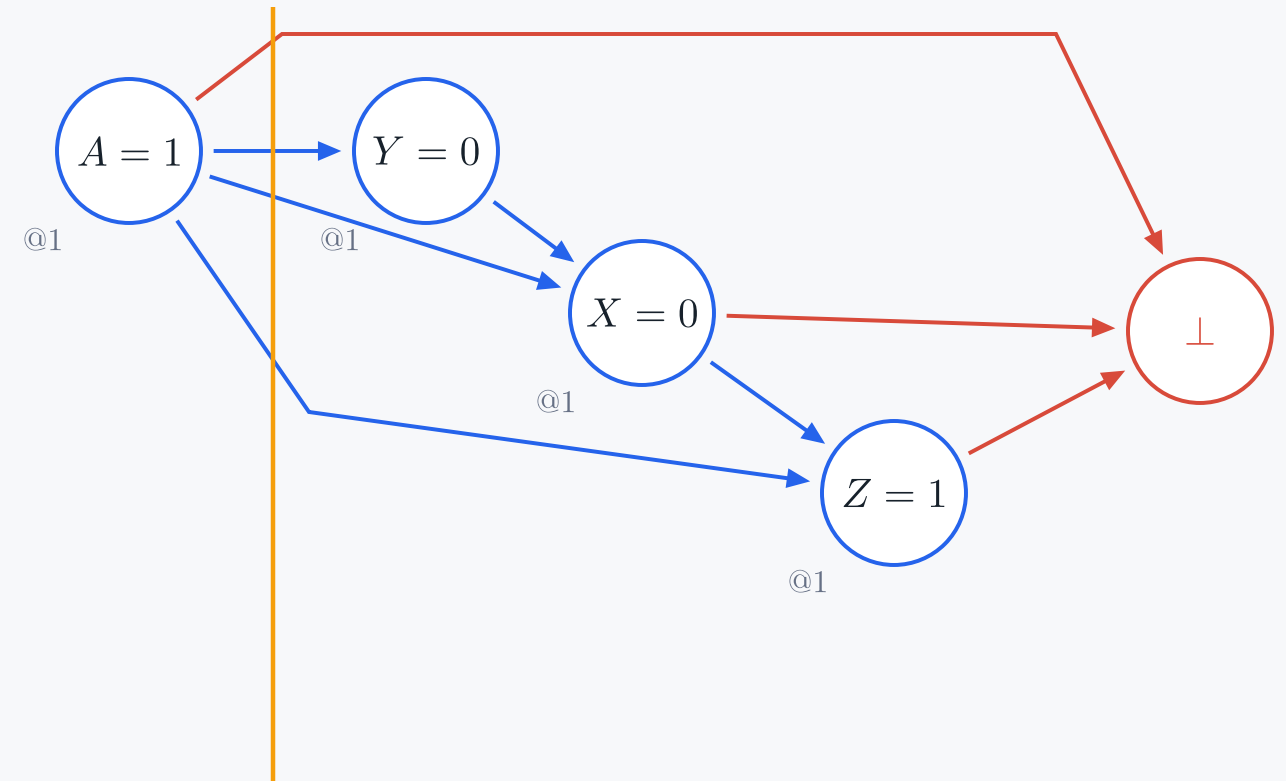
$$(\neg A \vee X \vee Z)$$

$$(\neg A \vee \neg Y \vee Z)$$

$$(\neg A \vee X \vee \neg Z) \Rightarrow \perp$$

$$(\neg A \vee \neg Y \vee \neg Z)$$

$$(\neg A \vee \neg Y)$$



$$\neg A$$

Conflict Analysis: Implication Graph

$(A \vee B)$

$(B \vee C)$

$(\neg A \vee \neg X \vee Y)$

$(\neg A \vee X \vee Z)$

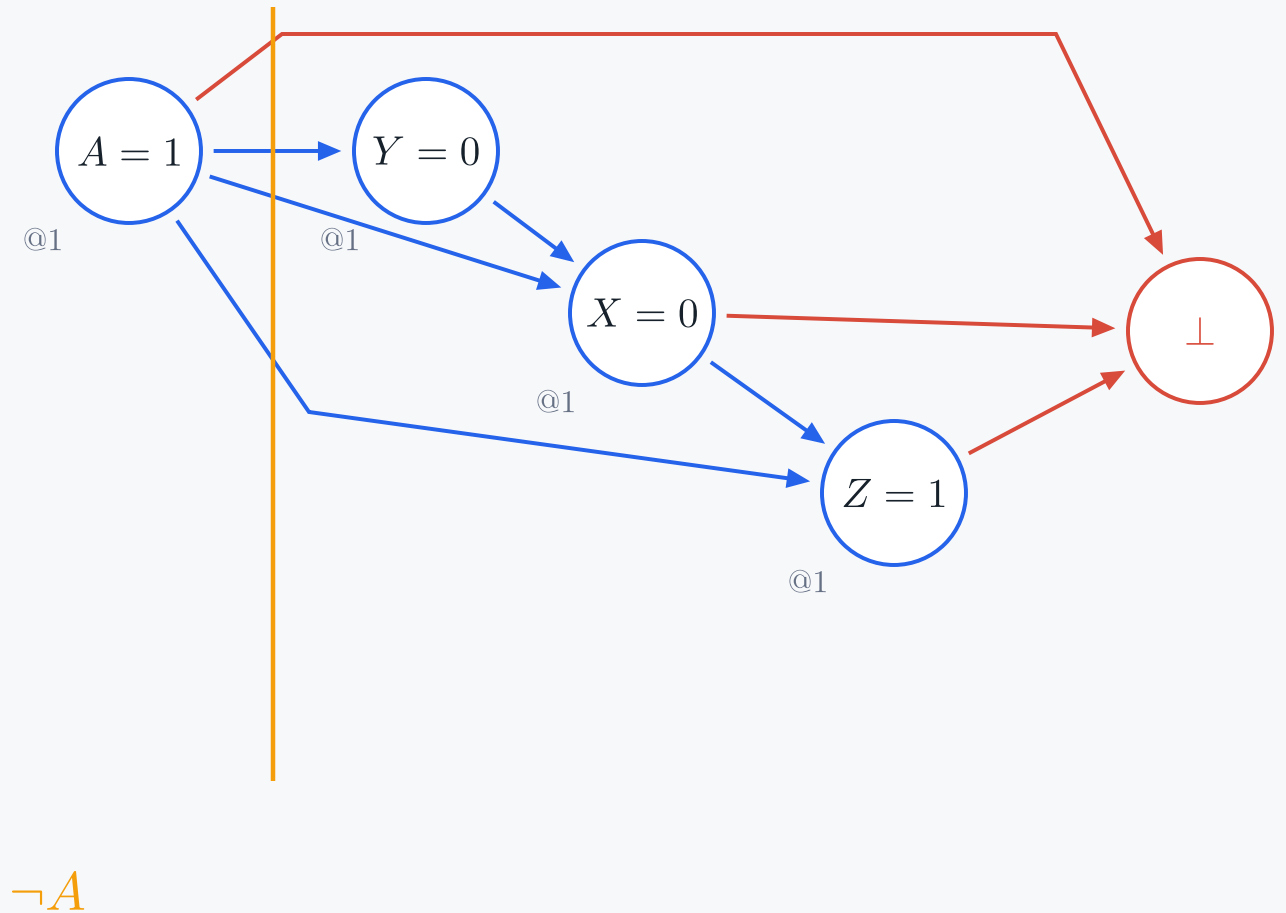
$(\neg A \vee \neg Y \vee Z)$

$(\neg A \vee X \vee \neg Z) \Rightarrow \perp$

$(\neg A \vee \neg Y \vee \neg Z)$

$(\neg A \vee \neg Y)$

$(\neg A)$



Conflict Analysis: Implication Graph

$(A \vee B)$

$(B \vee C)$

$(\neg A \vee \neg X \vee Y)$

$(\neg A \vee X \vee Z)$

$(\neg A \vee \neg Y \vee Z)$

$(\neg A \vee X \vee \neg Z)$

$(\neg A \vee \neg Y \vee \neg Z)$

$(\neg A \vee \neg Y)$

$(\neg A)$

Backjump to level 0.
Undo every assignment.

Conflict Analysis: Implication Graph

$$(A \vee B)$$

$$(B \vee C)$$

$$(\neg A \vee \neg X \vee Y)$$

$$(\neg A \vee X \vee Z)$$

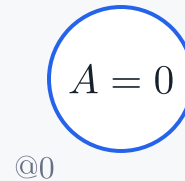
$$(\neg A \vee \neg Y \vee Z)$$

$$(\neg A \vee X \vee \neg Z)$$

$$(\neg A \vee \neg Y \vee \neg Z)$$

$$(\neg A \vee \neg Y)$$

$$(\neg A)$$



Propagate $A = 0$ at level 0.

Conflict Analysis: Implication Graph

$(A \vee B)$

$(B \vee C)$

$(\neg A \vee \neg X \vee Y)$

$(\neg A \vee X \vee Z)$

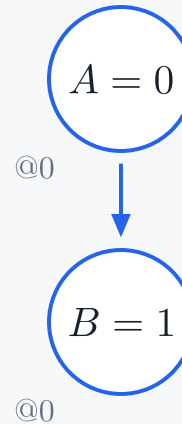
$(\neg A \vee \neg Y \vee Z)$

$(\neg A \vee X \vee \neg Z)$

$(\neg A \vee \neg Y \vee \neg Z)$

$(\neg A \vee \neg Y)$

$(\neg A)$



Propagate $B = 1$ at level 0.

Conflict Analysis: Implication Graph

$$(A \vee B)$$

$$(B \vee C)$$

$$(\neg A \vee \neg X \vee Y)$$

$$(\neg A \vee X \vee Z)$$

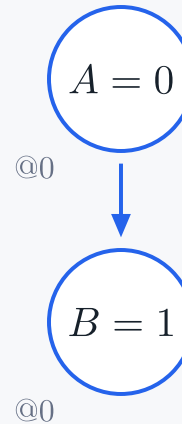
$$(\neg A \vee \neg Y \vee Z)$$

$$(\neg A \vee X \vee \neg Z)$$

$$(\neg A \vee \neg Y \vee \neg Z)$$

$$(\neg A \vee \neg Y)$$

$$(\neg A)$$



SAT: every clause is satisfied.

One complete satisfying assignment:

$$A = 0, \quad B = 1, \quad C = X = Y = Z = 0$$

How humans think...

```
def ProblemSolving(F: CNF) -> bool:
    Decisions = {} # Partial assignment
    while not all variables are assigned in Decisions:
        if conflict := UnitPropagation(F, Decisions):
            C,  $\ell$  = ConflictAnalysis(conflict)
            if C ==  $\perp$ : return UNSAT
            F.add(C)
            Decisions.drop_after( $\ell$ ) # Remove decisions after  $\ell$ 
        else:
             $\ell$  = make a decision
            Decisions.add( $\ell$ ) # Make a decision
    return SAT
```

How humans think...

```
def ProblemSolving(F: CNF) -> bool:
    Decisions = {} # Partial assignment
    while not all variables are assigned in Decisions:
        if conflict := UnitPropagation(F, Decisions):
            C,  $\ell$  = ConflictAnalysis(conflict)
            if C ==  $\perp$ : return UNSAT
            F.add(C)
            Decisions.drop_after( $\ell$ ) # Remove decisions after  $\ell$ 
        else:
             $\ell$  = DecisionHeuristic()
            Decisions.add( $\ell$ ) # Make a decision
    return SAT
```

Which variable to pick next?

Decision Heuristics: Pick an unassigned variable

MOMS

Too expensive to compute.

Maximum Occurrences in clauses of Minimum Size

Choose the variable occurring most often
in the shortest remaining clauses.

Decision Heuristics: Pick an unassigned variable

MOMS

Too expensive to compute.

Maximum Occurrences in clauses of Minimum Size

Choose the variable occurring most often
in the shortest remaining clauses.

Static

A, B, X, Y, Z

calculate once before solving

Decision Heuristics: Pick an unassigned variable

MOMS

Too expensive to compute.

Maximum Occurrences in clauses of Minimum Size

Choose the variable occurring most often
in the shortest remaining clauses.

Static

A, B, X, Y, Z

calculate once before solving

INC

$$s(v) \leftarrow s(v) + 1$$

EVSIDS

$$s(v) \leftarrow s(v) + \Delta$$

$$\Delta \leftarrow \Delta / \alpha \quad (0 < \alpha < 1)$$

Exponential Variable State Independent Decaying Sum

Decision Heuristics: Pick an unassigned variable

MOMS

Too expensive to compute.

Maximum Occurrences in clauses of Minimum Size

Choose the variable occurring most often
in the shortest remaining clauses.

Static

A, B, X, Y, Z

calculate once before solving

INC

$$s(v) \leftarrow s(v) + 1$$

EVSIDS

$$s(v) \leftarrow s(v) + \Delta$$

$$\Delta \leftarrow \Delta / \alpha \quad (0 < \alpha < 1)$$

Exponential Variable State Independent Decaying Sum



SAT Competition 2014

[Biere–Fröhlich, SAT '15]

Conflict-Driven Clause Learning

[Marques-Silva & Sakallah '96]

```
def ProblemSolving(F: CNF) -> bool:
    Decisions = {} # Partial assignment
    while not all variables are assigned in Decisions:
        if conflict := UnitPropagation(F, Decisions):
            C,  $\ell$  = ConflictAnalysis(conflict)
            if C ==  $\perp$ : return UNSAT
            F.add(C)
            Decisions.drop_after( $\ell$ ) # Remove decisions after  $\ell$ 
        else:
             $\ell$  = DecisionHeuristic()
            Decisions.add( $\ell$ ) # Make a decision
    return SAT
```

Conflict-Driven Clause Learning

[Marques-Silva & Sakallah '96]

```
def CDCL(F: CNF) -> bool:
    Decisions = {} # Partial assignment
    while not all variables are assigned in Decisions:
        if conflict := UnitPropagation(F, Decisions):
            C,  $\ell$  = ConflictAnalysis(conflict)
            if  $C == \perp$ : return UNSAT
            F.add(C)
            Decisions.drop_after( $\ell$ ) # Remove decisions after  $\ell$ 
        else:
             $\ell$  = DecisionHeuristic()
            Decisions.add( $\ell$ ) # Make a decision
    return SAT
```

Code from CaDiCaL

```
int Internal::cdcl_loop_with_inprocessing () {
    int res = 0;
    while (!res) {
        if (unsat) res = 20;
        else if (!propagate ()) analyze (); // propagate and analyze
        else if (iterating) iterate (); // report learned unit
        else if (satisfied ()) res = 10; // found model
        else if (terminating ()) break; // limit hit or async abort
        else if (restarting ()) restart (); // restart by backtracking
        else if (rephasing ()) rephase (); // reset variable phases
        else if (reducing ()) reduce (); // collect useless clauses
        else if (probing ()) probe (); // failed literal probing
        else if (subsuming ()) subsume (); // subsumption algorithm
        else if (eliminating ()) elim (); // variable elimination
        else if (compacting ()) compact (); // collect variables
        else res = decide (); // next decision
    }
    return res;
}
```

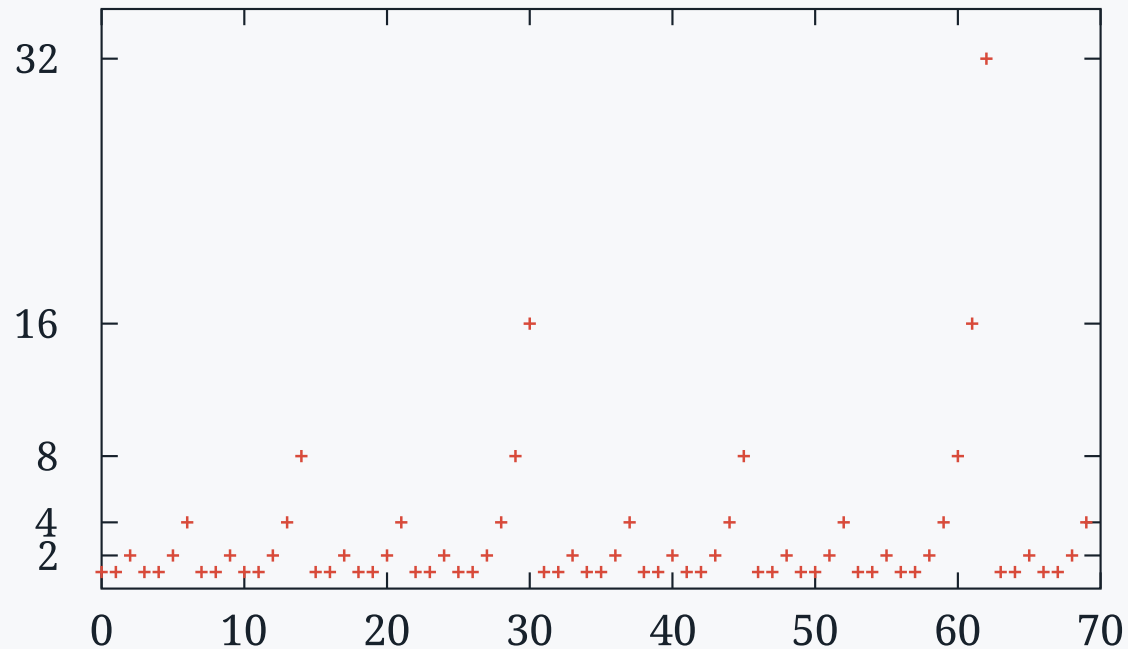
Code from CaDiCaL

```
int Internal::cdcl_loop_with_inprocessing () {
    int res = 0;
    while (!res) {
        if (unsat) res = 20;
        else if (!propagate ()) analyze (); // propagate and analyze
        else if (iterating) iterate (); // report learned unit
        else if (satisfied ()) res = 10; // found model
        else if (terminating ()) break; // limit hit or async abort
        else if (restarting ()) restart (); // restart by backtracking
        else if (rephasing ()) rephase (); // reset variable phases
        else if (reducing ()) reduce (); // collect useless clauses
        else if (probing ()) probe (); // failed literal probing
        else if (subsuming ()) subsume (); // subsumption algorithm
        else if (eliminating ()) elim (); // variable elimination
        else if (compacting ()) compact (); // collect variables
        else res = decide (); // next decision
    }
    return res;
}
```


Restarts

Restarting the search can help leave bad parts of the search space.

Luby sequence [Luby93] (provably optimal)



```
u, v = 1, 1
while True:
    print(v)
    u, v = (u + 1, 1) if (u & -u) == v
           else (u, 2 * v)
```

Donald Knuth [Knuth15]
TAOCP §7.2.2.2, p. 81, Eq. (131)
Draft of 23 September 2015

(1, 1), (2, 1), (2, 2), (3, 1), (4, 1), (4, 2), (4, 4), (5, 1), (6, 1), (6, 2), (7, 1), (8, 1), (8, 2), (8, 4), (8, 8), (9, 1), (10, 1), (10, 2), (10, 4), (10, 8), (11, 1), (12, 1), (12, 2), (12, 4), (13, 1), (14, 1), (14, 2), (14, 4), (15, 1), (16, 1), (16, 2), (16, 4), (17, 1), (18, 1), (18, 2), (18, 4), (19, 1), (20, 1), (20, 2), (20, 4), (21, 1), (22, 1), (22, 2), (22, 4), (23, 1), (24, 1), (24, 2), (24, 4), (25, 1), (26, 1), (26, 2), (26, 4), (27, 1), (28, 1), (28, 2), (28, 4), (29, 1), (30, 1), (30, 2), (30, 4), (30, 8), (31, 1), (32, 1), (32, 2), (32, 4), (33, 1), (34, 1), (34, 2), (34, 4), (35, 1), (36, 1), (36, 2), (36, 4), (37, 1), (38, 1), (38, 2), (38, 4), (39, 1), (40, 1), (40, 2), (40, 4), (41, 1), (42, 1), (42, 2), (42, 4), (43, 1), (44, 1), (44, 2), (44, 4), (45, 1), (46, 1), (46, 2), (46, 4), (47, 1), (48, 1), (48, 2), (48, 4), (49, 1), (50, 1), (50, 2), (50, 4), (51, 1), (52, 1), (52, 2), (52, 4), (53, 1), (54, 1), (54, 2), (54, 4), (55, 1), (56, 1), (56, 2), (56, 4), (57, 1), (58, 1), (58, 2), (58, 4), (59, 1), (60, 1), (60, 2), (60, 4), (60, 8), (61, 1), (61, 2), (61, 4), (61, 8), (61, 16), (61, 32), (62, 1), (63, 1), (63, 2), (63, 4), (64, 1), (65, 1), (65, 2), (65, 4), (66, 1), (67, 1), (67, 2), (67, 4), (68, 1), (69, 1), (69, 2), (69, 4), (70, 1).

Translating to CNF

Given an arbitrary propositional formula, how can we translate it into CNF?

Tseitin encoding

$$F = (x \wedge \neg y) \vee (z \vee (x \wedge \neg w))$$

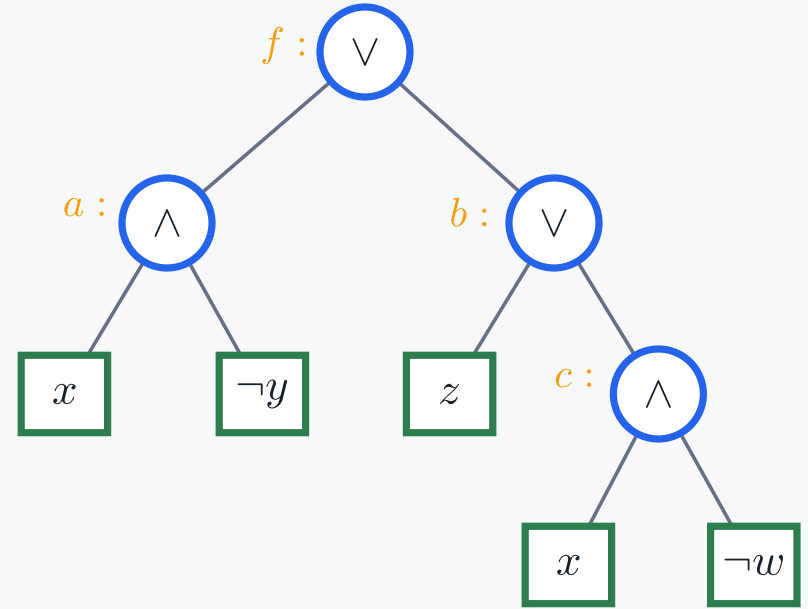
“Triplets” using subformulas

$$f \Leftrightarrow a \vee b, \quad a \Leftrightarrow x \wedge \neg y, \dots$$

Encode each triplet as an equivalence in CNF

$$(\neg f \vee a \vee b) \wedge (f \vee \neg a) \wedge (f \vee \neg b) \wedge \dots$$

Add the unit clause (f) to require the original formula F to be true.



One lecture is not enough...



Second edition 2021

Incremental SAT Solving: Assumptions and Constraints

- Incremental Solving: add additional clauses.
- Assumptions: Temporarily enforced Boolean assignment.
- Constraint: single temporary clause.

$$F_0$$

$$F_1 = F_0 \wedge C_1 \wedge C_2$$

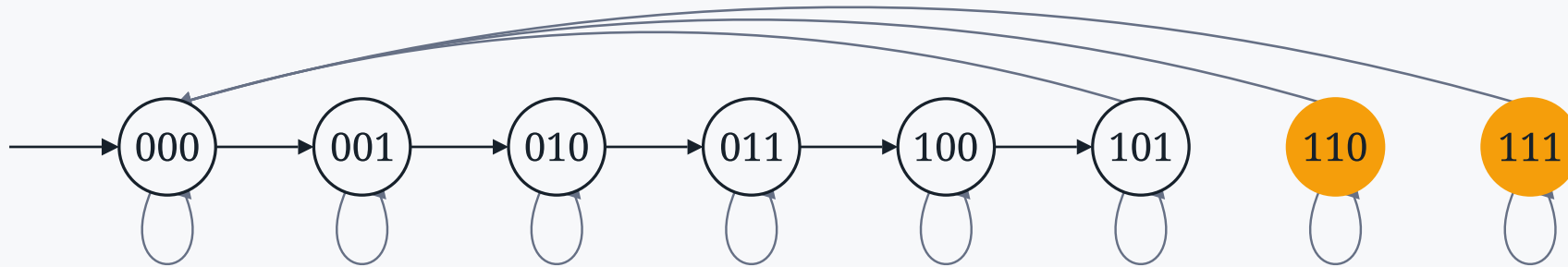
$$F_2 = F_1 \wedge C_3$$

$$\vdots$$

$$F_i \wedge \underbrace{\ell \wedge \bar{\ell'}}_{\text{Assumptions}} \wedge \underbrace{C}_{\text{Constraint}}$$

Introduction

- 3-bit ($b_2b_1b_0$) counter
- Property: Counter does not exceed 5



Introduction

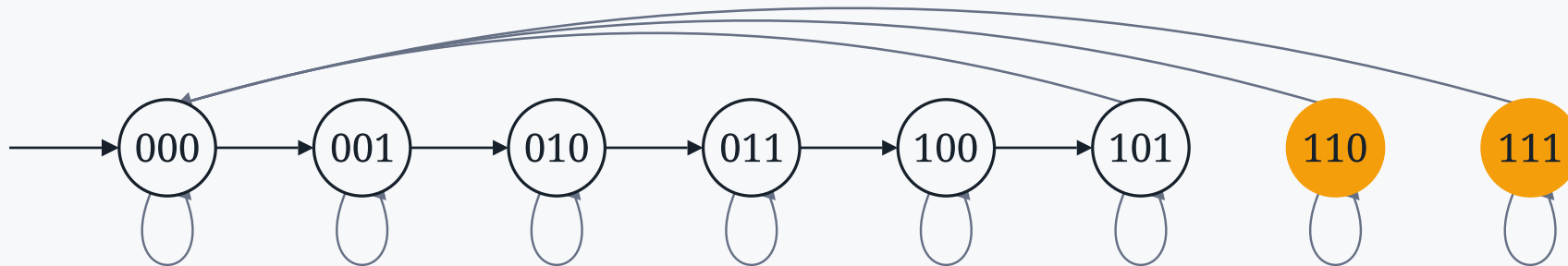
- 3-bit ($b_2b_1b_0$) counter
- Property: Counter does not exceed 5

$$\begin{aligned} T = (\bar{b}_2 \wedge \bar{b}_1 \wedge \bar{b}_0) &\Rightarrow ((\bar{b}'_2 \wedge \bar{b}'_1 \wedge \bar{b}'_0) \vee (\bar{b}'_2 \wedge \bar{b}'_1 \wedge b'_0)) \\ &\wedge (\bar{b}_2 \wedge \bar{b}_1 \wedge b_0) \Rightarrow ((\bar{b}'_2 \wedge \bar{b}'_1 \wedge b'_0) \vee (\bar{b}'_2 \wedge b'_1 \wedge \bar{b}'_0)) \\ &\wedge \dots \end{aligned}$$

- Can state 6 be reached?

$$T \wedge b'_2 \wedge b'_1 \wedge \bar{b}'_0$$

SAT



Introduction

- 3-bit ($b_2b_1b_0$) counter
- Property: Counter does not exceed 5

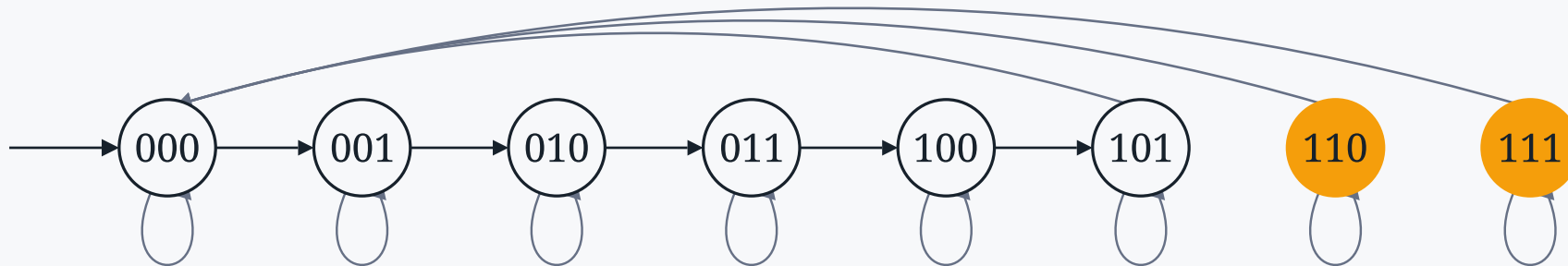
$$\begin{aligned} T &= (\bar{b}_2 \wedge \bar{b}_1 \wedge \bar{b}_0) \Rightarrow ((\bar{b}'_2 \wedge \bar{b}'_1 \wedge \bar{b}'_0) \vee (\bar{b}'_2 \wedge \bar{b}'_1 \wedge b'_0)) \\ &\quad \wedge (\bar{b}_2 \wedge \bar{b}_1 \wedge b_0) \Rightarrow ((\bar{b}'_2 \wedge \bar{b}'_1 \wedge b'_0) \vee (\bar{b}'_2 \wedge b'_1 \wedge \bar{b}'_0)) \\ &\quad \wedge \dots \end{aligned}$$

- Can state 6 be reached?

$$(\bar{b}_2 \vee \bar{b}_1 \vee b_0) \wedge T \wedge b'_2 \wedge b'_1 \wedge \bar{b}'_0$$

UNSAT

$$\Rightarrow T \wedge (\bar{b}'_2 \vee \bar{b}'_1 \vee b'_0)$$



Introduction

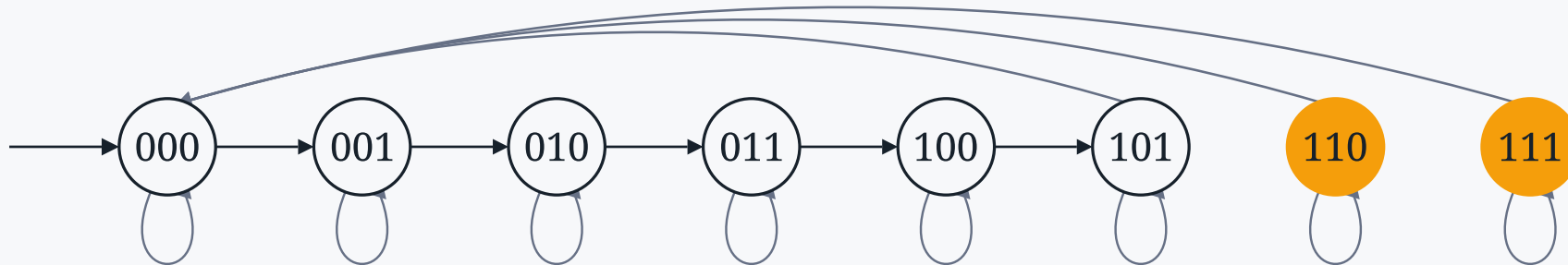
- 3-bit ($b_2b_1b_0$) counter
- Property: Counter does not exceed 5

$$\begin{aligned} T &= (\bar{b}_2 \wedge \bar{b}_1 \wedge \bar{b}_0) \Rightarrow ((\bar{b}'_2 \wedge \bar{b}'_1 \wedge \bar{b}'_0) \vee (\bar{b}'_2 \wedge \bar{b}'_1 \wedge b'_0)) \\ &\quad \wedge (\bar{b}_2 \wedge \bar{b}_1 \wedge b_0) \Rightarrow ((\bar{b}'_2 \wedge \bar{b}'_1 \wedge b'_0) \vee (\bar{b}'_2 \wedge b'_1 \wedge \bar{b}'_0)) \\ &\quad \wedge \dots \end{aligned}$$

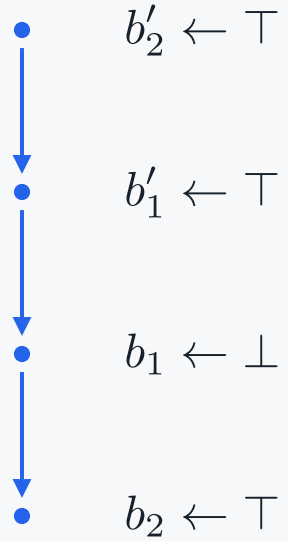
- Can state 7 be reached?

$$(\bar{b}_2 \vee \bar{b}_1 \vee \bar{b}_0) \wedge T \wedge b'_2 \wedge b'_1 \wedge b'_0$$

?



Conflict Driven Clause Learning



$$C_1 \quad \bar{b}_2 \vee b_1 \vee b_0 \vee \bar{b}'_1$$

$$C_2 \quad b_2 \vee b_1 \vee \bar{b}_0 \vee \bar{b}'_2$$

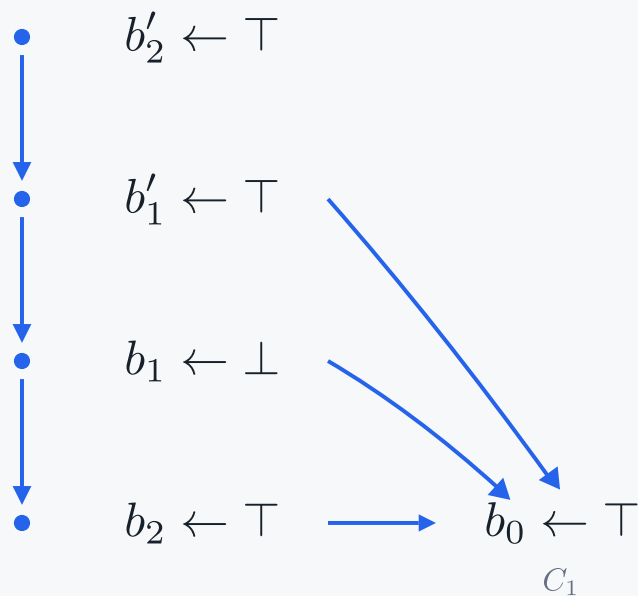
$$C_3 \quad \bar{b}_2 \vee b_1 \vee \bar{b}_0 \vee \bar{b}'_2 \vee \bar{b}'_1$$

$$C_4 \quad b_2 \vee b_1 \vee b_0 \vee \bar{b}'_2 \vee \bar{b}'_0$$

$$C_5 \quad b_2 \vee b_1 \vee b_0 \vee \bar{b}'_1 \vee b'_0$$

$\dots$

Conflict Driven Clause Learning



$$C_1 \quad \bar{b}_2 \vee b_1 \vee b_0 \vee \bar{b}'_1$$

$$C_2 \quad b_2 \vee b_1 \vee \bar{b}_0 \vee \bar{b}'_2$$

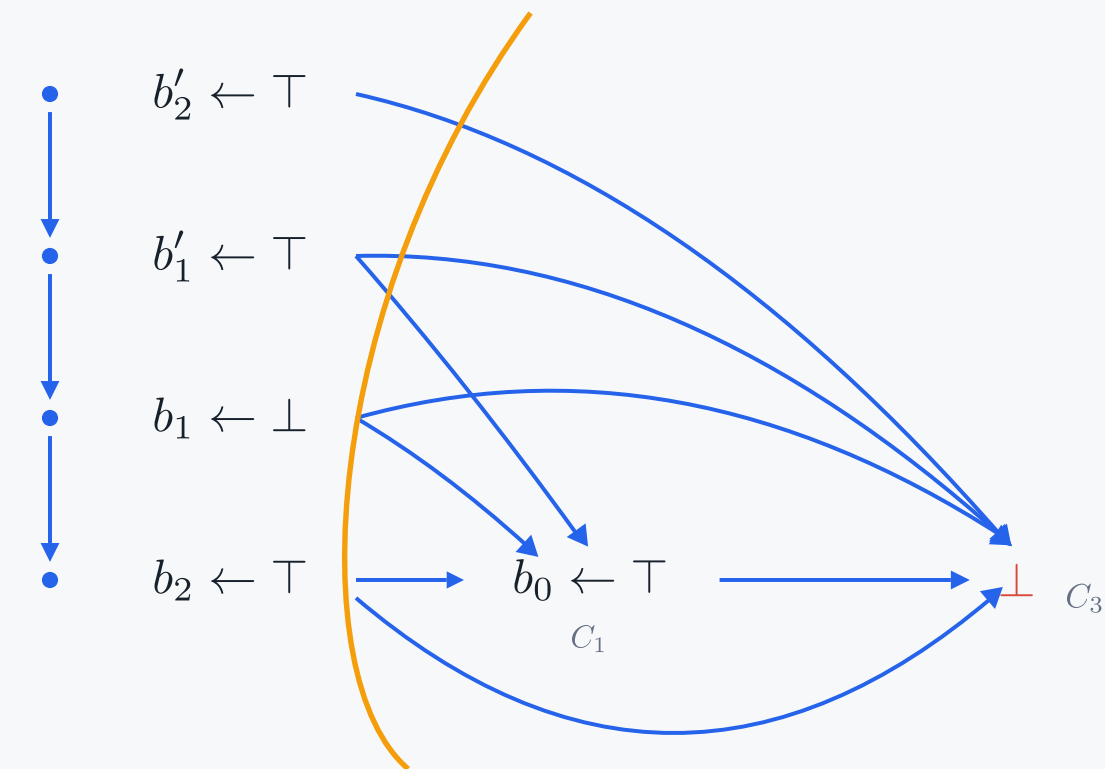
$$C_3 \quad \bar{b}_2 \vee b_1 \vee \bar{b}_0 \vee \bar{b}'_2 \vee \bar{b}'_1$$

$$C_4 \quad b_2 \vee b_1 \vee b_0 \vee \bar{b}'_2 \vee \bar{b}'_0$$

$$C_5 \quad b_2 \vee b_1 \vee b_0 \vee \bar{b}'_1 \vee b'_0$$

$\dots$

Conflict Driven Clause Learning



$$C_1 \quad \bar{b}_2 \vee b_1 \vee b_0 \vee \bar{b}'_1$$

$$C_2 \quad b_2 \vee b_1 \vee \bar{b}_0 \vee \bar{b}'_2$$

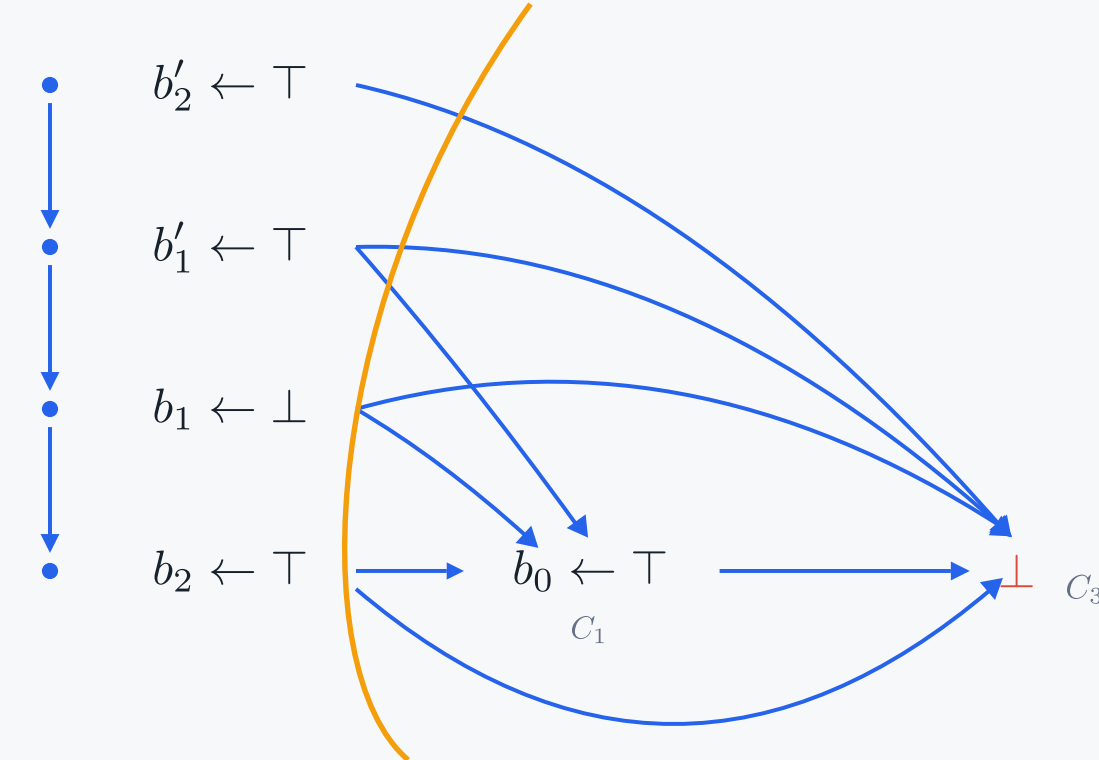
$$C_3 \quad \bar{b}_2 \vee b_1 \vee \bar{b}_0 \vee \bar{b}'_2 \vee \bar{b}'_1$$

$$C_4 \quad b_2 \vee b_1 \vee b_0 \vee \bar{b}'_2 \vee \bar{b}'_0$$

$$C_5 \quad b_2 \vee b_1 \vee b_0 \vee \bar{b}'_1 \vee b'_0$$

...

Conflict Driven Clause Learning



$$C_1 \quad \bar{b}_2 \vee b_1 \vee b_0 \vee \bar{b}'_1$$

$$C_2 \quad b_2 \vee b_1 \vee \bar{b}_0 \vee \bar{b}'_2$$

$$C_6 \quad \bar{b}'_2 \vee \bar{b}'_1 \vee b_1 \vee \bar{b}_2$$

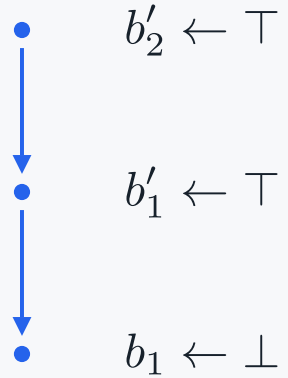
$$C_3 \quad \bar{b}_2 \vee b_1 \vee \bar{b}_0 \vee \bar{b}'_2 \vee \bar{b}'_1$$

$$C_4 \quad b_2 \vee b_1 \vee b_0 \vee \bar{b}'_2 \vee \bar{b}'_0$$

$$C_5 \quad b_2 \vee b_1 \vee b_0 \vee \bar{b}'_1 \vee b'_0$$

...

Conflict Driven Clause Learning



$$C_1 \quad \bar{b}_2 \vee b_1 \vee b_0 \vee \bar{b}'_1$$

$$C_2 \quad b_2 \vee b_1 \vee \bar{b}_0 \vee \bar{b}'_2$$

$$C_6 \quad \bar{b}'_2 \vee \bar{b}'_1 \vee b_1 \vee \bar{b}_2$$

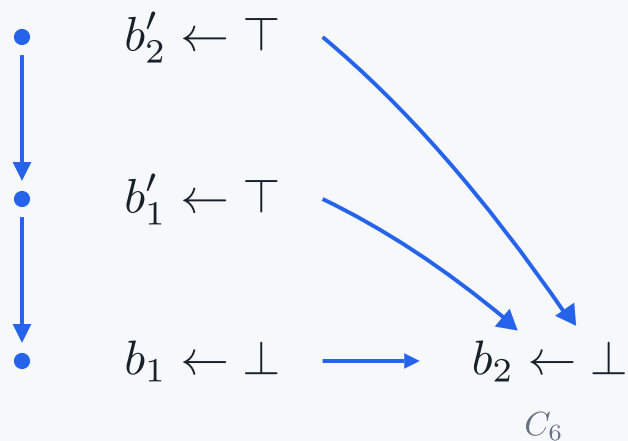
$$C_3 \quad \bar{b}_2 \vee b_1 \vee \bar{b}_0 \vee \bar{b}'_2 \vee \bar{b}'_1$$

$$C_4 \quad b_2 \vee b_1 \vee b_0 \vee \bar{b}'_2 \vee \bar{b}'_0$$

$$C_5 \quad b_2 \vee b_1 \vee b_0 \vee \bar{b}'_1 \vee b'_0$$

$\dots$

Conflict Driven Clause Learning



$$C_1 \quad \bar{b}_2 \vee b_1 \vee b_0 \vee \bar{b}'_1$$

$$C_2 \quad b_2 \vee b_1 \vee \bar{b}_0 \vee \bar{b}'_2$$

$$C_6 \quad \bar{b}'_2 \vee \bar{b}'_1 \vee b_1 \vee \bar{b}_2$$

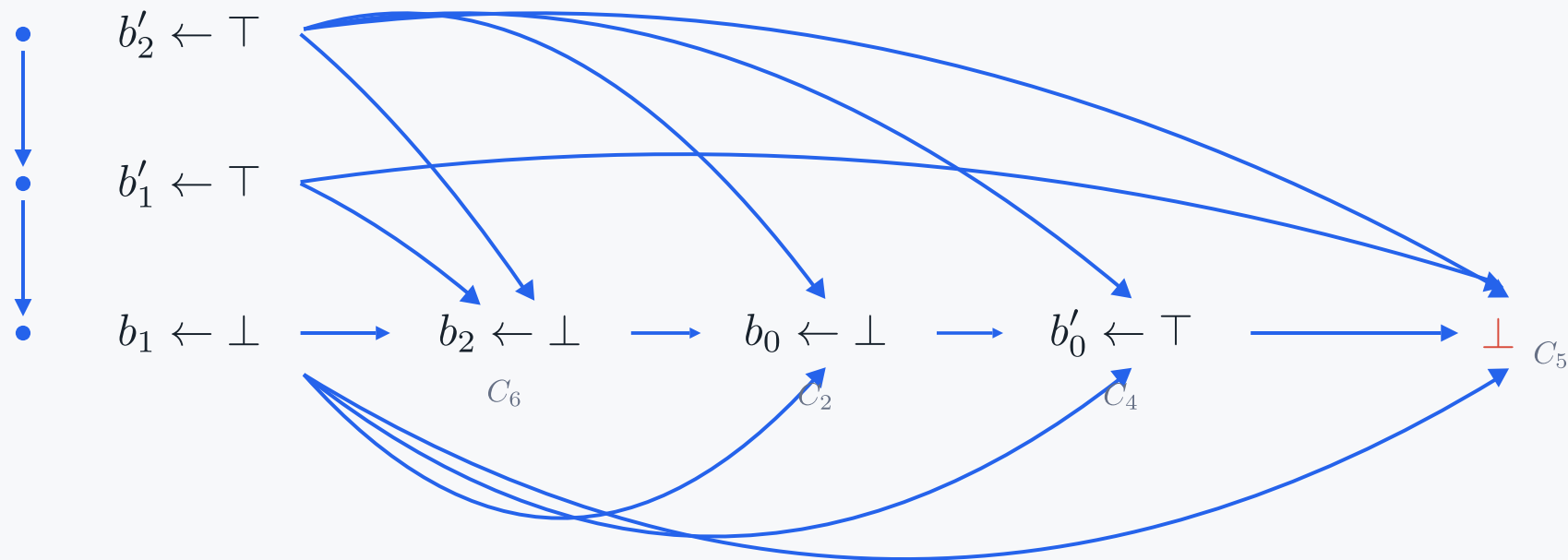
$$C_3 \quad \bar{b}_2 \vee b_1 \vee \bar{b}_0 \vee \bar{b}'_2 \vee \bar{b}'_1$$

$$C_4 \quad b_2 \vee b_1 \vee b_0 \vee \bar{b}'_2 \vee \bar{b}'_0$$

$$C_5 \quad b_2 \vee b_1 \vee b_0 \vee \bar{b}'_1 \vee b'_0$$

...

Conflict Driven Clause Learning



$$C_1 \quad \bar{b}_2 \vee b_1 \vee b_0 \vee \bar{b}_1'$$

$$C_2 \quad b_2 \vee b_1 \vee \bar{b}_0 \vee \bar{b}_2'$$

$$C_6 \quad \bar{b}_2' \vee \bar{b}_1' \vee b_1 \vee \bar{b}_2$$

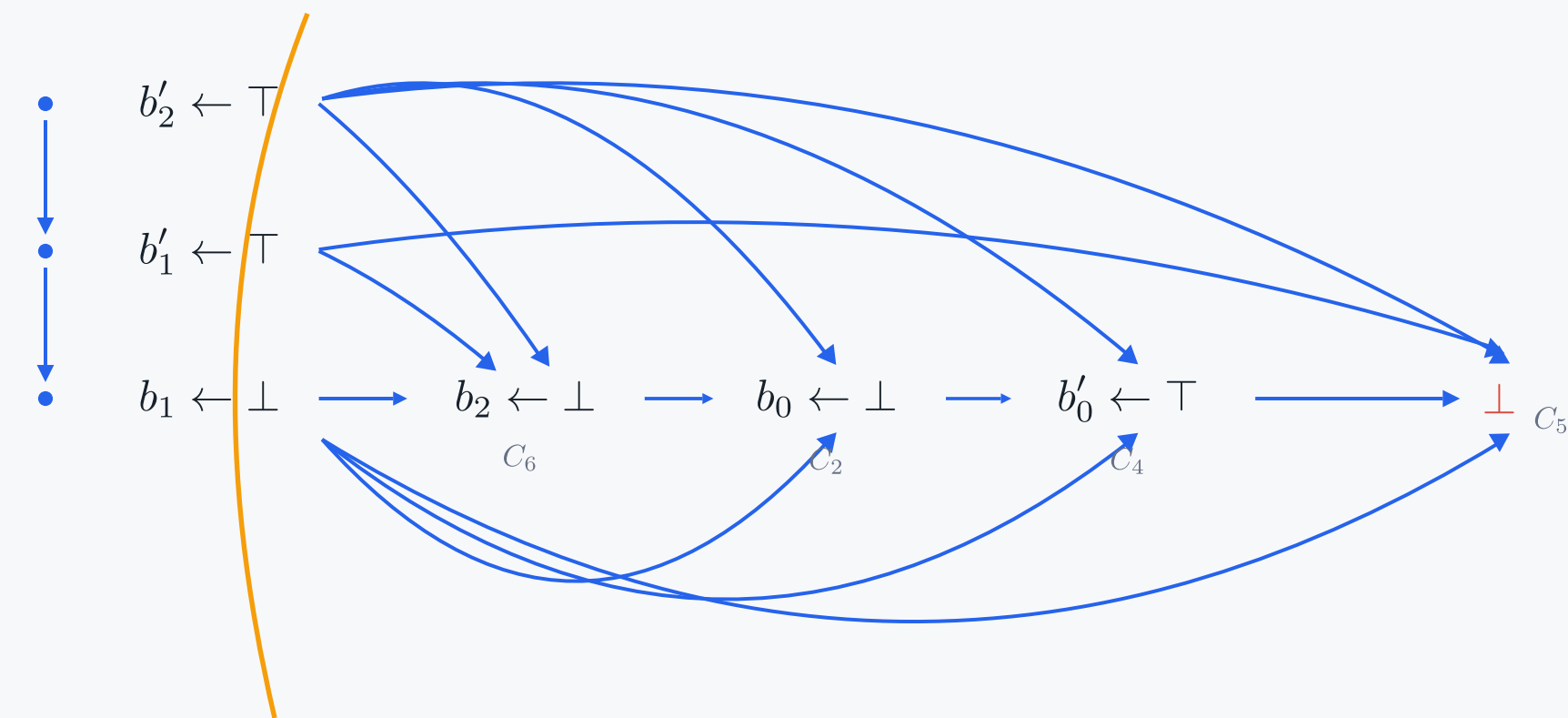
$$C_3 \quad \bar{b}_2 \vee b_1 \vee \bar{b}_0 \vee \bar{b}_2' \vee \bar{b}_1'$$

$$C_4 \quad b_2 \vee b_1 \vee b_0 \vee \bar{b}_2' \vee \bar{b}_0'$$

$$C_5 \quad b_2 \vee b_1 \vee b_0 \vee \bar{b}_1' \vee b_0'$$

...

Conflict Driven Clause Learning



$$C_1 \quad \bar{b}_2 \vee b_1 \vee b_0 \vee \bar{b}_1'$$

$$C_2 \quad b_2 \vee b_1 \vee \bar{b}_0 \vee \bar{b}_2'$$

$$C_6 \quad \bar{b}_2' \vee \bar{b}_1' \vee b_1 \vee \bar{b}_2$$

$$C_3 \quad \bar{b}_2 \vee b_1 \vee \bar{b}_0 \vee \bar{b}_2' \vee \bar{b}_1'$$

$$C_4 \quad b_2 \vee b_1 \vee b_0 \vee \bar{b}_2' \vee \bar{b}_0'$$

$$C_7 \quad \bar{b}_2' \vee \bar{b}_1' \vee b_1$$

$$C_5 \quad b_2 \vee b_1 \vee b_0 \vee \bar{b}_1' \vee b_0'$$

...

Counter Example

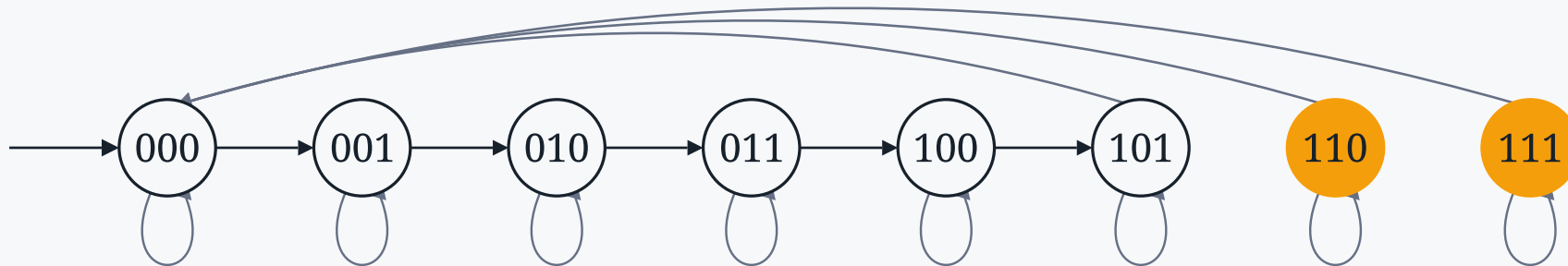
- Three-bit ($b_2b_1b_0$) counter
- Property: Counter does not exceed 5

$$\begin{aligned} T &= (\bar{b}_2 \wedge \bar{b}_1 \wedge \bar{b}_0) \Rightarrow ((\bar{b}'_2 \wedge \bar{b}'_1 \wedge \bar{b}'_0) \vee (\bar{b}'_2 \wedge \bar{b}'_1 \wedge b'_0)) \\ &\quad \wedge (\bar{b}_2 \wedge \bar{b}_1 \wedge b_0) \Rightarrow ((\bar{b}'_2 \wedge \bar{b}'_1 \wedge b'_0) \vee (\bar{b}'_2 \wedge b'_1 \wedge \bar{b}'_0)) \\ &\quad \wedge \dots \end{aligned}$$

- Can state 7 be reached?

$$(\bar{b}_2 \vee \bar{b}_1 \vee \bar{b}_0) \wedge T \wedge b'_2 \wedge b'_1 \wedge b'_0$$

?



Counter Example

- Three-bit ($b_2b_1b_0$) counter
- Property: Counter does not exceed 5

$$\begin{aligned} T &= (\bar{b}_2 \wedge \bar{b}_1 \wedge \bar{b}_0) \Rightarrow ((\bar{b}'_2 \wedge \bar{b}'_1 \wedge \bar{b}'_0) \vee (\bar{b}'_2 \wedge \bar{b}'_1 \wedge b'_0)) \\ &\quad \wedge (\bar{b}_2 \wedge \bar{b}_1 \wedge b_0) \Rightarrow ((\bar{b}'_2 \wedge \bar{b}'_1 \wedge b'_0) \vee (\bar{b}'_2 \wedge b'_1 \wedge \bar{b}'_0)) \\ &\quad \wedge \dots \end{aligned}$$

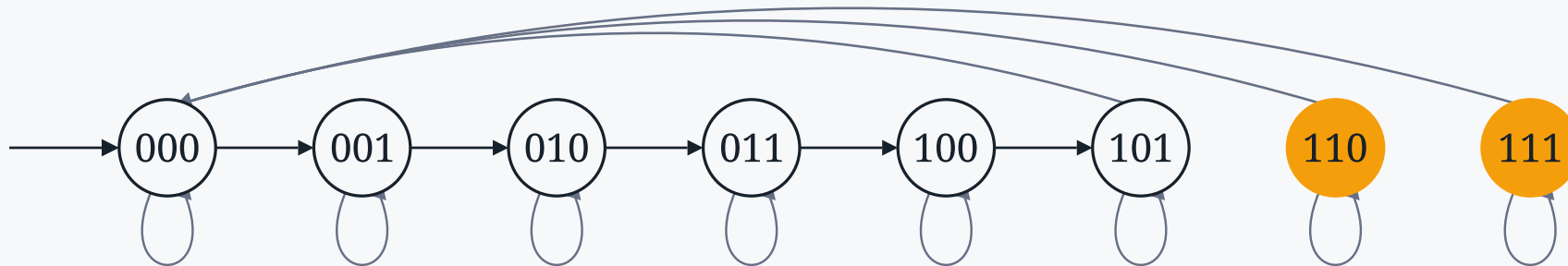
- Can the cube be reached?

$$110 \quad (\bar{b}_2 \vee \bar{b}_1 \vee b_0) \wedge T \wedge b'_2 \wedge b'_1 \wedge \bar{b}'_0$$

UNSAT

$$11_ \quad (\bar{b}_2 \vee \bar{b}_1) \wedge T \wedge b'_2 \wedge b'_1$$

UNSAT



Assumptions

$$\underbrace{(\bar{b}_2 \vee \bar{b}_1 \vee b_0)}_{\text{Constraint}} \wedge T \wedge \underbrace{b'_2 \wedge b'_1 \wedge \bar{b}'_0}_{\text{Assumptions}}$$

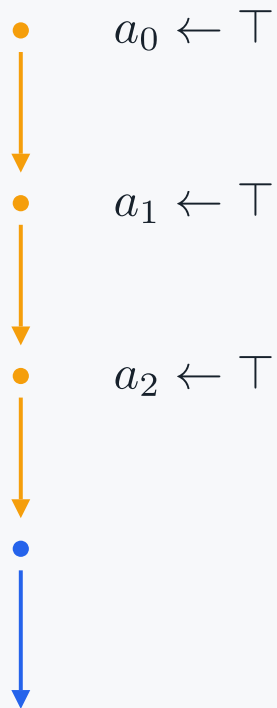
addClause (Clause c)

solve (list < Literal > assumptions)

“[...] we propose the following interface [...] very simple to implement [...] expressive enough to encompass several interesting incremental SAT-problems [...]”

Temporal Induction by Incremental SAT Solving
[Eén, Niklas and Sörensson, Niklas, 2003]

Assumptions: decisions



DECIDE ()

- 1 if level < |assumptions|
- 2 $\ell = \text{assumptions}[\text{level}]$
- 3 if $\text{val}(\ell) = \text{false}$ analyzeFinal(ℓ)
- 4 else decisions[level++] = ℓ
- 5 else decisions[level++] = heuristic()

My Contribution

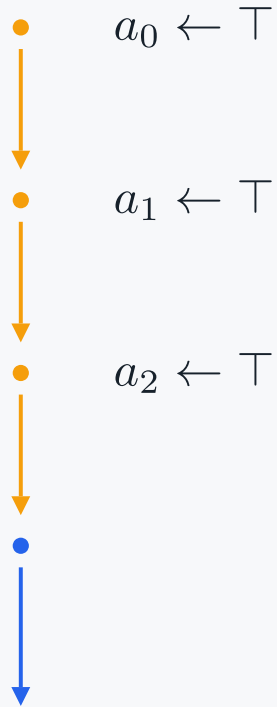
$$\underbrace{(\bar{b}_2 \vee \bar{b}_1 \vee b_0)}_{\text{Constraint}} \wedge T \wedge \underbrace{b'_2 \wedge b'_1 \wedge \bar{b}'_0}_{\text{Assumptions}}$$

addClause (Clause c)

solve (list < Literal > assumptions, Clause c)

- Easy to implement
- Expressive enough for IC3
- Learned clauses are save to keep

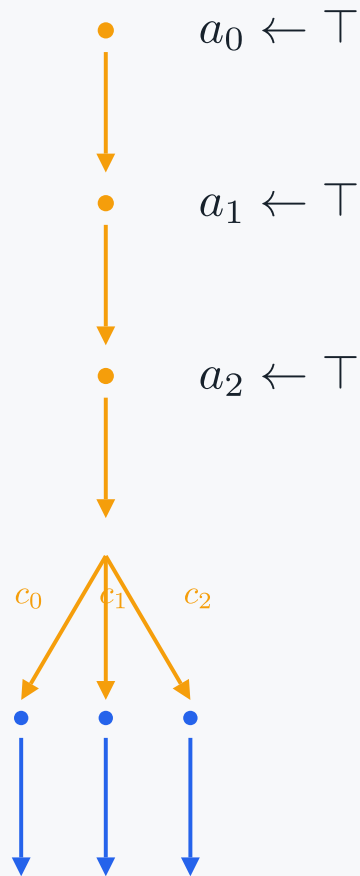
Constraint



DECIDE ()

```
1 if level < |assumptions|  
2    $\ell = \text{assumptions}[\text{level}]$   
3   if  $\text{val}(\ell) = \text{false}$   $\text{analyzeFinal}(\ell)$   
4   else  $\text{decisions}[\text{level}++] = \ell$   
5 else  $\text{decisions}[\text{level}++] = \text{heuristic}()$ 
```

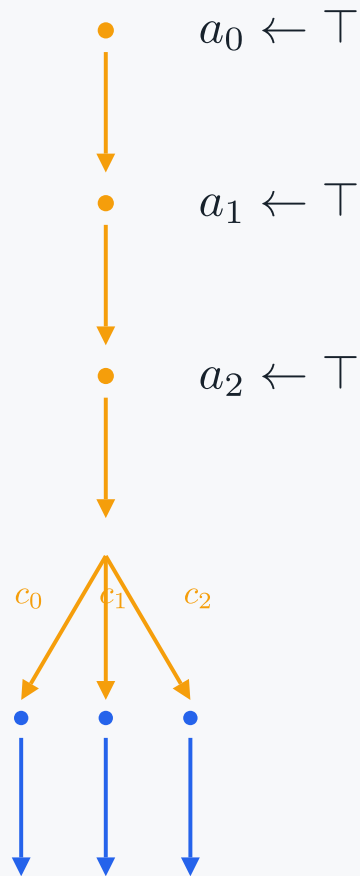
Constraint



DECIDE ()

```
1  if level < |assumptions|
2       $\ell = \text{assumptions}[\text{level}]$ 
3      if  $\text{val}(\ell) = \text{false}$  analyzeFinal( $\ell$ )
4      else  $\text{decisions}[\text{level}++] = \ell$ 
5  else if level = |assumptions|
6      unassignedLit = 0
7      for  $\ell$  in constraint
8          if  $\text{val}(\ell) = \text{true}$ 
9              constraint.moveToFront( $\ell$ )
10             level++
11         else if  $\text{val}(\ell) = \text{unassigned}$ 
12             unassignedLit =  $\ell$ 
13     if unassignedLit = 0
14         analyzeFinal(constraint)
15     else  $\text{decisions}[\text{level}++] = \text{unassignedLit}$ 
16 else  $\text{decisions}[\text{level}++] = \text{heuristic}()$ 
```

Constraint



DECIDE ()

```
1  if level < |assumptions|
2       $\ell$  = assumptions[level]
3      if val( $\ell$ ) = false analyzeFinal( $\ell$ )
4      else decisions[level++] =  $\ell$ 
5  else if level = |assumptions|
6      unassignedLit = 0
7      for  $\ell$  in constraint
8          if val( $\ell$ ) = true
9              constraint.moveToFront( $\ell$ )
10             level++
11         else if val( $\ell$ ) = unassigned
12             unassignedLit =  $\ell$ 
13     if unassignedLit = 0
14         analyzeFinal(constraint)
15     else decisions[level++] = unassignedLit
16 else decisions[level++] = heuristic()
```


Analyze Final

failed(Literal ℓ)

Queries if ℓ was used to proof unsatisfiability

If only b'_2 and b'_1 failed the second cube is free

$$110 \quad (\bar{b}_2 \vee \bar{b}_1 \vee b_0) \wedge T \wedge b'_2 \wedge b'_1 \wedge \cancel{\bar{b}'_0}$$

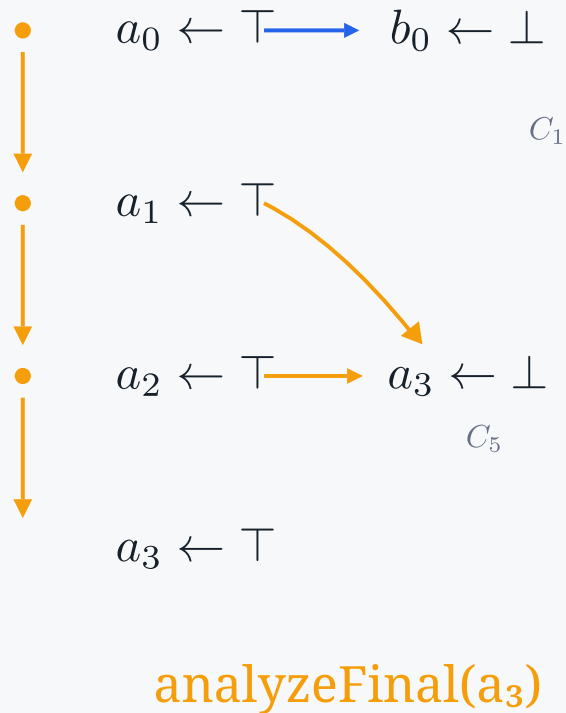
UNSAT



$$11_- \quad (\bar{b}_2 \vee \bar{b}_1) \wedge T \wedge b'_2 \wedge b'_1$$

UNSAT

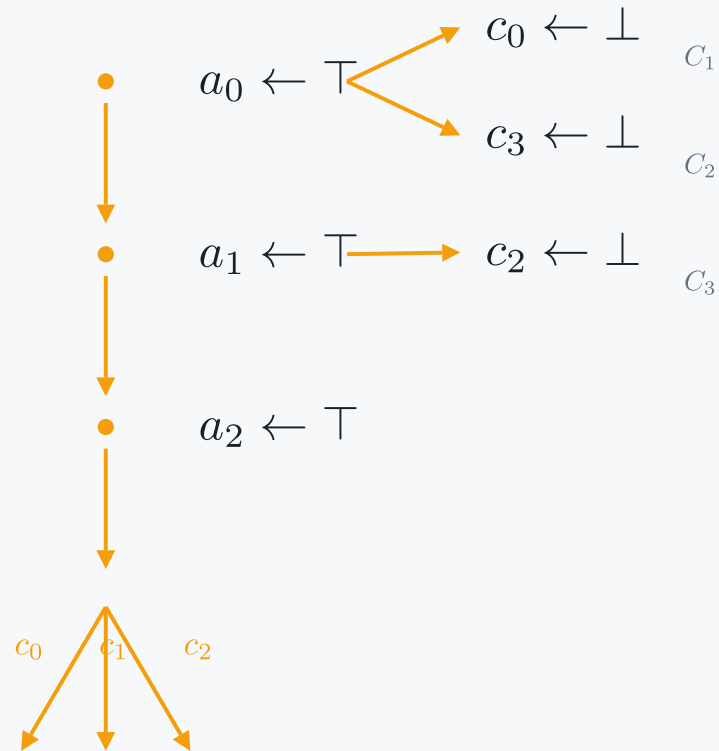
Analyze Final: algorithm



DECIDE ()

```
1  if level < |assumptions|
2     $\ell = \text{assumptions}[\text{level}]$ 
3    if val( $\ell$ ) = false analyzeFinal( $\ell$ )
4    else decisions[level++] =  $\ell$ 
5  else if level = |assumptions|
6    unassignedLit = 0
7    for  $\ell$  in constraint
8      if val( $\ell$ ) = true
9        constraint.moveToFront( $\ell$ )
10       level++
11     else if val( $\ell$ ) = unassigned
12       unassignedLit =  $\ell$ 
13   if unassignedLit = 0
14     analyzeFinal(constraint)
15   else decisions[level++] = unassignedLit
16 else decisions[level++] = heuristic()
```

Analyze Final: algorithm

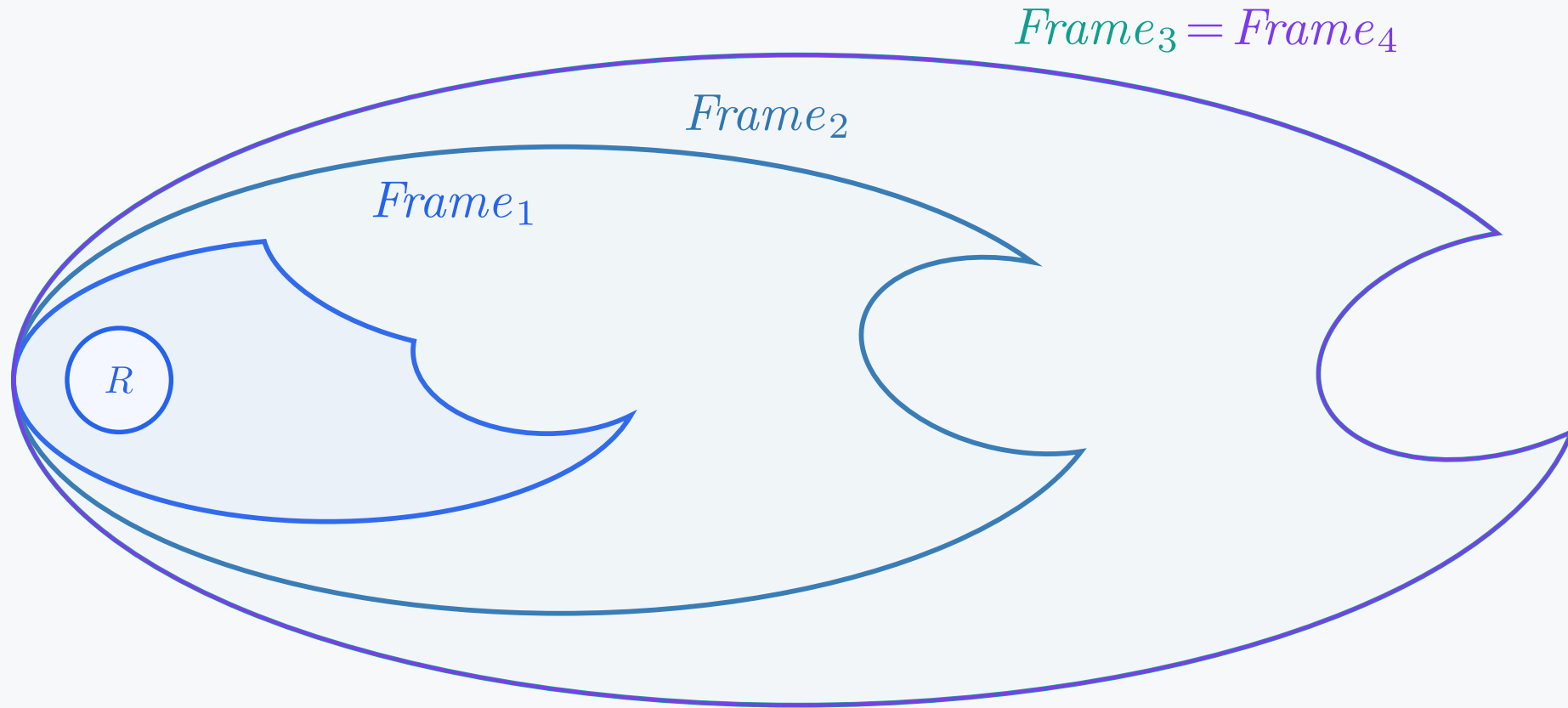


analyzeFinal(constraint)

DECIDE ()

```

1  if level < |assumptions|
2       $\ell = \text{assumptions}[\text{level}]$ 
3      if  $\text{val}(\ell) = \text{false}$  analyzeFinal( $\ell$ )
4      else decisions[level++] =  $\ell$ 
5  else if level = |assumptions|
6      unassignedLit = 0
7      for  $\ell$  in constraint
8          if  $\text{val}(\ell) = \text{true}$ 
9              constraint.moveToFront( $\ell$ )
10             level++
11         else if  $\text{val}(\ell) = \text{unassigned}$ 
12             unassignendLit =  $\ell$ 
13     if unassignendLit = 0
14         analyzeFinal(constraint)
15     else decisions[level++] = unassignendLit
16 else decisions[level++] = heuristic()
    
```



$$\underbrace{\bar{s}_{i-1}}_{\text{Constraint}} \wedge \underbrace{Frame_{i-1} \wedge F_{i-1,i}}_{\text{Clauses}} \wedge \underbrace{s_i}_{\text{Assumptions}}$$

CaDiCaL

IC3 is the first widespread verification method that requires highly efficient incremental solvers. An incremental interface for IC3 must allow single clauses to be pushed and popped; it must also allow literal assumptions. [It poses] many thousands to millions of queries in the course of an analysis, and so speed is crucial.

Understanding IC3 [Bradley, Aaron R., 2012]

github.com/arminbiere/cadical

Model-based testing: 1.5 Billion runs

Activation Literal

$$\underbrace{\bar{s}}_{\text{Constraint}} \wedge \underbrace{Frame_{i-1} \wedge T}_{\text{Clauses}} \wedge \underbrace{s'}_{\text{Assumptions}}$$

$$\underbrace{(a \vee \bar{s}) \wedge Frame_{i-1} \wedge T}_{\text{Clauses}} \wedge \underbrace{s' \wedge \bar{a}}_{\text{Assumptions}}$$

- Deactivated Clauses
- Unused activation literals
- Solver restarts — SAT solver management strategies in IC3

[Cabodi, G. and Camurati, P. E. and Mishchenko, A. and Palena, M. and Pasini, P., 2017]

- Worse Learned Clauses
 - Deleted by restarts
 - Tend to contain the activation literal

⇒ More failed assumptions ⇒ Worse generalization

Experimental Setup

- Bit-level model checker ABC

- Multi-engine
- Sophisticated preprocessing

- Replaced MiniSat with CaDiCaL

[Eén, Niklas and Sörensson, Niklas, 2004]

[Biere, Armin and Fleury, Mathias and Heisinger, Maximilian and Fazekas, Katalin, 2020]

- Replaced activation literals with constraints

- Benchmarks HWMCC'19

[Preiner, Mathias and Biere, Armin, 2019]

- 219 instances, 15 unsolved and removed

PAR-2 Score

	Di	Og	Ca	Co	De
Mean	80	46	16	8.93	8.21
beemTele6Int	136	7200	53	181	101
toyLock4	7200	483	1731	357	359
visArraysField5	7200	1.6	0.58	51	34
nan	208	421	163	158	140
beemColl6Int	241	258	322	133	108
cal110	213	168	130	110	122
cal109	179	197	102	117	86
cal93	186	136	121	118	140
cal94	127	160	115	95	131
cal100	112	42	67	67	54

Disable restarts, Original ABC, CaDiCaL,
Constraints, Defer model reconstruction

Time Per Call

	Restarts	Calls [10 ³]		TPC [ms]	
	CA	CA	CO	CA	CO
Mean	61	19	15	0.61	0.51
beemTele6Int	520	157	574	0.24	0.27
toyLock4	7459	2251	1098	0.42	0.25
visArraysField5	1	<1	113	0.53	0.41
nan	1381	420	423	0.29	0.32
beemColl6Int	398	123	91	2.31	1.24
cal110	191	59	42	1.96	2.39
cal109	110	34	44	2.71	2.44
cal93	206	63	58	1.69	1.8
cal94	171	52	41	1.94	2.1
cal100	148	45	44	1.23	1.29

Backbone

The backbone of a satisfiable formula is the set of literals that are true in all satisfying assignments.

$$a \wedge (\bar{a} \vee b) \wedge (\bar{a} \vee \bar{b} \vee c) \wedge$$

$$(d \vee e \vee f) \wedge (\bar{d} \vee e \vee f) \wedge (d \vee \bar{e} \vee f) \wedge (\bar{d} \vee \bar{e} \vee f) \wedge$$

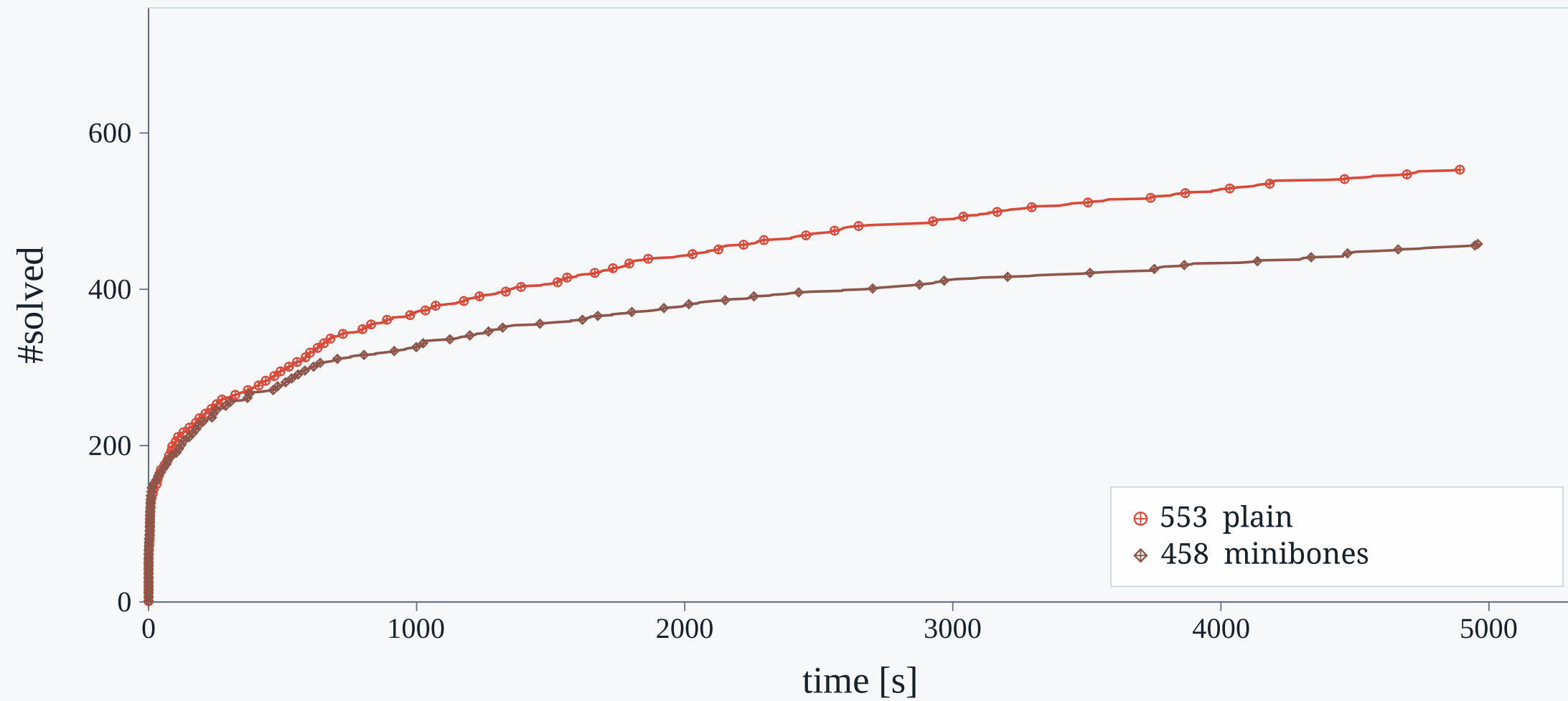
$$\text{PHP} \vee g$$

Backbone Extraction

backbone (CNF φ)

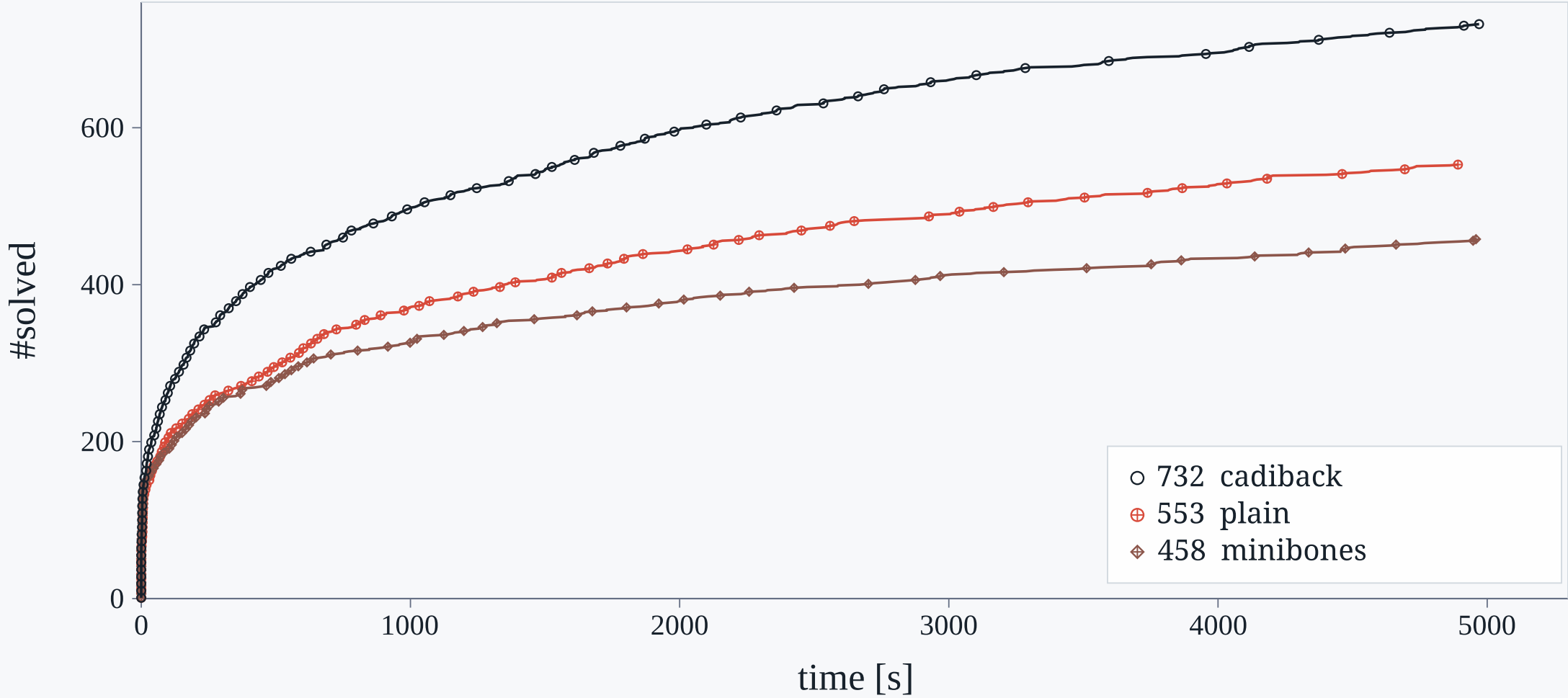
```
1   $(res, \sigma) \leftarrow \text{SAT}(\varphi)$ 
2   $\mathcal{B} \leftarrow \emptyset, \quad \Lambda \leftarrow \sigma$ 
3  while  $\Lambda \neq \emptyset$  do
4       $\ell \leftarrow$  pick literal from  $\Lambda$ 
5       $(res, \sigma) \leftarrow \text{SAT}(\varphi \mid \neg \ell)$ 
6      if  $res$  then // satisfiable
7           $\Lambda \leftarrow \Lambda \setminus \{\ell\}$ 
8      else // unsatisfiable
9           $\mathcal{B} \leftarrow \mathcal{B} \cup \{\ell\}, \quad \varphi = \varphi \wedge \ell$ 
10          $\Lambda \leftarrow \Lambda \setminus \{\ell\}$ 
11 return  $\mathcal{B}$ 
```

CadiBack performance



1798 satisfiable instances from SAT competitions 2004 – 2022

CadiBack performance



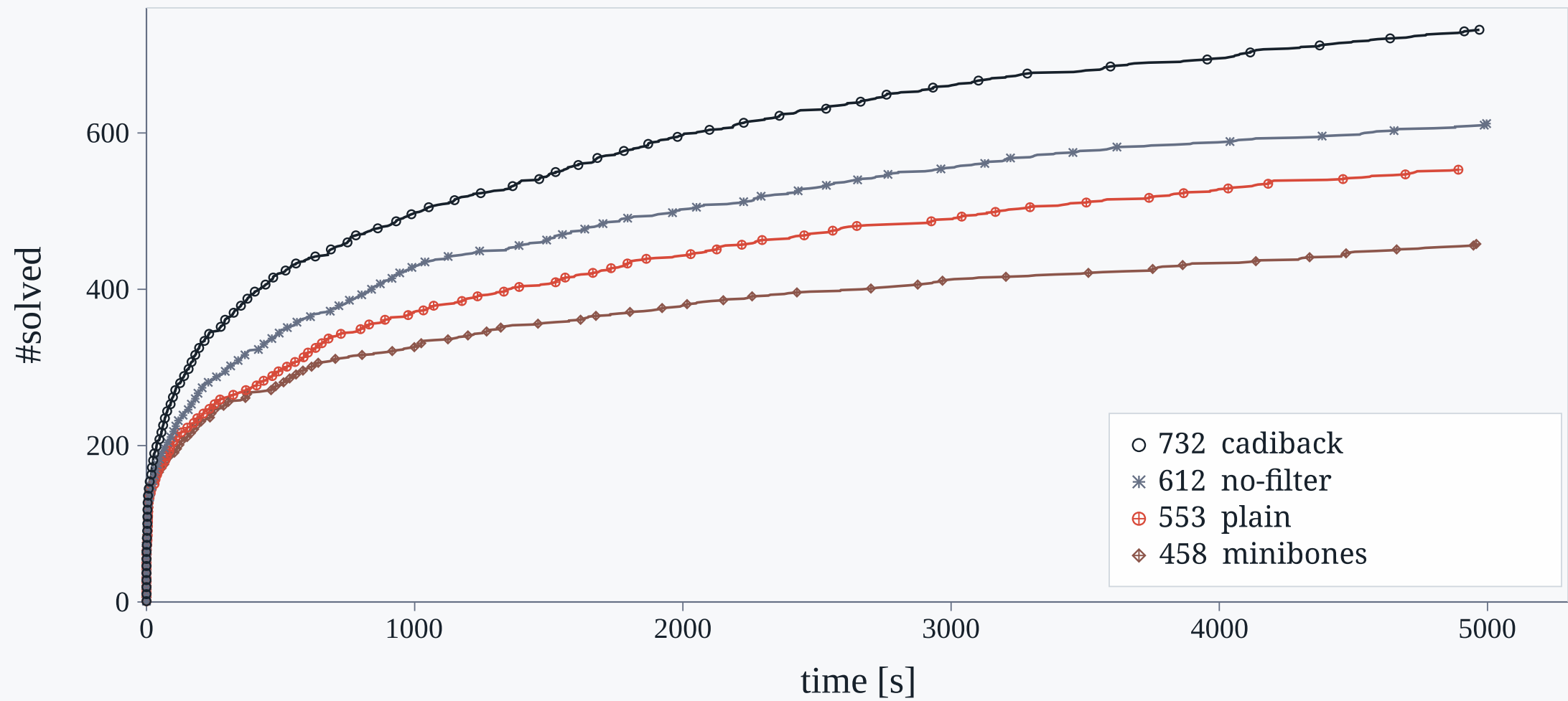
1798 satisfiable instances from SAT competitions 2004 – 2022

Model Filtering

backbone (CNF φ)

```
1   $(res, \sigma) \leftarrow \text{SAT}(\varphi)$ 
2   $\mathcal{B} \leftarrow \emptyset, \quad \Lambda \leftarrow \sigma$ 
3  while  $\Lambda \neq \emptyset$  do
4       $\ell \leftarrow$  pick literal from  $\Lambda$ 
5       $(res, \sigma) \leftarrow \text{SAT}(\varphi \mid \neg \ell)$ 
6      if  $res$  then // satisfiable
7           $\Lambda \leftarrow \cancel{\Lambda \setminus \{\ell\}} \Lambda \cap \sigma$ 
8      else // unsatisfiable
9           $\mathcal{B} \leftarrow \mathcal{B} \cup \{\ell\}, \quad \varphi = \varphi \wedge \ell$ 
10          $\Lambda \leftarrow \Lambda \setminus \{\ell\}$ 
11 return  $\mathcal{B}$ 
```

Model Filtering

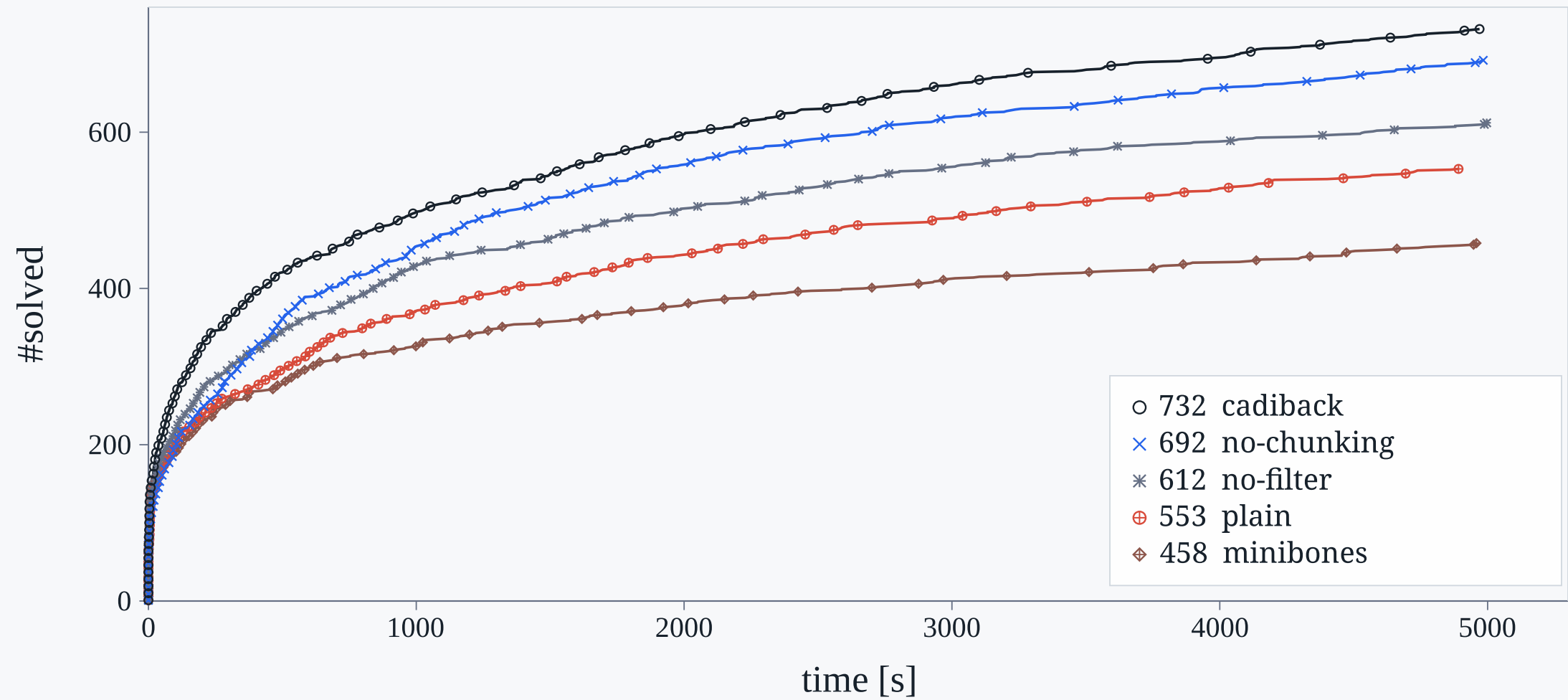


Chunking

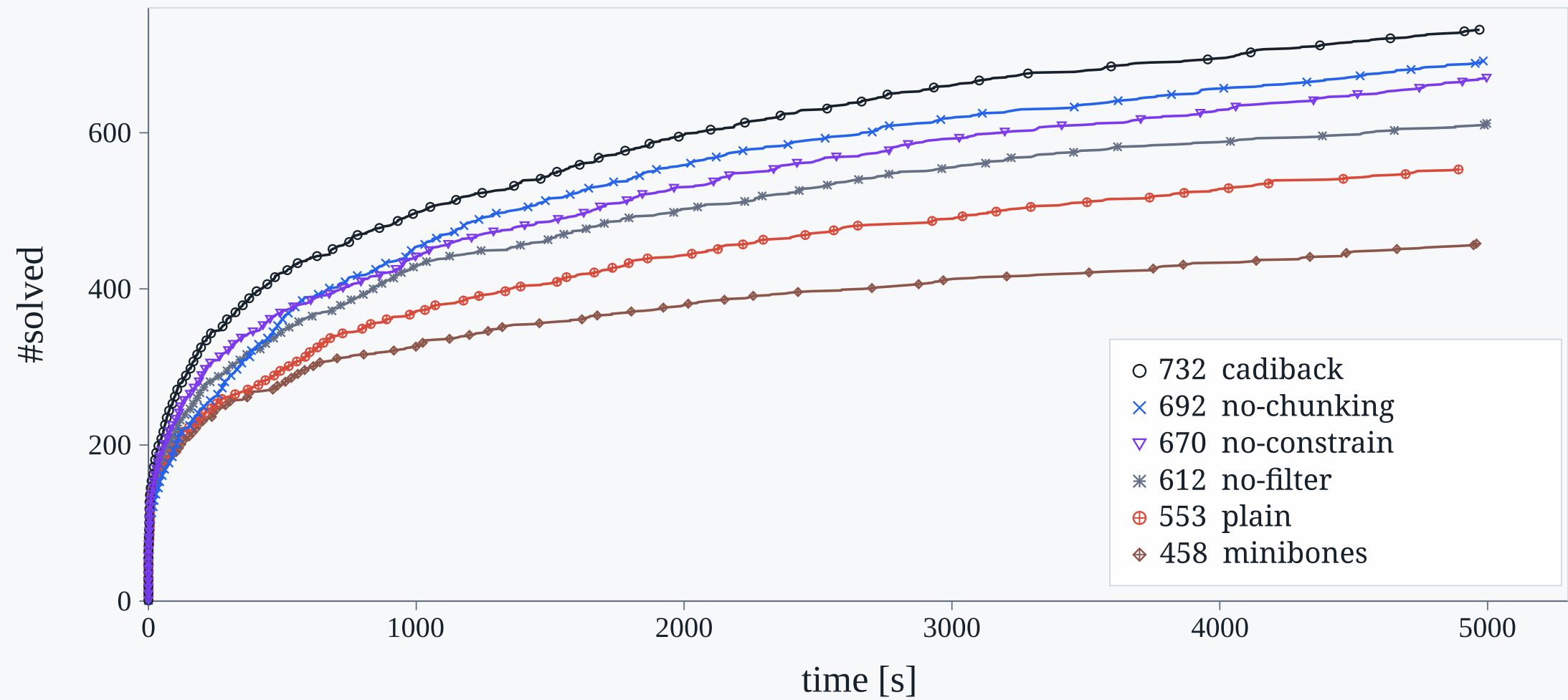
backbone (CNF φ)

```
1   $(res, \sigma) \leftarrow \text{SAT}(\varphi)$ 
2   $\mathcal{B} \leftarrow \emptyset, \quad \Lambda \leftarrow \sigma, \quad k \leftarrow 1$ 
3  while  $\Lambda \neq \emptyset$  do
4       $\Gamma \leftarrow$  pick  $k$  literals from  $\Lambda$  // chunk
5       $(res, \sigma) \leftarrow \text{SAT}(\varphi \mid \cancel{\neg \ell} \bigvee_{\ell \in \Gamma} \neg \ell)$ 
6      if  $res$  then // satisfiable
7           $\Lambda \leftarrow \Lambda \cap \sigma, \quad k \leftarrow 1$ 
8      else // unsatisfiable
9           $\mathcal{B} \leftarrow \mathcal{B} \cup \{\cancel{\ell}\} \Gamma, \quad \varphi = \varphi \wedge \ell$ 
10          $\Lambda \leftarrow \Lambda \setminus \{\cancel{\ell}\} \Gamma$ 
11          $k \leftarrow k \times K$  // geometric increase
12 return  $\mathcal{B}$ 
```


Chunking



Constraint performance



Anytime Algorithm

“How quickly do you find the majority of these backbones.
I would like to use this as preprocessing for model counting.”

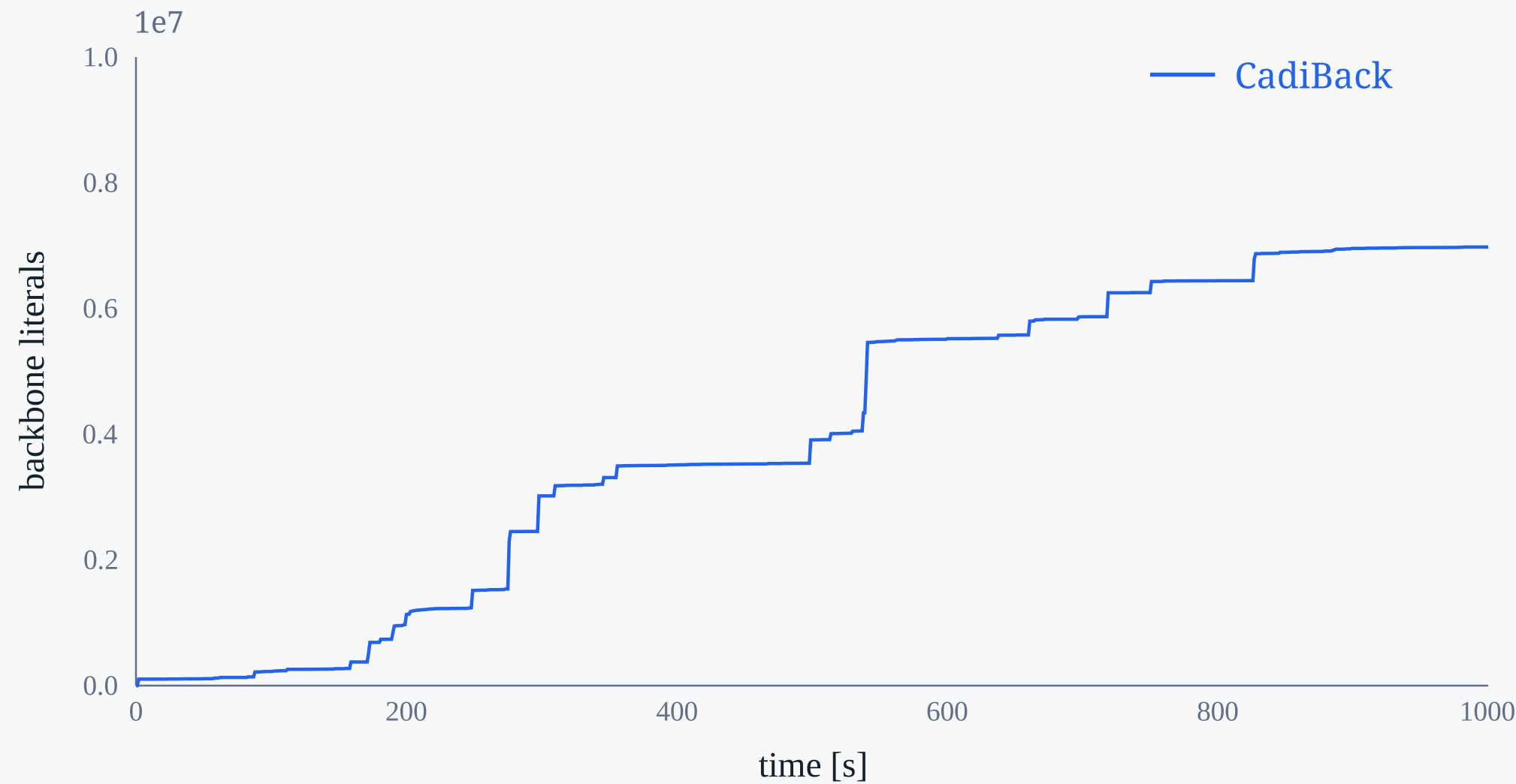
— Johannes Fichte

Organizer, Model Counting Competition

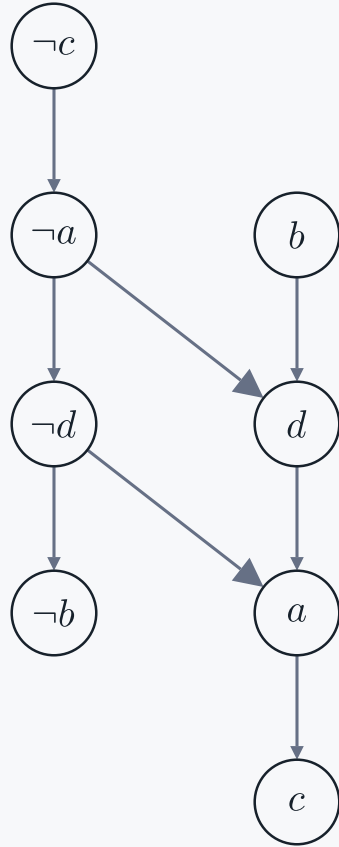
A bit after this paper was submitted

Backbone literals found over time

SC2020–SC2022



Focusing on the BIG



Binary Implication Graph

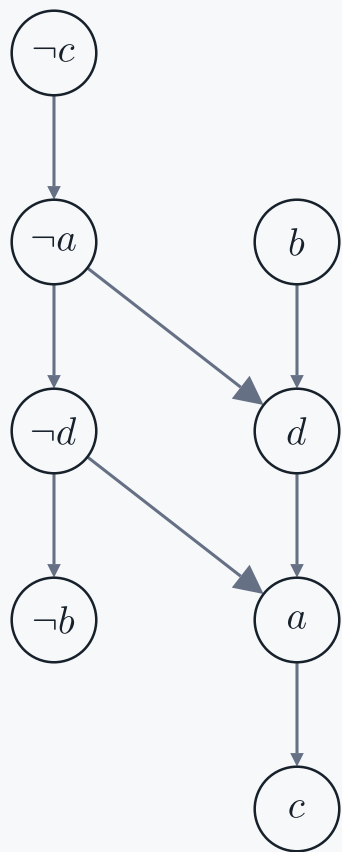
$$V = \mathcal{L}$$

$$E = \{(\neg u, v), (\neg v, u) \mid (u, v) \in \mathcal{F}\}$$

Symmetric (Contraposition)

$$\begin{aligned} \varphi = & (\neg a \vee c) \wedge (\neg d \vee a) \\ & \wedge (\neg b \vee d) \wedge (a \vee d) \end{aligned}$$

Focusing on the BIG



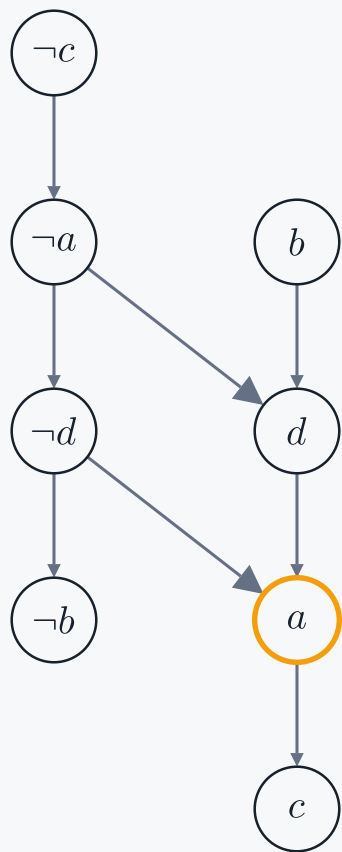
$\Lambda =$

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

```

1   $\mathcal{B} \leftarrow \emptyset, \quad \Lambda \leftarrow \mathcal{L}$ 
2  while  $\Lambda \neq \emptyset$ 
3       $\ell \leftarrow$  pick literal from  $\Lambda$ 
4       $\sigma \leftarrow \text{UnitPropagation}(\mathcal{F}, \{\ell\})$ 
5       $\Lambda \leftarrow \Lambda \setminus \{\ell\}$ 
6      if  $\perp \in \sigma$  then
7           $\mathcal{B} \leftarrow \mathcal{B} \cup \{\ell\}$ 
8           $\varphi \leftarrow \varphi \wedge \ell$ 
9  return  $\mathcal{B}$ 
    
```

Focusing on the BIG



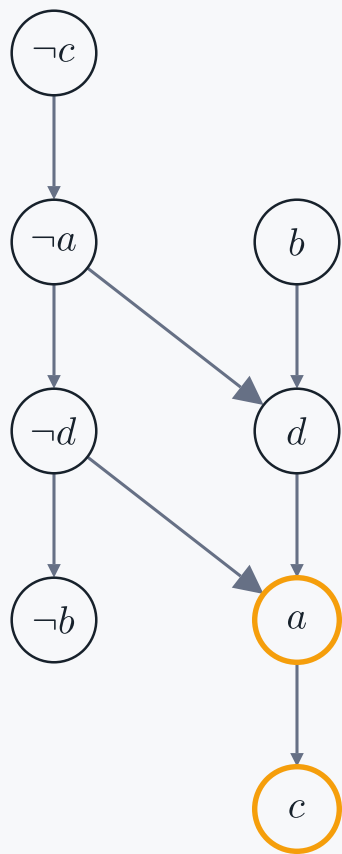
$\Lambda =$

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

```

1   $\mathcal{B} \leftarrow \emptyset, \quad \Lambda \leftarrow \mathcal{L}$ 
2  while  $\Lambda \neq \emptyset$ 
3       $\ell \leftarrow$  pick literal from  $\Lambda$ 
4       $\sigma \leftarrow \text{UnitPropagation}(\mathcal{F}, \{\ell\})$ 
5       $\Lambda \leftarrow \Lambda \setminus \{\ell\}$ 
6      if  $\perp \in \sigma$  then
7           $\mathcal{B} \leftarrow \mathcal{B} \cup \{\ell\}$ 
8           $\varphi \leftarrow \varphi \wedge \ell$ 
9  return  $\mathcal{B}$ 
  
```

Focusing on the BIG



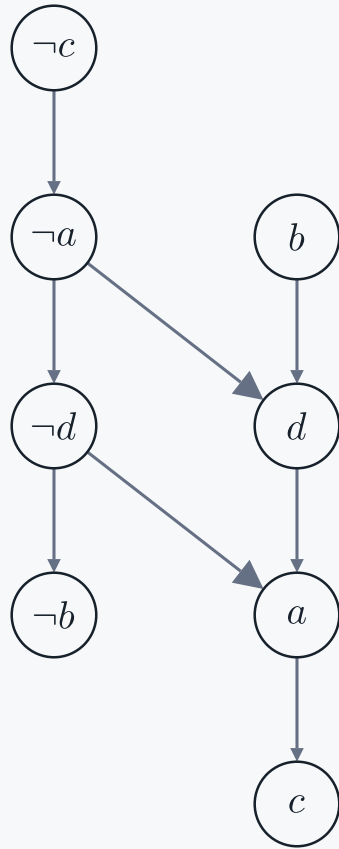
$\Lambda =$

a	b	c	d
¬a	¬b	¬c	¬d

```

1   $\mathcal{B} \leftarrow \emptyset, \quad \Lambda \leftarrow \mathcal{L}$ 
2  while  $\Lambda \neq \emptyset$ 
3       $\ell \leftarrow$  pick literal from  $\Lambda$ 
4       $\sigma \leftarrow \text{UnitPropagation}(\mathcal{F}, \{\ell\})$ 
5       $\Lambda \leftarrow \Lambda \setminus \{\ell\}$ 
6      if  $\perp \in \sigma$  then
7           $\mathcal{B} \leftarrow \mathcal{B} \cup \{\ell\}$ 
8           $\varphi \leftarrow \varphi \wedge \ell$ 
9  return  $\mathcal{B}$ 
  
```


Focusing on the BIG



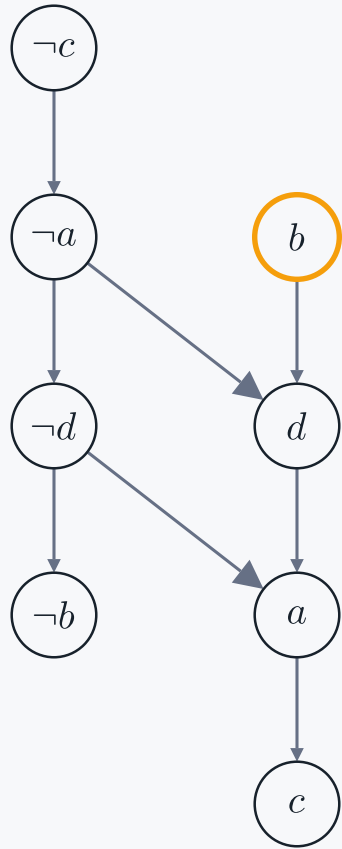
$\Lambda =$

a	b	c	d
¬a	¬b	¬c	¬d

```

1   $\mathcal{B} \leftarrow \emptyset, \quad \Lambda \leftarrow \mathcal{L}$ 
2  while  $\Lambda \neq \emptyset$ 
3       $\ell \leftarrow$  pick literal from  $\Lambda$ 
4       $\sigma \leftarrow \text{UnitPropagation}(\mathcal{F}, \{\ell\})$ 
5       $\Lambda \leftarrow \Lambda \setminus \{\ell\}$ 
6      if  $\perp \in \sigma$  then
7           $\mathcal{B} \leftarrow \mathcal{B} \cup \{\ell\}$ 
8           $\varphi \leftarrow \varphi \wedge \ell$ 
9  return  $\mathcal{B}$ 
  
```

Focusing on the BIG



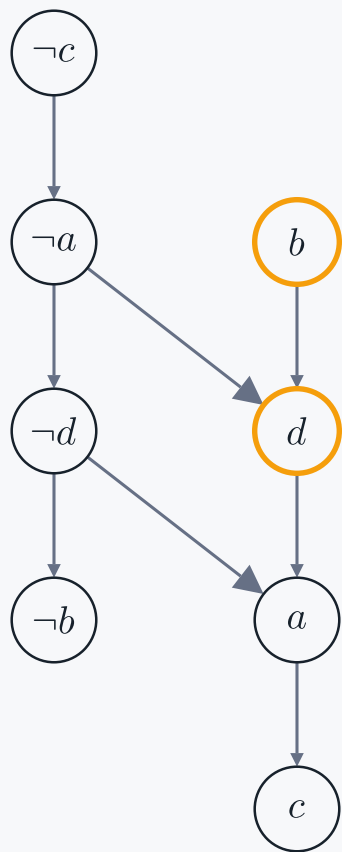
$\Lambda =$

a	b	c	d
¬a	¬b	¬c	¬d

```

1   $\mathcal{B} \leftarrow \emptyset, \quad \Lambda \leftarrow \mathcal{L}$ 
2  while  $\Lambda \neq \emptyset$ 
3       $\ell \leftarrow$  pick literal from  $\Lambda$ 
4       $\sigma \leftarrow \text{UnitPropagation}(\mathcal{F}, \{\ell\})$ 
5       $\Lambda \leftarrow \Lambda \setminus \{\ell\}$ 
6      if  $\perp \in \sigma$  then
7           $\mathcal{B} \leftarrow \mathcal{B} \cup \{\ell\}$ 
8           $\varphi \leftarrow \varphi \wedge \ell$ 
9  return  $\mathcal{B}$ 
    
```

Focusing on the BIG



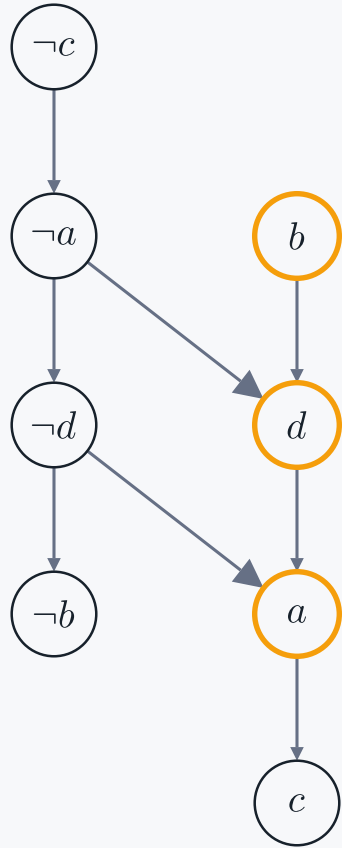
$\Lambda =$

a	b	c	d
¬a	¬b	¬c	¬d

```

1   $\mathcal{B} \leftarrow \emptyset, \quad \Lambda \leftarrow \mathcal{L}$ 
2  while  $\Lambda \neq \emptyset$ 
3       $\ell \leftarrow$  pick literal from  $\Lambda$ 
4       $\sigma \leftarrow \text{UnitPropagation}(\mathcal{F}, \{\ell\})$ 
5       $\Lambda \leftarrow \Lambda \setminus \{\ell\}$ 
6      if  $\perp \in \sigma$  then
7           $\mathcal{B} \leftarrow \mathcal{B} \cup \{\ell\}$ 
8           $\varphi \leftarrow \varphi \wedge \ell$ 
9  return  $\mathcal{B}$ 
  
```

Focusing on the BIG



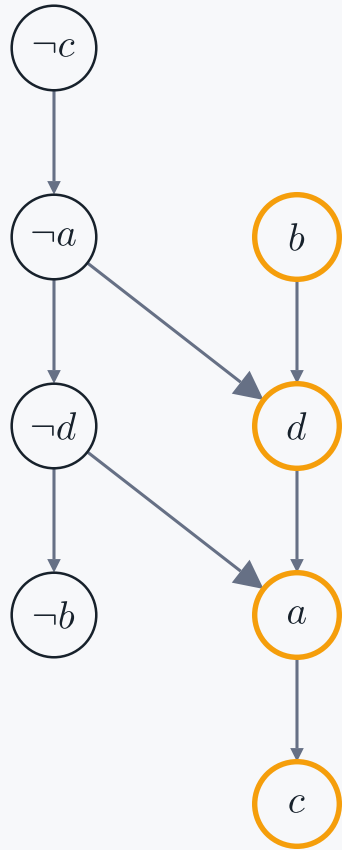
$\Lambda =$

a	b	c	d
¬a	¬b	¬c	¬d

```

1   $\mathcal{B} \leftarrow \emptyset, \quad \Lambda \leftarrow \mathcal{L}$ 
2  while  $\Lambda \neq \emptyset$ 
3       $\ell \leftarrow$  pick literal from  $\Lambda$ 
4       $\sigma \leftarrow \text{UnitPropagation}(\mathcal{F}, \{\ell\})$ 
5       $\Lambda \leftarrow \Lambda \setminus \{\ell\}$ 
6      if  $\perp \in \sigma$  then
7           $\mathcal{B} \leftarrow \mathcal{B} \cup \{\ell\}$ 
8           $\varphi \leftarrow \varphi \wedge \ell$ 
9  return  $\mathcal{B}$ 
    
```

Focusing on the BIG



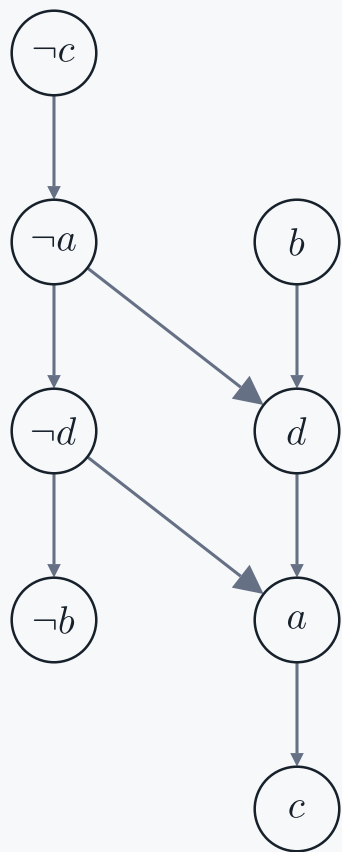
$\Lambda =$

a	b	c	d
¬a	¬b	¬c	¬d

```

1   $\mathcal{B} \leftarrow \emptyset, \quad \Lambda \leftarrow \mathcal{L}$ 
2  while  $\Lambda \neq \emptyset$ 
3       $\ell \leftarrow$  pick literal from  $\Lambda$ 
4       $\sigma \leftarrow \text{UnitPropagation}(\mathcal{F}, \{\ell\})$ 
5       $\Lambda \leftarrow \Lambda \setminus \{\ell\}$ 
6      if  $\perp \in \sigma$  then
7           $\mathcal{B} \leftarrow \mathcal{B} \cup \{\ell\}$ 
8           $\varphi \leftarrow \varphi \wedge \ell$ 
9  return  $\mathcal{B}$ 
    
```

Focusing on the BIG



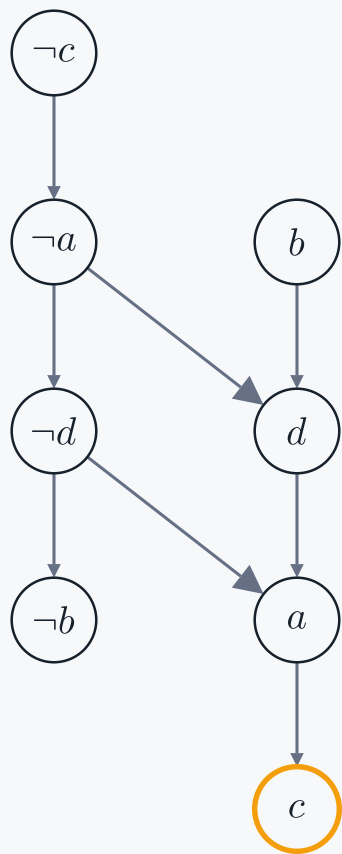
$\Lambda =$

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

```

1   $\mathcal{B} \leftarrow \emptyset, \quad \Lambda \leftarrow \mathcal{L}$ 
2  while  $\Lambda \neq \emptyset$ 
3       $\ell \leftarrow$  pick literal from  $\Lambda$ 
4       $\sigma \leftarrow \text{UnitPropagation}(\mathcal{F}, \{\ell\})$ 
5       $\Lambda \leftarrow \Lambda \setminus \{\ell\}$ 
6      if  $\perp \in \sigma$  then
7           $\mathcal{B} \leftarrow \mathcal{B} \cup \{\ell\}$ 
8           $\varphi \leftarrow \varphi \wedge \ell$ 
9  return  $\mathcal{B}$ 
  
```

Focusing on the BIG



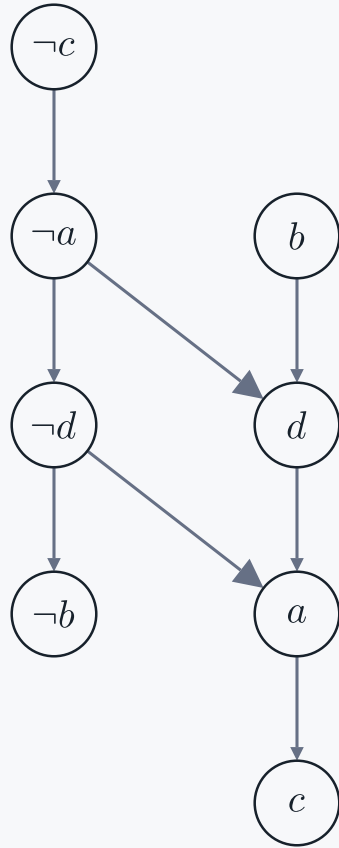
$\Lambda =$

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

```

1   $\mathcal{B} \leftarrow \emptyset, \quad \Lambda \leftarrow \mathcal{L}$ 
2  while  $\Lambda \neq \emptyset$ 
3       $\ell \leftarrow$  pick literal from  $\Lambda$ 
4       $\sigma \leftarrow \text{UnitPropagation}(\mathcal{F}, \{\ell\})$ 
5       $\Lambda \leftarrow \Lambda \setminus \{\ell\}$ 
6      if  $\perp \in \sigma$  then
7           $\mathcal{B} \leftarrow \mathcal{B} \cup \{\ell\}$ 
8           $\varphi \leftarrow \varphi \wedge \ell$ 
9  return  $\mathcal{B}$ 
  
```

Focusing on the BIG



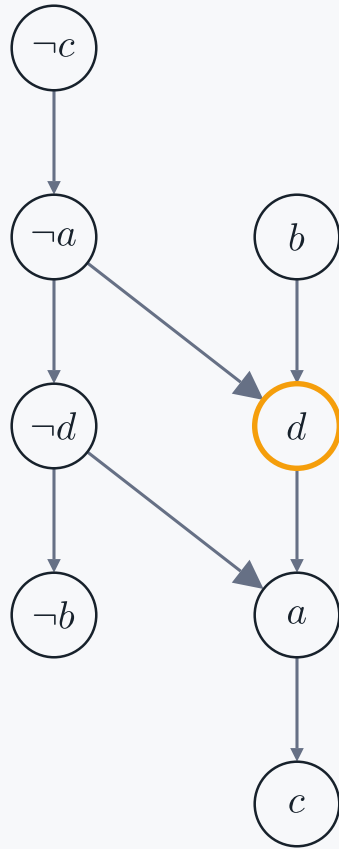
$\Lambda =$

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

```

1   $\mathcal{B} \leftarrow \emptyset, \quad \Lambda \leftarrow \mathcal{L}$ 
2  while  $\Lambda \neq \emptyset$ 
3       $\ell \leftarrow$  pick literal from  $\Lambda$ 
4       $\sigma \leftarrow \text{UnitPropagation}(\mathcal{F}, \{\ell\})$ 
5       $\Lambda \leftarrow \Lambda \setminus \{\ell\}$ 
6      if  $\perp \in \sigma$  then
7           $\mathcal{B} \leftarrow \mathcal{B} \cup \{\ell\}$ 
8           $\varphi \leftarrow \varphi \wedge \ell$ 
9  return  $\mathcal{B}$ 
  
```


Focusing on the BIG



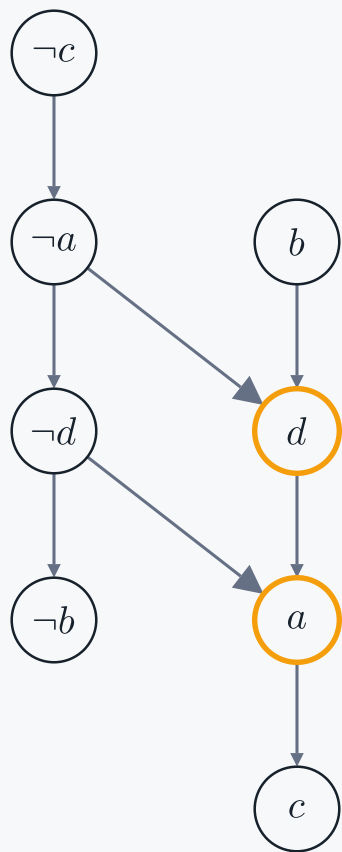
$\Lambda =$

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

```

1   $\mathcal{B} \leftarrow \emptyset, \quad \Lambda \leftarrow \mathcal{L}$ 
2  while  $\Lambda \neq \emptyset$ 
3       $\ell \leftarrow$  pick literal from  $\Lambda$ 
4       $\sigma \leftarrow \text{UnitPropagation}(\mathcal{F}, \{\ell\})$ 
5       $\Lambda \leftarrow \Lambda \setminus \{\ell\}$ 
6      if  $\perp \in \sigma$  then
7           $\mathcal{B} \leftarrow \mathcal{B} \cup \{\ell\}$ 
8           $\varphi \leftarrow \varphi \wedge \ell$ 
9  return  $\mathcal{B}$ 
    
```

Focusing on the BIG



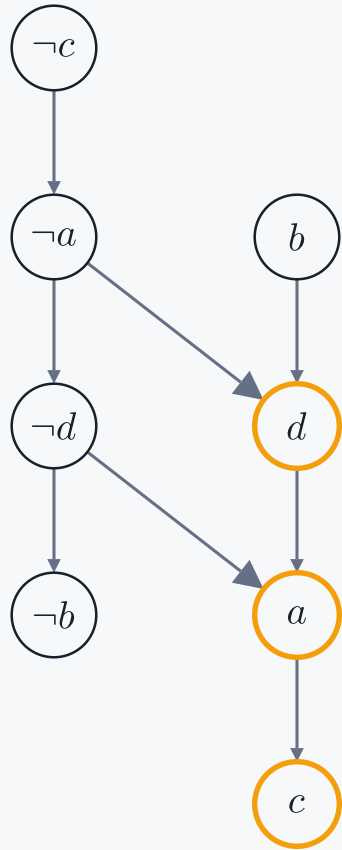
$\Lambda =$

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

```

1   $\mathcal{B} \leftarrow \emptyset, \quad \Lambda \leftarrow \mathcal{L}$ 
2  while  $\Lambda \neq \emptyset$ 
3       $\ell \leftarrow$  pick literal from  $\Lambda$ 
4       $\sigma \leftarrow \text{UnitPropagation}(\mathcal{F}, \{\ell\})$ 
5       $\Lambda \leftarrow \Lambda \setminus \{\ell\}$ 
6      if  $\perp \in \sigma$  then
7           $\mathcal{B} \leftarrow \mathcal{B} \cup \{\ell\}$ 
8           $\varphi \leftarrow \varphi \wedge \ell$ 
9  return  $\mathcal{B}$ 
    
```

Focusing on the BIG



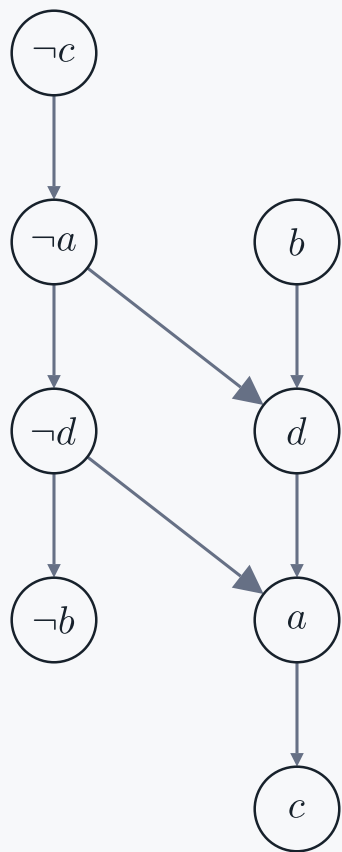
$\Lambda =$

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

```

1   $\mathcal{B} \leftarrow \emptyset, \quad \Lambda \leftarrow \mathcal{L}$ 
2  while  $\Lambda \neq \emptyset$ 
3       $\ell \leftarrow$  pick literal from  $\Lambda$ 
4       $\sigma \leftarrow \text{UnitPropagation}(\mathcal{F}, \{\ell\})$ 
5       $\Lambda \leftarrow \Lambda \setminus \{\ell\}$ 
6      if  $\perp \in \sigma$  then
7           $\mathcal{B} \leftarrow \mathcal{B} \cup \{\ell\}$ 
8           $\varphi \leftarrow \varphi \wedge \ell$ 
9  return  $\mathcal{B}$ 
    
```

Focusing on the BIG



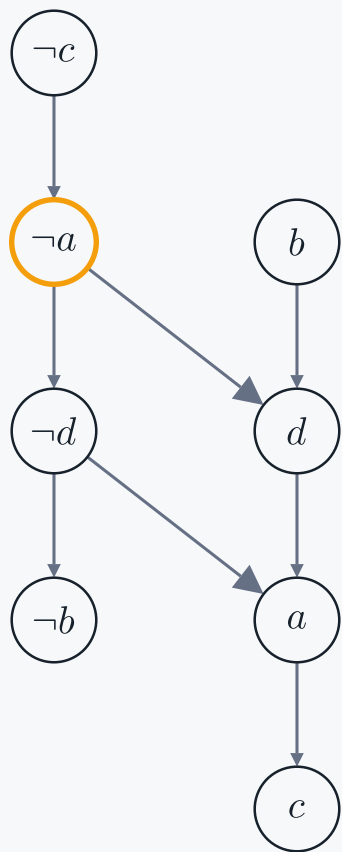
$\Lambda =$

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

```

1   $\mathcal{B} \leftarrow \emptyset, \quad \Lambda \leftarrow \mathcal{L}$ 
2  while  $\Lambda \neq \emptyset$ 
3       $\ell \leftarrow$  pick literal from  $\Lambda$ 
4       $\sigma \leftarrow \text{UnitPropagation}(\mathcal{F}, \{\ell\})$ 
5       $\Lambda \leftarrow \Lambda \setminus \{\ell\}$ 
6      if  $\perp \in \sigma$  then
7           $\mathcal{B} \leftarrow \mathcal{B} \cup \{\ell\}$ 
8           $\varphi \leftarrow \varphi \wedge \ell$ 
9  return  $\mathcal{B}$ 
    
```

Focusing on the BIG



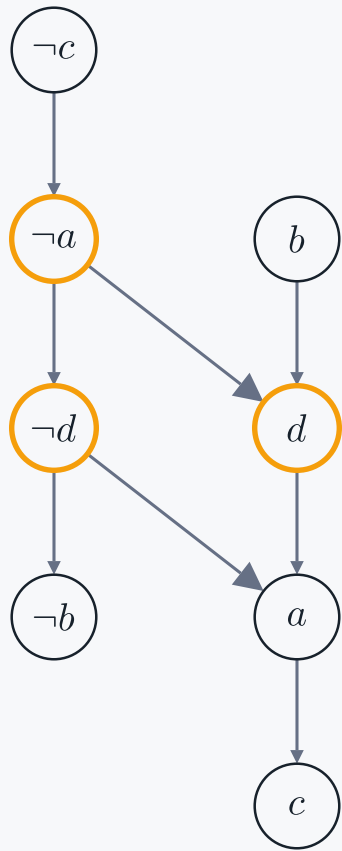
$\Lambda =$

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

```

1   $\mathcal{B} \leftarrow \emptyset, \quad \Lambda \leftarrow \mathcal{L}$ 
2  while  $\Lambda \neq \emptyset$ 
3       $\ell \leftarrow$  pick literal from  $\Lambda$ 
4       $\sigma \leftarrow \text{UnitPropagation}(\mathcal{F}, \{\ell\})$ 
5       $\Lambda \leftarrow \Lambda \setminus \{\ell\}$ 
6      if  $\perp \in \sigma$  then
7           $\mathcal{B} \leftarrow \mathcal{B} \cup \{\ell\}$ 
8           $\varphi \leftarrow \varphi \wedge \ell$ 
9  return  $\mathcal{B}$ 
  
```

Focusing on the BIG



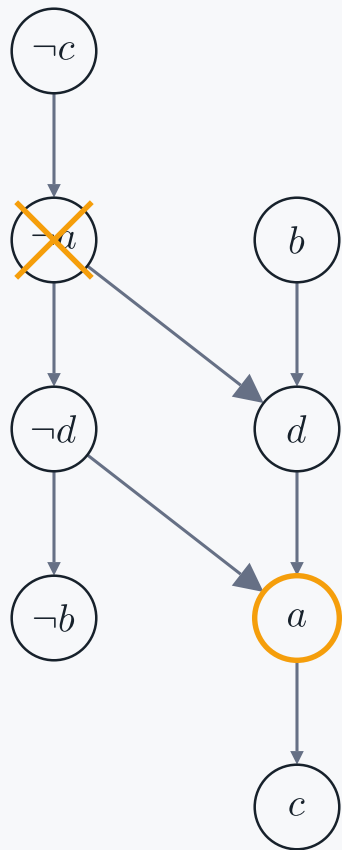
$\Lambda =$

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

```

1   $\mathcal{B} \leftarrow \emptyset, \quad \Lambda \leftarrow \mathcal{L}$ 
2  while  $\Lambda \neq \emptyset$ 
3       $\ell \leftarrow$  pick literal from  $\Lambda$ 
4       $\sigma \leftarrow \text{UnitPropagation}(\mathcal{F}, \{\ell\})$ 
5       $\Lambda \leftarrow \Lambda \setminus \{\ell\}$ 
6      if  $\perp \in \sigma$  then
7           $\mathcal{B} \leftarrow \mathcal{B} \cup \{\ell\}$ 
8           $\varphi \leftarrow \varphi \wedge \ell$ 
9  return  $\mathcal{B}$ 
    
```

Focusing on the BIG



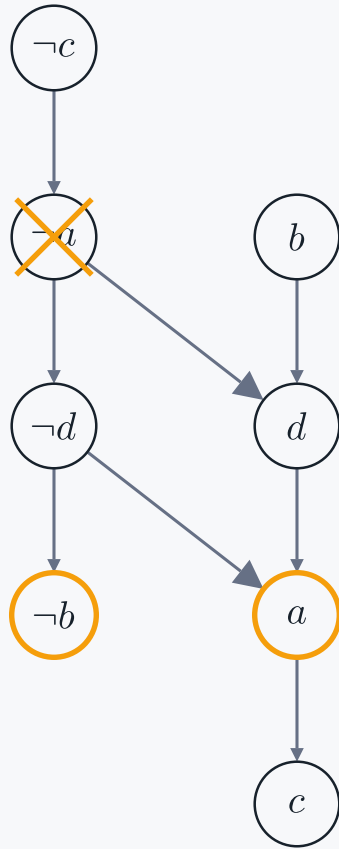
$\Lambda =$

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

```

1   $\mathcal{B} \leftarrow \emptyset, \quad \Lambda \leftarrow \mathcal{L}$ 
2  while  $\Lambda \neq \emptyset$ 
3       $\ell \leftarrow$  pick literal from  $\Lambda$ 
4       $\sigma \leftarrow \text{UnitPropagation}(\mathcal{F}, \{\ell\})$ 
5       $\Lambda \leftarrow \Lambda \setminus \{\ell\}$ 
6      if  $\perp \in \sigma$  then
7           $\mathcal{B} \leftarrow \mathcal{B} \cup \{\ell\}$ 
8           $\varphi \leftarrow \varphi \wedge \ell$ 
9  return  $\mathcal{B}$ 
  
```

Focusing on the BIG



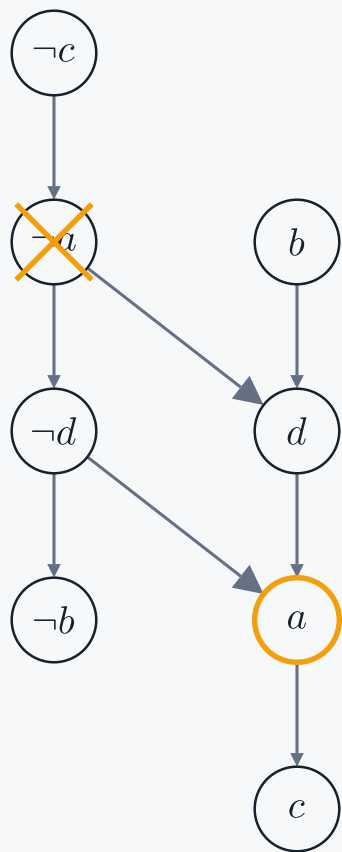
$\Lambda =$

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

```

1   $\mathcal{B} \leftarrow \emptyset, \quad \Lambda \leftarrow \mathcal{L}$ 
2  while  $\Lambda \neq \emptyset$ 
3       $\ell \leftarrow$  pick literal from  $\Lambda$ 
4       $\sigma \leftarrow \text{UnitPropagation}(\mathcal{F}, \{\ell\})$ 
5       $\Lambda \leftarrow \Lambda \setminus \{\ell\}$ 
6      if  $\perp \in \sigma$  then
7           $\mathcal{B} \leftarrow \mathcal{B} \cup \{\ell\}$ 
8           $\varphi \leftarrow \varphi \wedge \ell$ 
9  return  $\mathcal{B}$ 
  
```


Focusing on the BIG



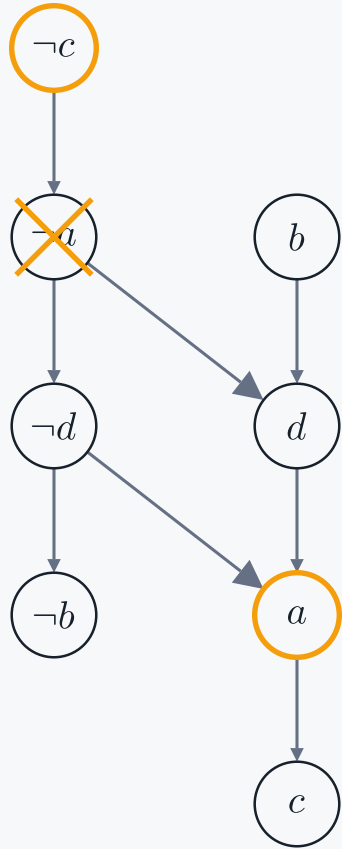
$\Lambda =$

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

```

1   $\mathcal{B} \leftarrow \emptyset, \quad \Lambda \leftarrow \mathcal{L}$ 
2  while  $\Lambda \neq \emptyset$ 
3       $\ell \leftarrow$  pick literal from  $\Lambda$ 
4       $\sigma \leftarrow \text{UnitPropagation}(\mathcal{F}, \{\ell\})$ 
5       $\Lambda \leftarrow \Lambda \setminus \{\ell\}$ 
6      if  $\perp \in \sigma$  then
7           $\mathcal{B} \leftarrow \mathcal{B} \cup \{\ell\}$ 
8           $\varphi \leftarrow \varphi \wedge \ell$ 
9  return  $\mathcal{B}$ 
  
```

Focusing on the BIG



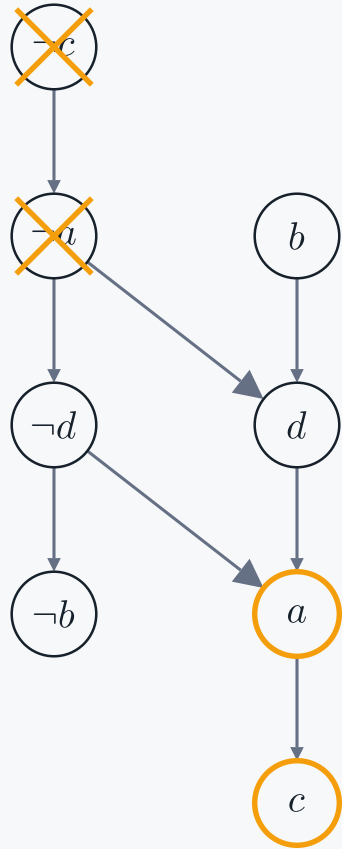
$\Lambda =$

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

```

1   $\mathcal{B} \leftarrow \emptyset, \quad \Lambda \leftarrow \mathcal{L}$ 
2  while  $\Lambda \neq \emptyset$ 
3       $\ell \leftarrow$  pick literal from  $\Lambda$ 
4       $\sigma \leftarrow \text{UnitPropagation}(\mathcal{F}, \{\ell\})$ 
5       $\Lambda \leftarrow \Lambda \setminus \{\ell\}$ 
6      if  $\perp \in \sigma$  then
7           $\mathcal{B} \leftarrow \mathcal{B} \cup \{\ell\}$ 
8           $\varphi \leftarrow \varphi \wedge \ell$ 
9  return  $\mathcal{B}$ 
  
```

Focusing on the BIG



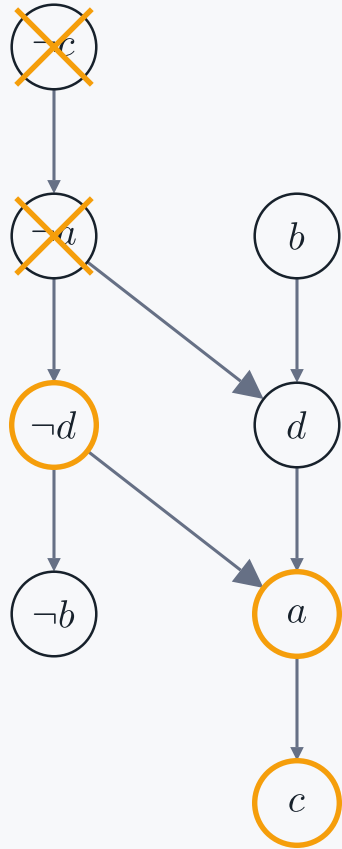
$\Lambda =$

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

```

1   $\mathcal{B} \leftarrow \emptyset, \quad \Lambda \leftarrow \mathcal{L}$ 
2  while  $\Lambda \neq \emptyset$ 
3       $\ell \leftarrow$  pick literal from  $\Lambda$ 
4       $\sigma \leftarrow \text{UnitPropagation}(\mathcal{F}, \{\ell\})$ 
5       $\Lambda \leftarrow \Lambda \setminus \{\ell\}$ 
6      if  $\perp \in \sigma$  then
7           $\mathcal{B} \leftarrow \mathcal{B} \cup \{\ell\}$ 
8           $\varphi \leftarrow \varphi \wedge \ell$ 
9  return  $\mathcal{B}$ 
  
```

Focusing on the BIG



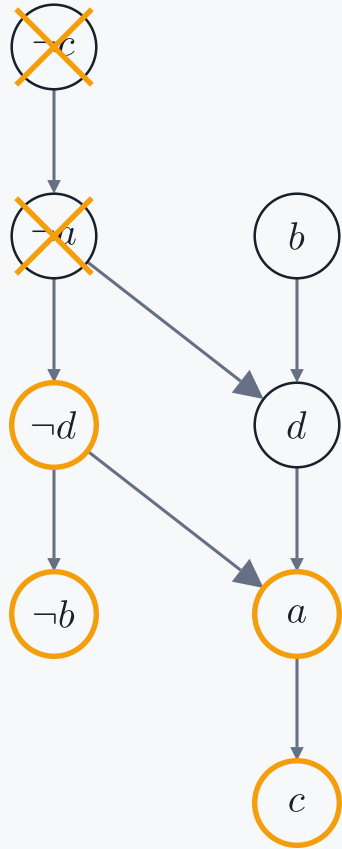
$\Lambda =$

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

```

1   $\mathcal{B} \leftarrow \emptyset, \quad \Lambda \leftarrow \mathcal{L}$ 
2  while  $\Lambda \neq \emptyset$ 
3       $\ell \leftarrow$  pick literal from  $\Lambda$ 
4       $\sigma \leftarrow \text{UnitPropagation}(\mathcal{F}, \{\ell\})$ 
5       $\Lambda \leftarrow \Lambda \setminus \{\ell\}$ 
6      if  $\perp \in \sigma$  then
7           $\mathcal{B} \leftarrow \mathcal{B} \cup \{\ell\}$ 
8           $\varphi \leftarrow \varphi \wedge \ell$ 
9  return  $\mathcal{B}$ 
  
```

Focusing on the BIG



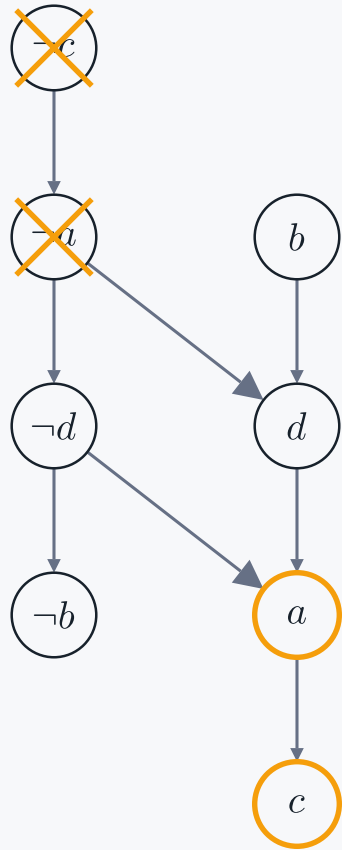
$\Lambda =$

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

```

1   $\mathcal{B} \leftarrow \emptyset, \quad \Lambda \leftarrow \mathcal{L}$ 
2  while  $\Lambda \neq \emptyset$ 
3       $\ell \leftarrow$  pick literal from  $\Lambda$ 
4       $\sigma \leftarrow \text{UnitPropagation}(\mathcal{F}, \{\ell\})$ 
5       $\Lambda \leftarrow \Lambda \setminus \{\ell\}$ 
6      if  $\perp \in \sigma$  then
7           $\mathcal{B} \leftarrow \mathcal{B} \cup \{\ell\}$ 
8           $\varphi \leftarrow \varphi \wedge \ell$ 
9  return  $\mathcal{B}$ 
    
```

Focusing on the BIG



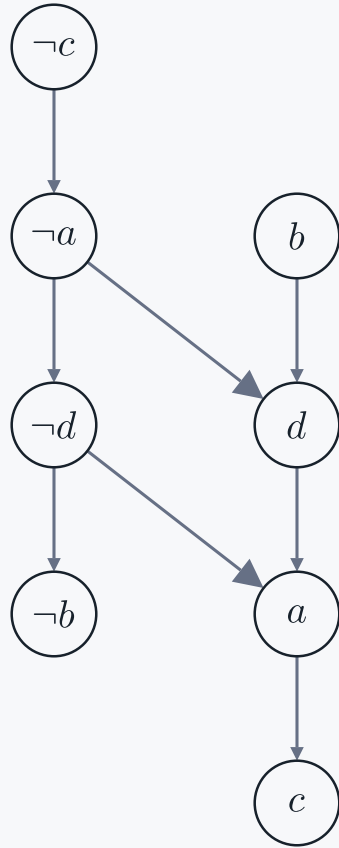
$\Lambda =$

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

```

1   $\mathcal{B} \leftarrow \emptyset, \quad \Lambda \leftarrow \mathcal{L}$ 
2  while  $\Lambda \neq \emptyset$ 
3       $\ell \leftarrow$  pick literal from  $\Lambda$ 
4       $\sigma \leftarrow \text{UnitPropagation}(\mathcal{F}, \{\ell\})$ 
5       $\Lambda \leftarrow \Lambda \setminus \{\ell\}$ 
6      if  $\perp \in \sigma$  then
7           $\mathcal{B} \leftarrow \mathcal{B} \cup \{\ell\}$ 
8           $\varphi \leftarrow \varphi \wedge \ell$ 
9  return  $\mathcal{B}$ 
  
```

Keep Assignment Big Backbone



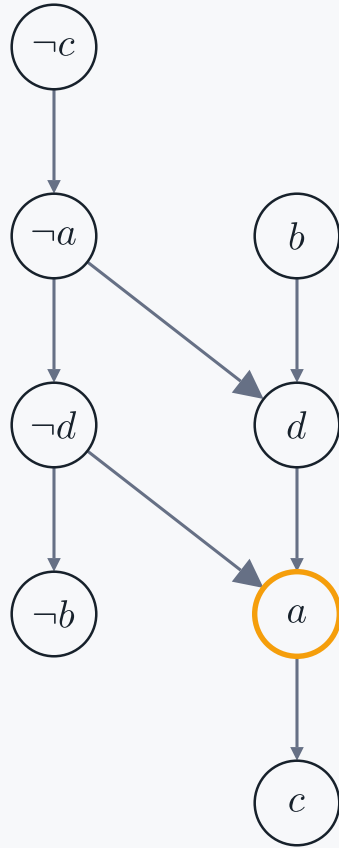
$\Lambda =$

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

```

1   $\mathcal{B} \leftarrow \emptyset, \quad \Lambda \leftarrow \mathcal{L}$ 
2  while  $\Lambda \neq \emptyset$ 
3       $\sigma \leftarrow \mathcal{B}, \quad \Delta \leftarrow \emptyset$ 
4      for  $\ell \in \Lambda$  // next candidate
5          if  $\neg \ell \in \sigma$  continue
6           $\Delta \leftarrow \Delta \cup \{\ell\}$ 
7          if  $\ell \in \sigma$  continue
8           $\sigma' \leftarrow \text{UnitPropagation}(\mathcal{F}, \sigma \cup \{\ell\})$ 
9          if  $\perp \in \sigma'$  then
10              $\mathcal{B} \leftarrow \mathcal{B} \cup \{\neg \ell\}$ 
11              $\Delta \leftarrow \Delta \cup \mathcal{B} \cup \neg \mathcal{B}$ 
12              $\sigma \leftarrow \sigma \cup \mathcal{B}$ 
13         else  $\sigma \leftarrow \sigma'$ 
14      $\Lambda \leftarrow \Lambda \setminus \Delta$ 
15 return  $\mathcal{B}$ 
  
```

Keep Assignment Big Backbone



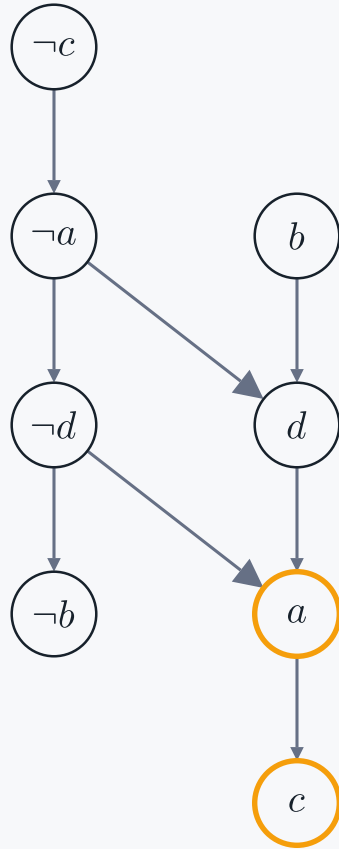
$\Lambda =$

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

```

1   $\mathcal{B} \leftarrow \emptyset, \quad \Lambda \leftarrow \mathcal{L}$ 
2  while  $\Lambda \neq \emptyset$ 
3       $\sigma \leftarrow \mathcal{B}, \quad \Delta \leftarrow \emptyset$ 
4      for  $\ell \in \Lambda$  // next candidate
5          if  $\neg \ell \in \sigma$  continue
6           $\Delta \leftarrow \Delta \cup \{\ell\}$ 
7          if  $\ell \in \sigma$  continue
8           $\sigma' \leftarrow \text{UnitPropagation}(\mathcal{F}, \sigma \cup \{\ell\})$ 
9          if  $\perp \in \sigma'$  then
10              $\mathcal{B} \leftarrow \mathcal{B} \cup \{\neg \ell\}$ 
11              $\Delta \leftarrow \Delta \cup \mathcal{B} \cup \neg \mathcal{B}$ 
12              $\sigma \leftarrow \sigma \cup \mathcal{B}$ 
13         else  $\sigma \leftarrow \sigma'$ 
14      $\Lambda \leftarrow \Lambda \setminus \Delta$ 
15 return  $\mathcal{B}$ 
    
```


Keep Assignment Big Backbone



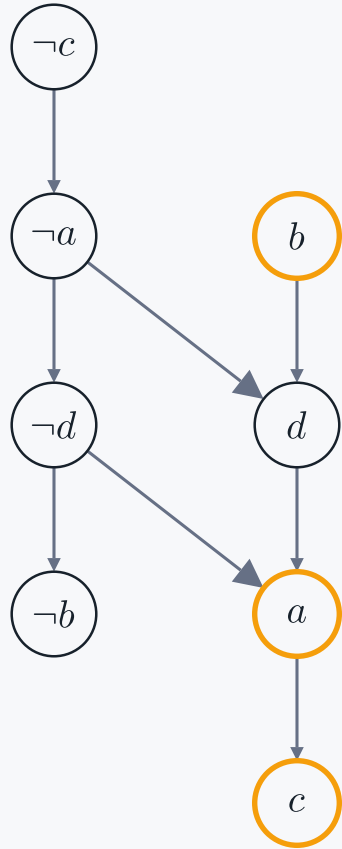
$\Lambda =$

a	b	c	d
¬a	¬b	¬c	¬d

```

1   $\mathcal{B} \leftarrow \emptyset, \quad \Lambda \leftarrow \mathcal{L}$ 
2  while  $\Lambda \neq \emptyset$ 
3       $\sigma \leftarrow \mathcal{B}, \quad \Delta \leftarrow \emptyset$ 
4      for  $\ell \in \Lambda$  // next candidate
5          if  $\neg \ell \in \sigma$  continue
6           $\Delta \leftarrow \Delta \cup \{\ell\}$ 
7          if  $\ell \in \sigma$  continue
8           $\sigma' \leftarrow \text{UnitPropagation}(\mathcal{F}, \sigma \cup \{\ell\})$ 
9          if  $\perp \in \sigma'$  then
10              $\mathcal{B} \leftarrow \mathcal{B} \cup \{\neg \ell\}$ 
11              $\Delta \leftarrow \Delta \cup \mathcal{B} \cup \neg \mathcal{B}$ 
12              $\sigma \leftarrow \sigma \cup \mathcal{B}$ 
13         else  $\sigma \leftarrow \sigma'$ 
14      $\Lambda \leftarrow \Lambda \setminus \Delta$ 
15 return  $\mathcal{B}$ 
    
```

Keep Assignment Big Backbone



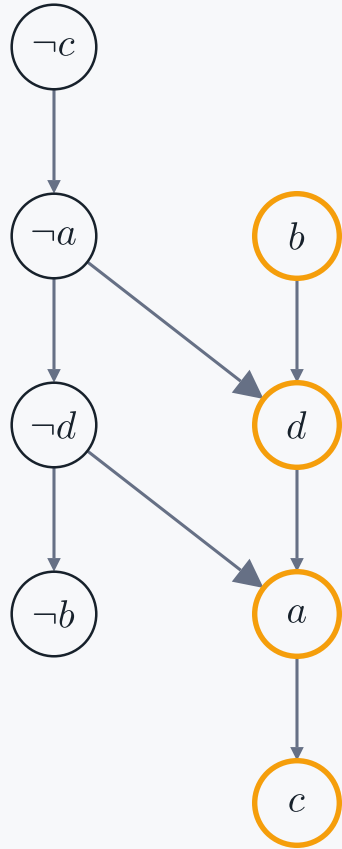
$\Lambda =$

a	b	c	d
¬a	¬b	¬c	¬d

```

1   $\mathcal{B} \leftarrow \emptyset, \quad \Lambda \leftarrow \mathcal{L}$ 
2  while  $\Lambda \neq \emptyset$ 
3       $\sigma \leftarrow \mathcal{B}, \quad \Delta \leftarrow \emptyset$ 
4      for  $\ell \in \Lambda$  // next candidate
5          if  $\neg \ell \in \sigma$  continue
6           $\Delta \leftarrow \Delta \cup \{\ell\}$ 
7          if  $\ell \in \sigma$  continue
8           $\sigma' \leftarrow \text{UnitPropagation}(\mathcal{F}, \sigma \cup \{\ell\})$ 
9          if  $\perp \in \sigma'$  then
10              $\mathcal{B} \leftarrow \mathcal{B} \cup \{\neg \ell\}$ 
11              $\Delta \leftarrow \Delta \cup \mathcal{B} \cup \neg \mathcal{B}$ 
12              $\sigma \leftarrow \sigma \cup \mathcal{B}$ 
13         else  $\sigma \leftarrow \sigma'$ 
14      $\Lambda \leftarrow \Lambda \setminus \Delta$ 
15 return  $\mathcal{B}$ 
    
```

Keep Assignment Big Backbone



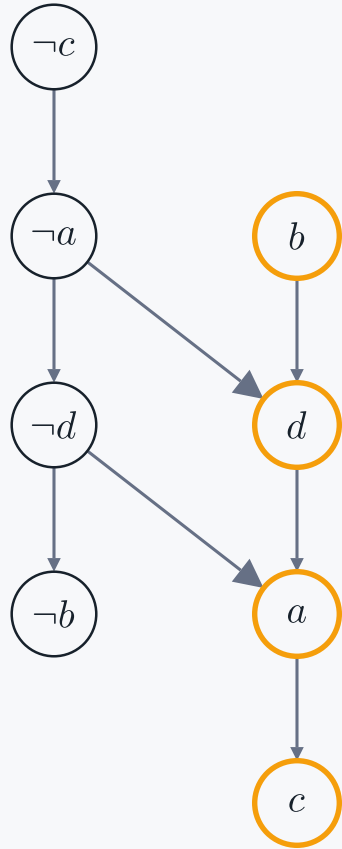
$\Lambda =$

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

```

1   $\mathcal{B} \leftarrow \emptyset, \quad \Lambda \leftarrow \mathcal{L}$ 
2  while  $\Lambda \neq \emptyset$ 
3       $\sigma \leftarrow \mathcal{B}, \quad \Delta \leftarrow \emptyset$ 
4      for  $\ell \in \Lambda$  // next candidate
5          if  $\neg \ell \in \sigma$  continue
6           $\Delta \leftarrow \Delta \cup \{\ell\}$ 
7          if  $\ell \in \sigma$  continue
8           $\sigma' \leftarrow \text{UnitPropagation}(\mathcal{F}, \sigma \cup \{\ell\})$ 
9          if  $\perp \in \sigma'$  then
10              $\mathcal{B} \leftarrow \mathcal{B} \cup \{\neg \ell\}$ 
11              $\Delta \leftarrow \Delta \cup \mathcal{B} \cup \neg \mathcal{B}$ 
12              $\sigma \leftarrow \sigma \cup \mathcal{B}$ 
13         else  $\sigma \leftarrow \sigma'$ 
14      $\Lambda \leftarrow \Lambda \setminus \Delta$ 
15 return  $\mathcal{B}$ 
    
```

Keep Assignment Big Backbone



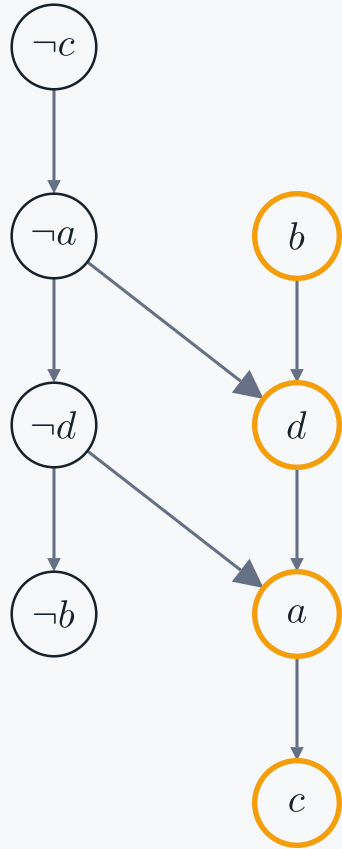
$\Lambda =$

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

```

1   $\mathcal{B} \leftarrow \emptyset, \quad \Lambda \leftarrow \mathcal{L}$ 
2  while  $\Lambda \neq \emptyset$ 
3       $\sigma \leftarrow \mathcal{B}, \quad \Delta \leftarrow \emptyset$ 
4      for  $\ell \in \Lambda$  // next candidate
5          if  $\neg \ell \in \sigma$  continue
6           $\Delta \leftarrow \Delta \cup \{\ell\}$ 
7          if  $\ell \in \sigma$  continue
8           $\sigma' \leftarrow \text{UnitPropagation}(\mathcal{F}, \sigma \cup \{\ell\})$ 
9          if  $\perp \in \sigma'$  then
10              $\mathcal{B} \leftarrow \mathcal{B} \cup \{\neg \ell\}$ 
11              $\Delta \leftarrow \Delta \cup \mathcal{B} \cup \neg \mathcal{B}$ 
12              $\sigma \leftarrow \sigma \cup \mathcal{B}$ 
13         else  $\sigma \leftarrow \sigma'$ 
14      $\Lambda \leftarrow \Lambda \setminus \Delta$ 
15 return  $\mathcal{B}$ 
    
```

Keep Assignment Big Backbone

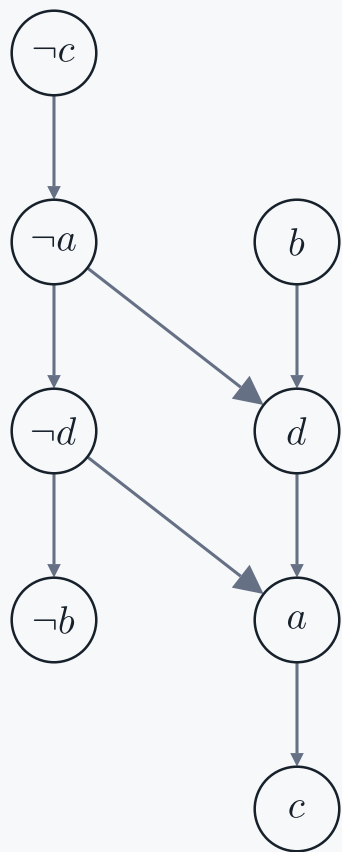


$\Lambda =$
 ~~a~~ ~~b~~ ~~c~~ ~~d~~
 $\neg a$ $\neg b$ $\neg c$ $\neg d$

```

1   $\mathcal{B} \leftarrow \emptyset, \quad \Lambda \leftarrow \mathcal{L}$ 
2  while  $\Lambda \neq \emptyset$ 
3       $\sigma \leftarrow \mathcal{B}, \quad \Delta \leftarrow \emptyset$ 
4      for  $\ell \in \Lambda$  // next candidate
5          if  $\neg \ell \in \sigma$  continue
6           $\Delta \leftarrow \Delta \cup \{\ell\}$ 
7          if  $\ell \in \sigma$  continue
8           $\sigma' \leftarrow \text{UnitPropagation}(\mathcal{F}, \sigma \cup \{\ell\})$ 
9          if  $\perp \in \sigma'$  then
10              $\mathcal{B} \leftarrow \mathcal{B} \cup \{\neg \ell\}$ 
11              $\Delta \leftarrow \Delta \cup \mathcal{B} \cup \neg \mathcal{B}$ 
12              $\sigma \leftarrow \sigma \cup \mathcal{B}$ 
13         else  $\sigma \leftarrow \sigma'$ 
14      $\Lambda \leftarrow \Lambda \setminus \Delta$ 
15 return  $\mathcal{B}$ 
  
```

Keep Assignment Big Backbone

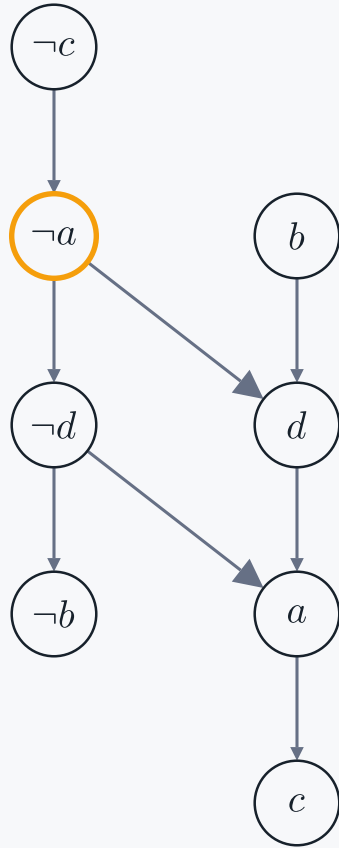


$\Lambda =$
~~a~~ ~~b~~ ~~c~~ ~~d~~
 $\neg a$ $\neg b$ $\neg c$ $\neg d$

```

1   $\mathcal{B} \leftarrow \emptyset, \quad \Lambda \leftarrow \mathcal{L}$ 
2  while  $\Lambda \neq \emptyset$ 
3       $\sigma \leftarrow \mathcal{B}, \quad \Delta \leftarrow \emptyset$ 
4      for  $\ell \in \Lambda$  // next candidate
5          if  $\neg \ell \in \sigma$  continue
6           $\Delta \leftarrow \Delta \cup \{\ell\}$ 
7          if  $\ell \in \sigma$  continue
8           $\sigma' \leftarrow \text{UnitPropagation}(\mathcal{F}, \sigma \cup \{\ell\})$ 
9          if  $\perp \in \sigma'$  then
10              $\mathcal{B} \leftarrow \mathcal{B} \cup \{\neg \ell\}$ 
11              $\Delta \leftarrow \Delta \cup \mathcal{B} \cup \neg \mathcal{B}$ 
12              $\sigma \leftarrow \sigma \cup \mathcal{B}$ 
13         else  $\sigma \leftarrow \sigma'$ 
14      $\Lambda \leftarrow \Lambda \setminus \Delta$ 
15 return  $\mathcal{B}$ 
  
```

Keep Assignment Big Backbone

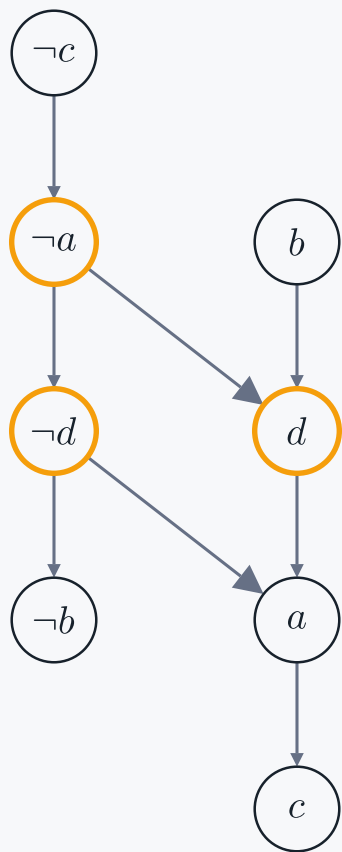


$\Lambda =$
 ~~a~~ ~~b~~ ~~c~~ ~~d~~
 $\neg a$ $\neg b$ $\neg c$ $\neg d$

```

1   $\mathcal{B} \leftarrow \emptyset, \quad \Lambda \leftarrow \mathcal{L}$ 
2  while  $\Lambda \neq \emptyset$ 
3       $\sigma \leftarrow \mathcal{B}, \quad \Delta \leftarrow \emptyset$ 
4      for  $\ell \in \Lambda$  // next candidate
5          if  $\neg \ell \in \sigma$  continue
6           $\Delta \leftarrow \Delta \cup \{\ell\}$ 
7          if  $\ell \in \sigma$  continue
8           $\sigma' \leftarrow \text{UnitPropagation}(\mathcal{F}, \sigma \cup \{\ell\})$ 
9          if  $\perp \in \sigma'$  then
10              $\mathcal{B} \leftarrow \mathcal{B} \cup \{\neg \ell\}$ 
11              $\Delta \leftarrow \Delta \cup \mathcal{B} \cup \neg \mathcal{B}$ 
12              $\sigma \leftarrow \sigma \cup \mathcal{B}$ 
13         else  $\sigma \leftarrow \sigma'$ 
14      $\Lambda \leftarrow \Lambda \setminus \Delta$ 
15 return  $\mathcal{B}$ 
    
```

Keep Assignment Big Backbone

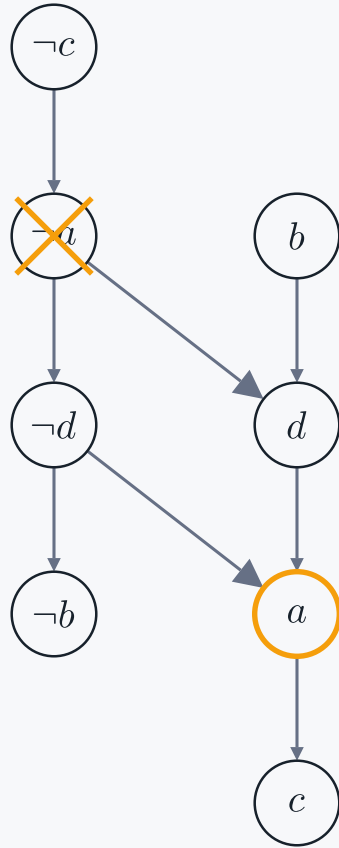


$\Lambda =$
 ~~a~~
 ~~b~~
 ~~c~~
 ~~d~~
 $\neg a$
 $\neg b$
 $\neg c$
 $\neg d$

```

1   $\mathcal{B} \leftarrow \emptyset, \quad \Lambda \leftarrow \mathcal{L}$ 
2  while  $\Lambda \neq \emptyset$ 
3       $\sigma \leftarrow \mathcal{B}, \quad \Delta \leftarrow \emptyset$ 
4      for  $\ell \in \Lambda$  // next candidate
5          if  $\neg \ell \in \sigma$  continue
6           $\Delta \leftarrow \Delta \cup \{\ell\}$ 
7          if  $\ell \in \sigma$  continue
8           $\sigma' \leftarrow \text{UnitPropagation}(\mathcal{F}, \sigma \cup \{\ell\})$ 
9          if  $\perp \in \sigma'$  then
10              $\mathcal{B} \leftarrow \mathcal{B} \cup \{\neg \ell\}$ 
11              $\Delta \leftarrow \Delta \cup \mathcal{B} \cup \neg \mathcal{B}$ 
12              $\sigma \leftarrow \sigma \cup \mathcal{B}$ 
13         else  $\sigma \leftarrow \sigma'$ 
14      $\Lambda \leftarrow \Lambda \setminus \Delta$ 
15 return  $\mathcal{B}$ 
  
```


Keep Assignment Big Backbone



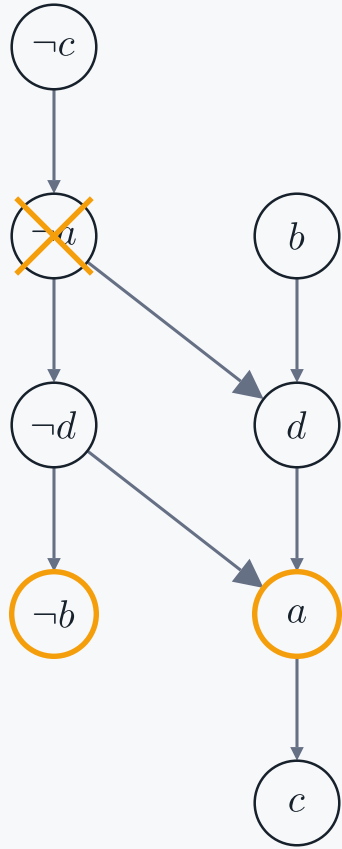
$\Lambda =$

a	b	c	d
¬a	¬b	¬c	¬d

```

1   $\mathcal{B} \leftarrow \emptyset, \quad \Lambda \leftarrow \mathcal{L}$ 
2  while  $\Lambda \neq \emptyset$ 
3       $\sigma \leftarrow \mathcal{B}, \quad \Delta \leftarrow \emptyset$ 
4      for  $\ell \in \Lambda$  // next candidate
5          if  $\neg \ell \in \sigma$  continue
6           $\Delta \leftarrow \Delta \cup \{\ell\}$ 
7          if  $\ell \in \sigma$  continue
8           $\sigma' \leftarrow \text{UnitPropagation}(\mathcal{F}, \sigma \cup \{\ell\})$ 
9          if  $\perp \in \sigma'$  then
10              $\mathcal{B} \leftarrow \mathcal{B} \cup \{\neg \ell\}$ 
11              $\Delta \leftarrow \Delta \cup \mathcal{B} \cup \neg \mathcal{B}$ 
12              $\sigma \leftarrow \sigma \cup \mathcal{B}$ 
13         else  $\sigma \leftarrow \sigma'$ 
14      $\Lambda \leftarrow \Lambda \setminus \Delta$ 
15 return  $\mathcal{B}$ 
    
```

Keep Assignment Big Backbone



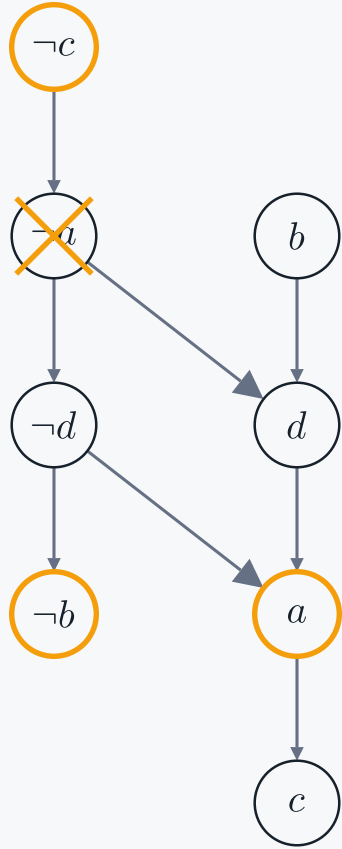
$\Lambda =$

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

```

1   $\mathcal{B} \leftarrow \emptyset, \quad \Lambda \leftarrow \mathcal{L}$ 
2  while  $\Lambda \neq \emptyset$ 
3       $\sigma \leftarrow \mathcal{B}, \quad \Delta \leftarrow \emptyset$ 
4      for  $\ell \in \Lambda$  // next candidate
5          if  $\neg \ell \in \sigma$  continue
6           $\Delta \leftarrow \Delta \cup \{\ell\}$ 
7          if  $\ell \in \sigma$  continue
8           $\sigma' \leftarrow \text{UnitPropagation}(\mathcal{F}, \sigma \cup \{\ell\})$ 
9          if  $\perp \in \sigma'$  then
10              $\mathcal{B} \leftarrow \mathcal{B} \cup \{\neg \ell\}$ 
11              $\Delta \leftarrow \Delta \cup \mathcal{B} \cup \neg \mathcal{B}$ 
12              $\sigma \leftarrow \sigma \cup \mathcal{B}$ 
13         else  $\sigma \leftarrow \sigma'$ 
14      $\Lambda \leftarrow \Lambda \setminus \Delta$ 
15 return  $\mathcal{B}$ 
  
```

Keep Assignment Big Backbone



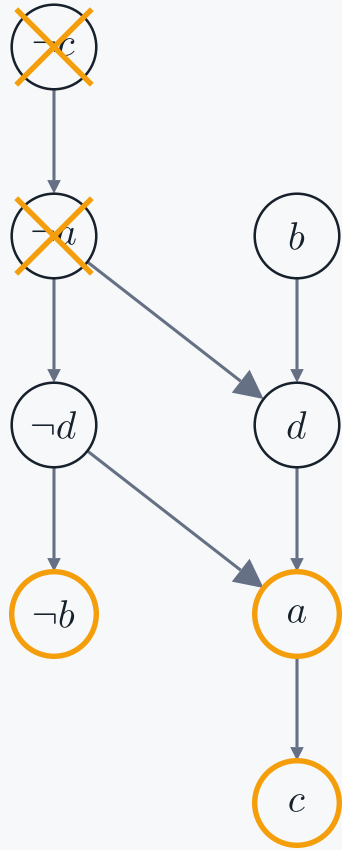
$\Lambda =$

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

```

1   $\mathcal{B} \leftarrow \emptyset, \quad \Lambda \leftarrow \mathcal{L}$ 
2  while  $\Lambda \neq \emptyset$ 
3       $\sigma \leftarrow \mathcal{B}, \quad \Delta \leftarrow \emptyset$ 
4      for  $\ell \in \Lambda$  // next candidate
5          if  $\neg \ell \in \sigma$  continue
6           $\Delta \leftarrow \Delta \cup \{\ell\}$ 
7          if  $\ell \in \sigma$  continue
8           $\sigma' \leftarrow \text{UnitPropagation}(\mathcal{F}, \sigma \cup \{\ell\})$ 
9          if  $\perp \in \sigma'$  then
10              $\mathcal{B} \leftarrow \mathcal{B} \cup \{\neg \ell\}$ 
11              $\Delta \leftarrow \Delta \cup \mathcal{B} \cup \neg \mathcal{B}$ 
12              $\sigma \leftarrow \sigma \cup \mathcal{B}$ 
13         else  $\sigma \leftarrow \sigma'$ 
14      $\Lambda \leftarrow \Lambda \setminus \Delta$ 
15 return  $\mathcal{B}$ 
  
```

Keep Assignment Big Backbone



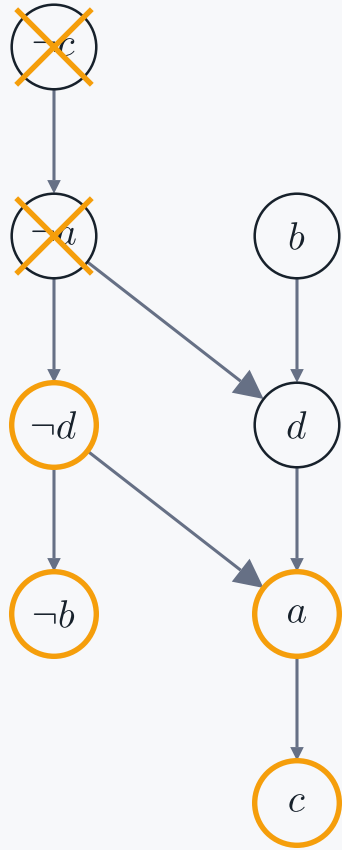
$\Lambda =$

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

```

1   $\mathcal{B} \leftarrow \emptyset, \quad \Lambda \leftarrow \mathcal{L}$ 
2  while  $\Lambda \neq \emptyset$ 
3       $\sigma \leftarrow \mathcal{B}, \quad \Delta \leftarrow \emptyset$ 
4      for  $\ell \in \Lambda$  // next candidate
5          if  $\neg \ell \in \sigma$  continue
6           $\Delta \leftarrow \Delta \cup \{\ell\}$ 
7          if  $\ell \in \sigma$  continue
8           $\sigma' \leftarrow \text{UnitPropagation}(\mathcal{F}, \sigma \cup \{\ell\})$ 
9          if  $\perp \in \sigma'$  then
10              $\mathcal{B} \leftarrow \mathcal{B} \cup \{\neg \ell\}$ 
11              $\Delta \leftarrow \Delta \cup \mathcal{B} \cup \neg \mathcal{B}$ 
12              $\sigma \leftarrow \sigma \cup \mathcal{B}$ 
13         else  $\sigma \leftarrow \sigma'$ 
14      $\Lambda \leftarrow \Lambda \setminus \Delta$ 
15 return  $\mathcal{B}$ 
  
```

Keep Assignment Big Backbone



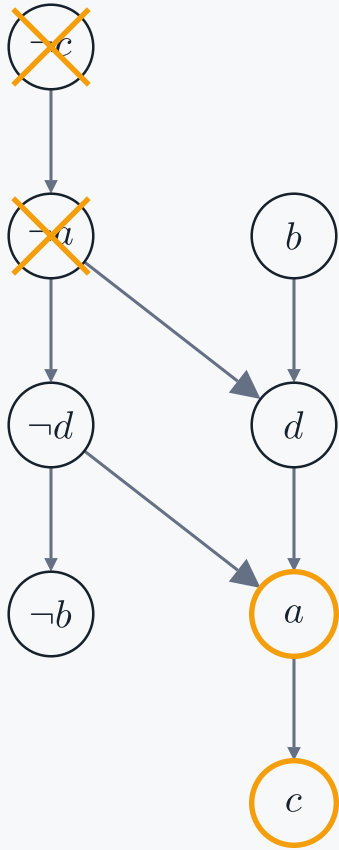
$\Lambda =$

a	b	c	d
¬a	¬b	¬c	¬d

```

1   $\mathcal{B} \leftarrow \emptyset, \quad \Lambda \leftarrow \mathcal{L}$ 
2  while  $\Lambda \neq \emptyset$ 
3       $\sigma \leftarrow \mathcal{B}, \quad \Delta \leftarrow \emptyset$ 
4      for  $\ell \in \Lambda$  // next candidate
5          if  $\neg \ell \in \sigma$  continue
6           $\Delta \leftarrow \Delta \cup \{\ell\}$ 
7          if  $\ell \in \sigma$  continue
8           $\sigma' \leftarrow \text{UnitPropagation}(\mathcal{F}, \sigma \cup \{\ell\})$ 
9          if  $\perp \in \sigma'$  then
10              $\mathcal{B} \leftarrow \mathcal{B} \cup \{\neg \ell\}$ 
11              $\Delta \leftarrow \Delta \cup \mathcal{B} \cup \neg \mathcal{B}$ 
12              $\sigma \leftarrow \sigma \cup \mathcal{B}$ 
13         else  $\sigma \leftarrow \sigma'$ 
14      $\Lambda \leftarrow \Lambda \setminus \Delta$ 
15 return  $\mathcal{B}$ 
  
```

Keep Assignment Big Backbone



$\Lambda =$

a	b	c	d
¬a	¬b	¬c	¬d

```

1   $\mathcal{B} \leftarrow \emptyset, \quad \Lambda \leftarrow \mathcal{L}$ 
2  while  $\Lambda \neq \emptyset$ 
3       $\sigma \leftarrow \mathcal{B}, \quad \Delta \leftarrow \emptyset$ 
4      for  $\ell \in \Lambda$  // next candidate
5          if  $\neg \ell \in \sigma$  continue
6           $\Delta \leftarrow \Delta \cup \{\ell\}$ 
7          if  $\ell \in \sigma$  continue
8           $\sigma' \leftarrow \text{UnitPropagation}(\mathcal{F}, \sigma \cup \{\ell\})$ 
9          if  $\perp \in \sigma'$  then
10              $\mathcal{B} \leftarrow \mathcal{B} \cup \{\neg \ell\}$ 
11              $\Delta \leftarrow \Delta \cup \mathcal{B} \cup \neg \mathcal{B}$ 
12              $\sigma \leftarrow \sigma \cup \mathcal{B}$ 
13         else  $\sigma \leftarrow \sigma'$ 
14      $\Lambda \leftarrow \Lambda \setminus \Delta$ 
15 return  $\mathcal{B}$ 
    
```

Only Sound on 2-CNF

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

$$\Lambda = a, b, \dots$$

Models

$$(\neg a \wedge b \wedge c), (a \wedge \neg b \vee \neg c), \dots$$

```
1   $\mathcal{B} \leftarrow \emptyset, \quad \Lambda \leftarrow \mathcal{L}$ 
2  while  $\Lambda \neq \emptyset$ 
3       $\sigma \leftarrow \mathcal{B}, \quad \Delta \leftarrow \emptyset$ 
4      for  $\ell \in \Lambda$  // next candidate
5          if  $\neg \ell \in \sigma$  continue
6           $\Delta \leftarrow \Delta \cup \{\ell\}$ 
7          if  $\ell \in \sigma$  continue
8           $\sigma' \leftarrow \text{UnitPropagation}(\mathcal{F}, \sigma \cup \{\ell\})$ 
9          if  $\perp \in \sigma'$  then
10              $\mathcal{B} \leftarrow \mathcal{B} \cup \{\neg \ell\}$ 
11              $\Delta \leftarrow \Delta \cup \mathcal{B} \cup \neg \mathcal{B}$ 
12              $\sigma \leftarrow \sigma \cup \mathcal{B}$ 
13         else  $\sigma \leftarrow \sigma'$ 
14      $\Lambda \leftarrow \Lambda \setminus \Delta$ 
15 return  $\mathcal{B}$ 
```

Only Sound on 2-CNF

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

$$\Lambda = a, b, \dots$$

Models

$$(\neg a \wedge b \wedge c), (a \wedge \neg b \vee \neg c), \dots$$

```
1   $\mathcal{B} \leftarrow \emptyset, \quad \Lambda \leftarrow \mathcal{L}$ 
2  while  $\Lambda \neq \emptyset$ 
3       $\sigma \leftarrow \mathcal{B}, \quad \Delta \leftarrow \emptyset$ 
4      for  $\ell \in \Lambda$  // next candidate
5          if  $\neg \ell \in \sigma$  continue
6           $\Delta \leftarrow \Delta \cup \{\ell\}$ 
7          if  $\ell \in \sigma$  continue
8           $\sigma' \leftarrow \text{UnitPropagation}(\mathcal{F}, \sigma \cup \{\ell\})$ 
9          if  $\perp \in \sigma'$  then
10              $\mathcal{B} \leftarrow \mathcal{B} \cup \{\neg \ell\}$ 
11              $\Delta \leftarrow \Delta \cup \mathcal{B} \cup \neg \mathcal{B}$ 
12              $\sigma \leftarrow \sigma \cup \mathcal{B}$ 
13         else  $\sigma \leftarrow \sigma'$ 
14      $\Lambda \leftarrow \Lambda \setminus \Delta$ 
15 return  $\mathcal{B}$ 
```


Only Sound on 2-CNF

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

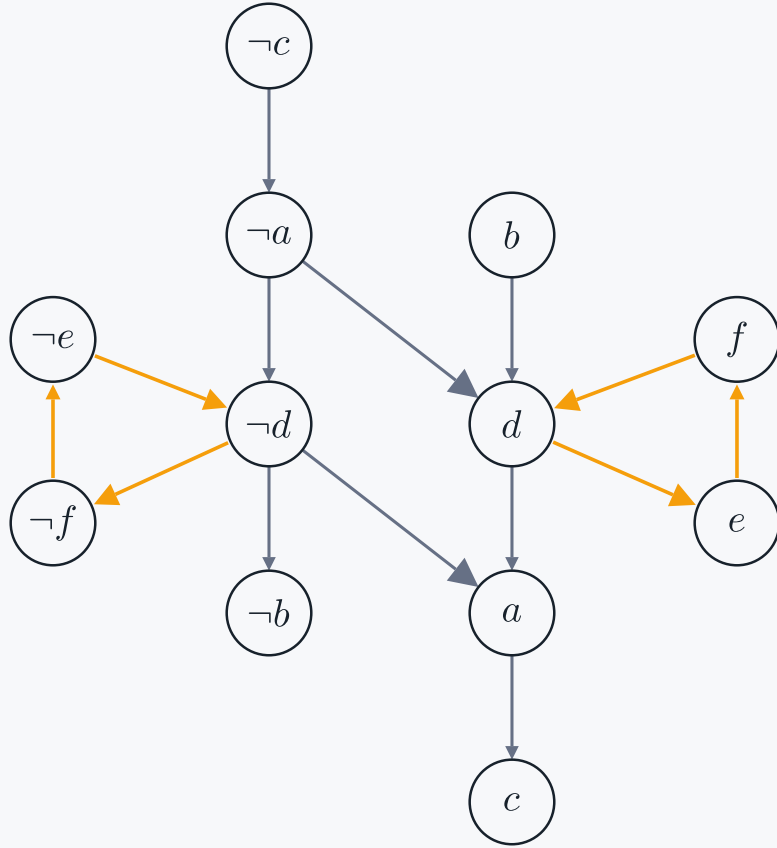
$$\Lambda = a, b, \dots$$

Models

$$(\neg a \wedge b \wedge c), (a \wedge \neg b \vee \neg c), \dots$$

```
1   $\mathcal{B} \leftarrow \emptyset, \quad \Lambda \leftarrow \mathcal{L}$ 
2  while  $\Lambda \neq \emptyset$ 
3       $\sigma \leftarrow \mathcal{B}, \quad \Delta \leftarrow \emptyset$ 
4      for  $\ell \in \Lambda$  // next candidate
5          if  $\neg \ell \in \sigma$  continue
6           $\Delta \leftarrow \Delta \cup \{\ell\}$ 
7          if  $\ell \in \sigma$  continue
8           $\sigma' \leftarrow \text{UnitPropagation}(\mathcal{F}, \sigma \cup \{\ell\})$ 
9          if  $\perp \in \sigma'$  then
10              $\mathcal{B} \leftarrow \mathcal{B} \cup \{\neg \ell\}$ 
11              $\Delta \leftarrow \Delta \cup \mathcal{B} \cup \neg \mathcal{B}$ 
12              $\sigma \leftarrow \sigma \cup \mathcal{B}$ 
13         else  $\sigma \leftarrow \sigma'$ 
14      $\Lambda \leftarrow \Lambda \setminus \Delta$ 
15 return  $\mathcal{B}$ 
```

Equivalent Literal Substitution

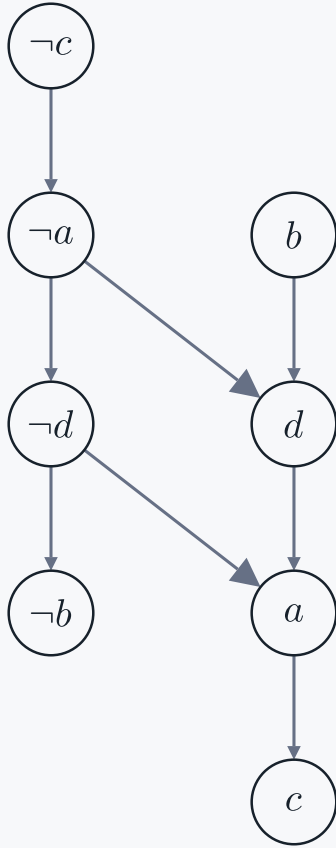


```

1   $\mathcal{B} \leftarrow \emptyset, \quad \Lambda \leftarrow \mathcal{L}$ 
2  while  $\Lambda \neq \emptyset$ 
3       $\sigma \leftarrow \mathcal{B}, \quad \Delta \leftarrow \emptyset$ 
4      for  $\ell \in \Lambda$  // next candidate
5          if  $\neg \ell \in \sigma$  continue
6           $\Delta \leftarrow \Delta \cup \{\ell\}$ 
7          if  $\ell \in \sigma$  continue
8           $\sigma' \leftarrow \text{UnitPropagation}(\mathcal{F}, \sigma \cup \{\ell\})$ 
9          if  $\perp \in \sigma'$  then
10              $\mathcal{B} \leftarrow \mathcal{B} \cup \{\neg \ell\}$ 
11              $\Delta \leftarrow \Delta \cup \mathcal{B} \cup \neg \mathcal{B}$ 
12              $\sigma \leftarrow \sigma \cup \mathcal{B}$ 
13         else  $\sigma \leftarrow \sigma'$ 
14      $\Lambda \leftarrow \Lambda \setminus \Delta$ 
15 return  $\mathcal{B}$ 

```

Roots Only

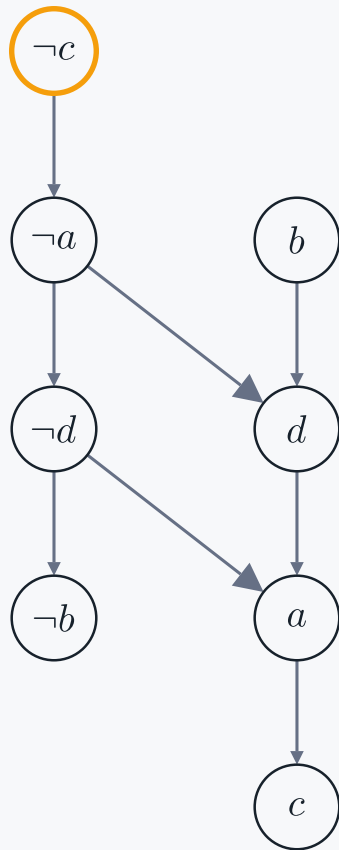


$\Lambda = \neg c, b$

```

1   $\mathcal{B} \leftarrow \emptyset, \quad \Lambda \leftarrow \mathcal{L}$ 
2  while  $\Lambda \neq \emptyset$ 
3       $\sigma \leftarrow \mathcal{B}, \quad \Delta \leftarrow \emptyset$ 
4      for  $\ell \in \Lambda$  // next candidate
5          if  $\neg \ell \in \sigma$  continue
6           $\Delta \leftarrow \Delta \cup \{\ell\}$ 
7          if  $\ell \in \sigma$  continue
8           $\sigma' \leftarrow \text{UnitPropagation}(\mathcal{F}, \sigma \cup \{\ell\})$ 
9          if  $\perp \in \sigma'$  then
10              $\mathcal{B} \leftarrow \mathcal{B} \cup \{\neg \ell\}$ 
11              $\Delta \leftarrow \Delta \cup \mathcal{B} \cup \neg \mathcal{B}$ 
12              $\sigma \leftarrow \sigma \cup \mathcal{B}$ 
13         else  $\sigma \leftarrow \sigma'$ 
14      $\Lambda \leftarrow \Lambda \setminus \Delta$ 
15 return  $\mathcal{B}$ 
  
```

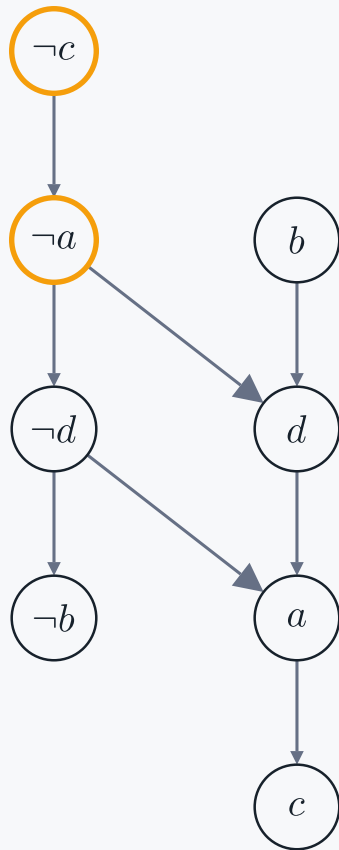
DFS vs BFS



$\Lambda = \neg c, b$

```
1   $\mathcal{B} \leftarrow \emptyset, \quad \Lambda \leftarrow \mathcal{L}$ 
2  while  $\Lambda \neq \emptyset$ 
3       $\sigma \leftarrow \mathcal{B}, \quad \Delta \leftarrow \emptyset$ 
4      for  $\ell \in \Lambda$  // next candidate
5          if  $\neg \ell \in \sigma$  continue
6           $\Delta \leftarrow \Delta \cup \{\ell\}$ 
7          if  $\ell \in \sigma$  continue
8           $\sigma' \leftarrow \text{UnitPropagation}(\mathcal{F}, \sigma \cup \{\ell\})$ 
9          if  $\perp \in \sigma'$  then
10              $\mathcal{B} \leftarrow \mathcal{B} \cup \{\neg \ell\}$ 
11              $\Delta \leftarrow \Delta \cup \mathcal{B} \cup \neg \mathcal{B}$ 
12              $\sigma \leftarrow \sigma \cup \mathcal{B}$ 
13         else  $\sigma \leftarrow \sigma'$ 
14      $\Lambda \leftarrow \Lambda \setminus \Delta$ 
15 return  $\mathcal{B}$ 
```

DFS vs BFS



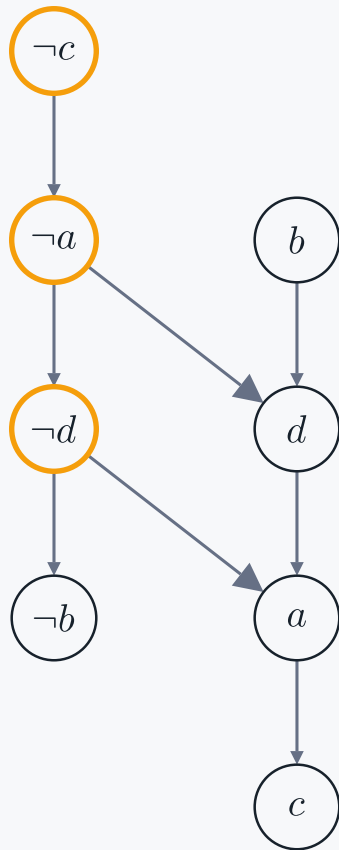
$\Lambda = \neg c, b$

```

1   $\mathcal{B} \leftarrow \emptyset, \quad \Lambda \leftarrow \mathcal{L}$ 
2  while  $\Lambda \neq \emptyset$ 
3       $\sigma \leftarrow \mathcal{B}, \quad \Delta \leftarrow \emptyset$ 
4      for  $\ell \in \Lambda$  // next candidate
5          if  $\neg \ell \in \sigma$  continue
6           $\Delta \leftarrow \Delta \cup \{\ell\}$ 
7          if  $\ell \in \sigma$  continue
8           $\sigma' \leftarrow \text{UnitPropagation}(\mathcal{F}, \sigma \cup \{\ell\})$ 
9          if  $\perp \in \sigma'$  then
10              $\mathcal{B} \leftarrow \mathcal{B} \cup \{\neg \ell\}$ 
11              $\Delta \leftarrow \Delta \cup \mathcal{B} \cup \neg \mathcal{B}$ 
12              $\sigma \leftarrow \sigma \cup \mathcal{B}$ 
13         else  $\sigma \leftarrow \sigma'$ 
14      $\Lambda \leftarrow \Lambda \setminus \Delta$ 
15 return  $\mathcal{B}$ 

```

DFS vs BFS



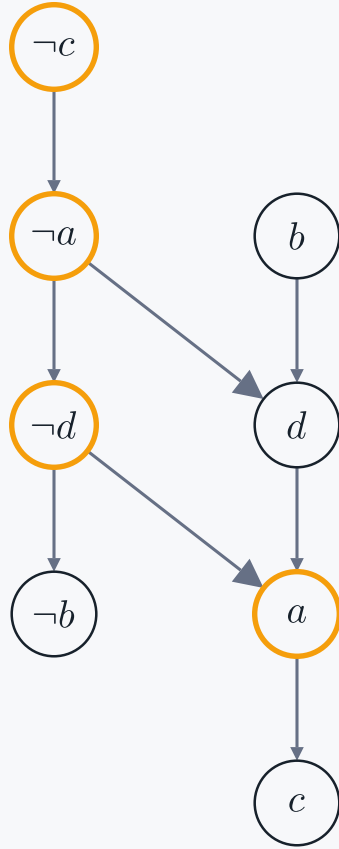
$\Lambda = \neg c, b$

```

1   $\mathcal{B} \leftarrow \emptyset, \quad \Lambda \leftarrow \mathcal{L}$ 
2  while  $\Lambda \neq \emptyset$ 
3       $\sigma \leftarrow \mathcal{B}, \quad \Delta \leftarrow \emptyset$ 
4      for  $\ell \in \Lambda$  // next candidate
5          if  $\neg \ell \in \sigma$  continue
6           $\Delta \leftarrow \Delta \cup \{\ell\}$ 
7          if  $\ell \in \sigma$  continue
8           $\sigma' \leftarrow \text{UnitPropagation}(\mathcal{F}, \sigma \cup \{\ell\})$ 
9          if  $\perp \in \sigma'$  then
10              $\mathcal{B} \leftarrow \mathcal{B} \cup \{\neg \ell\}$ 
11              $\Delta \leftarrow \Delta \cup \mathcal{B} \cup \neg \mathcal{B}$ 
12              $\sigma \leftarrow \sigma \cup \mathcal{B}$ 
13         else  $\sigma \leftarrow \sigma'$ 
14      $\Lambda \leftarrow \Lambda \setminus \Delta$ 
15 return  $\mathcal{B}$ 

```

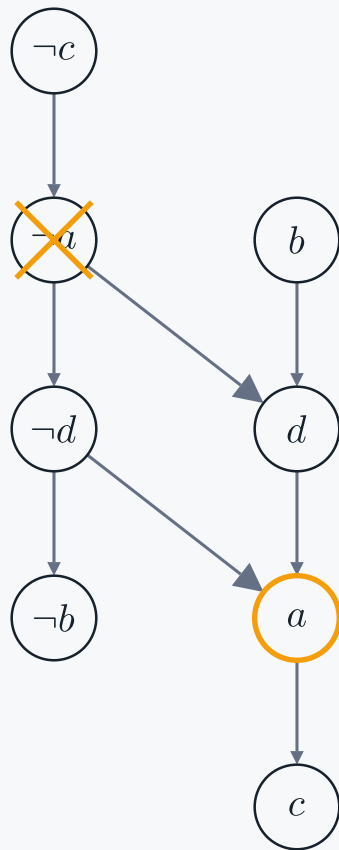
DFS vs BFS



$\Lambda = \neg c, b$

```
1   $\mathcal{B} \leftarrow \emptyset, \quad \Lambda \leftarrow \mathcal{L}$ 
2  while  $\Lambda \neq \emptyset$ 
3       $\sigma \leftarrow \mathcal{B}, \quad \Delta \leftarrow \emptyset$ 
4      for  $\ell \in \Lambda$  // next candidate
5          if  $\neg \ell \in \sigma$  continue
6           $\Delta \leftarrow \Delta \cup \{\ell\}$ 
7          if  $\ell \in \sigma$  continue
8           $\sigma' \leftarrow \text{UnitPropagation}(\mathcal{F}, \sigma \cup \{\ell\})$ 
9          if  $\perp \in \sigma'$  then
10              $\mathcal{B} \leftarrow \mathcal{B} \cup \{\neg \ell\}$ 
11              $\Delta \leftarrow \Delta \cup \mathcal{B} \cup \neg \mathcal{B}$ 
12              $\sigma \leftarrow \sigma \cup \mathcal{B}$ 
13         else  $\sigma \leftarrow \sigma'$ 
14      $\Lambda \leftarrow \Lambda \setminus \Delta$ 
15 return  $\mathcal{B}$ 
```

DFS vs BFS

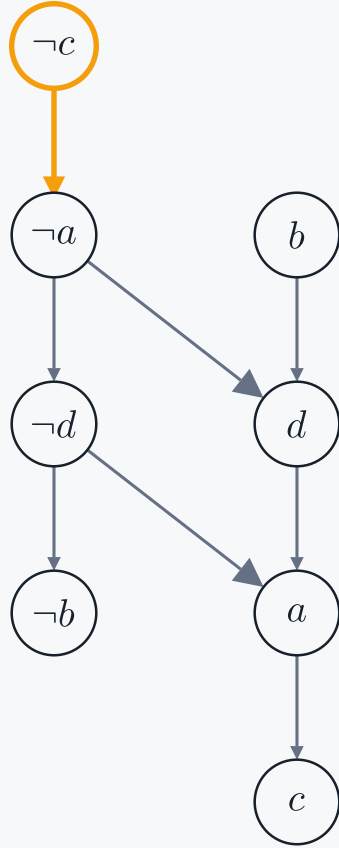


$\Lambda = \neg c, b$

```

1   $\mathcal{B} \leftarrow \emptyset, \quad \Lambda \leftarrow \mathcal{L}$ 
2  while  $\Lambda \neq \emptyset$ 
3       $\sigma \leftarrow \mathcal{B}, \quad \Delta \leftarrow \emptyset$ 
4      for  $\ell \in \Lambda$  // next candidate
5          if  $\neg \ell \in \sigma$  continue
6           $\Delta \leftarrow \Delta \cup \{\ell\}$ 
7          if  $\ell \in \sigma$  continue
8           $\sigma' \leftarrow \text{UnitPropagation}(\mathcal{F}, \sigma \cup \{\ell\})$ 
9          if  $\perp \in \sigma'$  then
10              $\mathcal{B} \leftarrow \mathcal{B} \cup \{\neg \ell\}$ 
11              $\Delta \leftarrow \Delta \cup \mathcal{B} \cup \neg \mathcal{B}$ 
12              $\sigma \leftarrow \sigma \cup \mathcal{B}$ 
13         else  $\sigma \leftarrow \sigma'$ 
14      $\Lambda \leftarrow \Lambda \setminus \Delta$ 
15 return  $\mathcal{B}$ 
  
```


Unique Implication Point



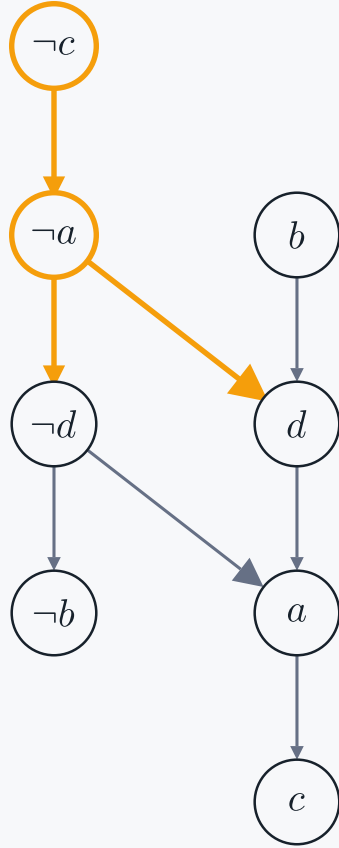
$$\Lambda = \neg c, b$$

```

1   $\mathcal{B} \leftarrow \emptyset, \quad \Lambda \leftarrow \mathcal{L}$ 
2  while  $\Lambda \neq \emptyset$ 
3       $\sigma \leftarrow \mathcal{B}, \quad \Delta \leftarrow \emptyset$ 
4      for  $\ell \in \Lambda$  // next candidate
5          if  $\neg \ell \in \sigma$  continue
6           $\Delta \leftarrow \Delta \cup \{\ell\}$ 
7          if  $\ell \in \sigma$  continue
8           $\sigma' \leftarrow \text{UnitPropagation}(\mathcal{F}, \sigma \cup \{\ell\})$ 
9          if  $\perp \in \sigma'$  then
10              $\mathcal{B} \leftarrow \mathcal{B} \cup \{\neg \ell\}$ 
11              $\Delta \leftarrow \Delta \cup \mathcal{B} \cup \neg \mathcal{B}$ 
12              $\sigma \leftarrow \sigma \cup \mathcal{B}$ 
13         else  $\sigma \leftarrow \sigma'$ 
14      $\Lambda \leftarrow \Lambda \setminus \Delta$ 
15 return  $\mathcal{B}$ 

```

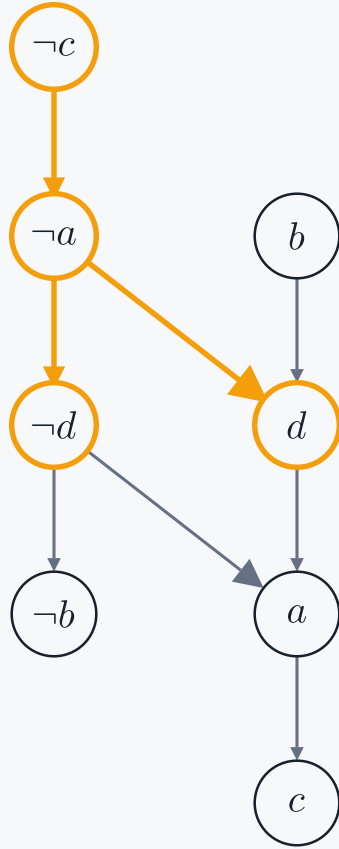
Unique Implication Point



$$\Lambda = \neg c, b$$

```
1   $\mathcal{B} \leftarrow \emptyset, \quad \Lambda \leftarrow \mathcal{L}$ 
2  while  $\Lambda \neq \emptyset$ 
3       $\sigma \leftarrow \mathcal{B}, \quad \Delta \leftarrow \emptyset$ 
4      for  $\ell \in \Lambda$  // next candidate
5          if  $\neg \ell \in \sigma$  continue
6           $\Delta \leftarrow \Delta \cup \{\ell\}$ 
7          if  $\ell \in \sigma$  continue
8           $\sigma' \leftarrow \text{UnitPropagation}(\mathcal{F}, \sigma \cup \{\ell\})$ 
9          if  $\perp \in \sigma'$  then
10              $\mathcal{B} \leftarrow \mathcal{B} \cup \{\neg \ell\}$ 
11              $\Delta \leftarrow \Delta \cup \mathcal{B} \cup \neg \mathcal{B}$ 
12              $\sigma \leftarrow \sigma \cup \mathcal{B}$ 
13         else  $\sigma \leftarrow \sigma'$ 
14      $\Lambda \leftarrow \Lambda \setminus \Delta$ 
15 return  $\mathcal{B}$ 
```

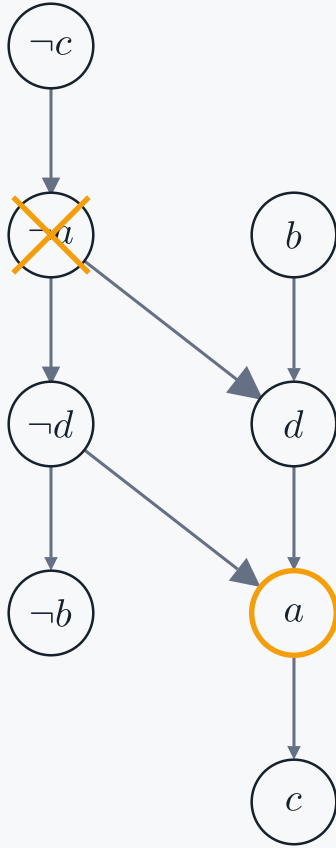
Unique Implication Point



$$\Lambda = \neg c, b$$

```
1   $\mathcal{B} \leftarrow \emptyset, \quad \Lambda \leftarrow \mathcal{L}$ 
2  while  $\Lambda \neq \emptyset$ 
3       $\sigma \leftarrow \mathcal{B}, \quad \Delta \leftarrow \emptyset$ 
4      for  $\ell \in \Lambda$  // next candidate
5          if  $\neg \ell \in \sigma$  continue
6           $\Delta \leftarrow \Delta \cup \{\ell\}$ 
7          if  $\ell \in \sigma$  continue
8           $\sigma' \leftarrow \text{UnitPropagation}(\mathcal{F}, \sigma \cup \{\ell\})$ 
9          if  $\perp \in \sigma'$  then
10              $\mathcal{B} \leftarrow \mathcal{B} \cup \{\neg \ell\}$ 
11              $\Delta \leftarrow \Delta \cup \mathcal{B} \cup \neg \mathcal{B}$ 
12              $\sigma \leftarrow \sigma \cup \mathcal{B}$ 
13         else  $\sigma \leftarrow \sigma'$ 
14      $\Lambda \leftarrow \Lambda \setminus \Delta$ 
15 return  $\mathcal{B}$ 
```

Unique Implication Point



$$\Lambda = \neg c, b$$

```

1   $\mathcal{B} \leftarrow \emptyset, \quad \Lambda \leftarrow \mathcal{L}$ 
2  while  $\Lambda \neq \emptyset$ 
3       $\sigma \leftarrow \mathcal{B}, \quad \Delta \leftarrow \emptyset$ 
4      for  $\ell \in \Lambda$  // next candidate
5          if  $\neg \ell \in \sigma$  continue
6           $\Delta \leftarrow \Delta \cup \{\ell\}$ 
7          if  $\ell \in \sigma$  continue
8           $\sigma' \leftarrow \text{UnitPropagation}(\mathcal{F}, \sigma \cup \{\ell\})$ 
9          if  $\perp \in \sigma'$  then
10              $\mathcal{B} \leftarrow \mathcal{B} \cup \{\neg \ell\}$ 
11              $\Delta \leftarrow \Delta \cup \mathcal{B} \cup \neg \mathcal{B}$ 
12              $\sigma \leftarrow \sigma \cup \mathcal{B}$ 
13         else  $\sigma \leftarrow \sigma'$ 
14      $\Lambda \leftarrow \Lambda \setminus \Delta$ 
15 return  $\mathcal{B}$ 

```

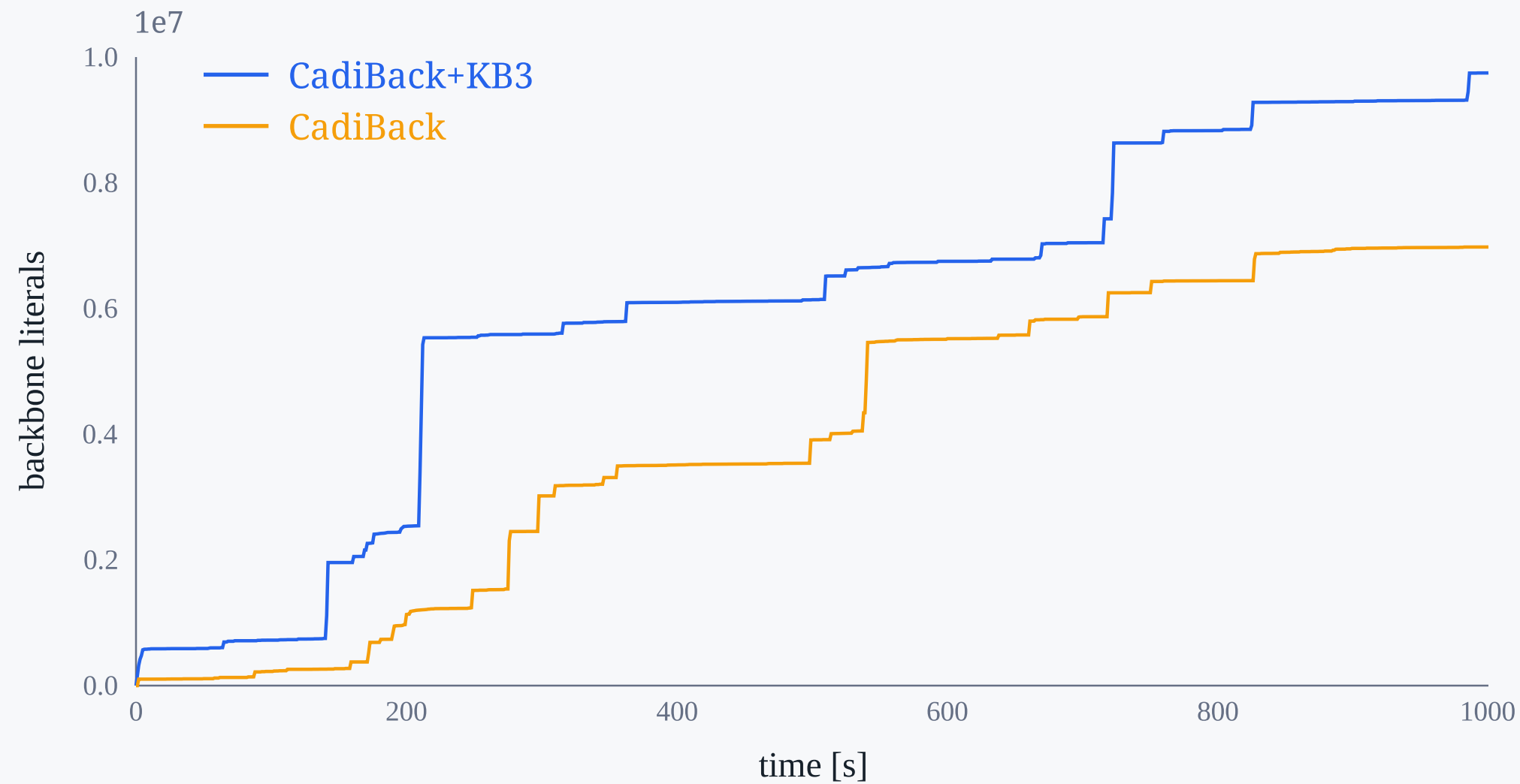
Runtime comparison

	backbone		KB3	
	BFS	DFS	BFS	DFS
Base	21 136	21 287	648	728
ELS	20 524	20 757	640	733
ELS+Roots	18 164	18 756	644	721
ELS+Roots+UIP			822	

- Total time in seconds on 1798 instances
- ELS takes 51 seconds

Anytime advantage with KB3

SC2020–SC2022



References

Biere, Armin; Fleury, Mathias; Heisinger, Maximilian; Fazekas, Katalin (2020).

Entering the SAT Competition 2020.
page 4.

Bradley, Aaron R. (2012).

Understanding IC3.

In Cimatti, Alessandro and Sebastiani, Roberto, editor, Theory and Applications of Satisfiability Testing – SAT 2012, Lecture Notes in Computer Science, pages 1–14, Berlin, Heidelberg. Springer.

Cabodi, G.; Camurati, P. E.; Mishchenko, A.; Palena, M.; Pasini, P. (2017).

SAT Solver Management Strategies in IC3: An Experimental Approach.

Formal Methods in System Design, 50:39–74.

Eén, Niklas and Sörensson, Niklas (2003).

Temporal Induction by Incremental SAT Solving.

Electronic Notes in Theoretical Computer Science, 89(4):543–560.

[Knuth15] Knuth, Donald E. (2015). The Art of Computer Programming, Vol. 4, Pre-Fascicle 6A: Satisfiability.

Draft of §7.2.2.2, 23 September 2015, p. 81, Eq. (131). <https://www-cs.stanford.edu/~knuth/fasc6a.ps.gz>

Eén, Niklas and Sörensson, Niklas (2004).

An Extensible SAT-Solver.

In Giunchiglia, Enrico and Tacchella, Armando, editor, Theory and Applications of Satisfiability Testing, Lecture Notes in Computer Science, pages 502–518, Berlin, Heidelberg. Springer.

Preiner, Mathias and Biere, Armin (2019).

Hardware Model Checking Competition 2019.

<http://fmv.jku.at/hwmcc19/>.

Fazekas, K., Biere, A., and Scholl, C. (2019).

Incremental Inprocessing in SAT Solving.

In Janota, M. and Lynce, I., editors, Theory and Applications of Satisfiability Testing – SAT 2019, Lecture Notes in Computer Science, pages 136–154, Cham. Springer International Publishing.

Froleyks, N. and Biere, A. (2021).

Single Clause Assumption without Activation Literals to Speed-up IC3.

In 2021 Formal Methods in Computer Aided Design (FMCAD), pages 72–76.

$$(x_1 \vee x_{13})$$

$$(\neg x_1 \vee \neg x_2 \vee x_{14})$$

$$(x_3 \vee x_{15})$$

$$(x_4 \vee x_{16})$$

$$(\neg x_5 \vee \neg x_3 \vee x_6)$$

$$(\neg x_5 \vee \neg x_7)$$

$$(\neg x_6 \vee x_7 \vee x_8)$$

$$(\neg x_4 \vee \neg x_8 \vee \neg x_9)$$

$$(\neg x_1 \vee x_9 \vee \neg x_{10})$$

$$(x_9 \vee x_{11} \vee \neg x_{14})$$

$$(x_{10} \vee \neg x_{11} \vee x_{12})$$

$$(\neg x_2 \vee \neg x_{11} \vee \neg x_{12})$$

$$(x_1 \vee x_{13})$$

$$(\neg x_1 \vee \neg x_2 \vee x_{14})$$

$$(x_3 \vee x_{15})$$

$$(x_4 \vee x_{16})$$

$$(\neg x_5 \vee \neg x_3 \vee x_6)$$

$$(\neg x_5 \vee \neg x_7)$$

$$(\neg x_6 \vee x_7 \vee x_8)$$

$$(\neg x_4 \vee \neg x_8 \vee \neg x_9)$$

$$(\neg x_1 \vee x_9 \vee \neg x_{10})$$

$$(x_9 \vee x_{11} \vee \neg x_{14})$$

$$(x_{10} \vee \neg x_{11} \vee x_{12})$$

$$(\neg x_2 \vee \neg x_{11} \vee \neg x_{12})$$

$x_1 = 1$

@1

$$(x_1 \vee x_{13})$$

$$(\neg x_1 \vee \neg x_2 \vee x_{14})$$

$$(x_3 \vee x_{15})$$

$$(x_4 \vee x_{16})$$

$$(\neg x_5 \vee \neg x_3 \vee x_6)$$

$$(\neg x_5 \vee \neg x_7)$$

$$(\neg x_6 \vee x_7 \vee x_8)$$

$$(\neg x_4 \vee \neg x_8 \vee \neg x_9)$$

$$(\neg x_1 \vee x_9 \vee \neg x_{10})$$

$$(x_9 \vee x_{11} \vee \neg x_{14})$$

$$(x_{10} \vee \neg x_{11} \vee x_{12})$$

$$(\neg x_2 \vee \neg x_{11} \vee \neg x_{12})$$

$$x_1 = 1$$

@1

$$x_2 = 1$$

@2

$$(x_1 \vee x_{13})$$

$$(\neg x_1 \vee \neg x_2 \vee x_{14})$$

$$(x_3 \vee x_{15})$$

$$(x_4 \vee x_{16})$$

$$(\neg x_5 \vee \neg x_3 \vee x_6)$$

$$(\neg x_5 \vee \neg x_7)$$

$$(\neg x_6 \vee x_7 \vee x_8)$$

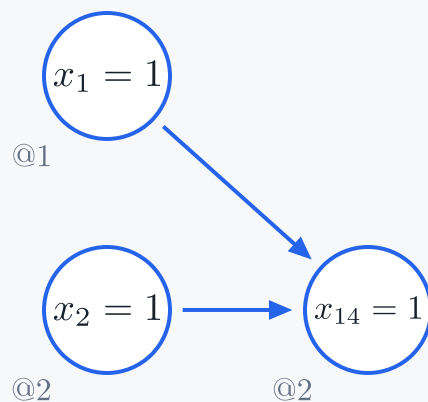
$$(\neg x_4 \vee \neg x_8 \vee \neg x_9)$$

$$(\neg x_1 \vee x_9 \vee \neg x_{10})$$

$$(x_9 \vee x_{11} \vee \neg x_{14})$$

$$(x_{10} \vee \neg x_{11} \vee x_{12})$$

$$(\neg x_2 \vee \neg x_{11} \vee \neg x_{12})$$



$$(x_1 \vee x_{13})$$

$$(\neg x_1 \vee \neg x_2 \vee x_{14})$$

$$(x_3 \vee x_{15})$$

$$(x_4 \vee x_{16})$$

$$(\neg x_5 \vee \neg x_3 \vee x_6)$$

$$(\neg x_5 \vee \neg x_7)$$

$$(\neg x_6 \vee x_7 \vee x_8)$$

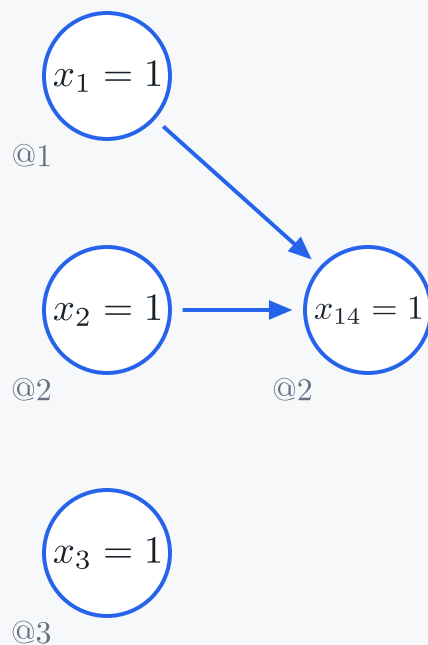
$$(\neg x_4 \vee \neg x_8 \vee \neg x_9)$$

$$(\neg x_1 \vee x_9 \vee \neg x_{10})$$

$$(x_9 \vee x_{11} \vee \neg x_{14})$$

$$(x_{10} \vee \neg x_{11} \vee x_{12})$$

$$(\neg x_2 \vee \neg x_{11} \vee \neg x_{12})$$



$$(x_1 \vee x_{13})$$

$$(\neg x_1 \vee \neg x_2 \vee x_{14})$$

$$(x_3 \vee x_{15})$$

$$(x_4 \vee x_{16})$$

$$(\neg x_5 \vee \neg x_3 \vee x_6)$$

$$(\neg x_5 \vee \neg x_7)$$

$$(\neg x_6 \vee x_7 \vee x_8)$$

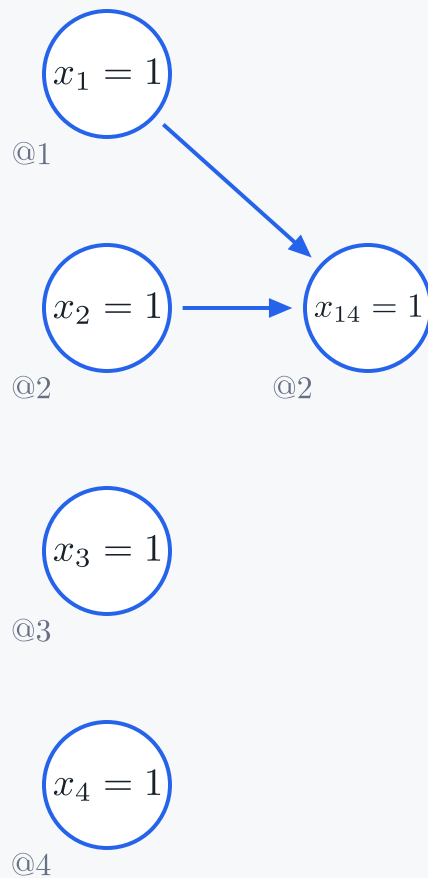
$$(\neg x_4 \vee \neg x_8 \vee \neg x_9)$$

$$(\neg x_1 \vee x_9 \vee \neg x_{10})$$

$$(x_9 \vee x_{11} \vee \neg x_{14})$$

$$(x_{10} \vee \neg x_{11} \vee x_{12})$$

$$(\neg x_2 \vee \neg x_{11} \vee \neg x_{12})$$



$$(x_1 \vee x_{13})$$

$$(\neg x_1 \vee \neg x_2 \vee x_{14})$$

$$(x_3 \vee x_{15})$$

$$(x_4 \vee x_{16})$$

$$(\neg x_5 \vee \neg x_3 \vee x_6)$$

$$(\neg x_5 \vee \neg x_7)$$

$$(\neg x_6 \vee x_7 \vee x_8)$$

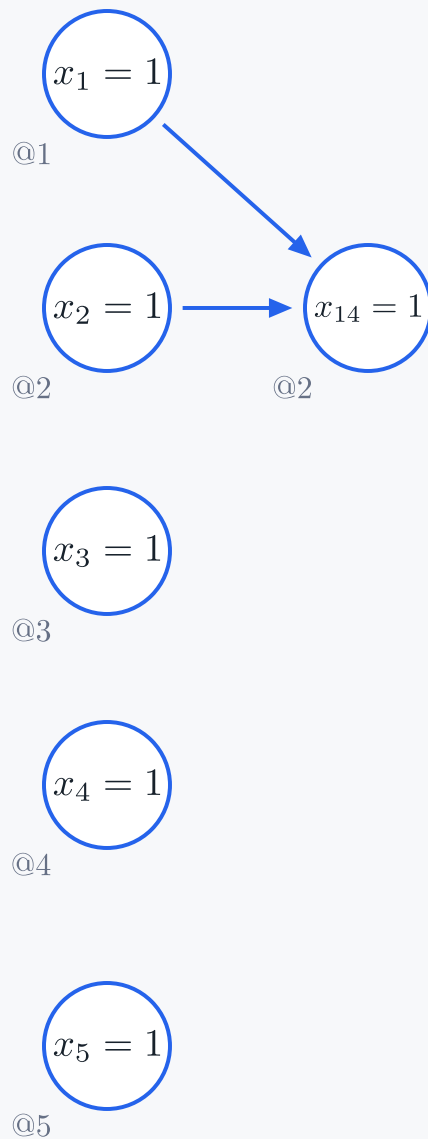
$$(\neg x_4 \vee \neg x_8 \vee \neg x_9)$$

$$(\neg x_1 \vee x_9 \vee \neg x_{10})$$

$$(x_9 \vee x_{11} \vee \neg x_{14})$$

$$(x_{10} \vee \neg x_{11} \vee x_{12})$$

$$(\neg x_2 \vee \neg x_{11} \vee \neg x_{12})$$



$$(x_1 \vee x_{13})$$

$$(\neg x_1 \vee \neg x_2 \vee x_{14})$$

$$(x_3 \vee x_{15})$$

$$(x_4 \vee x_{16})$$

$$(\neg x_5 \vee \neg x_3 \vee x_6)$$

$$(\neg x_5 \vee \neg x_7)$$

$$(\neg x_6 \vee x_7 \vee x_8)$$

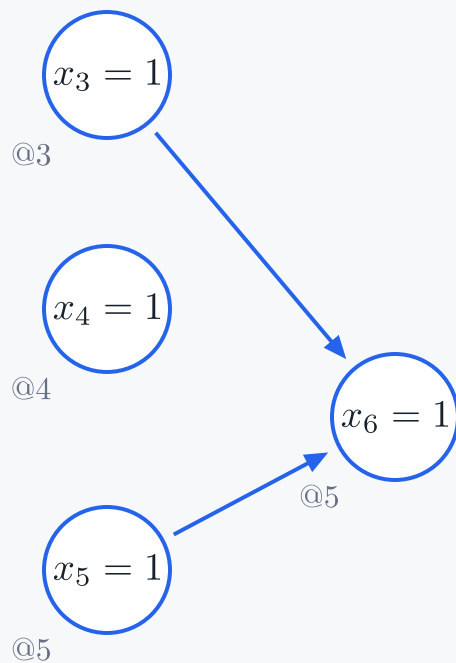
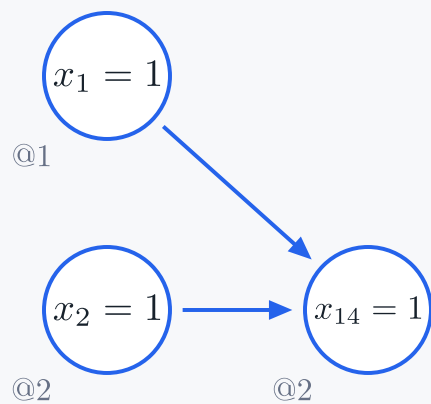
$$(\neg x_4 \vee \neg x_8 \vee \neg x_9)$$

$$(\neg x_1 \vee x_9 \vee \neg x_{10})$$

$$(x_9 \vee x_{11} \vee \neg x_{14})$$

$$(x_{10} \vee \neg x_{11} \vee x_{12})$$

$$(\neg x_2 \vee \neg x_{11} \vee \neg x_{12})$$



$$(x_1 \vee x_{13})$$

$$(\neg x_1 \vee \neg x_2 \vee x_{14})$$

$$(x_3 \vee x_{15})$$

$$(x_4 \vee x_{16})$$

$$(\neg x_5 \vee \neg x_3 \vee x_6)$$

$$(\neg x_5 \vee \neg x_7)$$

$$(\neg x_6 \vee x_7 \vee x_8)$$

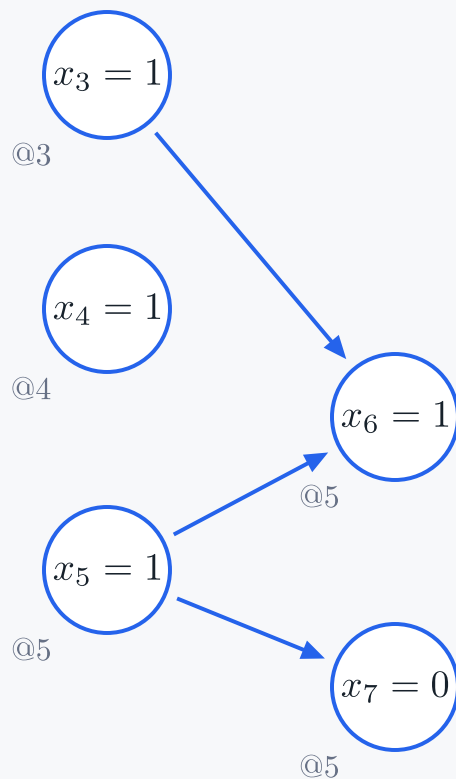
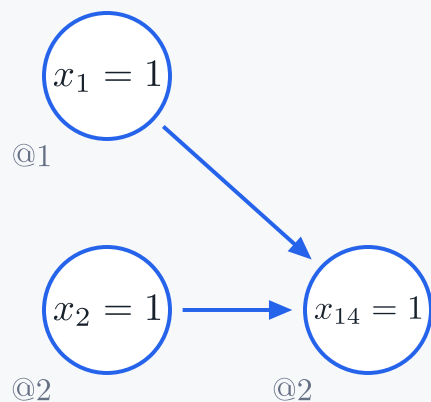
$$(\neg x_4 \vee \neg x_8 \vee \neg x_9)$$

$$(\neg x_1 \vee x_9 \vee \neg x_{10})$$

$$(x_9 \vee x_{11} \vee \neg x_{14})$$

$$(x_{10} \vee \neg x_{11} \vee x_{12})$$

$$(\neg x_2 \vee \neg x_{11} \vee \neg x_{12})$$



$$(x_1 \vee x_{13})$$

$$(\neg x_1 \vee \neg x_2 \vee x_{14})$$

$$(x_3 \vee x_{15})$$

$$(x_4 \vee x_{16})$$

$$(\neg x_5 \vee \neg x_3 \vee x_6)$$

$$(\neg x_5 \vee \neg x_7)$$

$$(\neg x_6 \vee x_7 \vee x_8)$$

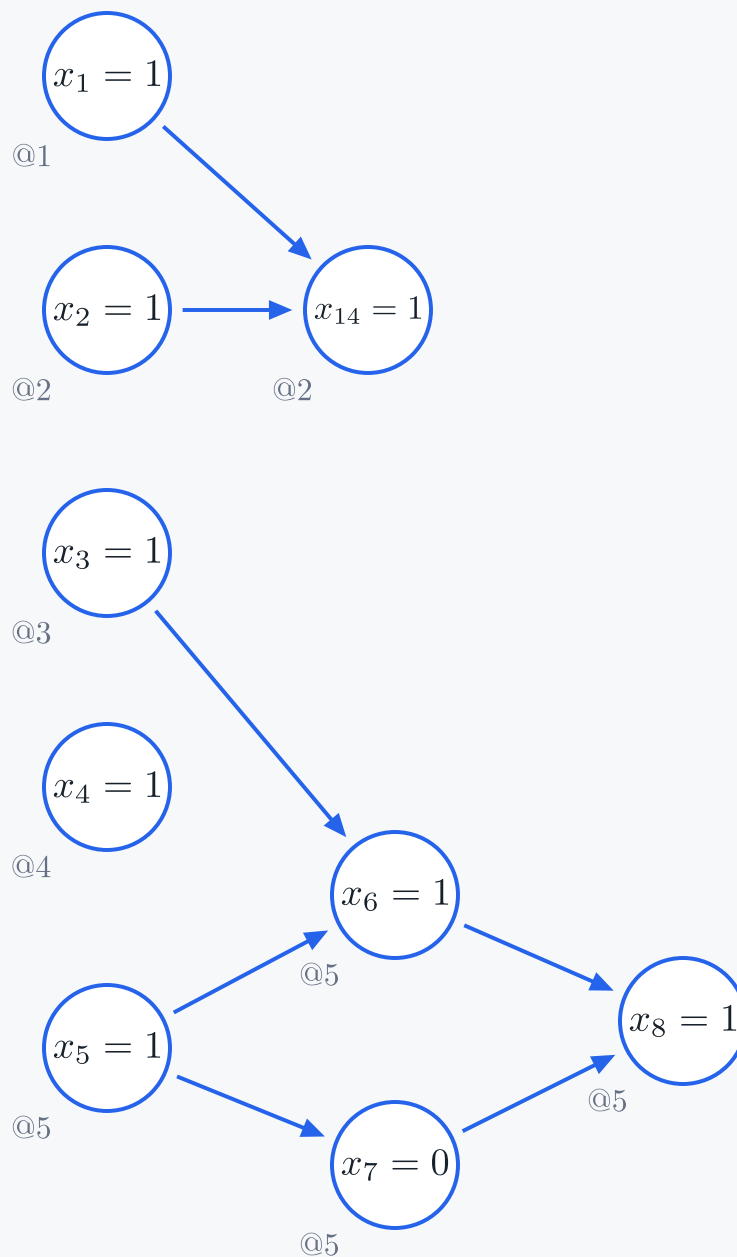
$$(\neg x_4 \vee \neg x_8 \vee \neg x_9)$$

$$(\neg x_1 \vee x_9 \vee \neg x_{10})$$

$$(x_9 \vee x_{11} \vee \neg x_{14})$$

$$(x_{10} \vee \neg x_{11} \vee x_{12})$$

$$(\neg x_2 \vee \neg x_{11} \vee \neg x_{12})$$



$$(x_1 \vee x_{13})$$

$$(\neg x_1 \vee \neg x_2 \vee x_{14})$$

$$(x_3 \vee x_{15})$$

$$(x_4 \vee x_{16})$$

$$(\neg x_5 \vee \neg x_3 \vee x_6)$$

$$(\neg x_5 \vee \neg x_7)$$

$$(\neg x_6 \vee x_7 \vee x_8)$$

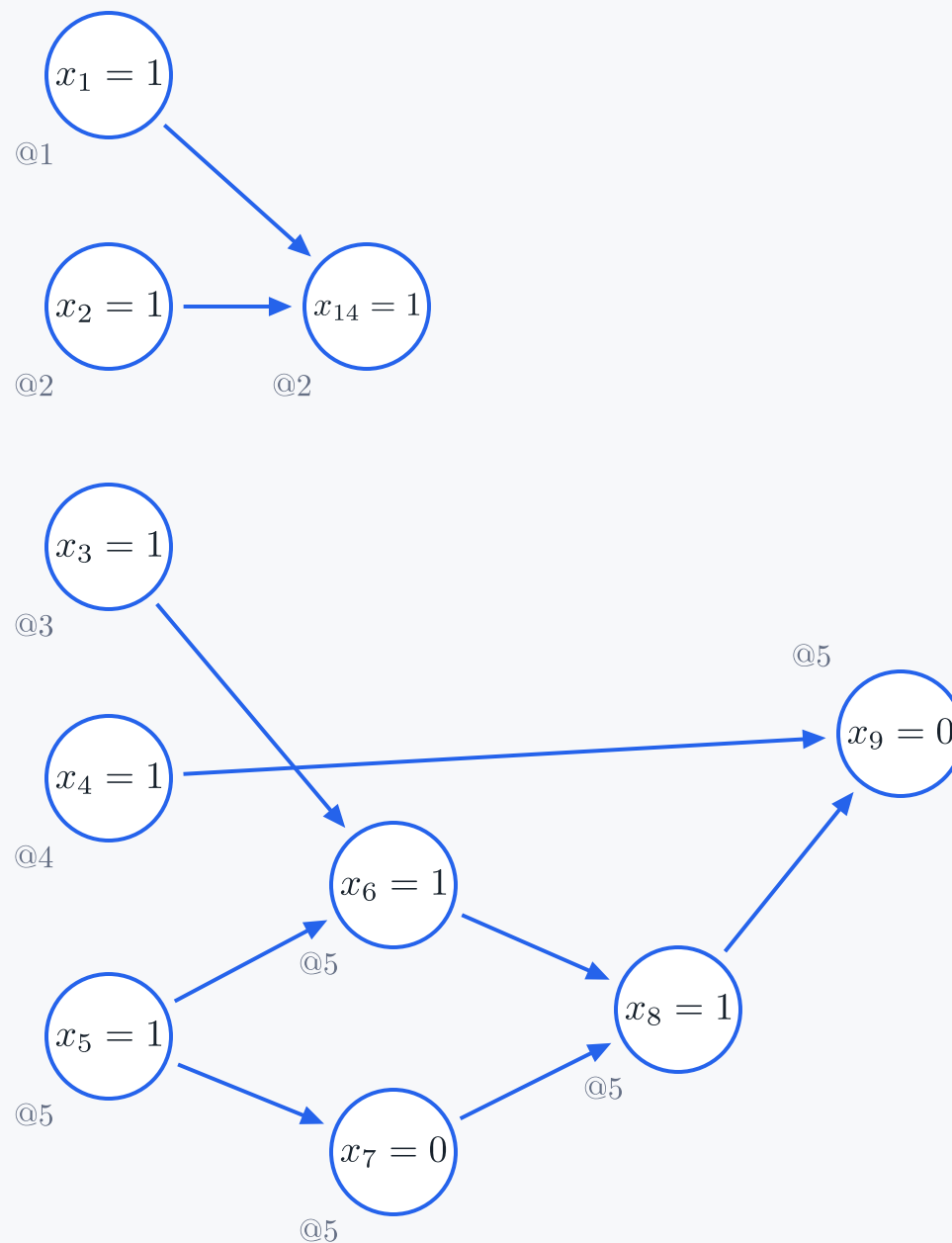
$$(\neg x_4 \vee \neg x_8 \vee \neg x_9)$$

$$(\neg x_1 \vee x_9 \vee \neg x_{10})$$

$$(x_9 \vee x_{11} \vee \neg x_{14})$$

$$(x_{10} \vee \neg x_{11} \vee x_{12})$$

$$(\neg x_2 \vee \neg x_{11} \vee \neg x_{12})$$



$$(x_1 \vee x_{13})$$

$$(\neg x_1 \vee \neg x_2 \vee x_{14})$$

$$(x_3 \vee x_{15})$$

$$(x_4 \vee x_{16})$$

$$(\neg x_5 \vee \neg x_3 \vee x_6)$$

$$(\neg x_5 \vee \neg x_7)$$

$$(\neg x_6 \vee x_7 \vee x_8)$$

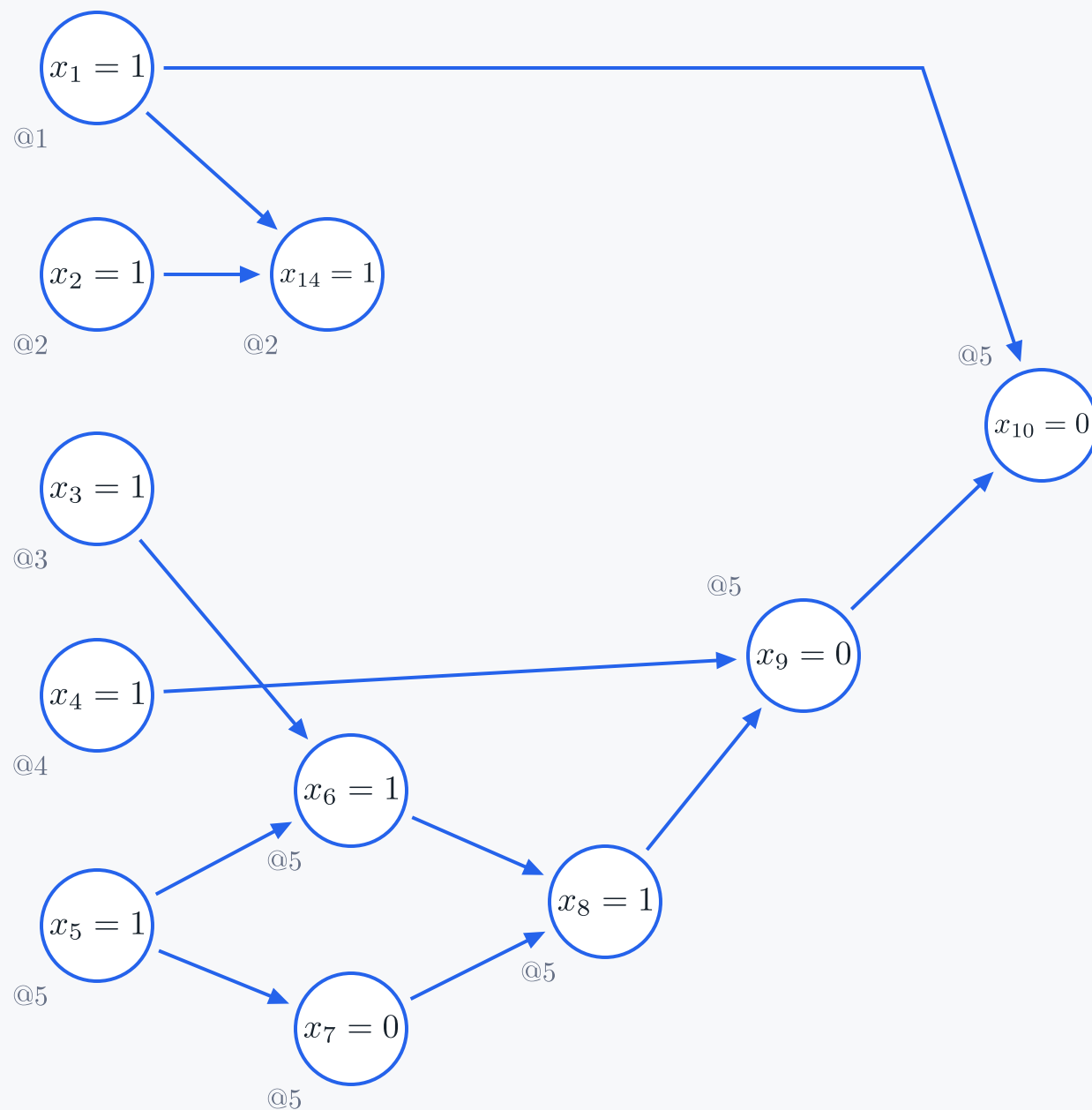
$$(\neg x_4 \vee \neg x_8 \vee \neg x_9)$$

$$(\neg x_1 \vee x_9 \vee \neg x_{10})$$

$$(x_9 \vee x_{11} \vee \neg x_{14})$$

$$(x_{10} \vee \neg x_{11} \vee x_{12})$$

$$(\neg x_2 \vee \neg x_{11} \vee \neg x_{12})$$



$$(x_1 \vee x_{13})$$

$$(\neg x_1 \vee \neg x_2 \vee x_{14})$$

$$(x_3 \vee x_{15})$$

$$(x_4 \vee x_{16})$$

$$(\neg x_5 \vee \neg x_3 \vee x_6)$$

$$(\neg x_5 \vee \neg x_7)$$

$$(\neg x_6 \vee x_7 \vee x_8)$$

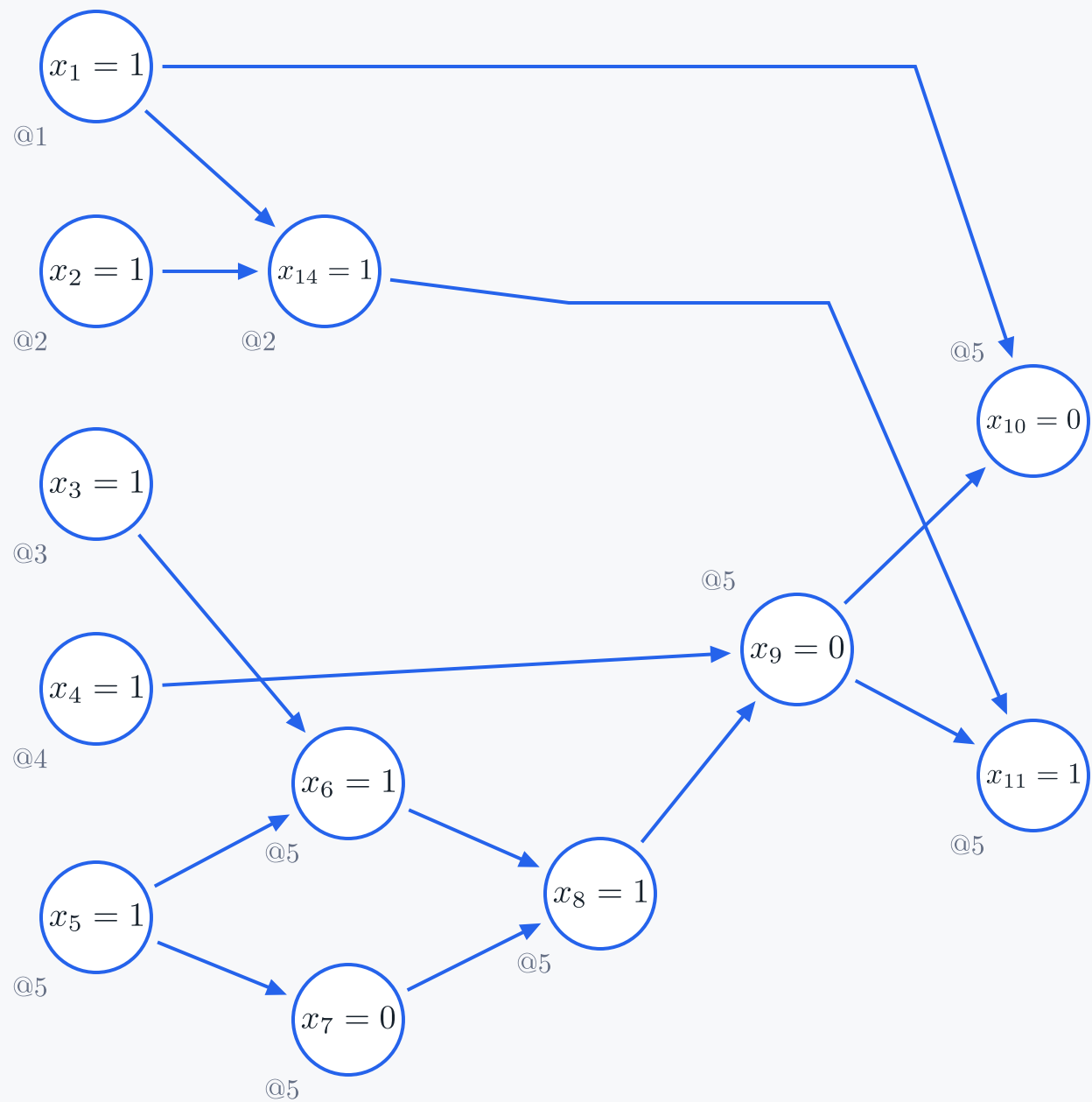
$$(\neg x_4 \vee \neg x_8 \vee \neg x_9)$$

$$(\neg x_1 \vee x_9 \vee \neg x_{10})$$

$$(x_9 \vee x_{11} \vee \neg x_{14})$$

$$(x_{10} \vee \neg x_{11} \vee x_{12})$$

$$(\neg x_2 \vee \neg x_{11} \vee \neg x_{12})$$



$$(x_1 \vee x_{13})$$

$$(\neg x_1 \vee \neg x_2 \vee x_{14})$$

$$(x_3 \vee x_{15})$$

$$(x_4 \vee x_{16})$$

$$(\neg x_5 \vee \neg x_3 \vee x_6)$$

$$(\neg x_5 \vee \neg x_7)$$

$$(\neg x_6 \vee x_7 \vee x_8)$$

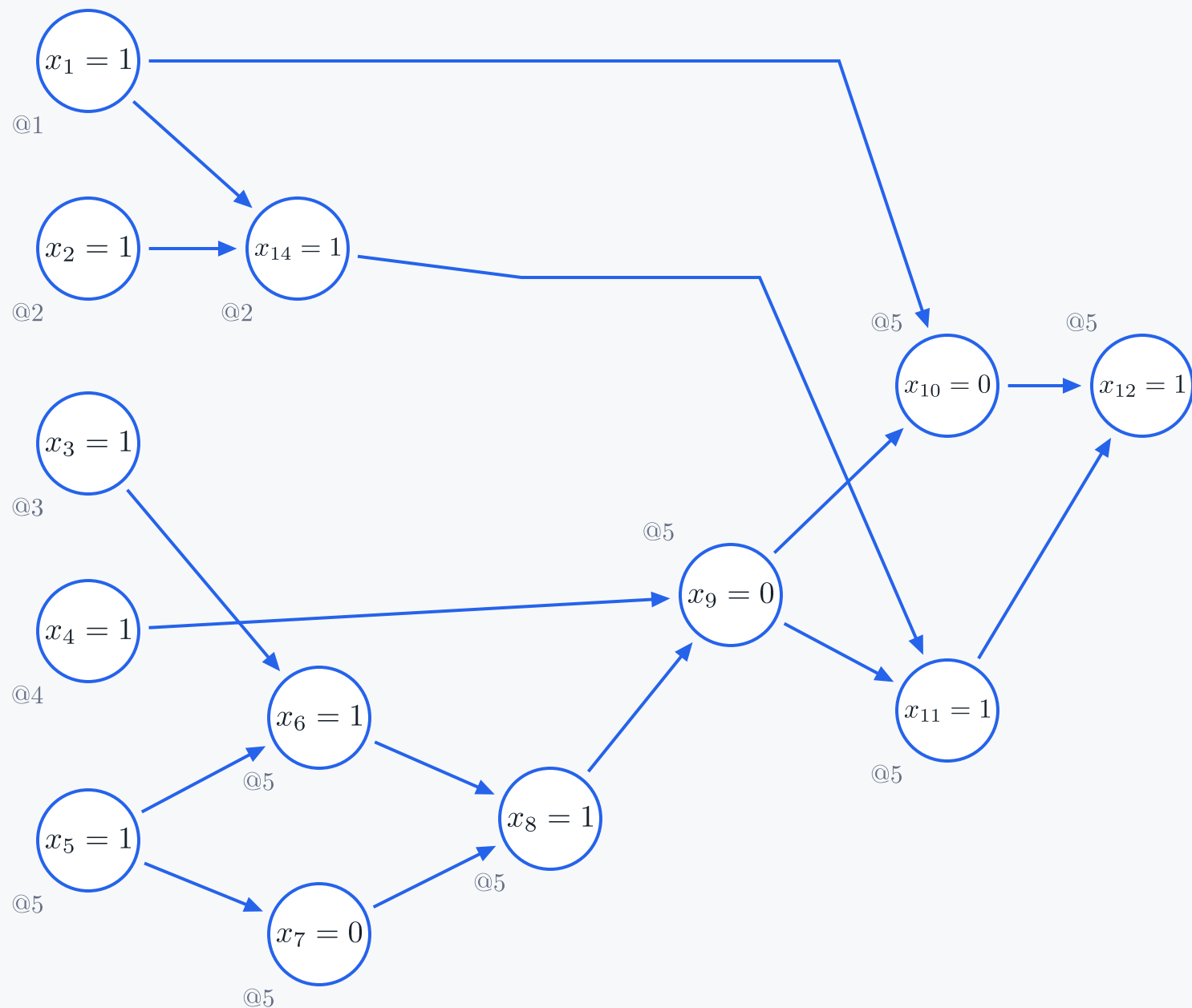
$$(\neg x_4 \vee \neg x_8 \vee \neg x_9)$$

$$(\neg x_1 \vee x_9 \vee \neg x_{10})$$

$$(x_9 \vee x_{11} \vee \neg x_{14})$$

$$(x_{10} \vee \neg x_{11} \vee x_{12})$$

$$(\neg x_2 \vee \neg x_{11} \vee \neg x_{12})$$



$$(x_1 \vee x_{13})$$

$$(\neg x_1 \vee \neg x_2 \vee x_{14})$$

$$(x_3 \vee x_{15})$$

$$(x_4 \vee x_{16})$$

$$(\neg x_5 \vee \neg x_3 \vee x_6)$$

$$(\neg x_5 \vee \neg x_7)$$

$$(\neg x_6 \vee x_7 \vee x_8)$$

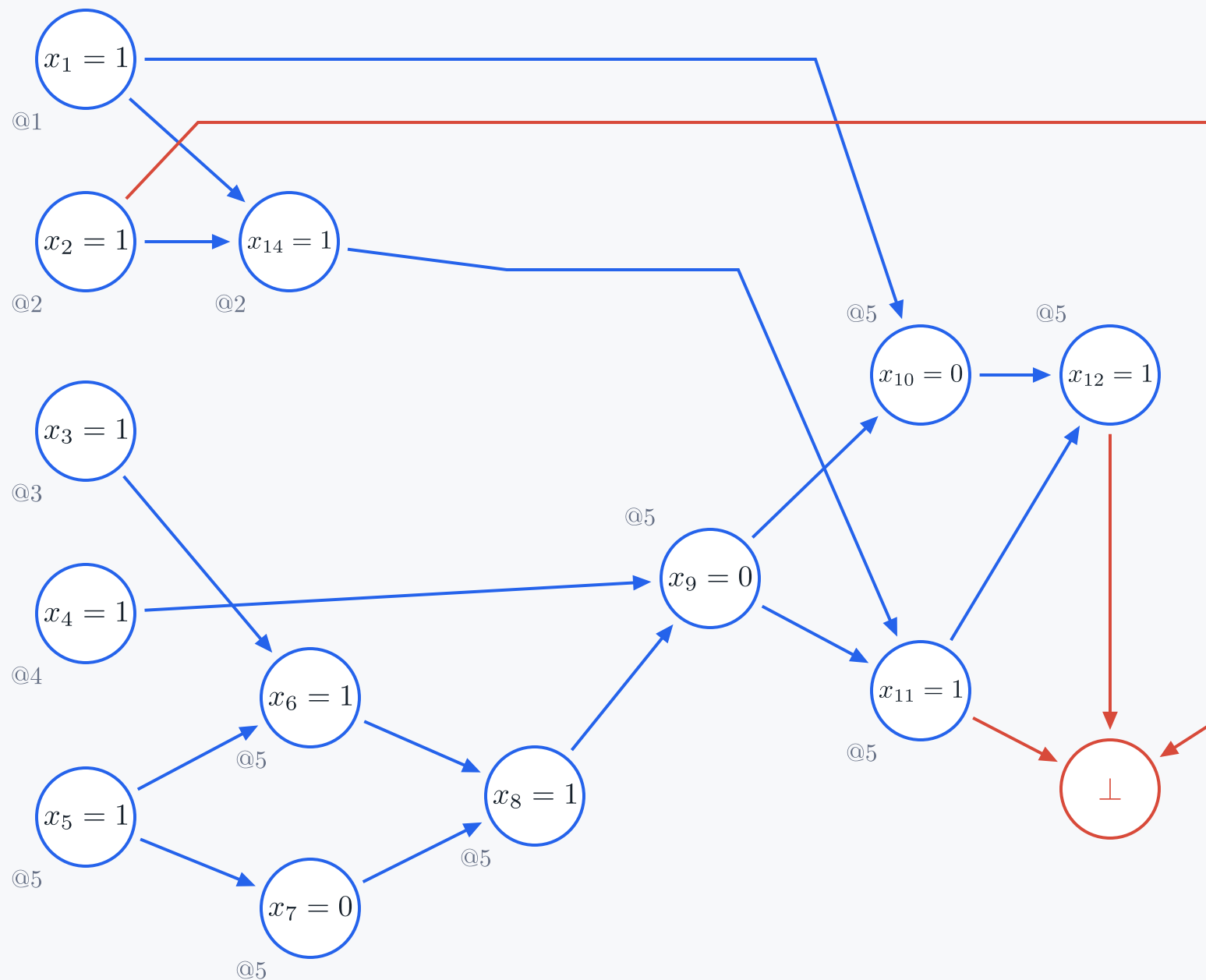
$$(\neg x_4 \vee \neg x_8 \vee \neg x_9)$$

$$(\neg x_1 \vee x_9 \vee \neg x_{10})$$

$$(x_9 \vee x_{11} \vee \neg x_{14})$$

$$(x_{10} \vee \neg x_{11} \vee x_{12})$$

$$(\neg x_2 \vee \neg x_{11} \vee \neg x_{12}) \Rightarrow \perp$$



$$(x_1 \vee x_{13})$$

$$(\neg x_1 \vee \neg x_2 \vee x_{14})$$

$$(x_3 \vee x_{15})$$

$$(x_4 \vee x_{16})$$

$$(\neg x_5 \vee \neg x_3 \vee x_6)$$

$$(\neg x_5 \vee \neg x_7)$$

$$(\neg x_6 \vee x_7 \vee x_8)$$

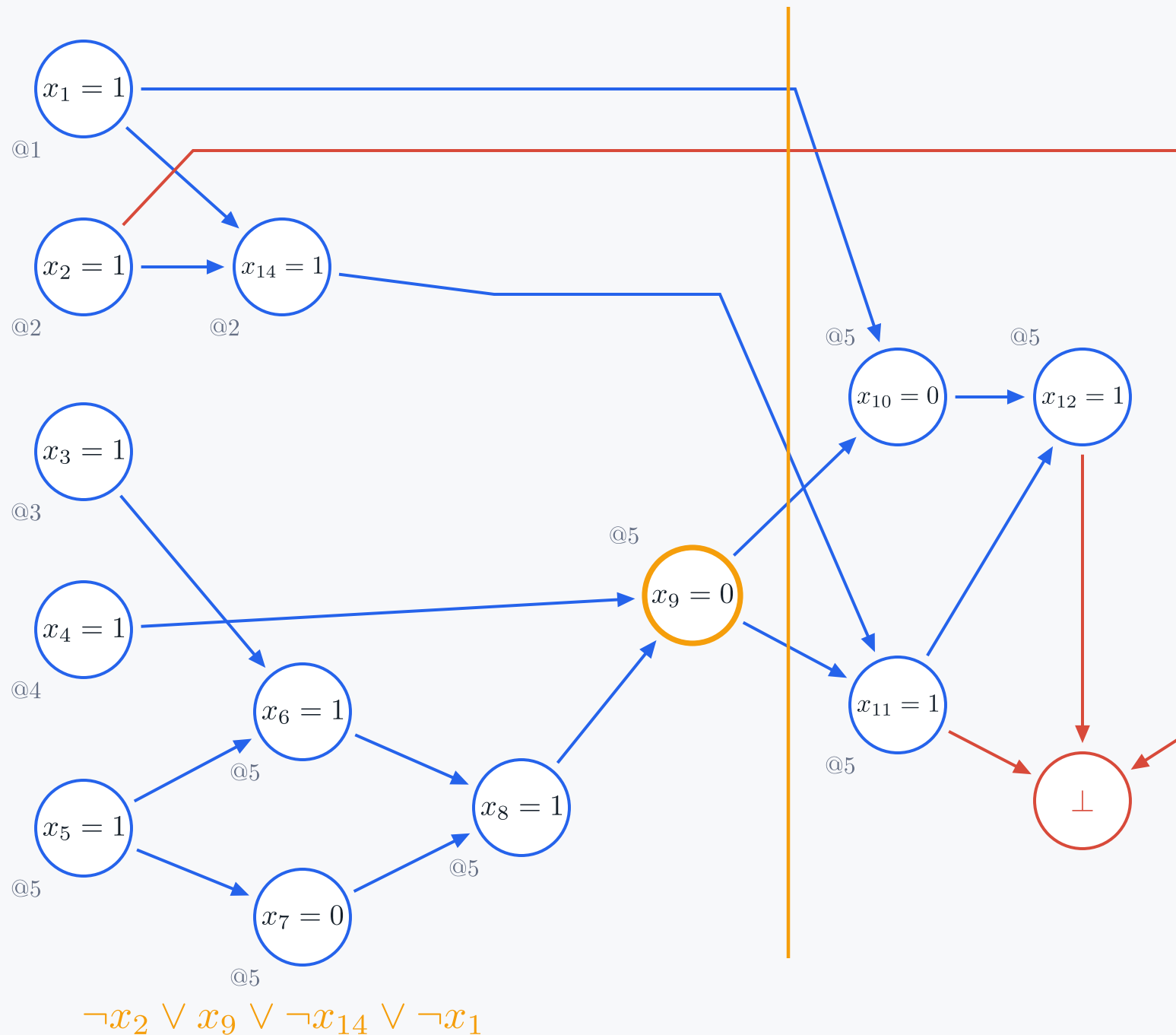
$$(\neg x_4 \vee \neg x_8 \vee \neg x_9)$$

$$(\neg x_1 \vee x_9 \vee \neg x_{10})$$

$$(x_9 \vee x_{11} \vee \neg x_{14})$$

$$(x_{10} \vee \neg x_{11} \vee x_{12})$$

$$(\neg x_2 \vee \neg x_{11} \vee \neg x_{12}) \Rightarrow \perp$$



$$(x_1 \vee x_{13})$$

$$(\neg x_1 \vee \neg x_2 \vee x_{14})$$

$$(x_3 \vee x_{15})$$

$$(x_4 \vee x_{16})$$

$$(\neg x_5 \vee \neg x_3 \vee x_6)$$

$$(\neg x_5 \vee \neg x_7)$$

$$(\neg x_6 \vee x_7 \vee x_8)$$

$$(\neg x_4 \vee \neg x_8 \vee \neg x_9)$$

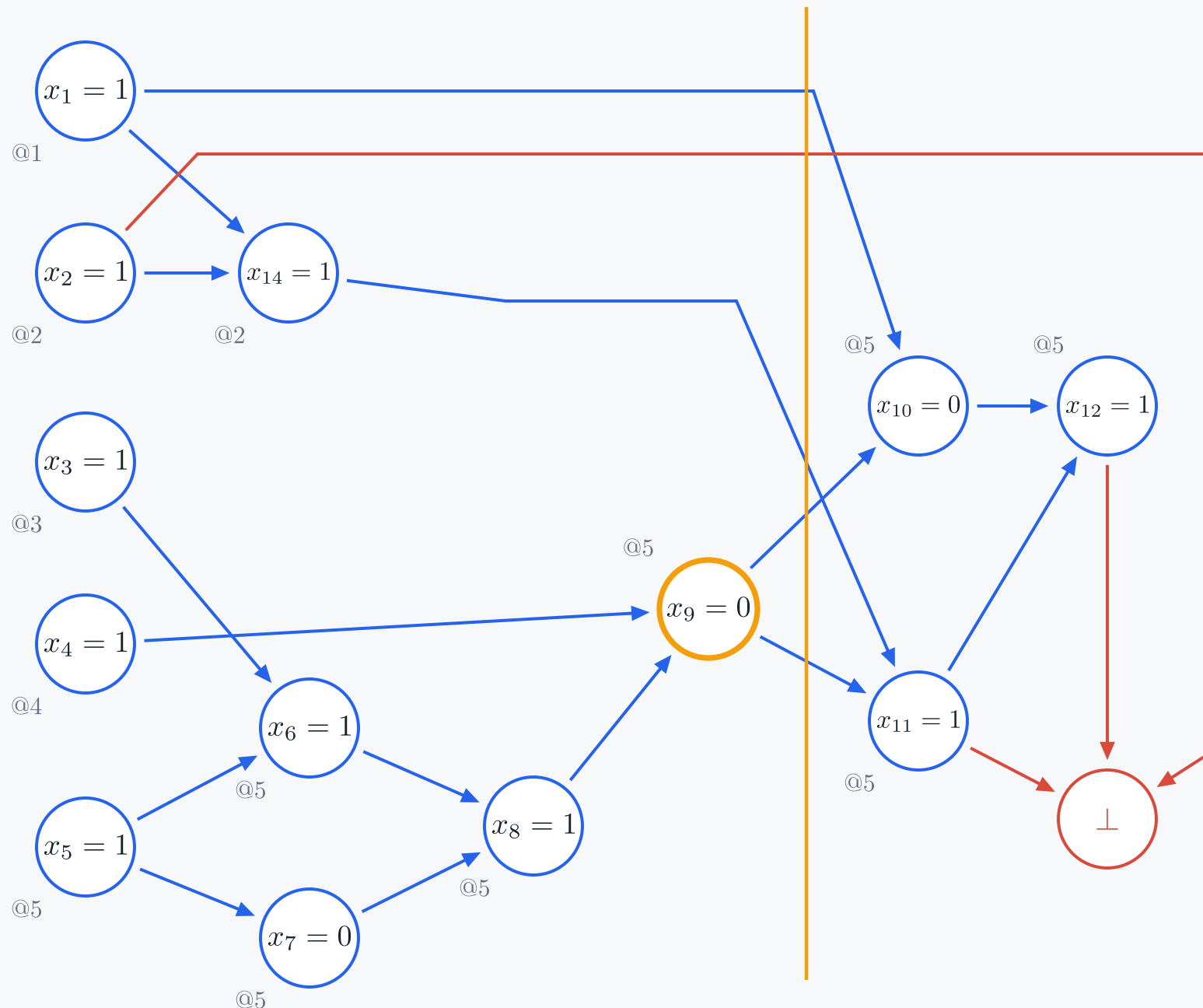
$$(\neg x_1 \vee x_9 \vee \neg x_{10})$$

$$(x_9 \vee x_{11} \vee \neg x_{14})$$

$$(x_{10} \vee \neg x_{11} \vee x_{12})$$

$$(\neg x_2 \vee \neg x_{11} \vee \neg x_{12}) \Rightarrow \perp$$

$$(\neg x_2 \vee x_9 \vee \neg x_{14} \vee \neg x_1)$$



$$\neg x_2 \vee x_9 \vee \neg x_{14} \vee \neg x_1$$

$$(x_1 \vee x_{13})$$

$$(\neg x_1 \vee \neg x_2 \vee x_{14})$$

$$(x_3 \vee x_{15})$$

$$(x_4 \vee x_{16})$$

$$(\neg x_5 \vee \neg x_3 \vee x_6)$$

$$(\neg x_5 \vee \neg x_7)$$

$$(\neg x_6 \vee x_7 \vee x_8)$$

$$(\neg x_4 \vee \neg x_8 \vee \neg x_9)$$

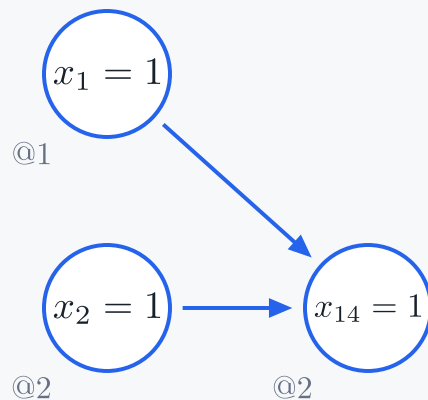
$$(\neg x_1 \vee x_9 \vee \neg x_{10})$$

$$(x_9 \vee x_{11} \vee \neg x_{14})$$

$$(x_{10} \vee \neg x_{11} \vee x_{12})$$

$$(\neg x_2 \vee \neg x_{11} \vee \neg x_{12})$$

$$(\neg x_2 \vee x_9 \vee \neg x_{14} \vee \neg x_1)$$



$$(x_1 \vee x_{13})$$

$$(\neg x_1 \vee \neg x_2 \vee x_{14})$$

$$(x_3 \vee x_{15})$$

$$(x_4 \vee x_{16})$$

$$(\neg x_5 \vee \neg x_3 \vee x_6)$$

$$(\neg x_5 \vee \neg x_7)$$

$$(\neg x_6 \vee x_7 \vee x_8)$$

$$(\neg x_4 \vee \neg x_8 \vee \neg x_9)$$

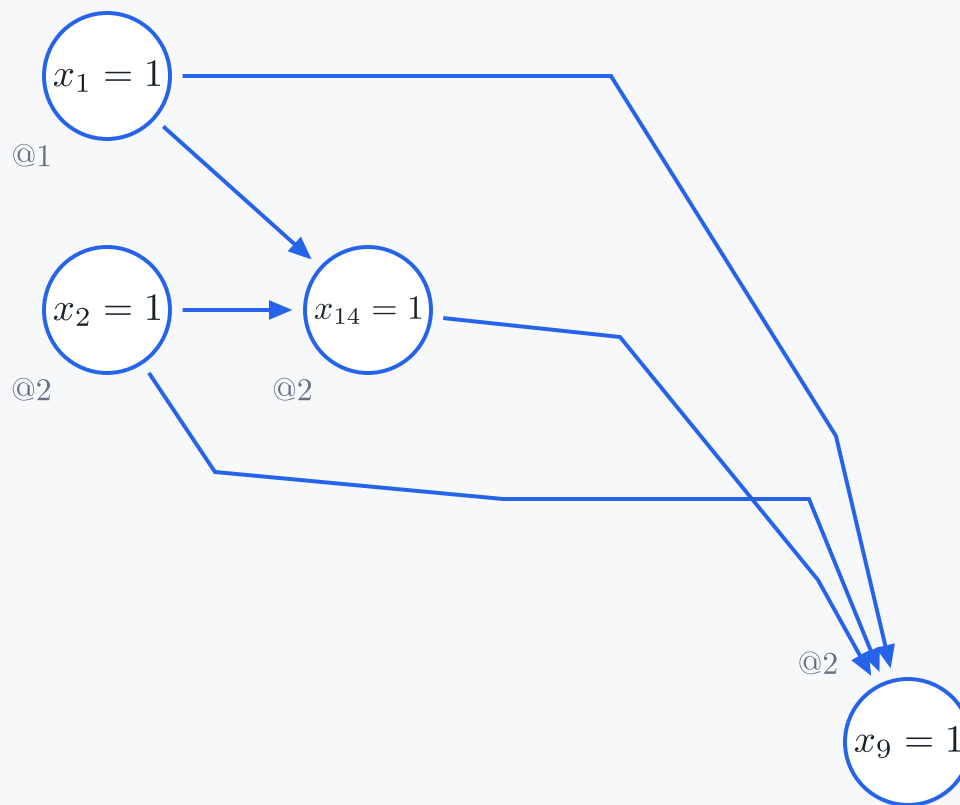
$$(\neg x_1 \vee x_9 \vee \neg x_{10})$$

$$(x_9 \vee x_{11} \vee \neg x_{14})$$

$$(x_{10} \vee \neg x_{11} \vee x_{12})$$

$$(\neg x_2 \vee \neg x_{11} \vee \neg x_{12})$$

$$(\neg x_2 \vee x_9 \vee \neg x_{14} \vee \neg x_1)$$



$$(x_1 \vee x_{13})$$

$$(\neg x_1 \vee \neg x_2 \vee x_{14})$$

$$(x_3 \vee x_{15})$$

$$(x_4 \vee x_{16})$$

$$(\neg x_5 \vee \neg x_3 \vee x_6)$$

$$(\neg x_5 \vee \neg x_7)$$

$$(\neg x_6 \vee x_7 \vee x_8)$$

$$(\neg x_4 \vee \neg x_8 \vee \neg x_9)$$

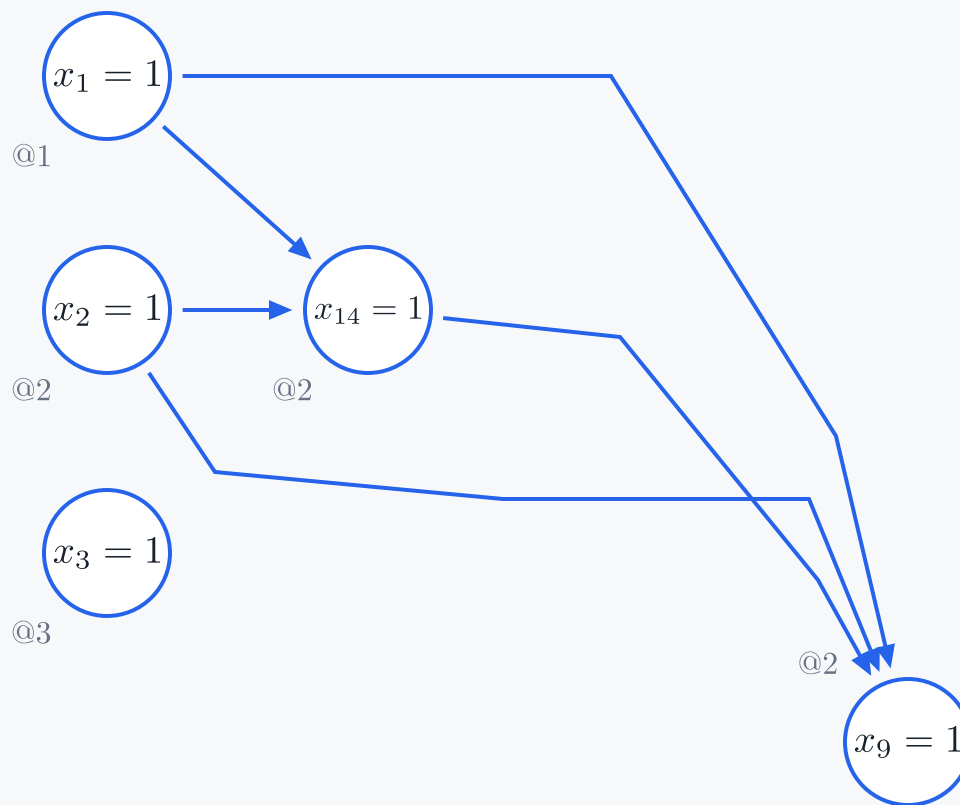
$$(\neg x_1 \vee x_9 \vee \neg x_{10})$$

$$(x_9 \vee x_{11} \vee \neg x_{14})$$

$$(x_{10} \vee \neg x_{11} \vee x_{12})$$

$$(\neg x_2 \vee \neg x_{11} \vee \neg x_{12})$$

$$(\neg x_2 \vee x_9 \vee \neg x_{14} \vee \neg x_1)$$



$$(x_1 \vee x_{13})$$

$$(\neg x_1 \vee \neg x_2 \vee x_{14})$$

$$(x_3 \vee x_{15})$$

$$(x_4 \vee x_{16})$$

$$(\neg x_5 \vee \neg x_3 \vee x_6)$$

$$(\neg x_5 \vee \neg x_7)$$

$$(\neg x_6 \vee x_7 \vee x_8)$$

$$(\neg x_4 \vee \neg x_8 \vee \neg x_9)$$

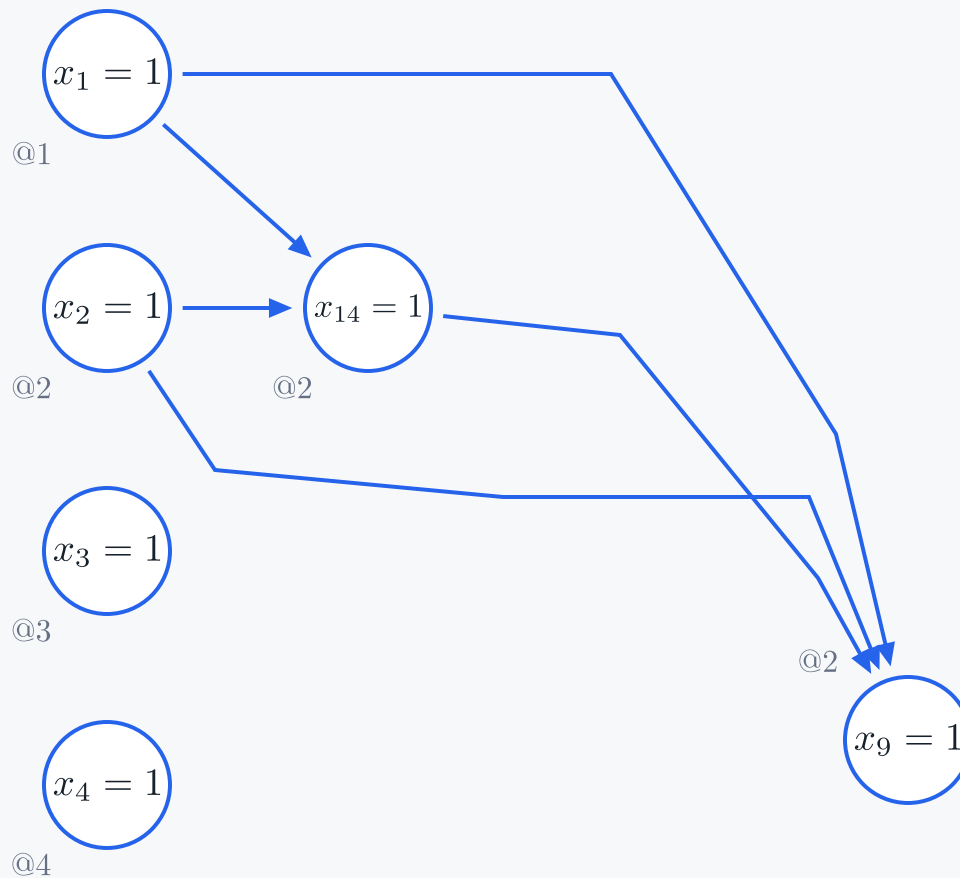
$$(\neg x_1 \vee x_9 \vee \neg x_{10})$$

$$(x_9 \vee x_{11} \vee \neg x_{14})$$

$$(x_{10} \vee \neg x_{11} \vee x_{12})$$

$$(\neg x_2 \vee \neg x_{11} \vee \neg x_{12})$$

$$(\neg x_2 \vee x_9 \vee \neg x_{14} \vee \neg x_1)$$



$$(x_1 \vee x_{13})$$

$$(\neg x_1 \vee \neg x_2 \vee x_{14})$$

$$(x_3 \vee x_{15})$$

$$(x_4 \vee x_{16})$$

$$(\neg x_5 \vee \neg x_3 \vee x_6)$$

$$(\neg x_5 \vee \neg x_7)$$

$$(\neg x_6 \vee x_7 \vee x_8)$$

$$(\neg x_4 \vee \neg x_8 \vee \neg x_9)$$

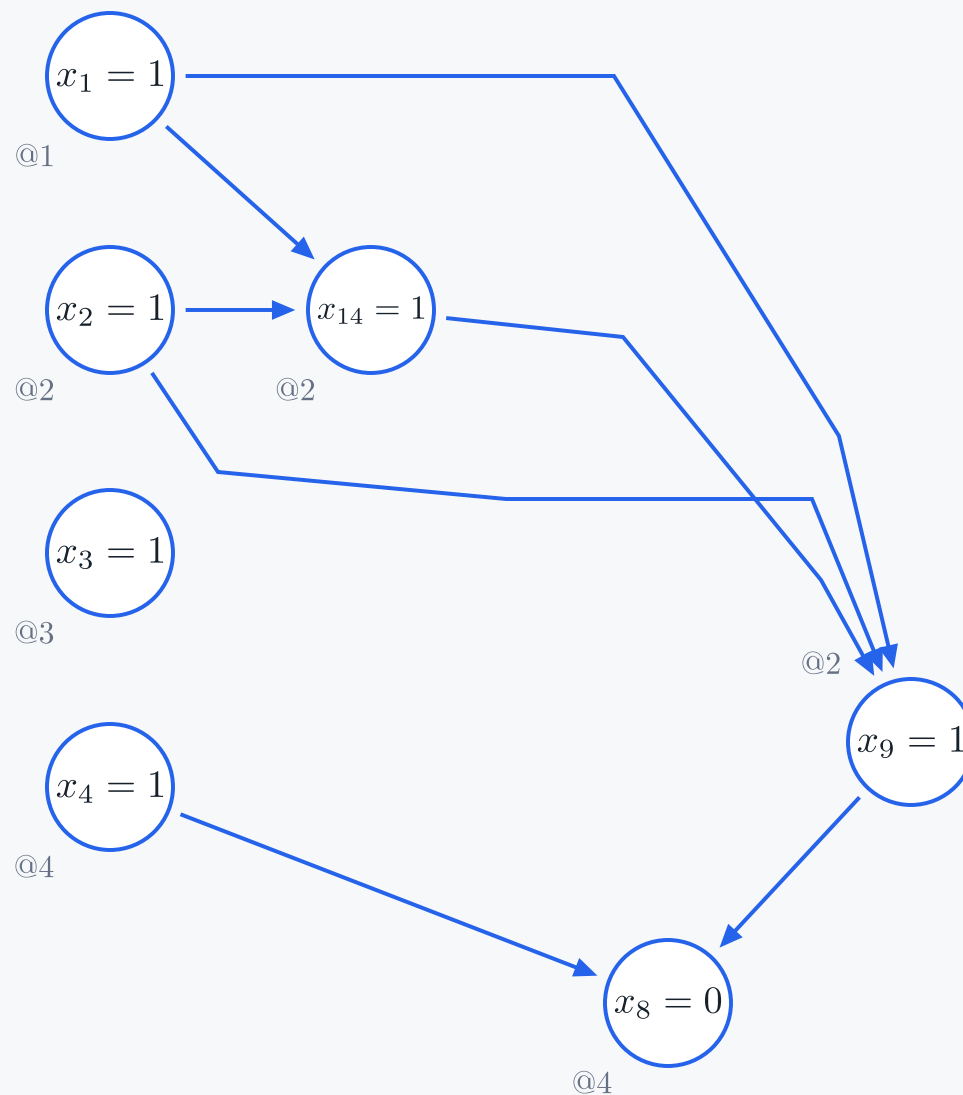
$$(\neg x_1 \vee x_9 \vee \neg x_{10})$$

$$(x_9 \vee x_{11} \vee \neg x_{14})$$

$$(x_{10} \vee \neg x_{11} \vee x_{12})$$

$$(\neg x_2 \vee \neg x_{11} \vee \neg x_{12})$$

$$(\neg x_2 \vee x_9 \vee \neg x_{14} \vee \neg x_1)$$



$$(x_1 \vee x_{13})$$

$$(\neg x_1 \vee \neg x_2 \vee x_{14})$$

$$(x_3 \vee x_{15})$$

$$(x_4 \vee x_{16})$$

$$(\neg x_5 \vee \neg x_3 \vee x_6)$$

$$(\neg x_5 \vee \neg x_7)$$

$$(\neg x_6 \vee x_7 \vee x_8)$$

$$(\neg x_4 \vee \neg x_8 \vee \neg x_9)$$

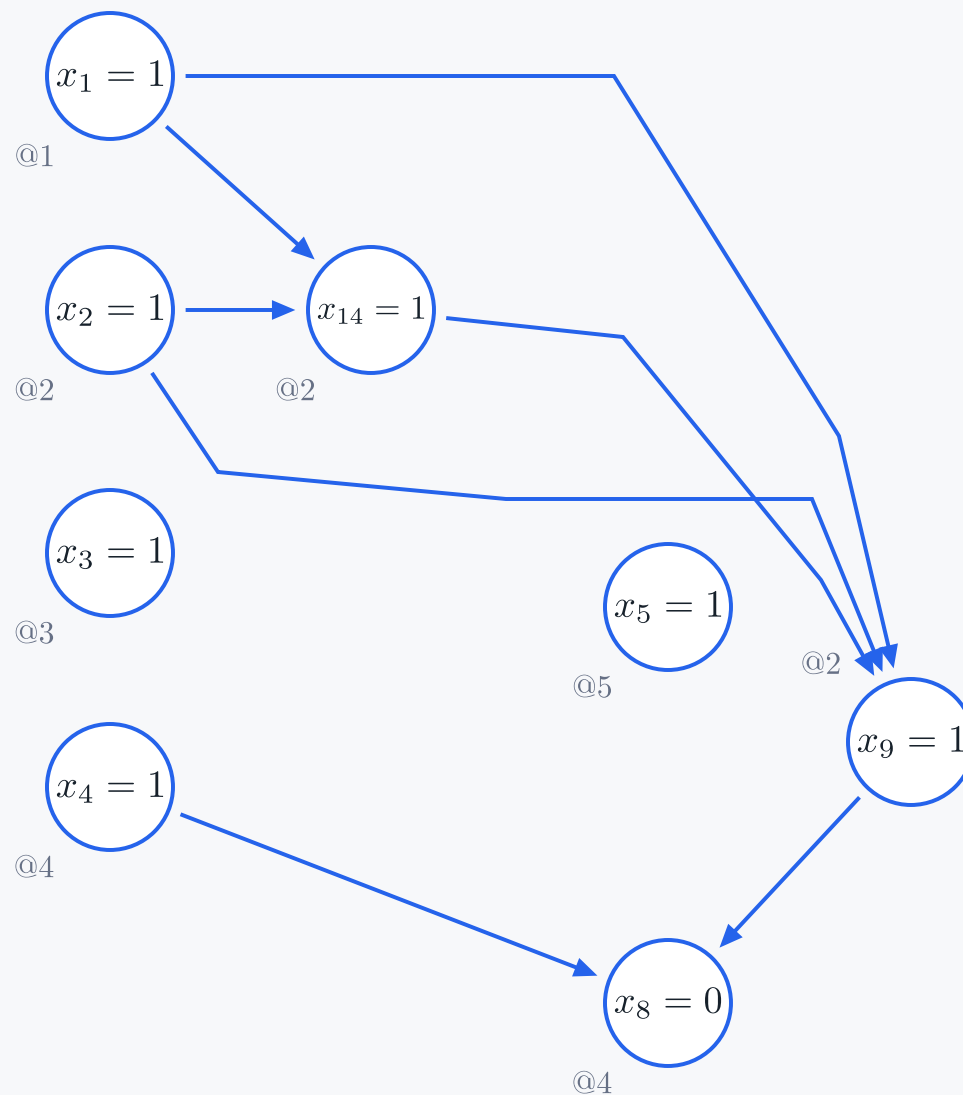
$$(\neg x_1 \vee x_9 \vee \neg x_{10})$$

$$(x_9 \vee x_{11} \vee \neg x_{14})$$

$$(x_{10} \vee \neg x_{11} \vee x_{12})$$

$$(\neg x_2 \vee \neg x_{11} \vee \neg x_{12})$$

$$(\neg x_2 \vee x_9 \vee \neg x_{14} \vee \neg x_1)$$



$$(x_1 \vee x_{13})$$

$$(\neg x_1 \vee \neg x_2 \vee x_{14})$$

$$(x_3 \vee x_{15})$$

$$(x_4 \vee x_{16})$$

$$(\neg x_5 \vee \neg x_3 \vee x_6)$$

$$(\neg x_5 \vee \neg x_7)$$

$$(\neg x_6 \vee x_7 \vee x_8)$$

$$(\neg x_4 \vee \neg x_8 \vee \neg x_9)$$

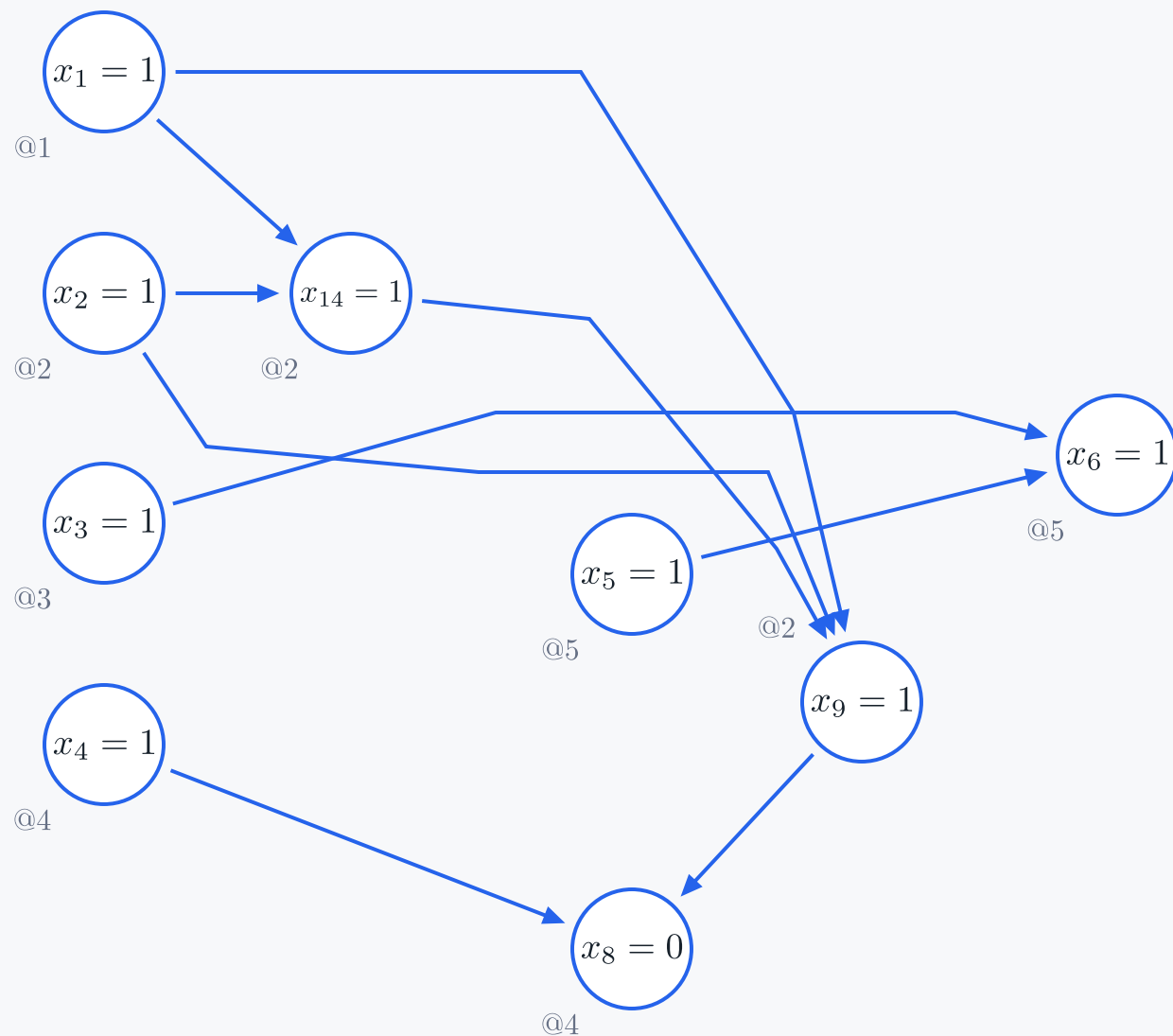
$$(\neg x_1 \vee x_9 \vee \neg x_{10})$$

$$(x_9 \vee x_{11} \vee \neg x_{14})$$

$$(x_{10} \vee \neg x_{11} \vee x_{12})$$

$$(\neg x_2 \vee \neg x_{11} \vee \neg x_{12})$$

$$(\neg x_2 \vee x_9 \vee \neg x_{14} \vee \neg x_1)$$



$$(x_1 \vee x_{13})$$

$$(\neg x_1 \vee \neg x_2 \vee x_{14})$$

$$(x_3 \vee x_{15})$$

$$(x_4 \vee x_{16})$$

$$(\neg x_5 \vee \neg x_3 \vee x_6)$$

$$(\neg x_5 \vee \neg x_7)$$

$$(\neg x_6 \vee x_7 \vee x_8)$$

$$(\neg x_4 \vee \neg x_8 \vee \neg x_9)$$

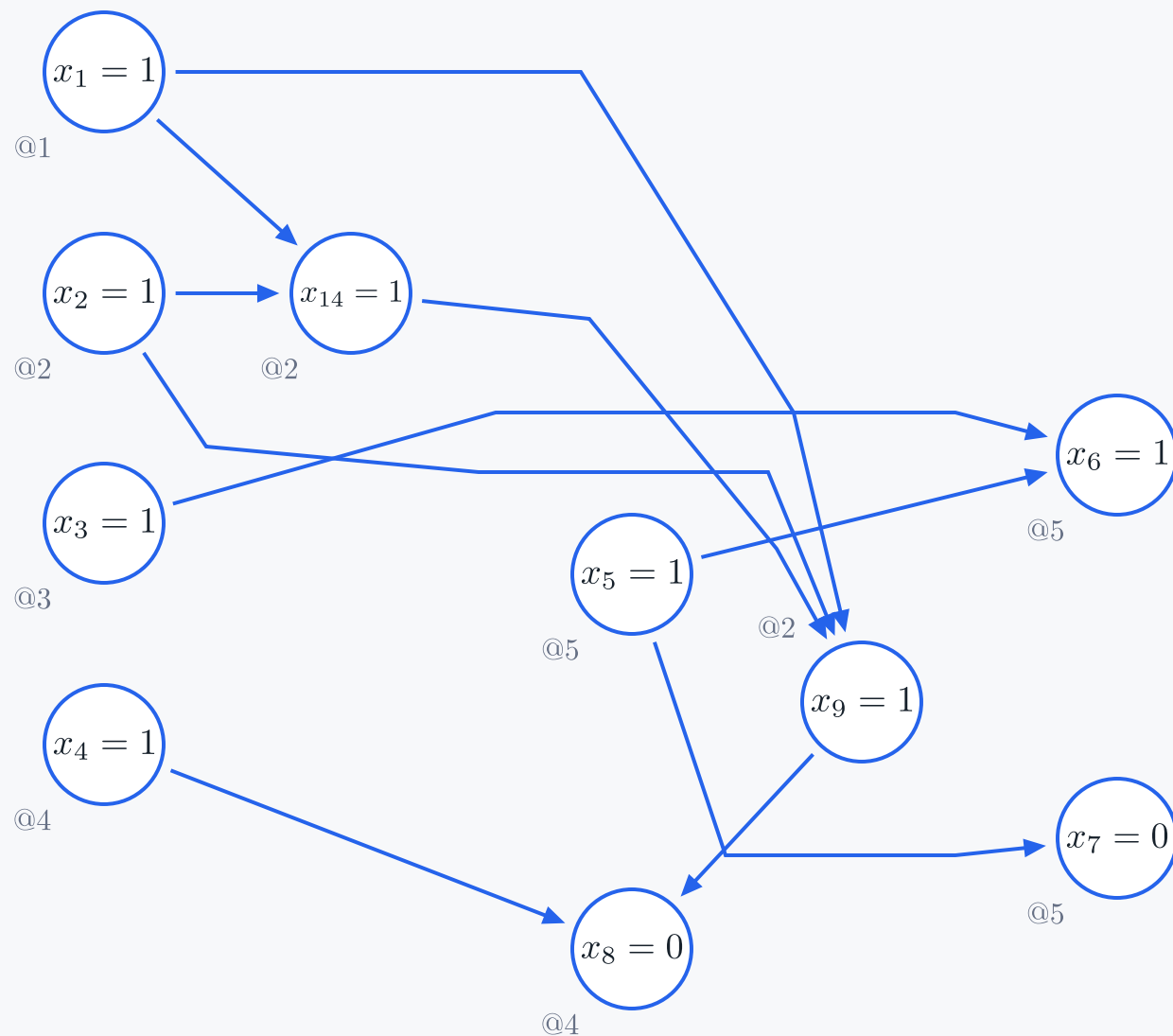
$$(\neg x_1 \vee x_9 \vee \neg x_{10})$$

$$(x_9 \vee x_{11} \vee \neg x_{14})$$

$$(x_{10} \vee \neg x_{11} \vee x_{12})$$

$$(\neg x_2 \vee \neg x_{11} \vee \neg x_{12})$$

$$(\neg x_2 \vee x_9 \vee \neg x_{14} \vee \neg x_1)$$



$$(x_1 \vee x_{13})$$

$$(\neg x_1 \vee \neg x_2 \vee x_{14})$$

$$(x_3 \vee x_{15})$$

$$(x_4 \vee x_{16})$$

$$(\neg x_5 \vee \neg x_3 \vee x_6)$$

$$(\neg x_5 \vee \neg x_7)$$

$$(\neg x_6 \vee x_7 \vee x_8) \Rightarrow \perp$$

$$(\neg x_4 \vee \neg x_8 \vee \neg x_9)$$

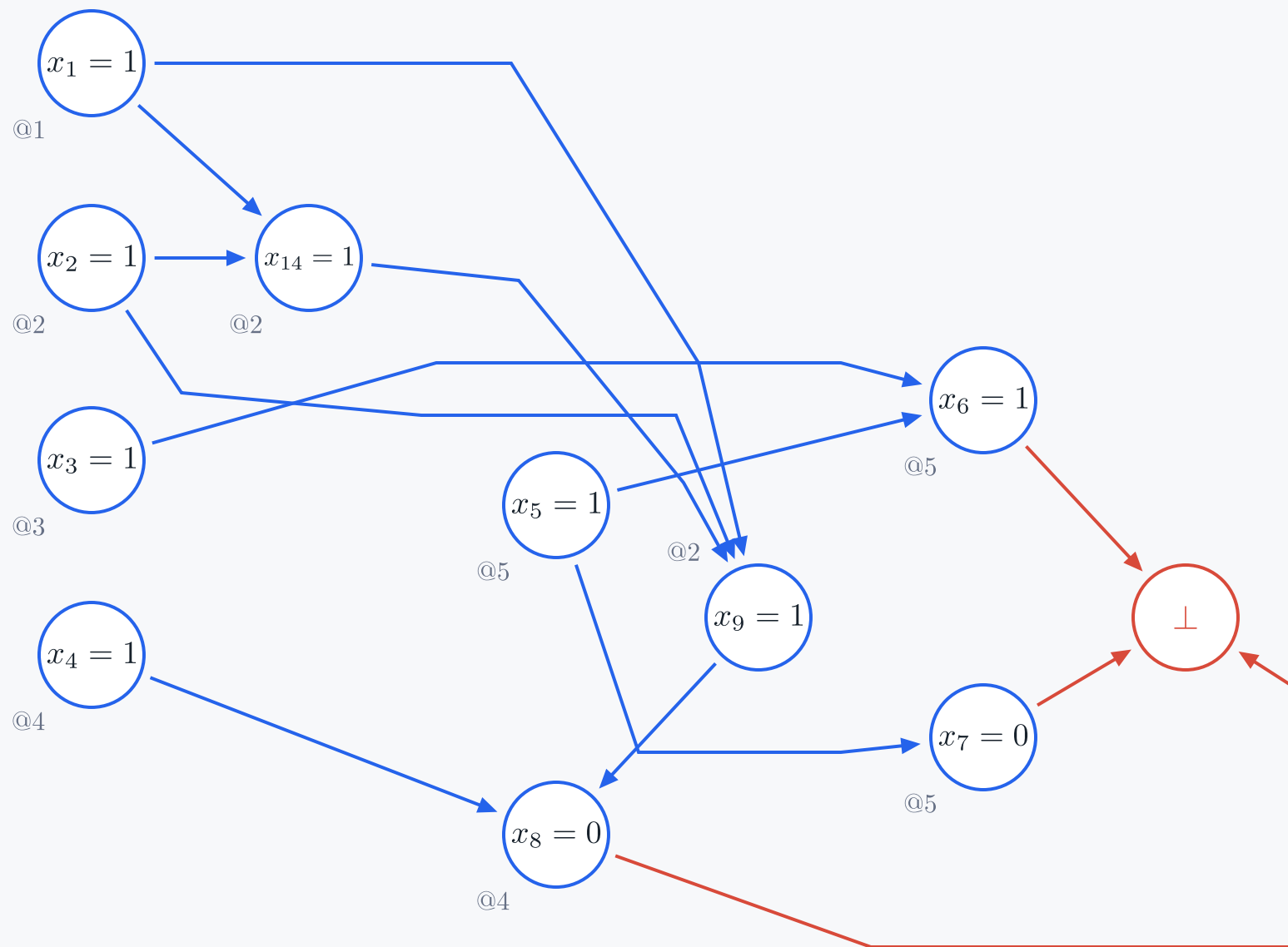
$$(\neg x_1 \vee x_9 \vee \neg x_{10})$$

$$(x_9 \vee x_{11} \vee \neg x_{14})$$

$$(x_{10} \vee \neg x_{11} \vee x_{12})$$

$$(\neg x_2 \vee \neg x_{11} \vee \neg x_{12})$$

$$(\neg x_2 \vee x_9 \vee \neg x_{14} \vee \neg x_1)$$



$$(x_1 \vee x_{13})$$

$$(\neg x_1 \vee \neg x_2 \vee x_{14})$$

$$(x_3 \vee x_{15})$$

$$(x_4 \vee x_{16})$$

$$(\neg x_5 \vee \neg x_3 \vee x_6)$$

$$(\neg x_5 \vee \neg x_7)$$

$$(\neg x_6 \vee x_7 \vee x_8) \Rightarrow \perp$$

$$(\neg x_4 \vee \neg x_8 \vee \neg x_9)$$

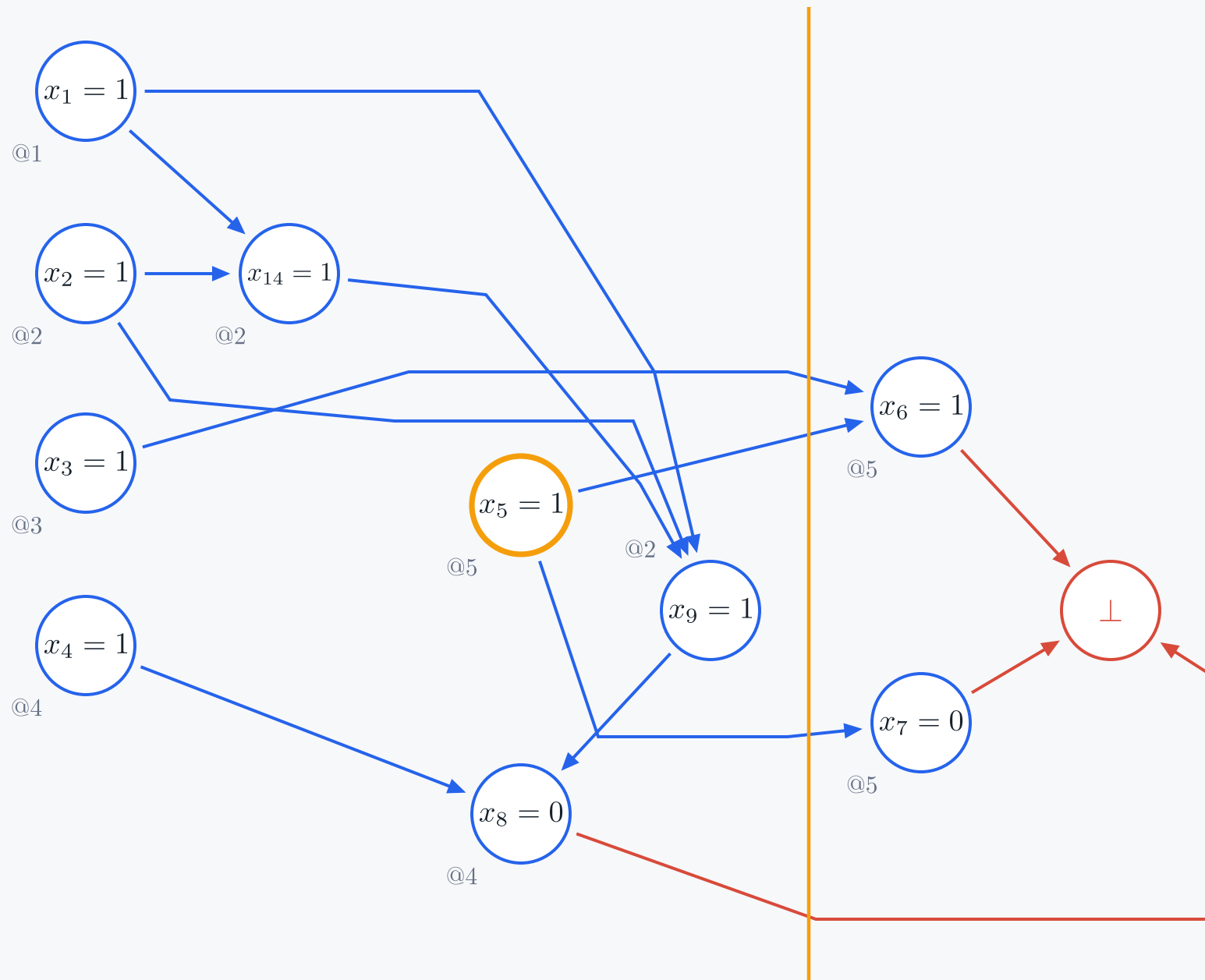
$$(\neg x_1 \vee x_9 \vee \neg x_{10})$$

$$(x_9 \vee x_{11} \vee \neg x_{14})$$

$$(x_{10} \vee \neg x_{11} \vee x_{12})$$

$$(\neg x_2 \vee \neg x_{11} \vee \neg x_{12})$$

$$(\neg x_2 \vee x_9 \vee \neg x_{14} \vee \neg x_1)$$



$$x_8 \vee \neg x_5 \vee \neg x_3$$

$$(x_1 \vee x_{13})$$

$$(\neg x_1 \vee \neg x_2 \vee x_{14})$$

$$(x_3 \vee x_{15})$$

$$(x_4 \vee x_{16})$$

$$(\neg x_5 \vee \neg x_3 \vee x_6)$$

$$(\neg x_5 \vee \neg x_7)$$

$$(\neg x_6 \vee x_7 \vee x_8) \Rightarrow \perp$$

$$(\neg x_4 \vee \neg x_8 \vee \neg x_9)$$

$$(\neg x_1 \vee x_9 \vee \neg x_{10})$$

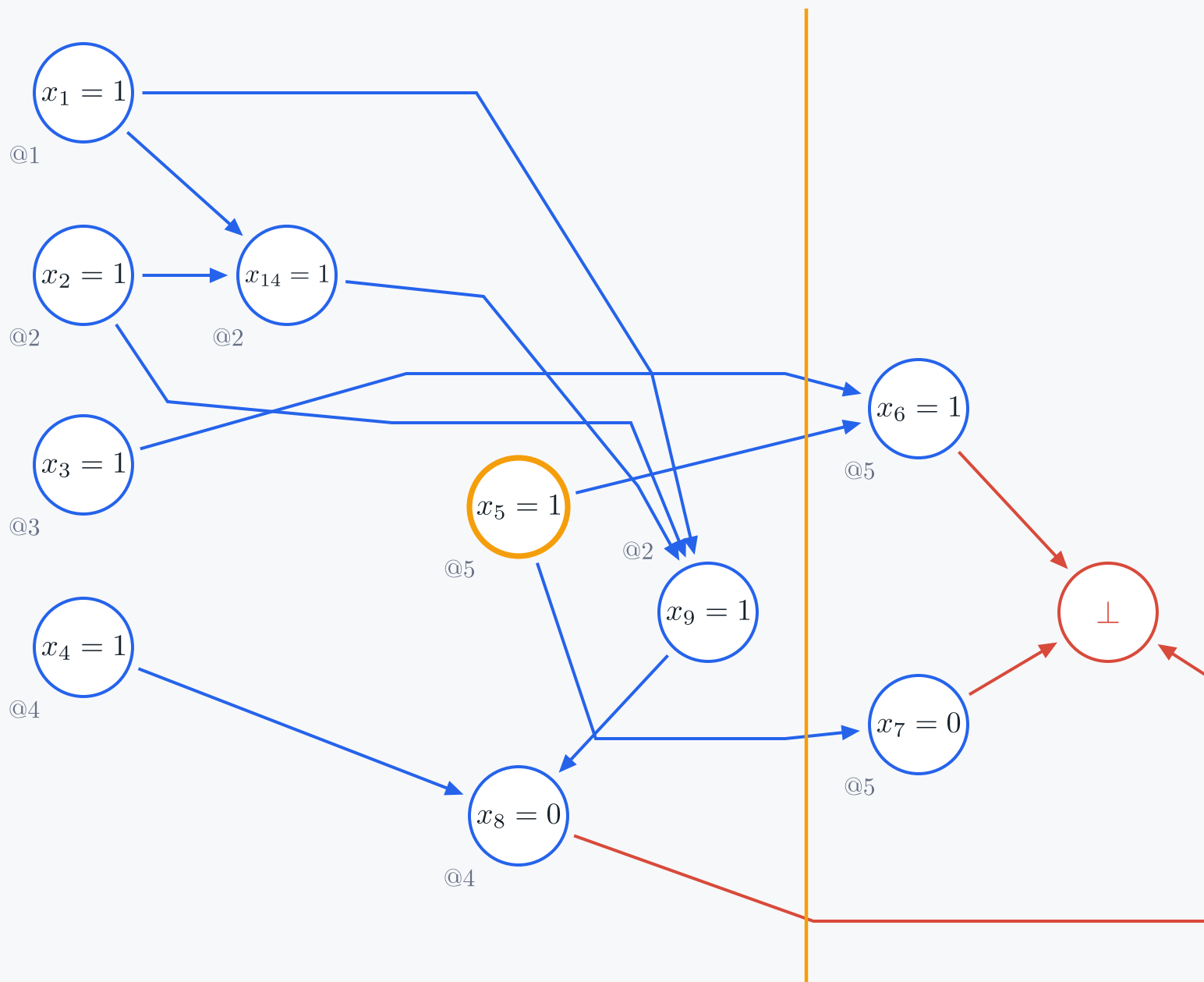
$$(x_9 \vee x_{11} \vee \neg x_{14})$$

$$(x_{10} \vee \neg x_{11} \vee x_{12})$$

$$(\neg x_2 \vee \neg x_{11} \vee \neg x_{12})$$

$$(\neg x_2 \vee x_9 \vee \neg x_{14} \vee \neg x_1)$$

$$(x_8 \vee \neg x_5 \vee \neg x_3)$$



$$x_8 \vee \neg x_5 \vee \neg x_3$$

$$(x_1 \vee x_{13})$$

$$(\neg x_1 \vee \neg x_2 \vee x_{14})$$

$$(x_3 \vee x_{15})$$

$$(x_4 \vee x_{16})$$

$$(\neg x_5 \vee \neg x_3 \vee x_6)$$

$$(\neg x_5 \vee \neg x_7)$$

$$(\neg x_6 \vee x_7 \vee x_8)$$

$$(\neg x_4 \vee \neg x_8 \vee \neg x_9)$$

$$(\neg x_1 \vee x_9 \vee \neg x_{10})$$

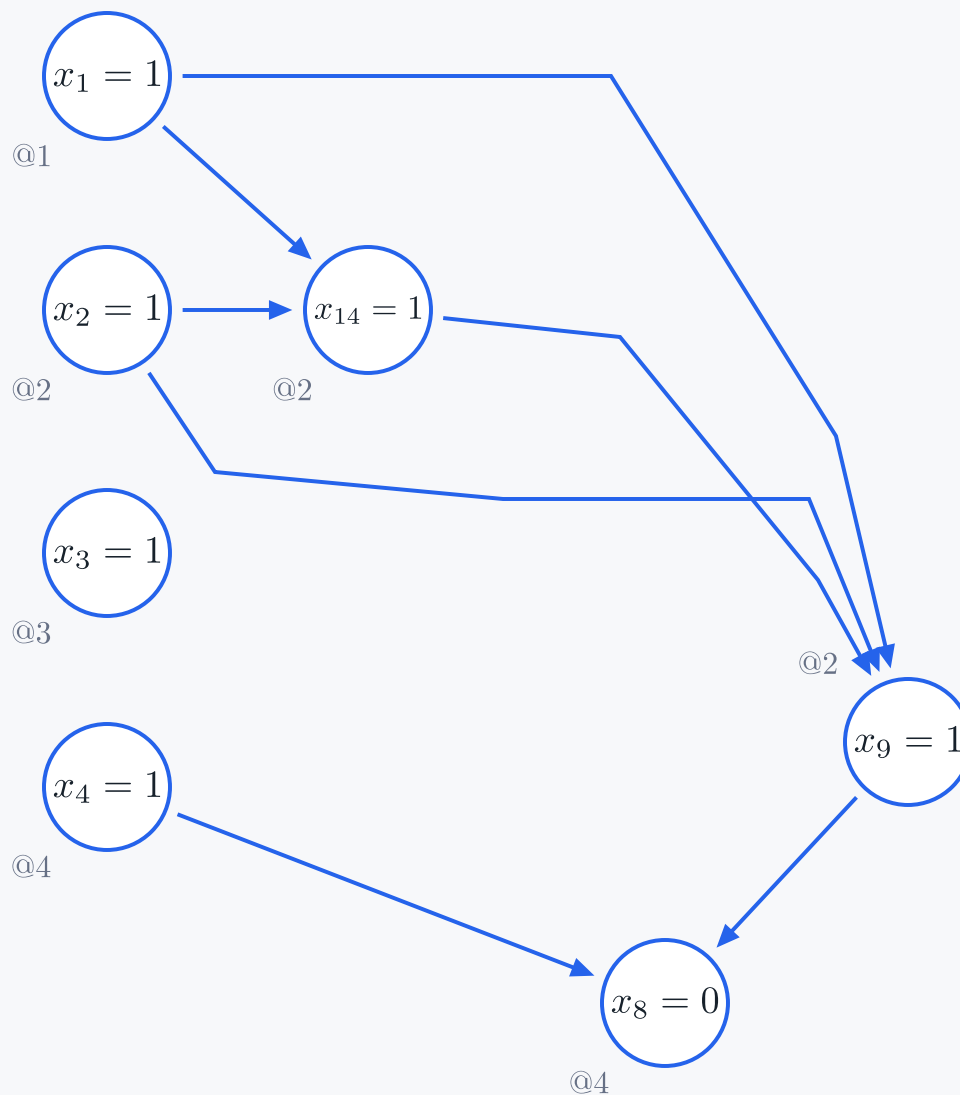
$$(x_9 \vee x_{11} \vee \neg x_{14})$$

$$(x_{10} \vee \neg x_{11} \vee x_{12})$$

$$(\neg x_2 \vee \neg x_{11} \vee \neg x_{12})$$

$$(\neg x_2 \vee x_9 \vee \neg x_{14} \vee \neg x_1)$$

$$(x_8 \vee \neg x_5 \vee \neg x_3)$$



$$(x_1 \vee x_{13})$$

$$(\neg x_1 \vee \neg x_2 \vee x_{14})$$

$$(x_3 \vee x_{15})$$

$$(x_4 \vee x_{16})$$

$$(\neg x_5 \vee \neg x_3 \vee x_6)$$

$$(\neg x_5 \vee \neg x_7)$$

$$(\neg x_6 \vee x_7 \vee x_8)$$

$$(\neg x_4 \vee \neg x_8 \vee \neg x_9)$$

$$(\neg x_1 \vee x_9 \vee \neg x_{10})$$

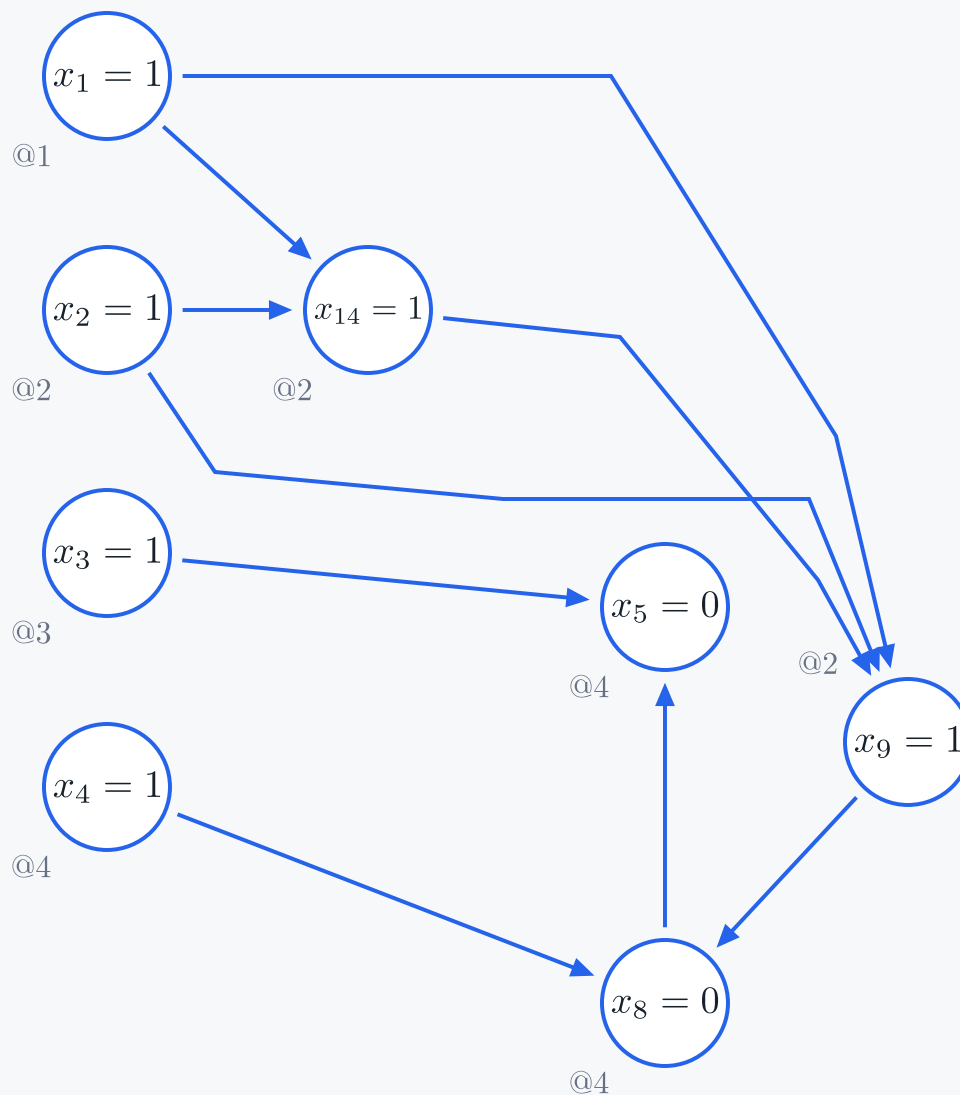
$$(x_9 \vee x_{11} \vee \neg x_{14})$$

$$(x_{10} \vee \neg x_{11} \vee x_{12})$$

$$(\neg x_2 \vee \neg x_{11} \vee \neg x_{12})$$

$$(\neg x_2 \vee x_9 \vee \neg x_{14} \vee \neg x_1)$$

$$(x_8 \vee \neg x_5 \vee \neg x_3)$$



$$(x_1 \vee x_{13})$$

$$(\neg x_1 \vee \neg x_2 \vee x_{14})$$

$$(x_3 \vee x_{15})$$

$$(x_4 \vee x_{16})$$

$$(\neg x_5 \vee \neg x_3 \vee x_6)$$

$$(\neg x_5 \vee \neg x_7)$$

$$(\neg x_6 \vee x_7 \vee x_8)$$

$$(\neg x_4 \vee \neg x_8 \vee \neg x_9)$$

$$(\neg x_1 \vee x_9 \vee \neg x_{10})$$

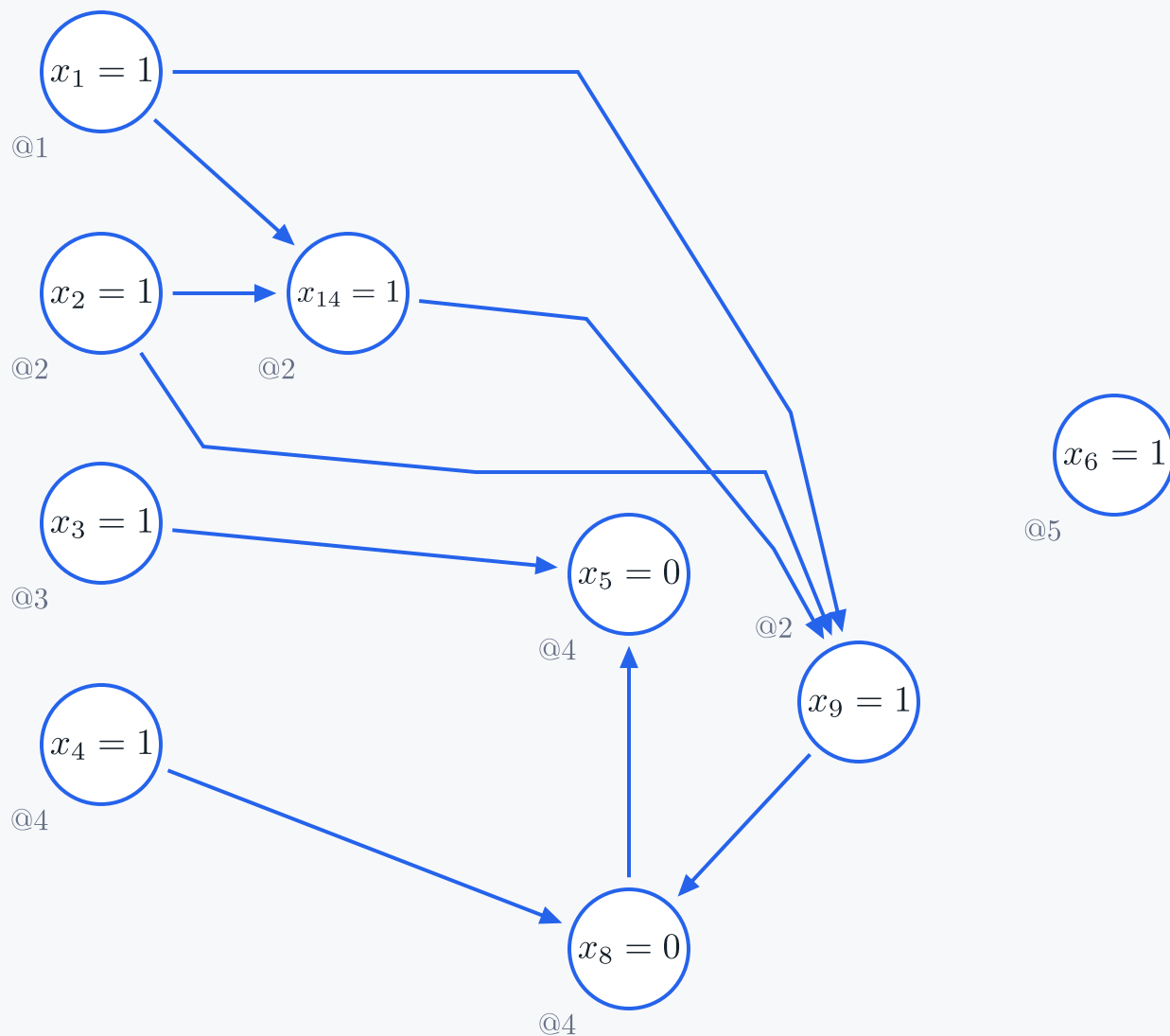
$$(x_9 \vee x_{11} \vee \neg x_{14})$$

$$(x_{10} \vee \neg x_{11} \vee x_{12})$$

$$(\neg x_2 \vee \neg x_{11} \vee \neg x_{12})$$

$$(\neg x_2 \vee x_9 \vee \neg x_{14} \vee \neg x_1)$$

$$(x_8 \vee \neg x_5 \vee \neg x_3)$$



$$(x_1 \vee x_{13})$$

$$(\neg x_1 \vee \neg x_2 \vee x_{14})$$

$$(x_3 \vee x_{15})$$

$$(x_4 \vee x_{16})$$

$$(\neg x_5 \vee \neg x_3 \vee x_6)$$

$$(\neg x_5 \vee \neg x_7)$$

$$(\neg x_6 \vee x_7 \vee x_8)$$

$$(\neg x_4 \vee \neg x_8 \vee \neg x_9)$$

$$(\neg x_1 \vee x_9 \vee \neg x_{10})$$

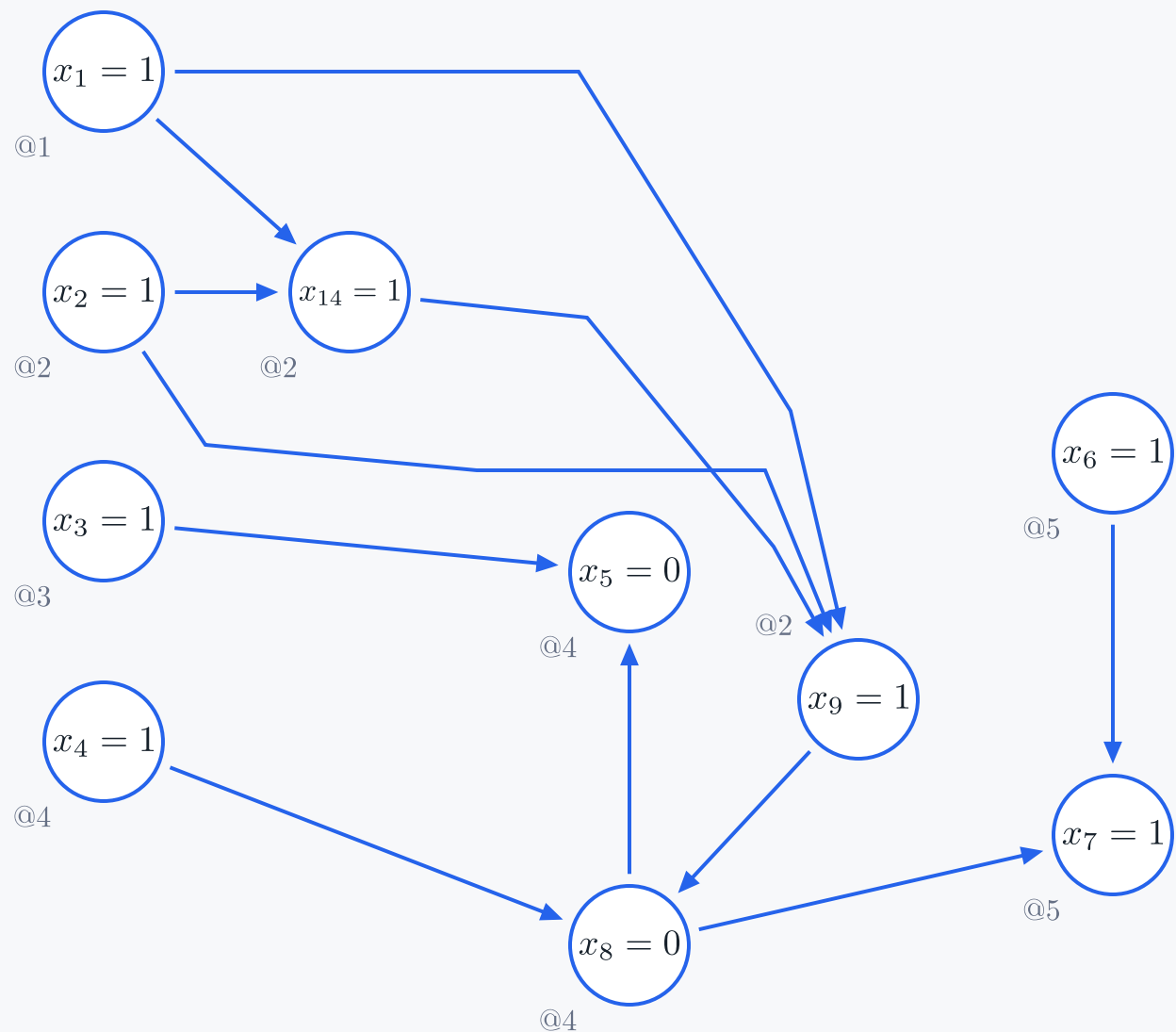
$$(x_9 \vee x_{11} \vee \neg x_{14})$$

$$(x_{10} \vee \neg x_{11} \vee x_{12})$$

$$(\neg x_2 \vee \neg x_{11} \vee \neg x_{12})$$

$$(\neg x_2 \vee x_9 \vee \neg x_{14} \vee \neg x_1)$$

$$(x_8 \vee \neg x_5 \vee \neg x_3)$$



$$(x_1 \vee x_{13})$$

$$(\neg x_1 \vee \neg x_2 \vee x_{14})$$

$$(x_3 \vee x_{15})$$

$$(x_4 \vee x_{16})$$

$$(\neg x_5 \vee \neg x_3 \vee x_6)$$

$$(\neg x_5 \vee \neg x_7)$$

$$(\neg x_6 \vee x_7 \vee x_8)$$

$$(\neg x_4 \vee \neg x_8 \vee \neg x_9)$$

$$(\neg x_1 \vee x_9 \vee \neg x_{10})$$

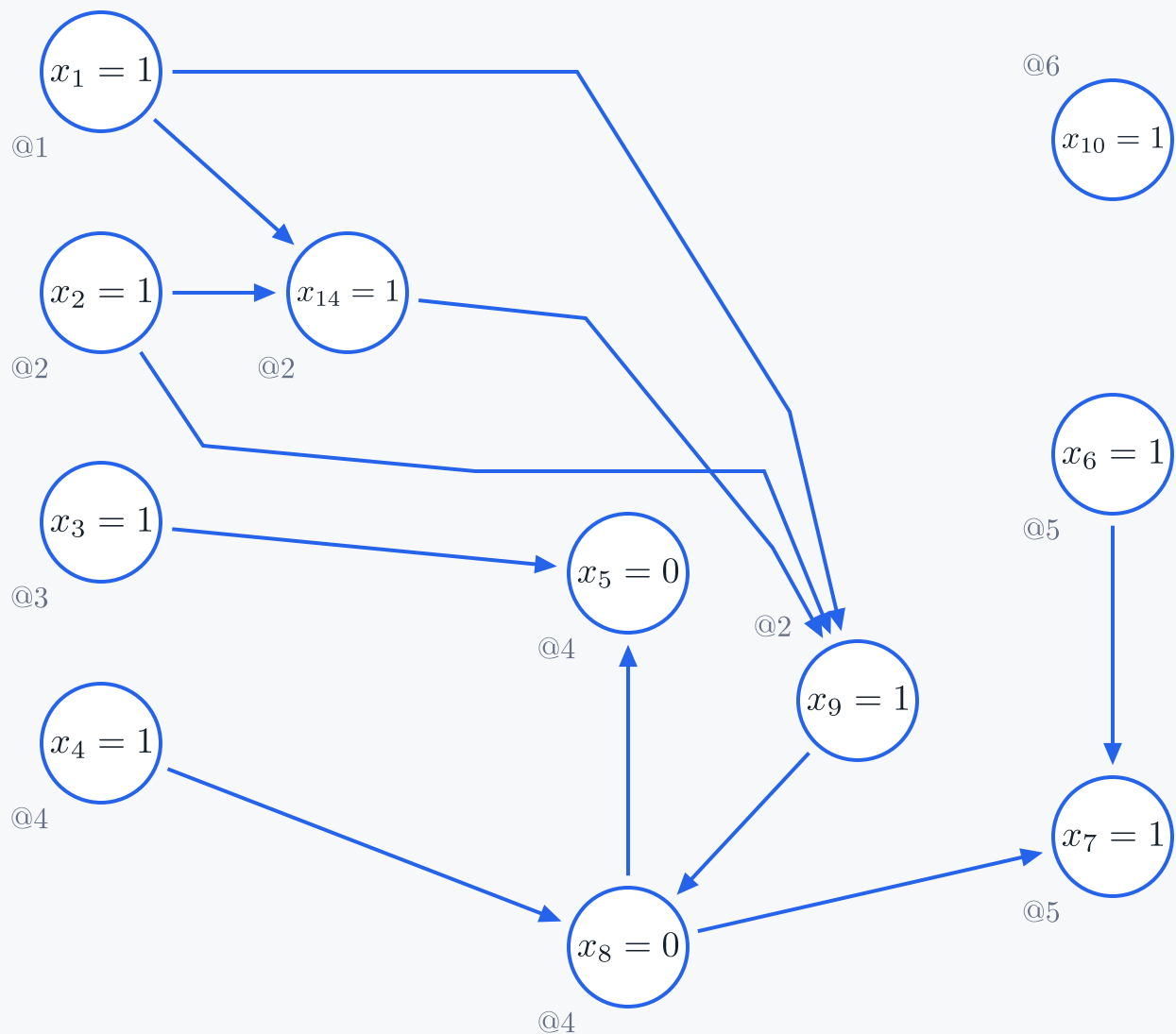
$$(x_9 \vee x_{11} \vee \neg x_{14})$$

$$(x_{10} \vee \neg x_{11} \vee x_{12})$$

$$(\neg x_2 \vee \neg x_{11} \vee \neg x_{12})$$

$$(\neg x_2 \vee x_9 \vee \neg x_{14} \vee \neg x_1)$$

$$(x_8 \vee \neg x_5 \vee \neg x_3)$$



$$(x_1 \vee x_{13})$$

$$(\neg x_1 \vee \neg x_2 \vee x_{14})$$

$$(x_3 \vee x_{15})$$

$$(x_4 \vee x_{16})$$

$$(\neg x_5 \vee \neg x_3 \vee x_6)$$

$$(\neg x_5 \vee \neg x_7)$$

$$(\neg x_6 \vee x_7 \vee x_8)$$

$$(\neg x_4 \vee \neg x_8 \vee \neg x_9)$$

$$(\neg x_1 \vee x_9 \vee \neg x_{10})$$

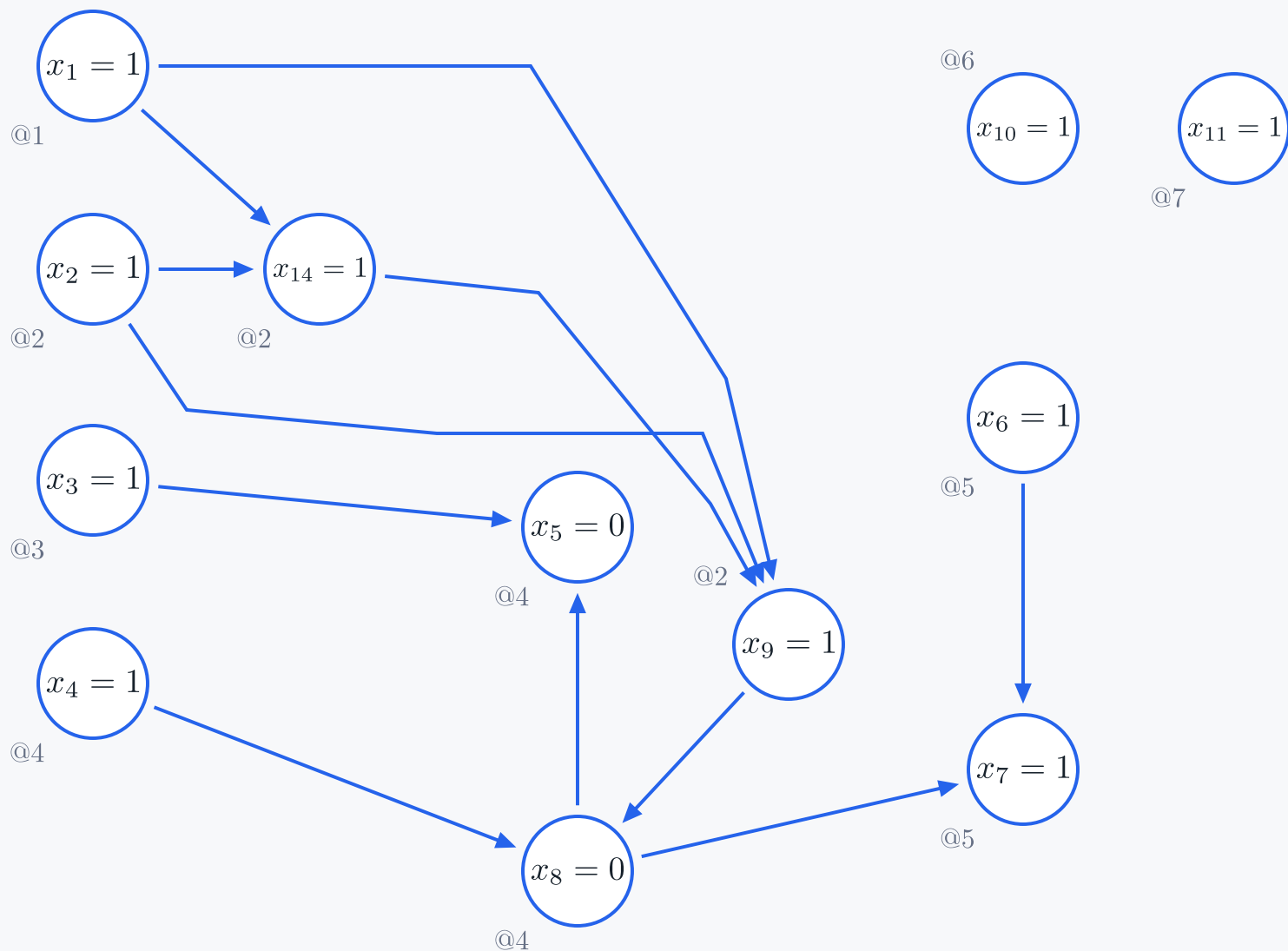
$$(x_9 \vee x_{11} \vee \neg x_{14})$$

$$(x_{10} \vee \neg x_{11} \vee x_{12})$$

$$(\neg x_2 \vee \neg x_{11} \vee \neg x_{12})$$

$$(\neg x_2 \vee x_9 \vee \neg x_{14} \vee \neg x_1)$$

$$(x_8 \vee \neg x_5 \vee \neg x_3)$$



$$(x_1 \vee x_{13})$$

$$(\neg x_1 \vee \neg x_2 \vee x_{14})$$

$$(x_3 \vee x_{15})$$

$$(x_4 \vee x_{16})$$

$$(\neg x_5 \vee \neg x_3 \vee x_6)$$

$$(\neg x_5 \vee \neg x_7)$$

$$(\neg x_6 \vee x_7 \vee x_8)$$

$$(\neg x_4 \vee \neg x_8 \vee \neg x_9)$$

$$(\neg x_1 \vee x_9 \vee \neg x_{10})$$

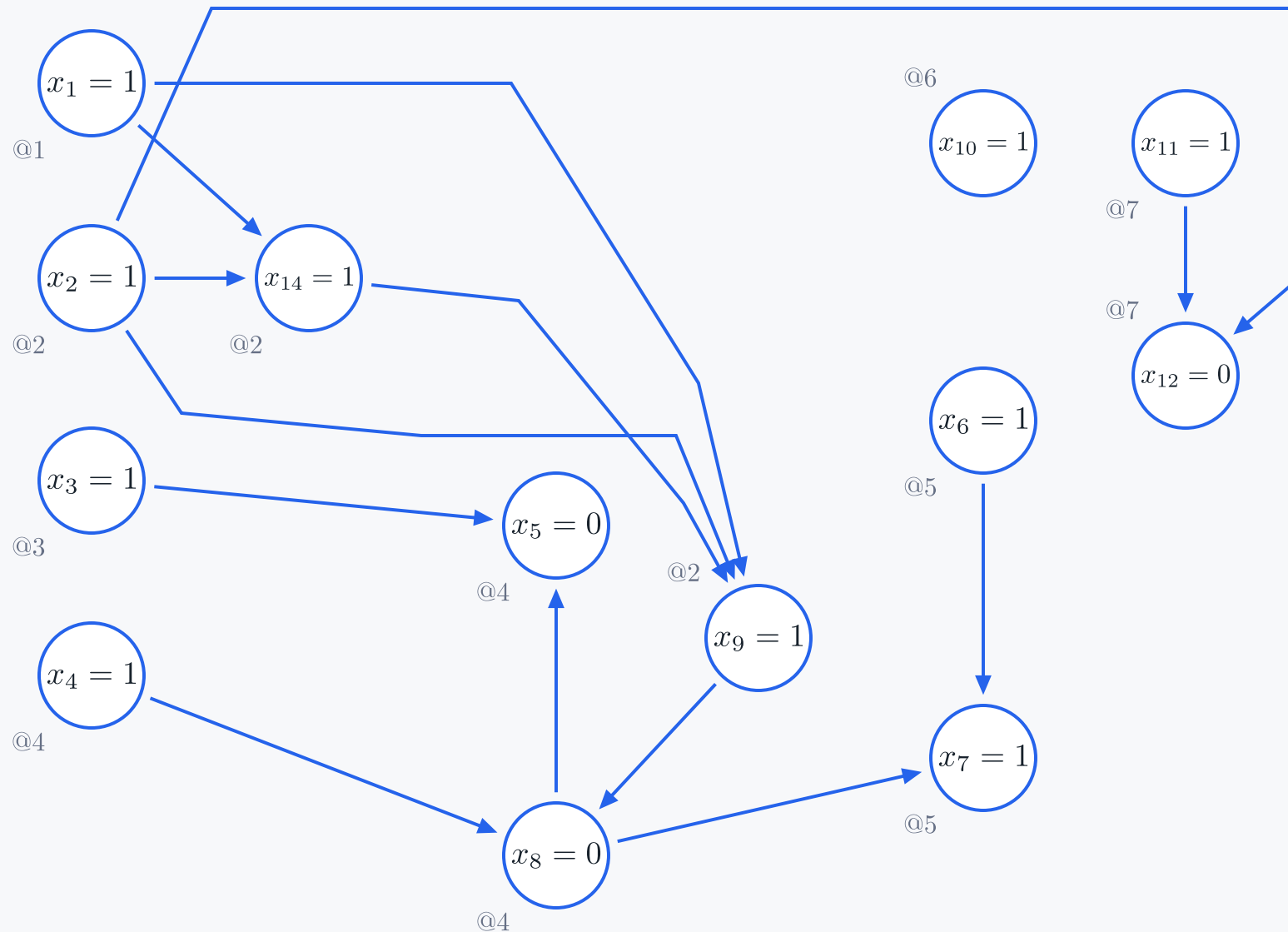
$$(x_9 \vee x_{11} \vee \neg x_{14})$$

$$(x_{10} \vee \neg x_{11} \vee x_{12})$$

$$(\neg x_2 \vee \neg x_{11} \vee \neg x_{12})$$

$$(\neg x_2 \vee x_9 \vee \neg x_{14} \vee \neg x_1)$$

$$(x_8 \vee \neg x_5 \vee \neg x_3)$$



$$(x_1 \vee x_{13})$$

$$(\neg x_1 \vee \neg x_2 \vee x_{14})$$

$$(x_3 \vee x_{15})$$

$$(x_4 \vee x_{16})$$

$$(\neg x_5 \vee \neg x_3 \vee x_6)$$

$$(\neg x_5 \vee \neg x_7)$$

$$(\neg x_6 \vee x_7 \vee x_8)$$

$$(\neg x_4 \vee \neg x_8 \vee \neg x_9)$$

$$(\neg x_1 \vee x_9 \vee \neg x_{10})$$

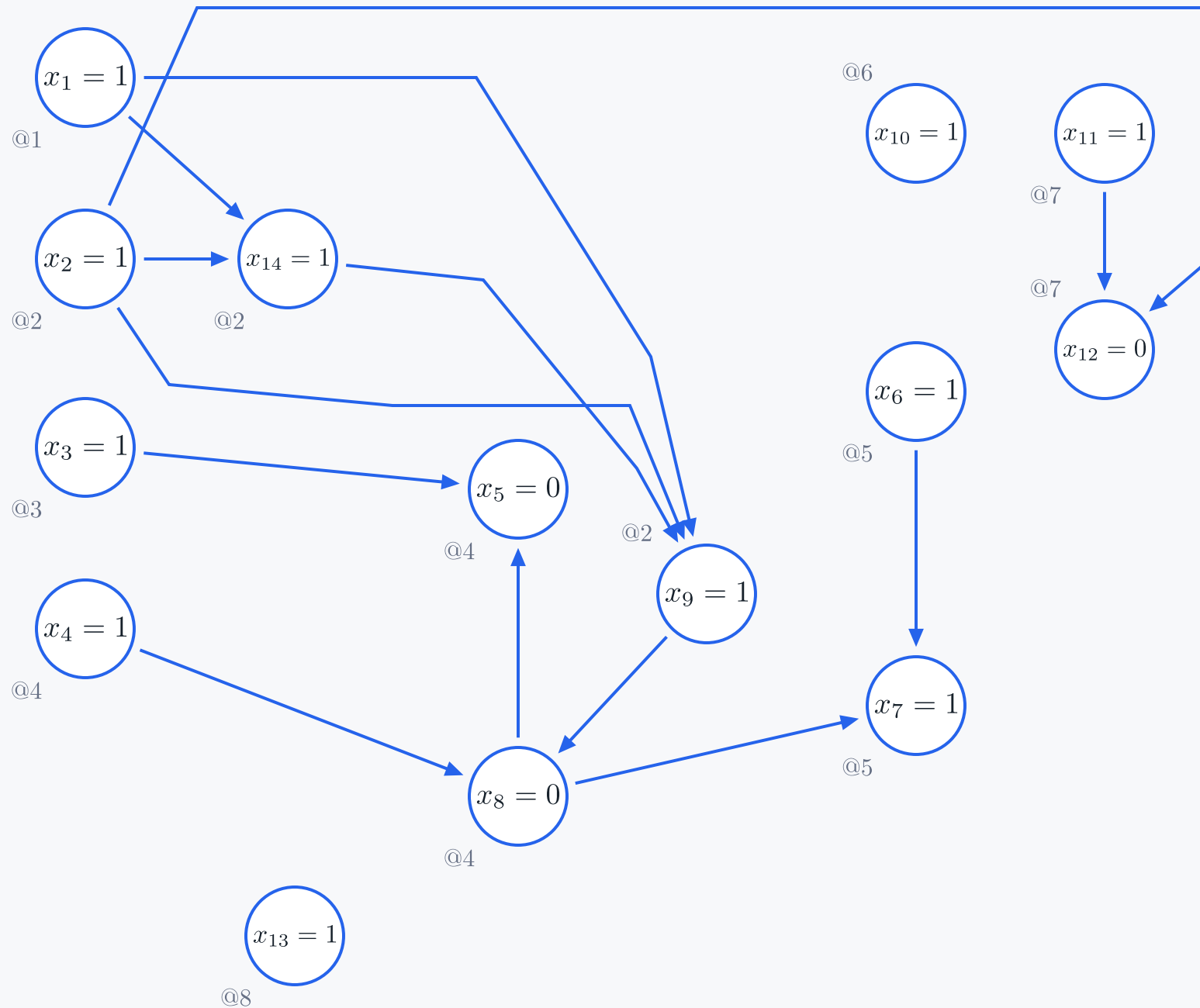
$$(x_9 \vee x_{11} \vee \neg x_{14})$$

$$(x_{10} \vee \neg x_{11} \vee x_{12})$$

$$(\neg x_2 \vee \neg x_{11} \vee \neg x_{12})$$

$$(\neg x_2 \vee x_9 \vee \neg x_{14} \vee \neg x_1)$$

$$(x_8 \vee \neg x_5 \vee \neg x_3)$$



$$(x_1 \vee x_{13})$$

$$(\neg x_1 \vee \neg x_2 \vee x_{14})$$

$$(x_3 \vee x_{15})$$

$$(x_4 \vee x_{16})$$

$$(\neg x_5 \vee \neg x_3 \vee x_6)$$

$$(\neg x_5 \vee \neg x_7)$$

$$(\neg x_6 \vee x_7 \vee x_8)$$

$$(\neg x_4 \vee \neg x_8 \vee \neg x_9)$$

$$(\neg x_1 \vee x_9 \vee \neg x_{10})$$

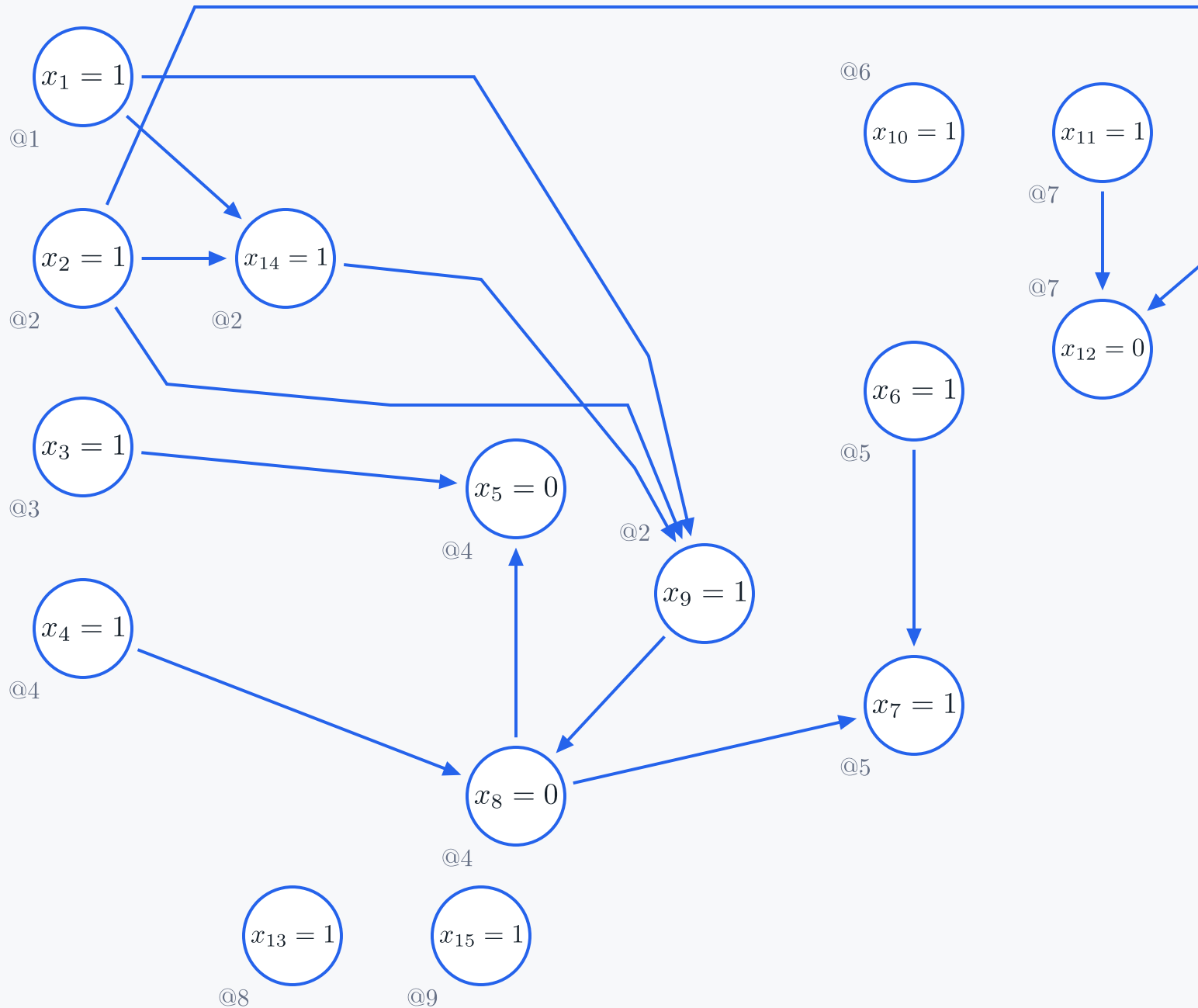
$$(x_9 \vee x_{11} \vee \neg x_{14})$$

$$(x_{10} \vee \neg x_{11} \vee x_{12})$$

$$(\neg x_2 \vee \neg x_{11} \vee \neg x_{12})$$

$$(\neg x_2 \vee x_9 \vee \neg x_{14} \vee \neg x_1)$$

$$(x_8 \vee \neg x_5 \vee \neg x_3)$$



$$(x_1 \vee x_{13})$$

$$(\neg x_1 \vee \neg x_2 \vee x_{14})$$

$$(x_3 \vee x_{15})$$

$$(x_4 \vee x_{16})$$

$$(\neg x_5 \vee \neg x_3 \vee x_6)$$

$$(\neg x_5 \vee \neg x_7)$$

$$(\neg x_6 \vee x_7 \vee x_8)$$

$$(\neg x_4 \vee \neg x_8 \vee \neg x_9)$$

$$(\neg x_1 \vee x_9 \vee \neg x_{10})$$

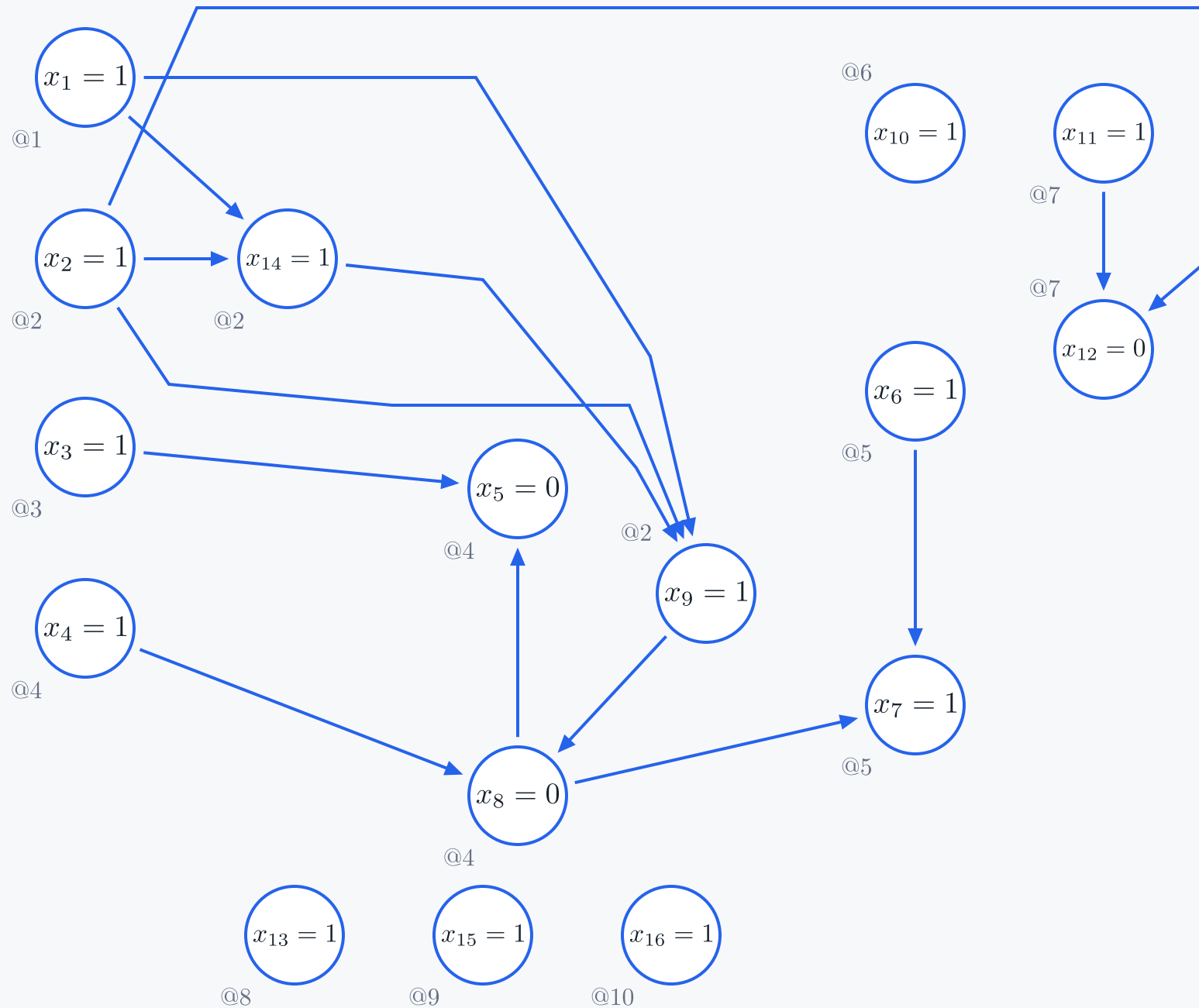
$$(x_9 \vee x_{11} \vee \neg x_{14})$$

$$(x_{10} \vee \neg x_{11} \vee x_{12})$$

$$(\neg x_2 \vee \neg x_{11} \vee \neg x_{12})$$

$$(\neg x_2 \vee x_9 \vee \neg x_{14} \vee \neg x_1)$$

$$(x_8 \vee \neg x_5 \vee \neg x_3)$$



$$(x_8 \vee \neg x_5 \vee \neg x_3)$$

SAT