SAT Solving: Introduction

Priyanka Golia

pgolia@cse.iitd.ac.in

Satisfiability

Boolean Satisfiability: Given a Boolean formula, is there a solution? Assignment of 0's and 1's to the variables that makes the formula equal 1.

$F(x_1, x_2, x_3) : x_1 \vee x_2 \vee x_3$

Is it satisfiable?

Satisfiability

Boolean Satisfiability: Given a Boolean formula, is there a solution? Assignment of 0's and 1's to the variables that makes the formula equal 1.

$F(x_1, x_2, x_3) : x_1 \vee x_2 \vee x_3$

Is it satisfiable?

Yes: $\sigma = < x_1 = 0, x_2 = 0, x_3 = 1 >$

$\sigma \models F(x_1, x_2, x_3)$: is called a satisfying assignment.

Satisfiability

$$F(X) = (x_1 \vee x_2) \wedge (\neg x_1 \vee x_2) \wedge (x_1 \vee \neg x_2) \wedge (\neg x_1 \vee \neg x_2)$$

Satisfiability

$$F(X) = (x_1 \lor x_2) \land (\neg x_1 \lor x_2) \land (x_1 \lor \neg x_2) \land (\neg x_1 \lor \neg x_2)$$

Is it satisfiable?

Satisfiability

$$F(X) = (x_1 \vee x_2) \wedge (\neg x_1 \vee x_2) \wedge (x_1 \vee \neg x_2) \wedge (\neg x_1 \vee \neg x_2)$$

Is it satisfiable?

No, F(X) is UNSAT

Satisfiability

$$F(X) = (x_1 \lor x_2) \land (\neg x_1 \lor x_2) \land (x_1 \lor \neg x_2) \land (\neg x_1 \lor \neg x_2)$$

Is it satisfiable?

No, F(X) is UNSAT

$$F(X) = (x_1 \lor x_2 \lor x_3) \land (\neg x_1 \lor x_2 \lor x_3) \land (x_1 \lor \neg x_2 \lor x_3) \land (x_1 \lor x_2 \lor \neg x_3)$$

Satisfiability

$$F(X) = (x_1 \vee x_2) \wedge (\neg x_1 \vee x_2) \wedge (x_1 \vee \neg x_2) \wedge (\neg x_1 \vee \neg x_2)$$

Is it satisfiable?

No, F(X) is UNSAT

$$F(X) = (x_1 \vee x_2 \vee x_3) \wedge (\neg x_1 \vee x_2 \vee x_3) \wedge (x_1 \vee \neg x_2 \vee x_3) \wedge (x_1 \vee x_2 \vee \neg x_3)$$

Is it satisfiable?

Satisfiability

$$F(X) = (x_1 \vee x_2) \wedge (\neg x_1 \vee x_2) \wedge (x_1 \vee \neg x_2) \wedge (\neg x_1 \vee \neg x_2)$$

Is it satisfiable?

No, F(X) is UNSAT

$$F(X) = (x_1 \vee x_2 \vee x_3) \wedge (\neg x_1 \vee x_2 \vee x_3) \wedge (x_1 \vee \neg x_2 \vee x_3) \wedge (x_1 \vee x_2 \vee \neg x_3)$$

Is it satisfiable?

Yes, F(X) is SAT, $\sigma = < x_1 = 0, x_2 = 1, x_3 = 1 >$

SAT solvers

- Boolean formulas -> SAT Solvers

If formula is SAT, gives an satisfying assignment

Otherwise, UNSAT

Boolean Satisfiability (SAT) Simple to State, Rich in Structure

Despite its simplicity, it captures a vast range of real-world problems.

Boolean Satisfiability (SAT) Simple to State, Rich in Structure

Despite its simplicity, it captures a vast range of real-world problems.

Different Problems

Boolean Satisfiability (SAT) Simple to State, Rich in Structure

Despite its simplicity, it captures a vast range of real-world problems.

Different Problems

Scheduling
Planning

Boolean Satisfiability (SAT) Simple to State, Rich in Structure

Despite its simplicity, it captures a vast range of real-world problems.

Different Problems

Scheduling
Planning

Graph coloring
Vertex cover

Boolean Satisfiability (SAT) Simple to State, Rich in Structure

Despite its simplicity, it captures a vast range of real-world problems.

Different Problems

Scheduling
Planning

Graph coloring
Vertex cover

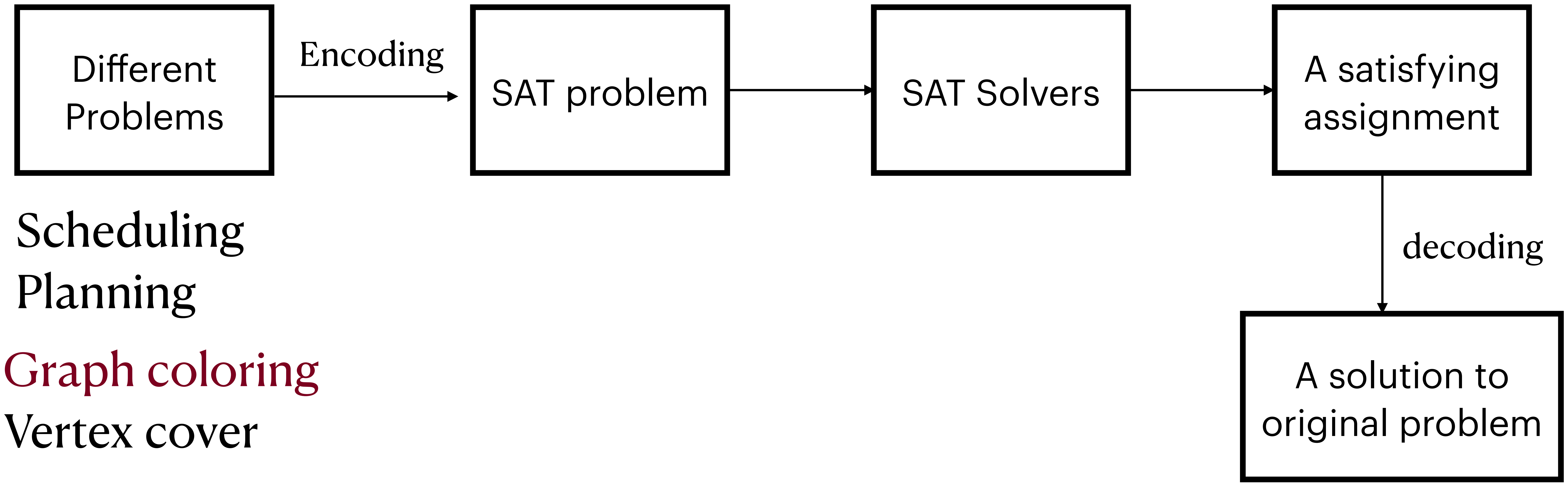
Does there exists an envy free allocation?

Does there exists a fair committee?

....

Boolean Satisfiability (SAT) Simple to State, Rich in Structure

Despite its simplicity, it captures a vast range of real-world problems.



Scheduling
Planning

Graph coloring
Vertex cover

Does there exists an envy free allocation?

Does there exists a fair committee?

....

Boolean Satisfiability (SAT) Simple to State, Rich in Structure

Despite its simplicity, it captures a vast range of real-world problems.

Different Problems $\xrightarrow{\text{Encoding}}$ SAT problem $\rightarrow$ SAT Solvers $\rightarrow$ A satisfying assignment $\xrightarrow{\text{decoding}}$ A solution to original problem

Scheduling
Planning

Graph coloring
Vertex cover

Does there exists an envy free allocation?

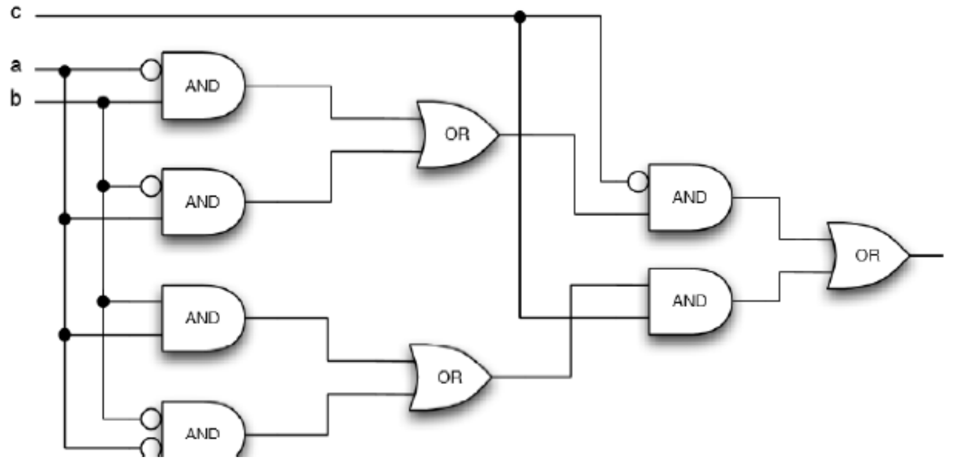
Does there exists a fair committee?

....

Problems in NP!

Boolean satisfiability (SAT) In Formal Verification

```
PC1 (char[] SP, char[] UI) {
        for (int i=0; i<UI.length(); i++) {
                if(SP[i] != UI[i]) return No;
        }
        return Yes;
}
```



System

$\models$

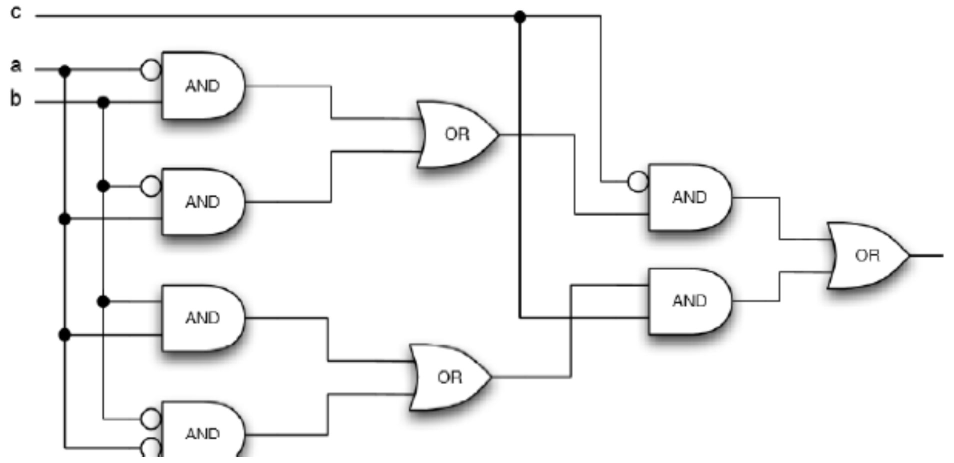
Satisfies

Properties

$$S(I,O) \models P(I,O)$$

Boolean satisfiability (SAT) In Formal Verification

```
PC1 (char[] SP, char[] UI) {
        for (int i=0; i<UI.length(); i++) {
                if(SP[i] != UI[i]) return No;
        }
        return Yes;
}
```



System

$\models$

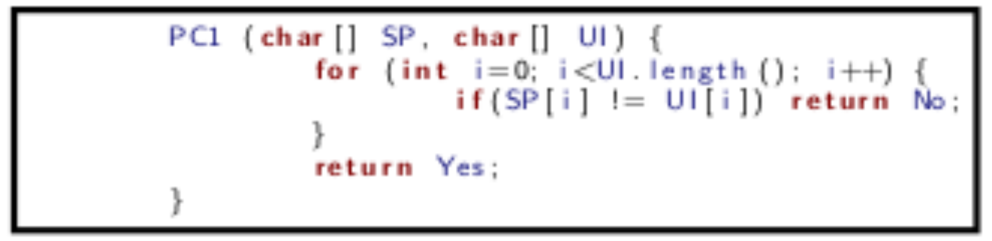
Satisfies

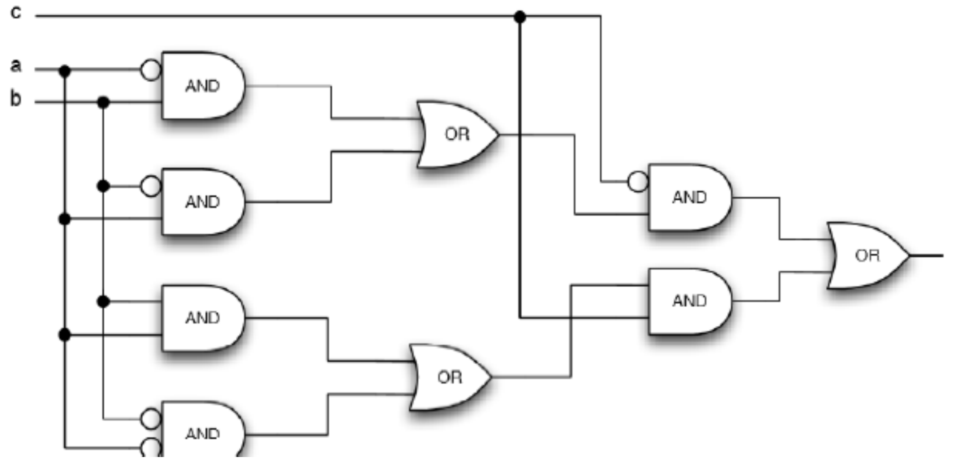
Properties

$$S(I,O) \models P(I,O)$$

Is the always the case that S satisfies Property P?

Boolean satisfiability (SAT) In Formal Verification





$\models$

System

Satisfies

Properties

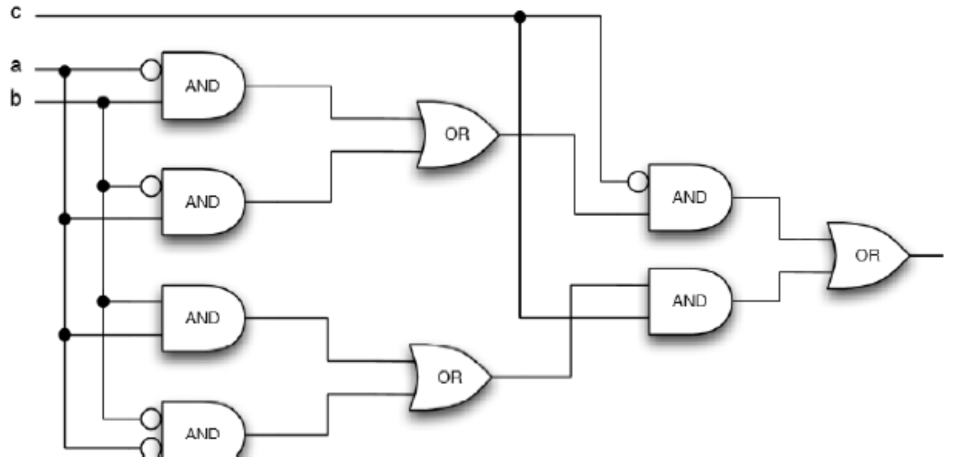
$$S(I,O) \models P(I,O)$$

Is the always the case that S satisfies Property P?

How often S satisfies P?

Boolean satisfiability (SAT) In Formal Verification

```
PC1 (char[] SP, char[] UI) {
        for (int i=0; i<UI.length(); i++) {
                if(SP[i] != UI[i]) return No;
        }
        return Yes;
}
```



$\models$

System

Satisfies

Properties

$$S(I,O) \models P(I,O)$$

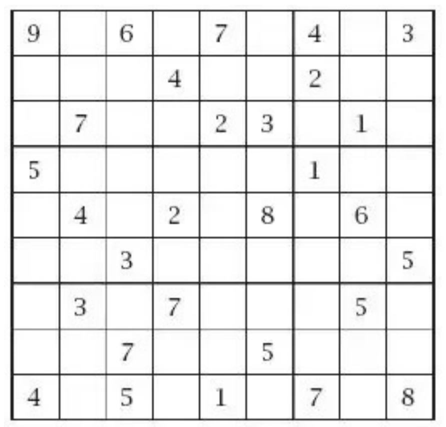
Is the always the case that S satisfies Property P?

How often S satisfies P?

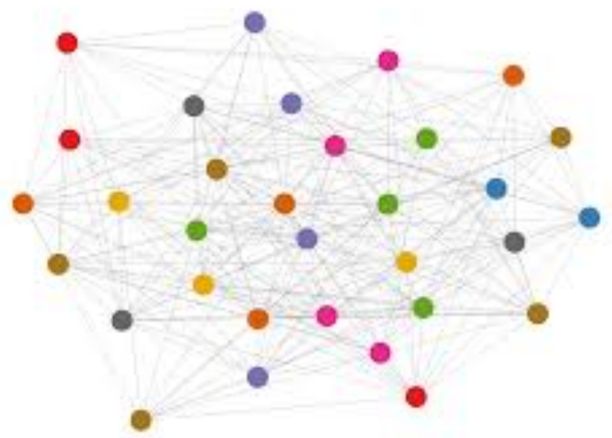
Why S doesn't satisfy P?

Outline

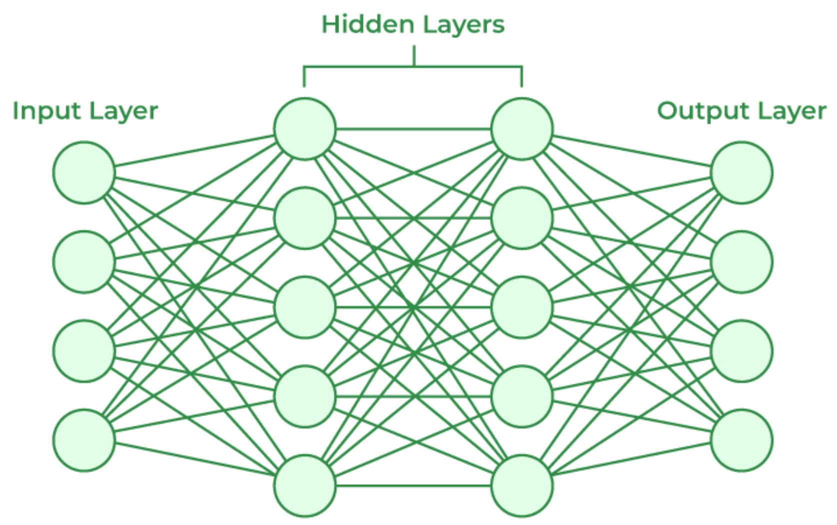
- Basic of propositional logic, and constraints encoding !



Sudoku



Graph Coloring

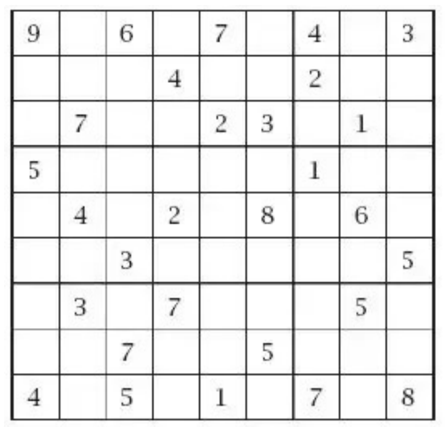


Neural Networks

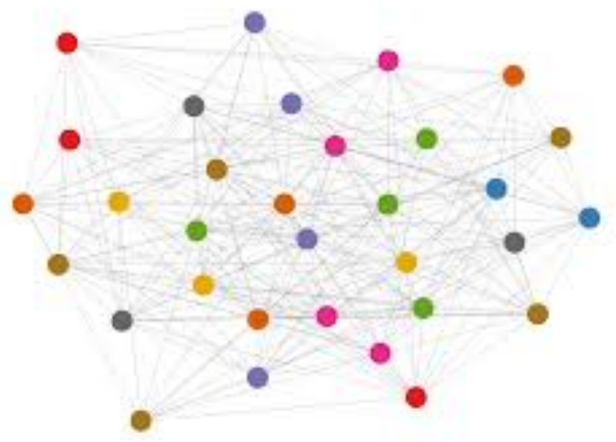
If time permits

Outline

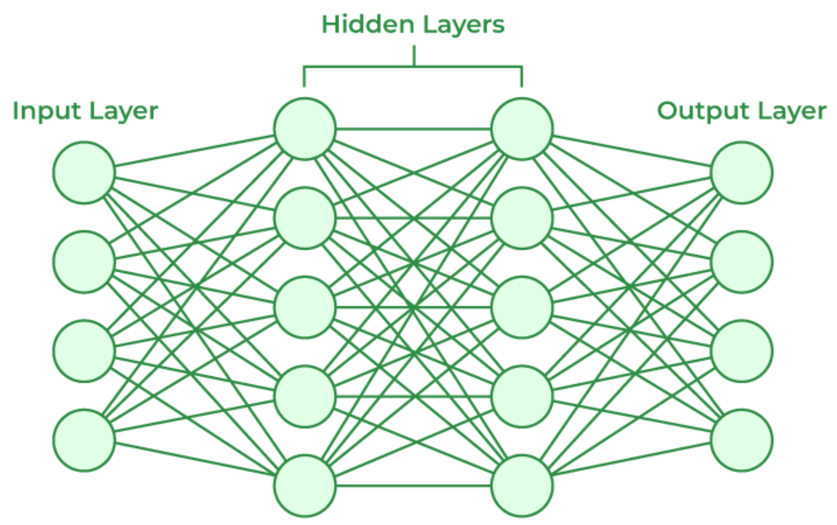
- Basic of propositional logic, and constraints encoding !



Sudoku



Graph Coloring



Neural Networks

- How does SAT solver works? What makes them fast? If time permits

Propositional Logic

(	Left parenthesis
)	Right parenthesis
$\neg$	Negation
$\wedge$	Or
$\vee$	And
$\rightarrow$	Condition
$\leftrightarrow$	Bi-Condition
P_1	Propositional variables
P_2	
P_n	

Logical Symbols: The meaning of logical symbols is always the same.

Non logical Symbols/Propositional Symbols: The meaning of nonlogical symbols depends on the context.

Propositional Logic

$TakeML \vee TakeFM$

$\neg FirstSucceed \rightarrow TryAgain$

$IsWinter \wedge IsSnow$

(	Left parenthesis
)	Right parenthesis
$\neg$	Negation
$\wedge$	Or
$\vee$	And
$\rightarrow$	Condition
$\leftrightarrow$	Bi-Condition
P_1	Propositional variables
P_2	
P_n	

Logical Symbols: The meaning of logical symbols is always the same.

Non logical Symbols/Propositional Symbols: The meaning of nonlogical symbols depends on the context.

Propositional Logic: Semantics

- τ is a function that maps proposition variables of a propositional formula to {0,1}.

$$F = ((p \vee q) \vee r)$$

$$\tau : \{p \mapsto 1, q \mapsto 0, r \mapsto 1\}$$

We call τ a truth assignment.

Propositional Logic: Semantics

- τ is a function that maps proposition variables of a propositional formula to {0,1}.

$$F = ((p \vee q) \vee r)$$

$$\tau : \{p \mapsto 1, q \mapsto 0, r \mapsto 1\}$$

We call τ a truth assignment.

- How many such τ (truth assignments) can exists ?

Propositional Logic: Semantics

- τ is a function that maps proposition variables of a propositional formula to {0,1}.

$$F = ((p \vee q) \vee r)$$

$$\tau : \{p \mapsto 1, q \mapsto 0, r \mapsto 1\}$$

We call τ a truth assignment.

- How many such τ (truth assignments) can exists ?

p	q	r
0	0	0
0	0	1
0	1	0
0	1	1
1	0	0
1	0	1
1	1	0
1	1	1

Propositional Logic: Semantics

- τ is a function that maps proposition variables of a propositional formula to {0,1}.

$$F = ((p \vee q) \vee r)$$

$$\tau : \{p \mapsto 1, q \mapsto 0, r \mapsto 1\}$$

We call τ a truth assignment.

- How many such τ (truth assignments) can exists ?

$$2^{variables(F)}$$

p	q	r
0	0	0
0	0	1
0	1	0
0	1	1
1	0	0
1	0	1
1	1	0
1	1	1

Propositional Logic: Semantics

- τ is a function that maps proposition variables of a propositional formula to {0,1}.

$$F = ((p \vee q) \vee r)$$

$$\tau : \{p \mapsto 1, q \mapsto 0, r \mapsto 1\}$$

We call τ a truth assignment.

- How many such τ (truth assignments) can exists ?

$2^{variables(F)}$

p	q	r
0	0	0
0	0	1
0	1	0
0	1	1
1	0	0
1	0	1
1	1	0
1	1	1

- τ satisfies formula F if and only if $F(\tau)$ is 1, such a τ is called satisfying assignment

Propositional Logic: Semantics

- τ is a function that maps proposition variables of a propositional formula to {0,1}.

$$F = ((p \vee q) \vee r)$$

$$\tau : \{p \mapsto 1, q \mapsto 0, r \mapsto 1\}$$

We call τ a truth assignment.

- How many such τ (truth assignments) can exists ? $2^{variables(F)}$

p	q	r
0	0	0
0	0	1
0	1	0
0	1	1
1	0	0
1	0	1
1	1	0
1	1	1

- τ satisfies formula F if and only if $F(\tau)$ is 1, such a τ is called satisfying assignment

$$F(\tau) : ((1 \vee 0) \vee 1) = 1$$

Propositional Logic: Semantics

- τ is a function that maps proposition variables of a propositional formula to {0,1}.

$$F = ((p \vee q) \vee r)$$

$$\tau : \{p \mapsto 1, q \mapsto 0, r \mapsto 1\}$$

We call τ a truth assignment.

- How many such τ (truth assignments) can exists ? $2^{variables(F)}$

p	q	r
0	0	0
0	0	1
0	1	0
0	1	1
1	0	0
1	0	1
1	1	0
1	1	1

- τ satisfies formula F if and only if $F(\tau)$ is 1,
  such a τ is called satisfying assignment

$$F(\tau) : ((1 \vee 0) \vee 1) = 1$$

- We use $\tau \models F$ to represent.

Propositional Logic: Semantics

- If there exists a τ such that $\tau \vDash F$, we say that F is satisfiable.

$F = ((p \vee q) \vee r)$ $\quad \tau : \{p \mapsto 1, q \mapsto 0, r \mapsto 1\}$ F is satisfiable

Propositional Logic: Semantics

- If there exists a τ such that $\tau \models F$, we say that F is satisfiable.

$F = ((p \vee q) \vee r) \qquad \tau : \{p \mapsto 1, q \mapsto 0, r \mapsto 1\}$ F is satisfiable

- If for all τ in $2^{variables(F)}$, $F(\tau)$ is 1, then F is valid.

Propositional Logic: Semantics

- If there exists a τ such that $\tau \models F$, we say that F is satisfiable.

$F = ((p \vee q) \vee r) \qquad \tau : \{p \mapsto 1, q \mapsto 0, r \mapsto 1\}$ F is satisfiable

- If for all τ in $2^{variables(F)}$, $F(\tau)$ is 1, then F is valid.

Is $F = ((p \vee q) \vee r)$ is valid ?

Propositional Logic: Semantics

- If there exists a τ such that $\tau \vDash F$, we say that F is satisfiable.

$F = ((p \vee q) \vee r)$ $\quad$ $\tau : \{p \mapsto 1, q \mapsto 0, r \mapsto 1\}$ $\quad$ F is satisfiable

- If for all τ in $2^{variables(F)}$, $F(\tau)$ is 1, then F is valid.

Is $F = ((p \vee q) \vee r)$ is valid ? $\quad$ Is $F = (p \vee \neg p)$ is valid ?

Propositional Logic: Semantics

- If there exists a τ such that $\tau \vDash F$, we say that F is satisfiable.

$F = ((p \vee q) \vee r) \qquad \tau : \{p \mapsto 1, q \mapsto 0, r \mapsto 1\}$ F is satisfiable

- If for all τ in $2^{variables(F)}$, $F(\tau)$ is 1, then F is valid.

Is $F = ((p \vee q) \vee r)$ is valid ? Is $F = (p \vee \neg p)$ is valid ?

- If there does not exists a τ in $2^{variables(F)}$ such that $F(\tau)$ is 1, then F is unsatisfiable.

Propositional Logic: Semantics

- If there exists a τ such that $\tau \models F$, we say that F is satisfiable.

$F = ((p \vee q) \vee r)$ $\tau : \{p \mapsto 1, q \mapsto 0, r \mapsto 1\}$ F is satisfiable

- If for all τ in $2^{variables(F)}$, $F(\tau)$ is 1, then F is valid.

Is $F = ((p \vee q) \vee r)$ is valid ? Is $F = (p \vee \neg p)$ is valid ?

- If there does not exists a τ in $2^{variables(F)}$ such that $F(\tau)$ is 1, then F is unsatisfiable.

Is $F = ((p \vee q) \vee r)$ is unsatisfiable?

Propositional Logic: Semantics

- If there exists a τ such that $\tau \vDash F$, we say that F is satisfiable.

$F = ((p \vee q) \vee r)$ $\quad$ $\tau : \{p \mapsto 1, q \mapsto 0, r \mapsto 1\}$ $\quad$ F is satisfiable

- If for all τ in $2^{variables(F)}$, $F(\tau)$ is 1, then F is valid.

Is $F = ((p \vee q) \vee r)$ is valid ? $\quad$ Is $F = (p \vee \neg p)$ is valid ?

- If there does not exists a τ in $2^{variables(F)}$ such that $F(\tau)$ is 1, then F is unsatisfiable.

Is $F = ((p \vee q) \vee r)$ is unsatisfiable? $\quad$ Is $F = (p \wedge \neg p)$ is unsatisfiable ?

Conjunction Normal Form (CNF)

- $F = (x_1 \vee x_2) \wedge (\neg x_1 \vee x_3)$

Clauses Literals : $x_1, \neg x_1, x_2, \neg x_2, x_3, \neg x_3$

CNF: $F = C_1 \wedge C_2 \wedge C_3 \ldots \wedge C_m$

where $C_i = (l_1 \vee l_2 \vee \ldots \vee l_k)$

where $l_j = p; l_j = \neg p$

Where p is propositional variable

Conjunction Normal Form (CNF)

- $F = (x_1 \vee x_2) \wedge (\neg x_1 \vee x_3)$

Clauses

Literals : $x_1, \neg x_1, x_2, \neg x_2, x_3, \neg x_3$

CNF: $F = C_1 \wedge C_2 \wedge C_3 \ldots \wedge C_m$

where $C_i = (l_1 \vee l_2 \vee \ldots \vee l_k)$

where $l_j = p; l_j = \neg p$

Where p is propositional variable

SAT solvers takes CNF formulas as input.

Is every Boolean formula F expressible in conjunctive normal form (CNF)?

Is every Boolean formula F expressible in conjunctive normal form (CNF)?

Is every Boolean formula F expressible in conjunctive normal form (CNF)?

Yes, every F can be represented in F_{CNF}, such that $F = F_{CNF}$

Same set of satisfying assignments

Is every Boolean formula F expressible in conjunctive normal form (CNF)?

Yes, every F can be represented in F_{CNF}, such that $F = F_{CNF}$

Same set of satisfying assignments

$$F = ((x_1 \wedge \neg x_2) \vee (x_3 \wedge x_4))$$

Can you convert F into F_{CNF}?

Is every Boolean formula F expressible in conjunctive normal form (CNF)?

Yes, every F can be represented in F_{CNF}, such that $F = F_{CNF}$

Same set of satisfying assignments

$F = ((x_1 \wedge \neg x_2) \vee (x_3 \wedge x_4))$

Can you convert F into F_{CNF}?

$F_{CNF} = (x_1 \vee x_3) \wedge (x_1 \vee x_4) \wedge (\neg x_2 \vee x_3) \wedge (\neg x_2 \vee x_4)$

Is every Boolean formula F expressible in conjunctive normal form (CNF)?

Yes, every F can be represented in F_{CNF}, such that $F = F_{CNF}$

Same set of satisfying assignments

$F = ((x_1 \wedge \neg x_2) \vee (x_3 \wedge x_4))$ Can you convert F into F_{CNF}?

$$F_{CNF} = (x_1 \vee x_3) \wedge (x_1 \vee x_4) \wedge (\neg x_2 \vee x_3) \wedge (\neg x_2 \vee x_4)$$

$$F = (x_1 \wedge y_1) \vee \ldots \vee (x_n \wedge y_n)$$

How many clauses are there in the F_{CNF}

Is every Boolean formula F expressible in conjunctive normal form (CNF)?

Yes, every F can be represented in F_{CNF}, such that $F = F_{CNF}$

Same set of satisfying assignments

$F = ((x_1 \wedge \neg x_2) \vee (x_3 \wedge x_4))$

Can you convert F into F_{CNF}?

$F_{CNF} = (x_1 \vee x_3) \wedge (x_1 \vee x_4) \wedge (\neg x_2 \vee x_3) \wedge (\neg x_2 \vee x_4)$

$F = (x_1 \wedge y_1) \vee \ldots \vee (x_n \wedge y_n)$

How many clauses are there in the F_{CNF} 2^n

Is every Boolean formula F expressible in conjunctive normal form (CNF)?

Yes, every F can be represented in F_{CNF}, such that $F = F_{CNF}$

Same set of satisfying assignments

$F = ((x_1 \wedge \neg x_2) \vee (x_3 \wedge x_4))$

Can you convert F into F_{CNF}?

$$F_{CNF} = (x_1 \vee x_3) \wedge (x_1 \vee x_4) \wedge (\neg x_2 \vee x_3) \wedge (\neg x_2 \vee x_4)$$

$$F = (x_1 \wedge y_1) \vee \ldots \vee (x_n \wedge y_n)$$

How many clauses are there in the F_{CNF} 2^n

Can we do better?

Equisatisfiable Formulas

Boolean formulas F and G are equisatisfiable if the following holds:

$Vars(G) \subseteq Vars(F)$

- Every satisfying assignment of G can be extended to the satisfying assignment of F.
  - For every $\tau \models G$, there is a τ' such that τ' extends τ to $Vars(F/G)$, and $\tau' \models F$
- Every satisfying assignment of F can be projected on $Vars(G)$ to get the satisfying assignment of G.
  - For every $\tau' \models F$, there is a τ such that $\tau = \tau'_{\downarrow Vars(G)}$ and $\tau \models G$

Equisatisfiable Formulas

$$F = (p \vee \alpha) \wedge (\neg p \vee \beta) \quad \text{and} \quad G = (\alpha \vee \beta)$$

$$Models(F) := \{(p \mapsto 1, \alpha \mapsto 0, \beta \mapsto 1), (p \mapsto 1, \alpha \mapsto 1, \beta \mapsto 1), (p \mapsto 0, \alpha \mapsto 1, \beta \mapsto 0), (p \mapsto 0, \alpha \mapsto 1, \beta \mapsto 1)\}$$

$$Models(F)_{\downarrow Vars(G)} := \{(\alpha \mapsto 0, \beta \mapsto 1), (\alpha \mapsto 1, \beta \mapsto 1), (\alpha \mapsto 1, \beta \mapsto 0)\}$$

$$Models(F)_{\downarrow Vars(G)} := Models(G)$$

For every $\tau \vDash G$, there is a τ' such that τ' extends τ to $Vars(F/G)$, and $\tau' \vDash F$

For every $\tau' \vDash F$, there is a τ such that $\tau = \tau'_{\downarrow Vars(G)}$ and $\tau \vDash G$

Equisatisfiable Formulas

$G = p \lor (q \land r)$

$F = (p \lor t) \land (t \leftrightarrow q \land r)$

$F' = (p \lor t) \land (t \rightarrow q \land r)$

Is F and G equisatisfiable?

Is F' and G equisatisfiable?

Equisatisfiable Formulas

Equisatisfiable Formulas

$$F = ((x_1 \land \neg x_2) \lor (x_3 \land x_4))$$

Equisatisfiable Formulas

$F = ((x_1 \wedge \neg x_2) \vee (x_3 \wedge x_4))$ Can you convert F into equisatisfiable F_{CNF}?

Equisatisfiable Formulas

$F = ((x_1 \wedge \neg x_2) \vee (x_3 \wedge x_4))$ Can you convert F into equisatisfiable F_{CNF}?

$\equiv (t_1 \leftrightarrow (x_1 \wedge \neg x_2)) \wedge (t_2 \leftrightarrow (x_3 \vee x_4)) \wedge (t_1 \vee t_2)$

Equisatisfiable Formulas

$F = ((x_1 \wedge \neg x_2) \vee (x_3 \wedge x_4))$ Can you convert F into equisatisfiable F_{CNF}?

$$\equiv (t_1 \leftrightarrow (x_1 \wedge \neg x_2)) \wedge (t_2 \leftrightarrow (x_3 \vee x_4)) \wedge (t_1 \vee t_2)$$

$$\equiv (\neg t_1 \vee x_1) \wedge (\neg t_1 \vee \neg x_2) \wedge (\neg x_1 \vee x_2 \vee t_1) \wedge (\neg t_2 \vee x_3) \wedge (\neg t_2 \vee x_4) \wedge (\neg x_3 \vee \neg x_4 \vee t_2) \wedge (t_1 \vee t_2)$$

Equisatisfiable Formulas

$F = ((x_1 \land \neg x_2) \lor (x_3 \land x_4))$ Can you convert F into equisatisfiable F_{CNF}?

$$\equiv (t_1 \leftrightarrow (x_1 \land \neg x_2)) \land (t_2 \leftrightarrow (x_3 \lor x_4)) \land (t_1 \lor t_2)$$

$$\equiv (\neg t_1 \lor x_1) \land (\neg t_1 \lor \neg x_2) \land (\neg x_1 \lor x_2 \lor t_1) \land (\neg t_2 \lor x_3) \land (\neg t_2 \lor x_4) \land (\neg x_3 \lor \neg x_4 \lor t_2) \land (t_1 \lor t_2)$$

$$\equiv (\neg t_1 \lor (x_1 \land \neg x_2)) \land (\neg x_1 \lor x_2 \lor t_1) \land (\neg t_2 \lor (x_3 \land x_4)) \land (\neg x_3 \lor \neg x_4 \lor t_2) \land (t_1 \lor t_2)$$

Equisatisfiable Formulas

$F = ((x_1 \wedge \neg x_2) \vee (x_3 \wedge x_4))$ Can you convert F into equisatisfiable F_{CNF}?

$$\equiv (t_1 \leftrightarrow (x_1 \wedge \neg x_2)) \wedge (t_2 \leftrightarrow (x_3 \vee x_4)) \wedge (t_1 \vee t_2)$$

$$\equiv (\neg t_1 \vee x_1) \wedge (\neg t_1 \vee \neg x_2) \wedge (\neg x_1 \vee x_2 \vee t_1) \wedge (\neg t_2 \vee x_3) \wedge (\neg t_2 \vee x_4) \wedge (\neg x_3 \vee \neg x_4 \vee t_2) \wedge (t_1 \vee t_2)$$

$$\equiv (\neg t_1 \vee (x_1 \wedge \neg x_2)) \wedge (\neg x_1 \vee x_2 \vee t_1) \wedge (\neg t_2 \vee (x_3 \wedge x_4)) \wedge (\neg x_3 \vee \neg x_4 \vee t_2) \wedge (t_1 \vee t_2)$$

$$\equiv F_{CNF}$$

Equisatisfiable Formulas

$F = ((x_1 \wedge \neg x_2) \vee (x_3 \wedge x_4))$ Can you convert F into equisatisfiable F_{CNF}?

$\equiv (t_1 \leftrightarrow (x_1 \wedge \neg x_2)) \wedge (t_2 \leftrightarrow (x_3 \vee x_4)) \wedge (t_1 \vee t_2)$

$\equiv (\neg t_1 \vee x_1) \wedge (\neg t_1 \vee \neg x_2) \wedge (\neg x_1 \vee x_2 \vee t_1) \wedge (\neg t_2 \vee x_3) \wedge (\neg t_2 \vee x_4) \wedge (\neg x_3 \vee \neg x_4 \vee t_2) \wedge (t_1 \vee t_2)$

$\equiv (\neg t_1 \vee (x_1 \wedge \neg x_2)) \wedge (\neg x_1 \vee x_2 \vee t_1) \wedge (\neg t_2 \vee (x_3 \wedge x_4)) \wedge (\neg x_3 \vee \neg x_4 \vee t_2) \wedge (t_1 \vee t_2)$

$\equiv F_{CNF}$ Tseitin transformation

Equisatisfiable Formulas

$F = ((x_1 \wedge \neg x_2) \vee (x_3 \wedge x_4))$ Can you convert F into equisatisfiable F_{CNF}?

$\equiv (t_1 \leftrightarrow (x_1 \wedge \neg x_2)) \wedge (t_2 \leftrightarrow (x_3 \vee x_4)) \wedge (t_1 \vee t_2)$

$\equiv (\neg t_1 \vee x_1) \wedge (\neg t_1 \vee \neg x_2) \wedge (\neg x_1 \vee x_2 \vee t_1) \wedge (\neg t_2 \vee x_3) \wedge (\neg t_2 \vee x_4) \wedge (\neg x_3 \vee \neg x_4 \vee t_2) \wedge (t_1 \vee t_2)$

$\equiv (\neg t_1 \vee (x_1 \wedge \neg x_2)) \wedge (\neg x_1 \vee x_2 \vee t_1) \wedge (\neg t_2 \vee (x_3 \wedge x_4)) \wedge (\neg x_3 \vee \neg x_4 \vee t_2) \wedge (t_1 \vee t_2)$

$\equiv F_{CNF}$ Tseitin transformation

$F = (x_1 \wedge y_1) \vee \ldots \vee (x_n \wedge y_n)$

How many clauses are there in equisatisfiable F_{CNF}

Equisatisfiable Formulas

$F = ((x_1 \wedge \neg x_2) \vee (x_3 \wedge x_4))$ Can you convert F into equisatisfiable F_{CNF}?

$\equiv (t_1 \leftrightarrow (x_1 \wedge \neg x_2)) \wedge (t_2 \leftrightarrow (x_3 \vee x_4)) \wedge (t_1 \vee t_2)$

$\equiv (\neg t_1 \vee x_1) \wedge (\neg t_1 \vee \neg x_2) \wedge (\neg x_1 \vee x_2 \vee t_1) \wedge (\neg t_2 \vee x_3) \wedge (\neg t_2 \vee x_4) \wedge (\neg x_3 \vee \neg x_4 \vee t_2) \wedge (t_1 \vee t_2)$

$\equiv (\neg t_1 \vee (x_1 \wedge \neg x_2)) \wedge (\neg x_1 \vee x_2 \vee t_1) \wedge (\neg t_2 \vee (x_3 \wedge x_4)) \wedge (\neg x_3 \vee \neg x_4 \vee t_2) \wedge (t_1 \vee t_2)$

$\equiv F_{CNF}$ Tseitin transformation

$F = (x_1 \wedge y_1) \vee \ldots \vee (x_n \wedge y_n)$

How many clauses are there in equisatisfiable F_{CNF} $2n + n + 1$

Every Boolean formula F can be converted into a CNF formula F_{CNF} of polynomial size, such that F is satisfiable if and only if F_{CNF} is satisfiable

K-SAT

CNF: $F = C_1 \wedge C_2 \wedge C_3 \ldots \wedge C_m$

where $C_i = (l_1 \vee l_2 \vee \ldots \vee l_k)$

where $l_j = p; l_j = \neg p$

Where p is propositional variable

K-SAT

CNF: $F = C_1 \wedge C_2 \wedge C_3 \ldots \wedge C_m$

where $C_i = (l_1 \vee l_2 \vee \ldots \vee l_k)$

where $l_j = p; l_j = \neg p$

Where p is propositional variable

$K - SAT$ if every clause in F has exactly K literals.

K-SAT

CNF: $F = C_1 \wedge C_2 \wedge C_3 \ldots \wedge C_m$

where $C_i = (l_1 \vee l_2 \vee \ldots \vee l_k)$

where $l_j = p; l_j = \neg p$

Where p is propositional variable

$K - SAT$ if every clause in F has exactly K literals.

If $K = 2, F = (x_1 \vee \neg x_2) \wedge (x_3 \vee x_4)$

K-SAT

CNF: $F = C_1 \wedge C_2 \wedge C_3 \ldots \wedge C_m$

where $C_i = (l_1 \vee l_2 \vee \ldots \vee l_k)$

where $l_j = p; l_j = \neg p$

Where p is propositional variable

$K - SAT$ if every clause in F has exactly K literals.

If $K = 2, F = (x_1 \vee \neg x_2) \wedge (x_3 \vee x_4)$

If $K = 3, \ \ F = (x_1 \vee \neg x_2 \vee x_3) \wedge (x_1 \vee x_3 \vee x_4)$

Can you convert given $4 - SAT$ formula into an equisatisfiable $3 - SAT$ formula?

Can you convert given $3 - SAT$ formula into an equisatisfiable $2 - SAT$ formula?

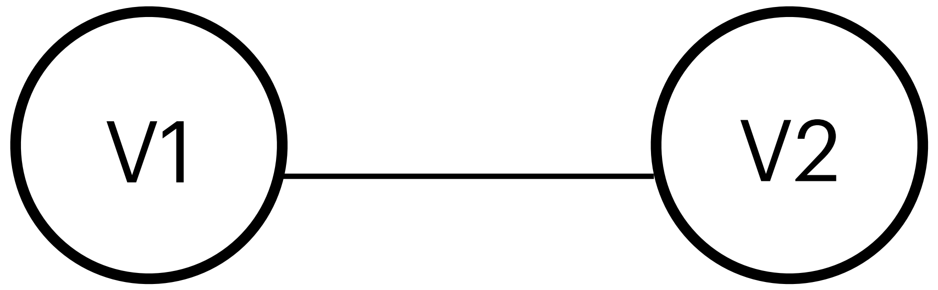
Constraint Encoding

Encoding of Graph Coloring to SAT

- Proper coloring: An assignment of colors to the vertices of a graph such that no two adjacent vertices have same color.
- K-color: A proper coloring involving a total of K colors.
- Is the following graphs 2-colorable?

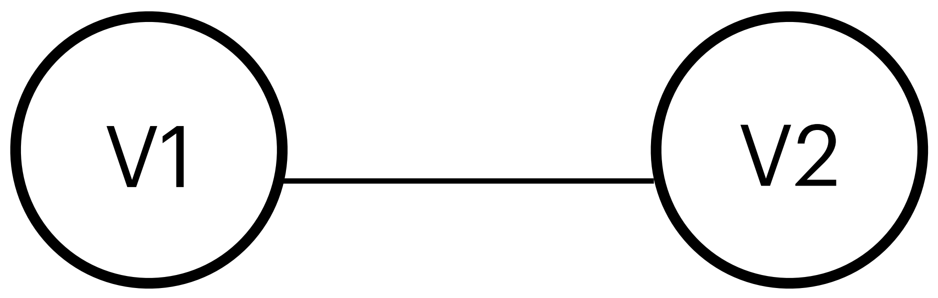
Encoding of Graph Coloring to SAT

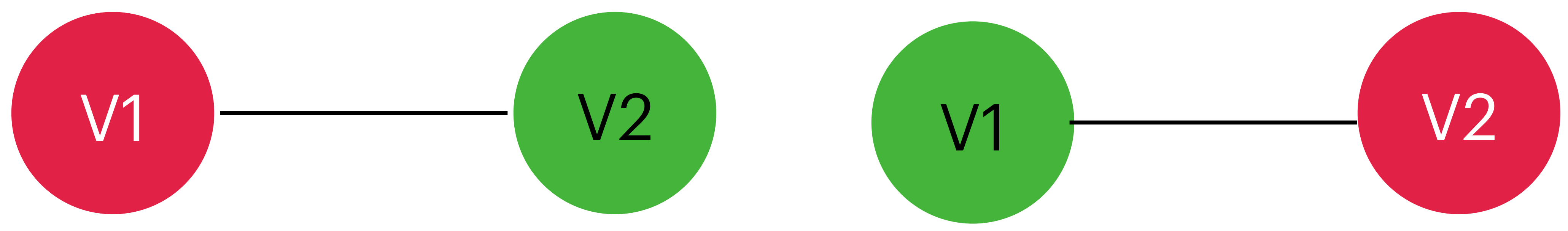
- Proper coloring: An assignment of colors to the vertices of a graph such that no two adjacent vertices have same color.
- K-color: A proper coloring involving a total of K colors.
- Is the following graphs 2-colorable?



Encoding of Graph Coloring to SAT

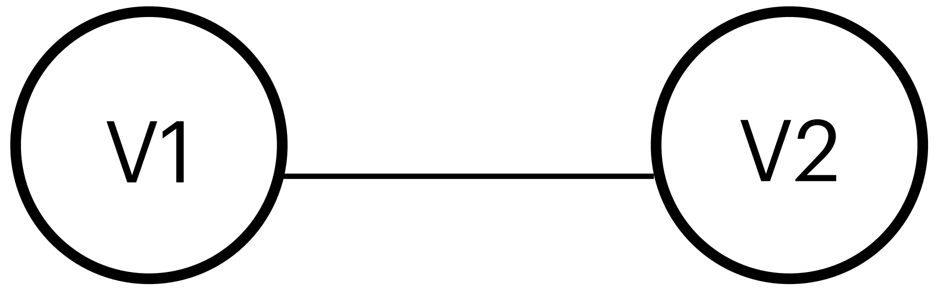
- Proper coloring: An assignment of colors to the vertices of a graph such that no two adjacent vertices have same color.
- K-color: A proper coloring involving a total of K colors.
- Is the following graphs 2-colorable?

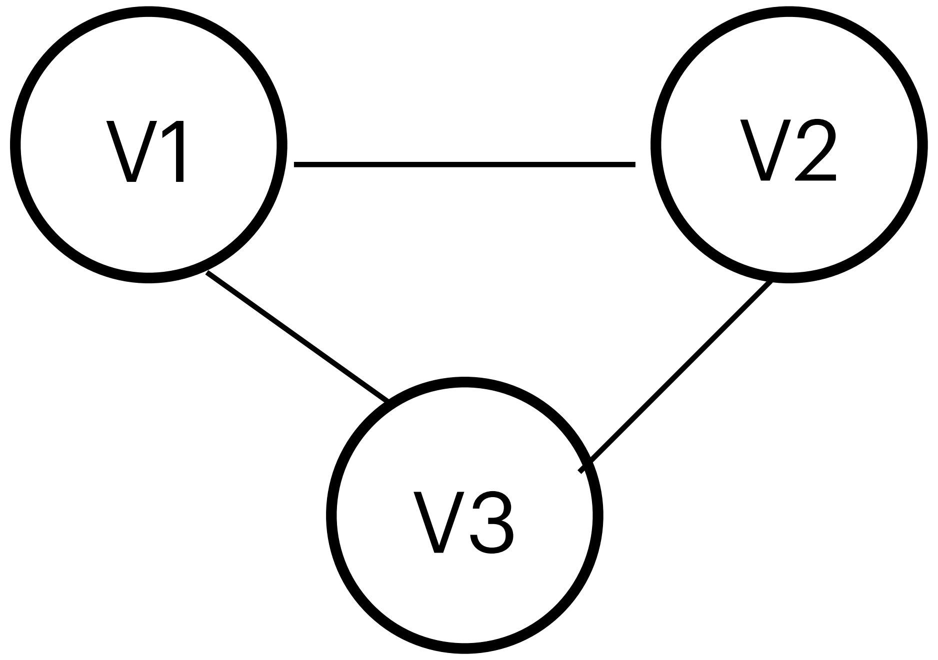




Encoding of Graph Coloring to SAT

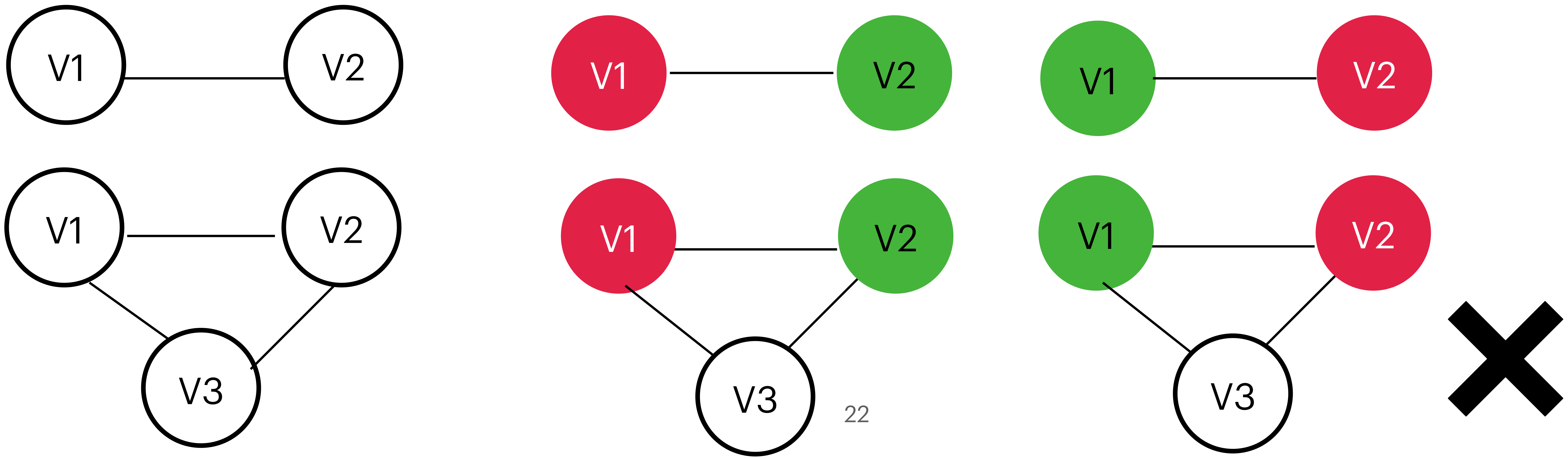
- Proper coloring: An assignment of colors to the vertices of a graph such that no two adjacent vertices have same color.
- K-color: A proper coloring involving a total of K colors.
- Is the following graphs 2-colorable?





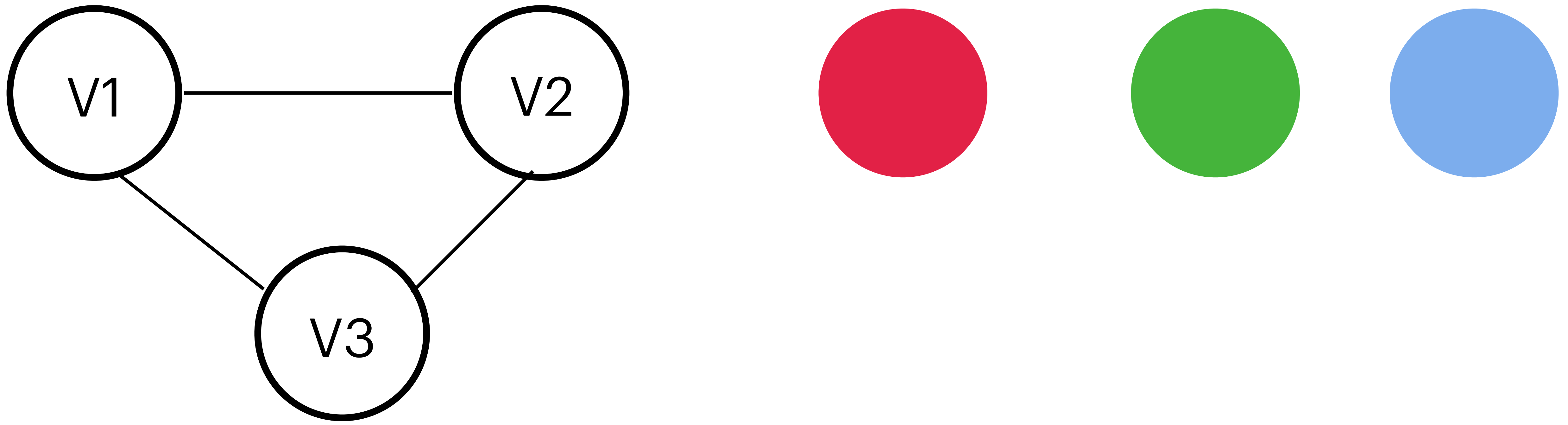
Encoding of Graph Coloring to SAT

- Proper coloring: An assignment of colors to the vertices of a graph such that no two adjacent vertices have same color.
- K-color: A proper coloring involving a total of K colors.
- Is the following graphs 2-colorable?



Encoding of Graph Coloring to SAT

Given a graph G(V,E) with V as a set of vertices and E as a set of edges, and an integer K (representing the number of colors), can we encode the proper graph coloring into a CNF formula such that the formula is satisfiable (SAT) if and only if the graph is K-colorable.

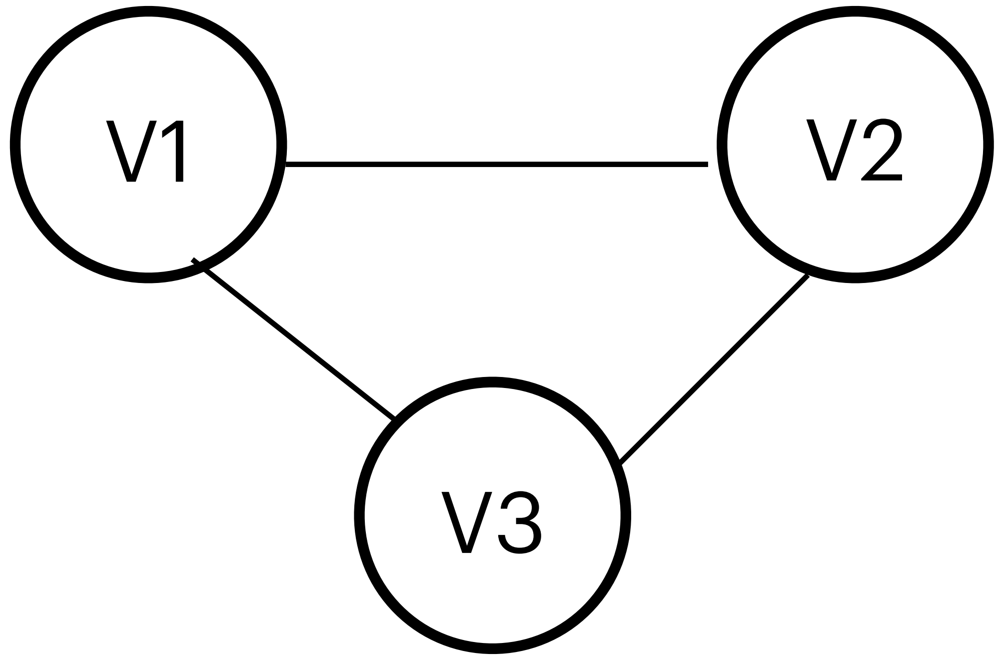


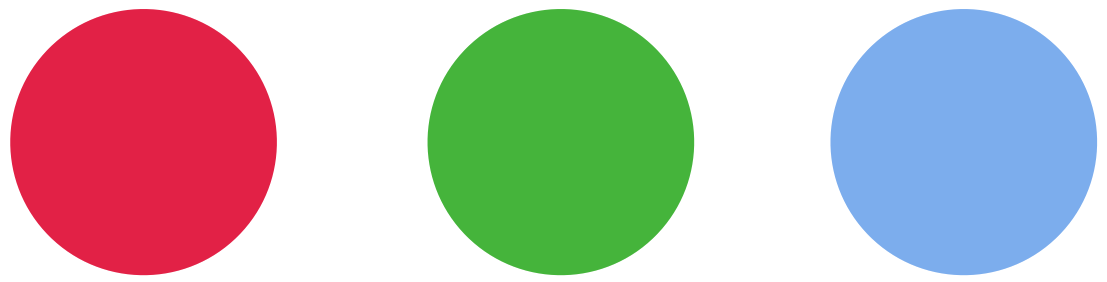
We want to encode that:

- No two adjacent vertices share the same color.
- Each vertex has exactly one color.

Step 1: Propositional Variables

- Use propositional variables $v_{i,g}$, where $i \in \{1,2,3\}, g \in \{R, G, B\}$
- $v_{i,g}$ is True, if and only if, vertex i is assigned g color.





$v_{1,G}, v_{1,R}, v_{1,B}$ $v_{2,G}, v_{2,R}, v_{2,B}$ $v_{3,G}, v_{3,R}, v_{3,B}$

Step 2: Encoding Constraints

- Each vertex must have exactly one color.
  - Each vertex must have at least one color, and each vertex must have at most one color

Step 2: Encoding Constraints

- Each vertex must have exactly one color.
  - Each vertex must have at least one color, and each vertex must have at most one color

How are we going to encode, each vertex must have at least one color:

Step 2: Encoding Constraints

- Each vertex must have exactly one color.
    - Each vertex must have at least one color, and each vertex must have at most one color

How are we going to encode, each vertex must have at least one color:

For vertex $V_1 : v_{1,G} \vee v_{1,R} \vee v_{1,B}$ $\quad V_2 : v_{2,G} \vee v_{2,R} \vee v_{2,B}$ $\quad V_3 : v_{3,G} \vee v_{3,R} \vee v_{3,B}$

Step 2: Encoding Constraints

- Each vertex must have exactly one color.
    - Each vertex must have at least one color, and each vertex must have at most one color

How are we going to encode, each vertex must have at least one color:

For vertex $V_1 : v_{1,G} \vee v_{1,R} \vee v_{1,B}$ $\quad V_2 : v_{2,G} \vee v_{2,R} \vee v_{2,B}$ $\quad V_3 : v_{3,G} \vee v_{3,R} \vee v_{3,B}$

How are we going to encode, each vertex must have at most one color:

Step 2: Encoding Constraints

- Each vertex must have exactly one color.
  - Each vertex must have at least one color, and each vertex must have at most one color

How are we going to encode, each vertex must have at least one color:

For vertex $V_1 : v_{1,G} \vee v_{1,R} \vee v_{1,B}$ $V_2 : v_{2,G} \vee v_{2,R} \vee v_{2,B}$ $V_3 : v_{3,G} \vee v_{3,R} \vee v_{3,B}$

How are we going to encode, each vertex must have at most one color:

$V_1 : (\neg v_{1,G} \vee \neg v_{1,R}) \wedge$

Step 2: Encoding Constraints

- Each vertex must have exactly one color.
    - Each vertex must have at least one color, and each vertex must have at most one color

How are we going to encode, each vertex must have at least one color:

For vertex $V_1 : v_{1,G} \vee v_{1,R} \vee v_{1,B}$ $\quad V_2 : v_{2,G} \vee v_{2,R} \vee v_{2,B}$ $\quad V_3 : v_{3,G} \vee v_{3,R} \vee v_{3,B}$

How are we going to encode, each vertex must have at most one color:

$$V_1 : (\neg v_{1,G} \vee \neg v_{1,R}) \wedge$$

$$(\neg v_{1,G} \vee \neg v_{1,B}) \wedge$$

Step 2: Encoding Constraints

- Each vertex must have exactly one color.
  - Each vertex must have at least one color, and each vertex must have at most one color

How are we going to encode, each vertex must have at least one color:

For vertex $V_1 : v_{1,G} \vee v_{1,R} \vee v_{1,B}$ $\quad V_2 : v_{2,G} \vee v_{2,R} \vee v_{2,B}$ $\quad V_3 : v_{3,G} \vee v_{3,R} \vee v_{3,B}$

How are we going to encode, each vertex must have at most one color:

$$V_1 : (\neg v_{1,G} \vee \neg v_{1,R}) \wedge$$

$$(\neg v_{1,G} \vee \neg v_{1,B}) \wedge$$

$$(\neg v_{1,R} \vee \neg v_{1,B}) \wedge$$

Step 2: Encoding Constraints

- Each vertex must have exactly one color.
  - Each vertex must have at least one color, and each vertex must have at most one color

How are we going to encode, each vertex must have at least one color:

For vertex $V_1 : v_{1,G} \vee v_{1,R} \vee v_{1,B}$ $\quad V_2 : v_{2,G} \vee v_{2,R} \vee v_{2,B}$ $\quad V_3 : v_{3,G} \vee v_{3,R} \vee v_{3,B}$

How are we going to encode, each vertex must have at most one color:

$V_1 : (\neg v_{1,G} \vee \neg v_{1,R}) \wedge$ $\quad V_2 : (\neg v_{2,G} \vee \neg v_{2,R}) \wedge$

$(\neg v_{1,G} \vee \neg v_{1,B}) \wedge$

$(\neg v_{1,R} \vee \neg v_{1,B}) \wedge$

Step 2: Encoding Constraints

- Each vertex must have exactly one color.
  - Each vertex must have at least one color, and each vertex must have at most one color

How are we going to encode, each vertex must have at least one color:

For vertex $V_1 : v_{1,G} \vee v_{1,R} \vee v_{1,B}$ $\quad V_2 : v_{2,G} \vee v_{2,R} \vee v_{2,B}$ $\quad V_3 : v_{3,G} \vee v_{3,R} \vee v_{3,B}$

How are we going to encode, each vertex must have at most one color:

$$V_1 : (\neg v_{1,G} \vee \neg v_{1,R}) \wedge$$
$$(\neg v_{1,G} \vee \neg v_{1,B}) \wedge$$
$$(\neg v_{1,R} \vee \neg v_{1,B}) \wedge$$

$$V_2 : (\neg v_{2,G} \vee \neg v_{2,R}) \wedge$$
$$(\neg v_{2,G} \vee \neg v_{2,B}) \wedge$$

Step 2: Encoding Constraints

- Each vertex must have exactly one color.
  - Each vertex must have at least one color, and each vertex must have at most one color

How are we going to encode, each vertex must have at least one color:

For vertex $V_1 : v_{1,G} \vee v_{1,R} \vee v_{1,B}$ $\quad V_2 : v_{2,G} \vee v_{2,R} \vee v_{2,B}$ $\quad V_3 : v_{3,G} \vee v_{3,R} \vee v_{3,B}$

How are we going to encode, each vertex must have at most one color:

$$V_1 : (\neg v_{1,G} \vee \neg v_{1,R}) \wedge \qquad V_2 : (\neg v_{2,G} \vee \neg v_{2,R}) \wedge$$

$$(\neg v_{1,G} \vee \neg v_{1,B}) \wedge \qquad (\neg v_{2,G} \vee \neg v_{2,B}) \wedge$$

$$(\neg v_{1,R} \vee \neg v_{1,B}) \wedge \qquad (\neg v_{2,R} \vee \neg v_{2,B}) \wedge$$

Step 2: Encoding Constraints

- Each vertex must have exactly one color.
    - Each vertex must have at least one color, and each vertex must have at most one color

How are we going to encode, each vertex must have at least one color:

For vertex $V_1 : v_{1,G} \vee v_{1,R} \vee v_{1,B}$ $\quad V_2 : v_{2,G} \vee v_{2,R} \vee v_{2,B}$ $\quad V_3 : v_{3,G} \vee v_{3,R} \vee v_{3,B}$

How are we going to encode, each vertex must have at most one color:

$$V_1 : (\neg v_{1,G} \vee \neg v_{1,R}) \wedge$$
$$(\neg v_{1,G} \vee \neg v_{1,B}) \wedge$$
$$(\neg v_{1,R} \vee \neg v_{1,B}) \wedge$$

$$V_2 : (\neg v_{2,G} \vee \neg v_{2,R}) \wedge$$
$$(\neg v_{2,G} \vee \neg v_{2,B}) \wedge$$
$$(\neg v_{2,R} \vee \neg v_{2,B}) \wedge$$

$$V_3 : (\neg v_{3,G} \vee \neg v_{3,R}) \wedge$$

Step 2: Encoding Constraints

- Each vertex must have exactly one color.
    - Each vertex must have at least one color, and each vertex must have at most one color

How are we going to encode, each vertex must have at least one color:

For vertex $V_1 : v_{1,G} \vee v_{1,R} \vee v_{1,B}$ $\quad V_2 : v_{2,G} \vee v_{2,R} \vee v_{2,B}$ $\quad V_3 : v_{3,G} \vee v_{3,R} \vee v_{3,B}$

How are we going to encode, each vertex must have at most one color:

$V_1 : (\neg v_{1,G} \vee \neg v_{1,R}) \wedge$ $\quad V_2 : (\neg v_{2,G} \vee \neg v_{2,R}) \wedge$ $\quad V_3 : (\neg v_{3,G} \vee \neg v_{3,R}) \wedge$

$(\neg v_{1,G} \vee \neg v_{1,B}) \wedge$ $\quad (\neg v_{2,G} \vee \neg v_{2,B}) \wedge$ $\quad (\neg v_{3,G} \vee \neg v_{3,B}) \wedge$

$(\neg v_{1,R} \vee \neg v_{1,B}) \wedge$ $\quad (\neg v_{2,R} \vee \neg v_{2,B}) \wedge$

Step 2: Encoding Constraints

- Each vertex must have exactly one color.
  - Each vertex must have at least one color, and each vertex must have at most one color

How are we going to encode, each vertex must have at least one color:

For vertex $V_1 : v_{1,G} \vee v_{1,R} \vee v_{1,B}$ $\quad V_2 : v_{2,G} \vee v_{2,R} \vee v_{2,B}$ $\quad V_3 : v_{3,G} \vee v_{3,R} \vee v_{3,B}$

How are we going to encode, each vertex must have at most one color:

$V_1 : (\neg v_{1,G} \vee \neg v_{1,R}) \wedge$ $\quad V_2 : (\neg v_{2,G} \vee \neg v_{2,R}) \wedge$ $\quad V_3 : (\neg v_{3,G} \vee \neg v_{3,R}) \wedge$

$(\neg v_{1,G} \vee \neg v_{1,B}) \wedge$ $\quad (\neg v_{2,G} \vee \neg v_{2,B}) \wedge$ $\quad (\neg v_{3,G} \vee \neg v_{3,B}) \wedge$

$(\neg v_{1,R} \vee \neg v_{1,B}) \wedge$ $\quad (\neg v_{2,R} \vee \neg v_{2,B}) \wedge$ $\quad (\neg v_{3,R} \vee \neg v_{3,B}) \wedge$

Step 2: Encoding Constraints

- No two adjacent vertex have the same color.

For V_1 and V_2:

$(\neg v_{1,R} \lor \neg v_{2,R}) \land$

$(\neg v_{1,G} \lor \neg v_{2,G}) \land$

$(\neg v_{1,B} \lor \neg v_{2,B}) \land$

For V_1 and V_3:

$(\neg v_{1,R} \lor \neg v_{3,R}) \land$

$(\neg v_{1,G} \lor \neg v_{3,G}) \land$

$(\neg v_{1,B} \lor \neg v_{3,B}) \land$

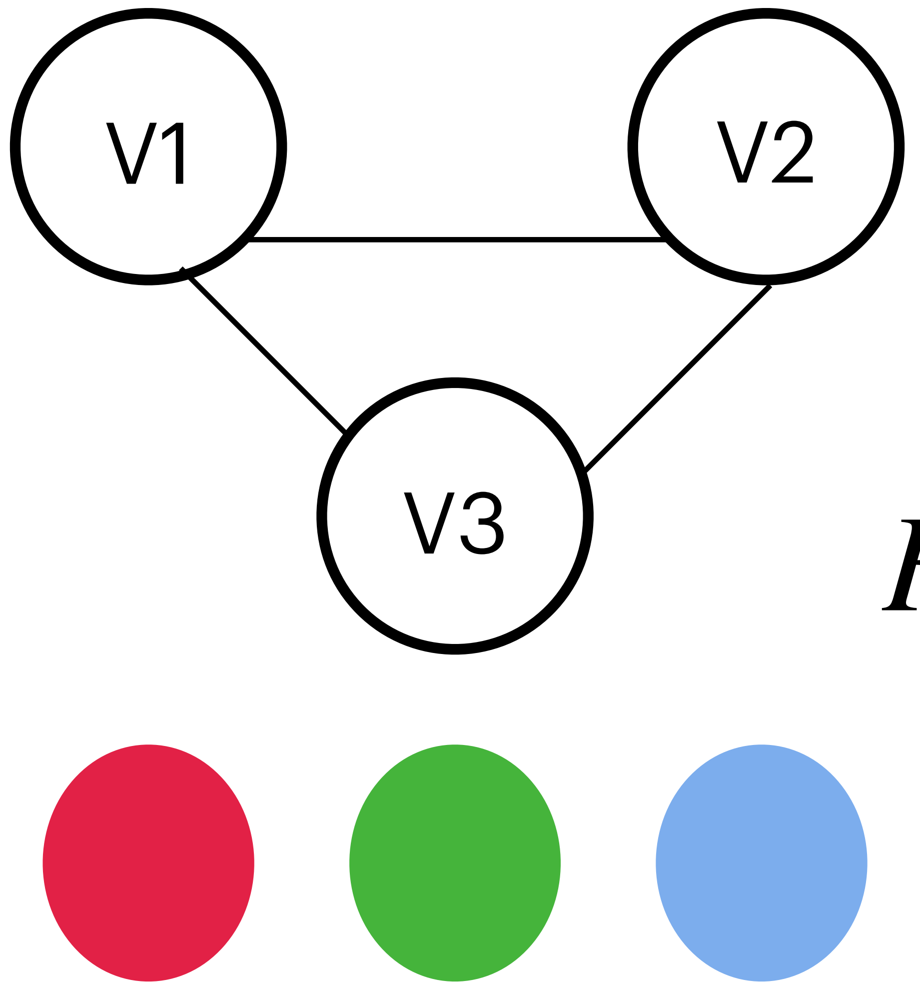
For V_2 and V_3:

$(\neg v_{2,R} \lor \neg v_{3,R}) \land$

$(\neg v_{2,G} \lor \neg v_{3,G}) \land$

$(\neg v_{2,B} \lor \neg v_{3,B})$

Proper Coloring to SAT



$$
\begin{aligned}
F_{CNF} = {} & (v_{1,G} \lor v_{1,R} \lor v_{1,B}) \land (v_{2,G} \lor v_{2,R} \lor v_{2,B}) \land (v_{3,G} \lor v_{3,R} \lor v_{3,B}) \land \\
& (\neg v_{1,G} \lor \neg v_{1,R}) \land (\neg v_{1,G} \lor \neg v_{1,B}) \land (\neg v_{1,R} \lor \neg v_{1,B}) \land \\
& (\neg v_{2,G} \lor \neg v_{2,R}) \land (\neg v_{2,G} \lor \neg v_{2,B}) \land (\neg v_{2,R} \lor \neg v_{2,B}) \land \\
& (\neg v_{3,G} \lor \neg v_{3,R}) \land (\neg v_{3,R} \lor \neg v_{3,B}) \land (\neg v_{3,R} \lor \neg v_{3,B}) \land \\
& (\neg v_{1,R} \lor \neg v_{2,R}) \land (\neg v_{1,G} \lor \neg v_{2,G}) \land (\neg v_{1,B} \lor \neg v_{2,B}) \land \\
& (\neg v_{1,R} \lor \neg v_{3,R}) \land (\neg v_{1,G} \lor \neg v_{3,G}) \land (\neg v_{1,B} \lor \neg v_{3,B}) \land \\
& (\neg v_{2,R} \lor \neg v_{3,R}) \land (\neg v_{2,G} \lor \neg v_{3,G}) \land (\neg v_{2,B} \lor \neg v_{3,B})
\end{aligned}
$$

Boolean Satisfiability (SAT) Simple to State, Rich in Structure

Despite its simplicity, it captures a vast range of real-world problems.

Different Problems $\xrightarrow{\text{Encoding}}$ SAT problem $\rightarrow$ SAT Solvers $\rightarrow$ A satisfying assignment $\xrightarrow{\text{decoding}}$ A solution to original problem

Scheduling
Planning

Graph coloring
Vertex cover

Does there exists an envy free allocation?

Does there exists a fair committee?

....

Problems in NP!

SAT solvers

- Boolean formulas -> SAT Solvers

If formula is SAT, gives an satisfying assignment

Otherwise, UNSAT

DPLL algorithm (Davis -Putnam-Logemann-Loveland 1960)

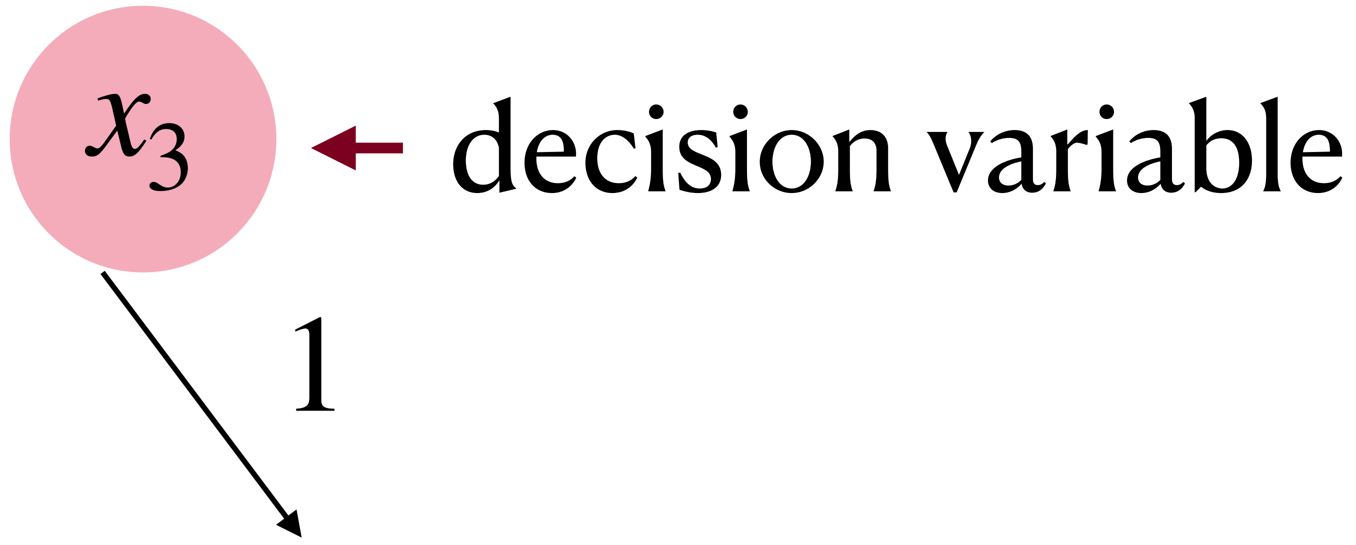
1. Maintains a partial model, initially $\varnothing$
2. Assign unassigned variables either 0 or 1
    1. (Randomly one after the other)
3. Sometime forced to make a decision due to unit clause

DPLL

$$F = (\neg x_1 \vee \neg x_2 \vee \neg x_3) \wedge (\neg x_3 \vee x_2) \wedge (\neg x_3 \vee x_1)$$

DPLL

$$F = (\neg x_1 \vee \neg x_2 \vee \neg x_3) \wedge (\neg x_3 \vee x_2) \wedge (\neg x_3 \vee x_1)$$



Pick a variable, say x_3, and assign it a Boolean value, say 1.

Partial model $m = \{x_3 \mapsto 1\}$

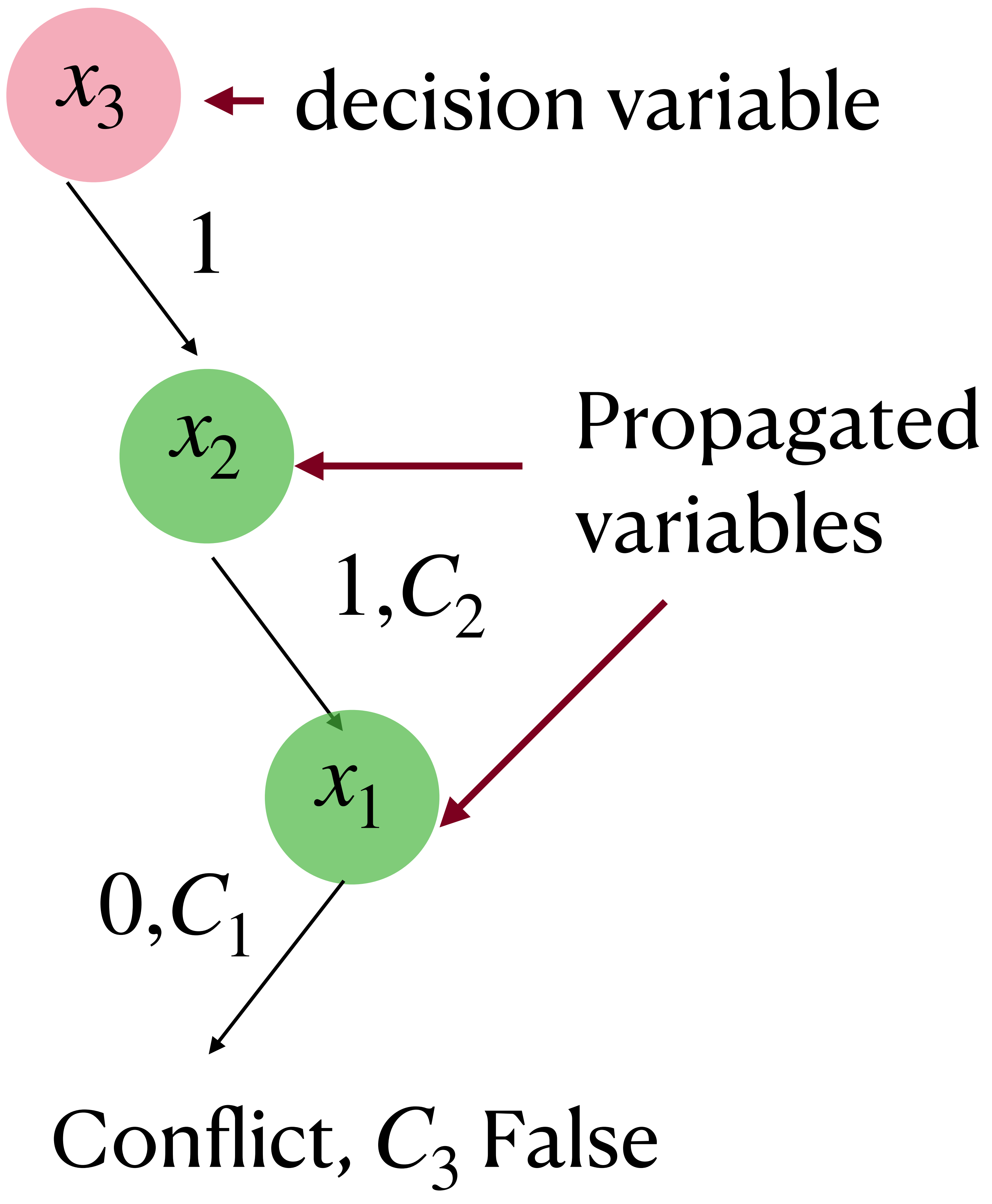
DPLL

$F = (\neg x_1 \vee \neg x_2 \vee \neg x_3) \wedge (\neg x_3 \vee x_2) \wedge (\neg x_3 \vee x_1)$

Pick a variable, say x_3, and assign it a Boolean value, say 1.

Partial model $m = \{x_3 \mapsto 1\}$

$(\neg x_3 \vee x_2) \wedge (\neg x_3 \vee x_1)$ — unit clauses



DPLL

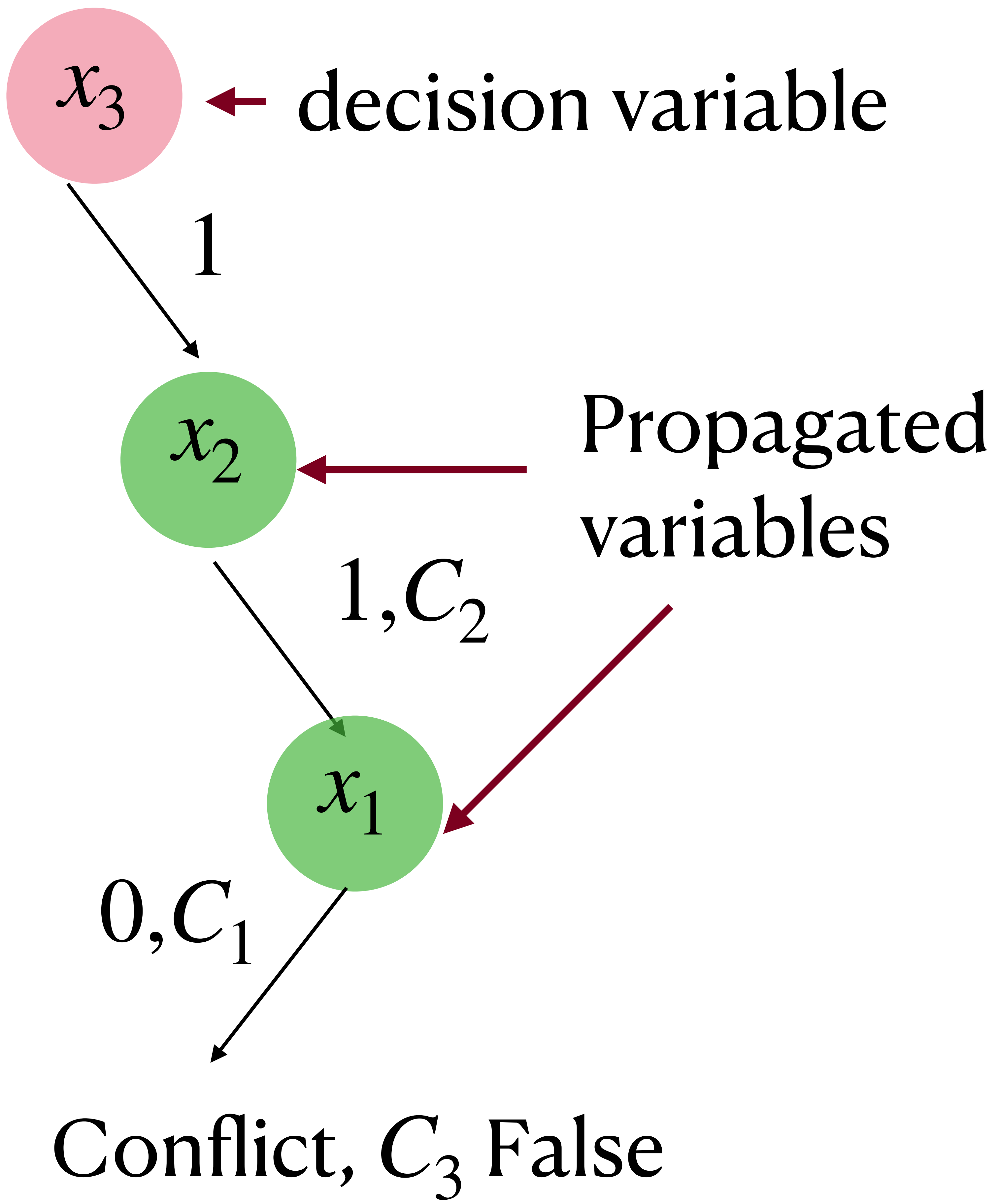
$F = (\neg x_1 \vee \neg x_2 \vee \neg x_3) \wedge (\neg x_3 \vee x_2) \wedge (\neg x_3 \vee x_1)$

Pick a variable, say x_3, and assign it a Boolean value, say 1.

Partial model $m = \{x_3 \mapsto 1\}$

$(\neg x_3 \vee x_2) \wedge (\neg x_3 \vee x_1)$ — unit clauses

What to do if F is False under partial model m?

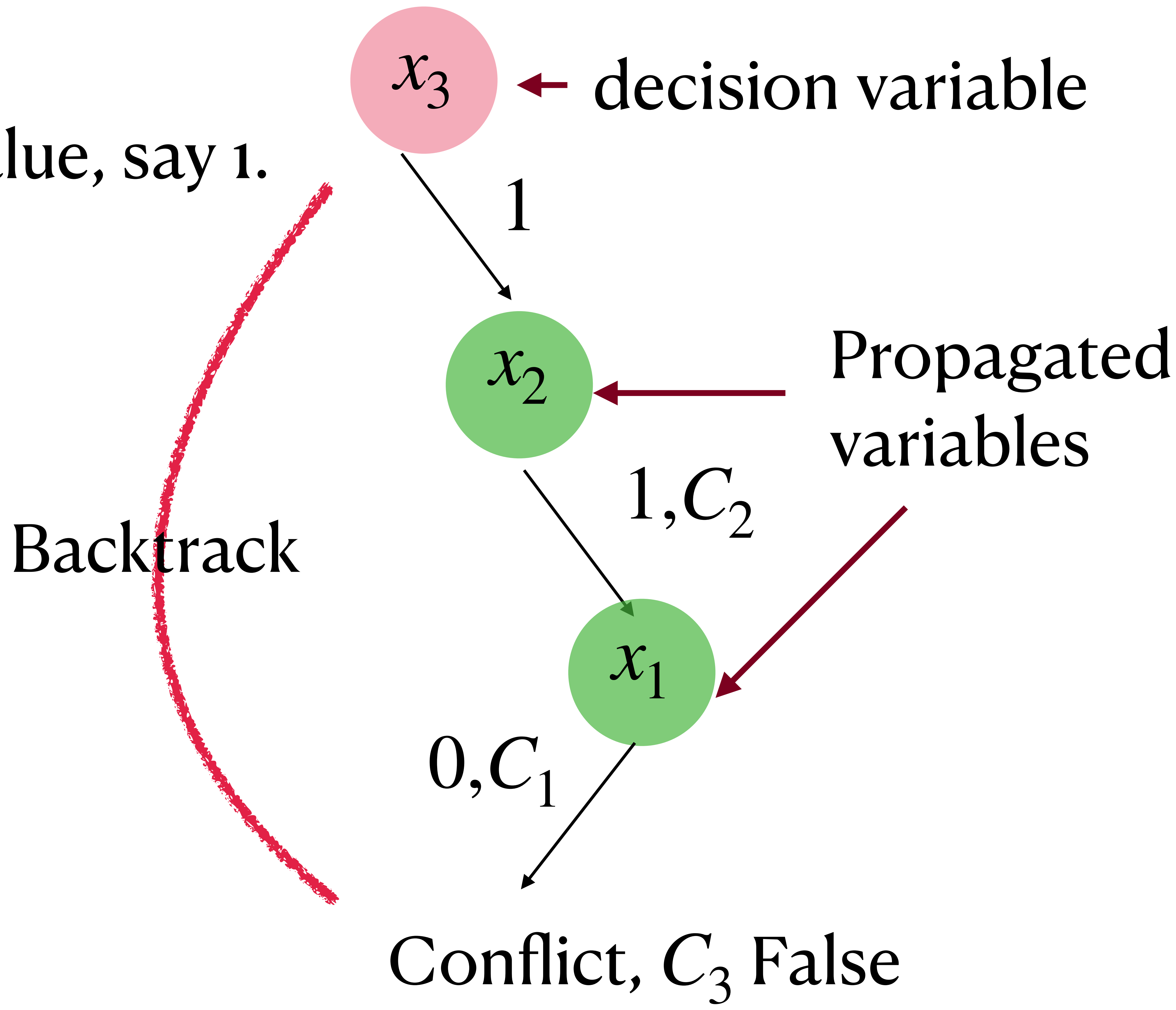


DPLL Backtracking

$F = (\neg x_1 \vee \neg x_2 \vee \neg x_3) \wedge (\neg x_3 \vee x_2) \wedge (\neg x_3 \vee x_1)$

Pick a variable, say x_3, and assign it a Boolean value, say 1.
Partial model $m = \{x_3 \mapsto 1\}$

$(\neg x_3 \vee x_2) \wedge (\neg x_3 \vee x_1)$ — unit clauses



DPLL Backtracking

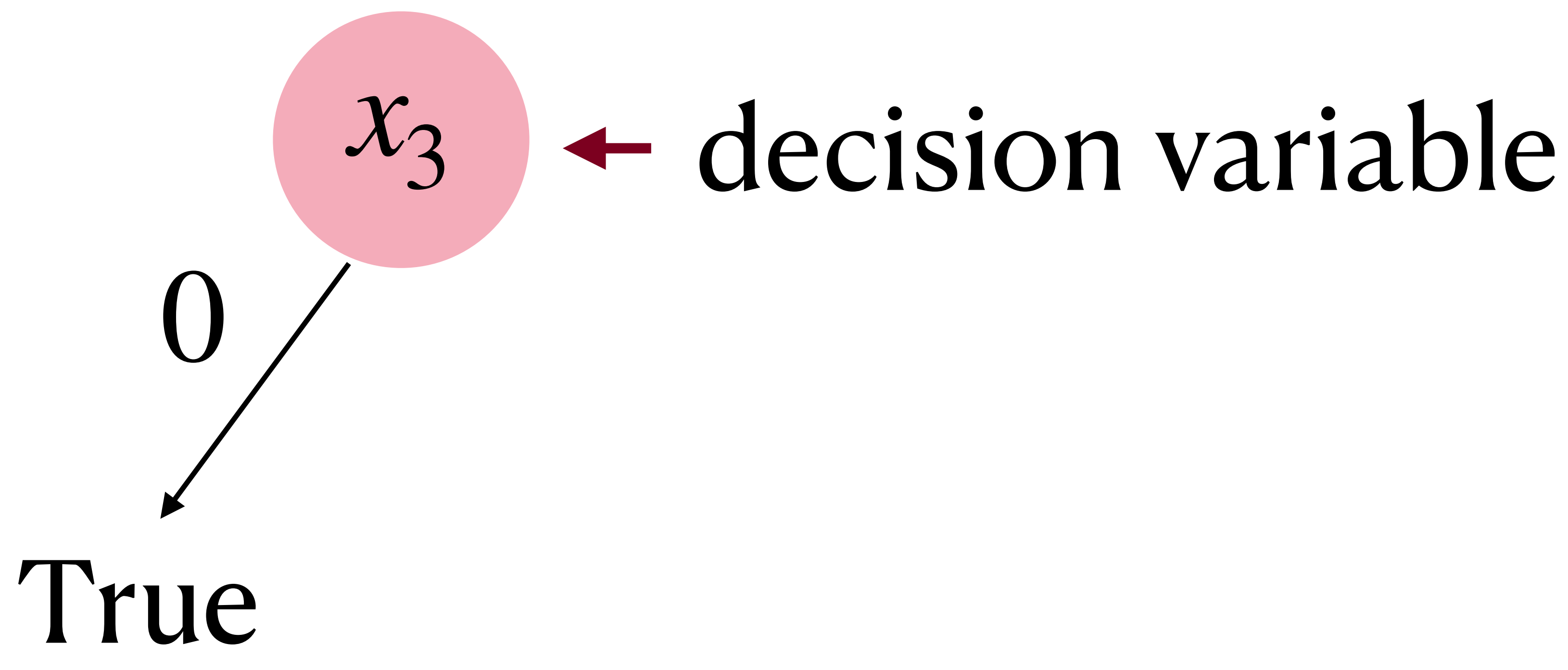
$$F = (\neg x_1 \vee \neg x_2 \vee \neg x_3) \wedge (\neg x_3 \vee x_2) \wedge (\neg x_3 \vee x_1)$$

0

True

DPLL Backtracking

$$F = (\neg x_1 \vee \neg x_2 \vee \neg x_3) \wedge (\neg x_3 \vee x_2) \wedge (\neg x_3 \vee x_1)$$

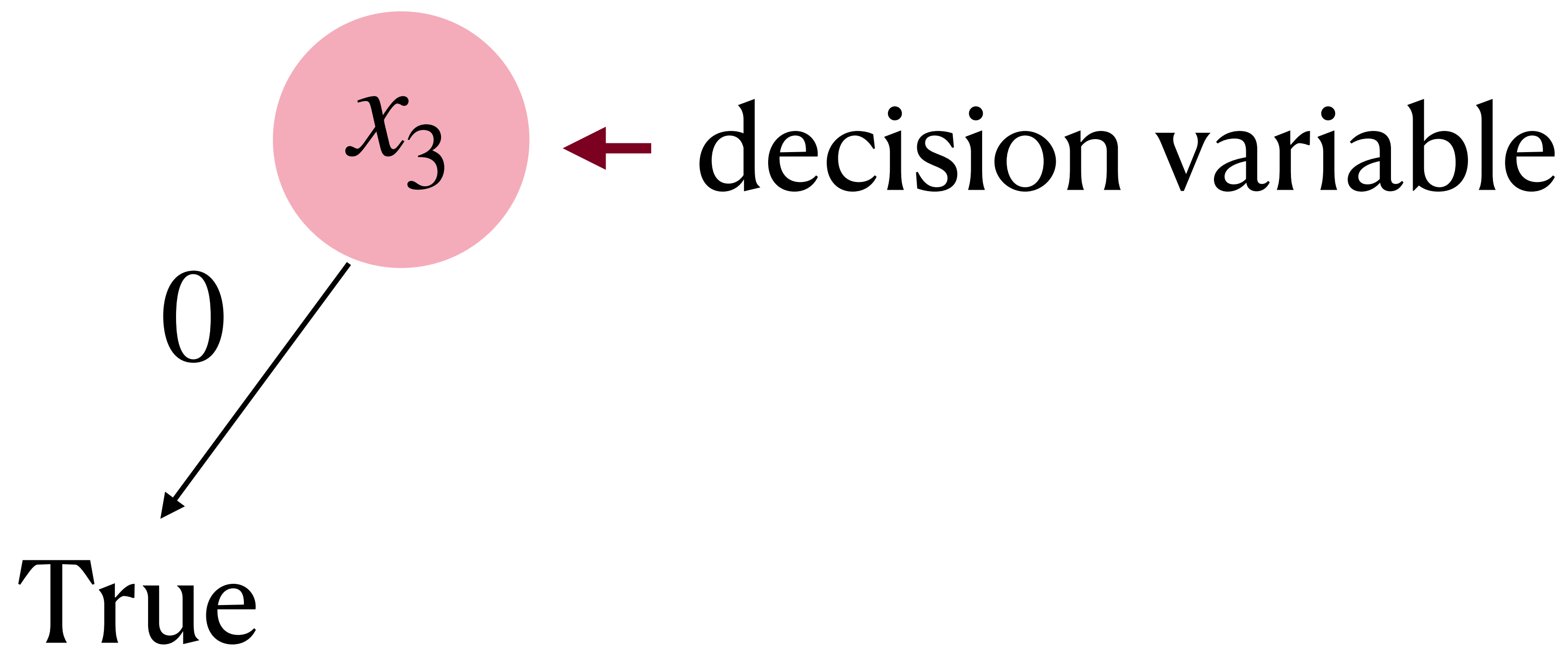


DPLL Backtracking

$$F = (\neg x_1 \vee \neg x_2 \vee \neg x_3) \wedge (\neg x_3 \vee x_2) \wedge (\neg x_3 \vee x_1)$$

Backtrack to last decision, and change the polarity.

Partial model $m = \{x_3 \mapsto 0\}$

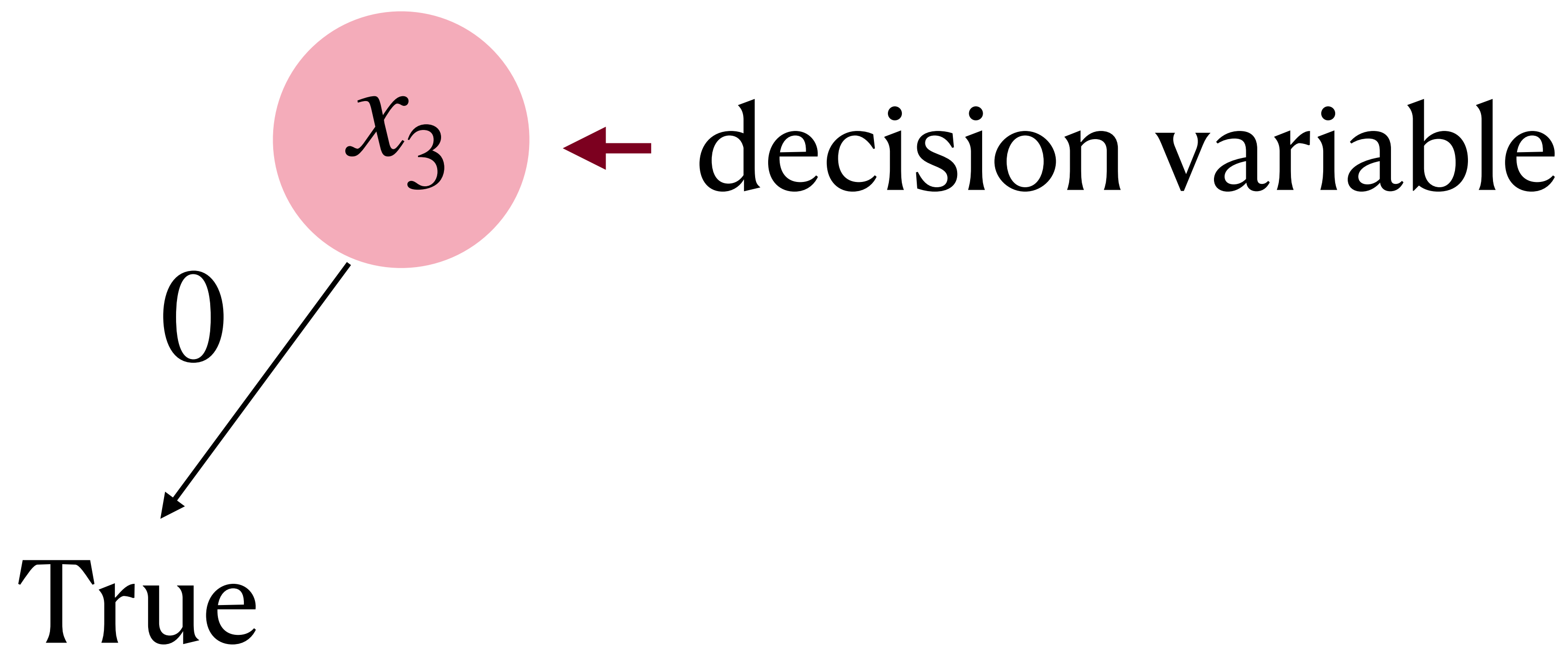


DPLL Backtracking

$$F = (\neg x_1 \lor \neg x_2 \lor \neg x_3) \land (\neg x_3 \lor x_2) \land (\neg x_3 \lor x_1)$$

Backtrack to last decision, and change the polarity.

Partial model $m = \{x_3 \mapsto 0\}$



All clauses are True, hence F is True

DPLL algorithm (Davis -Putnam-Logemann-Loveland 1960)

1. Maintains a partial model, initially $\emptyset$
2. Assign unassigned variables either 0 or 1
    1. (Randomly one after the other)
3. Sometime forced to make a decision due to unit clause

DPLL run consists of

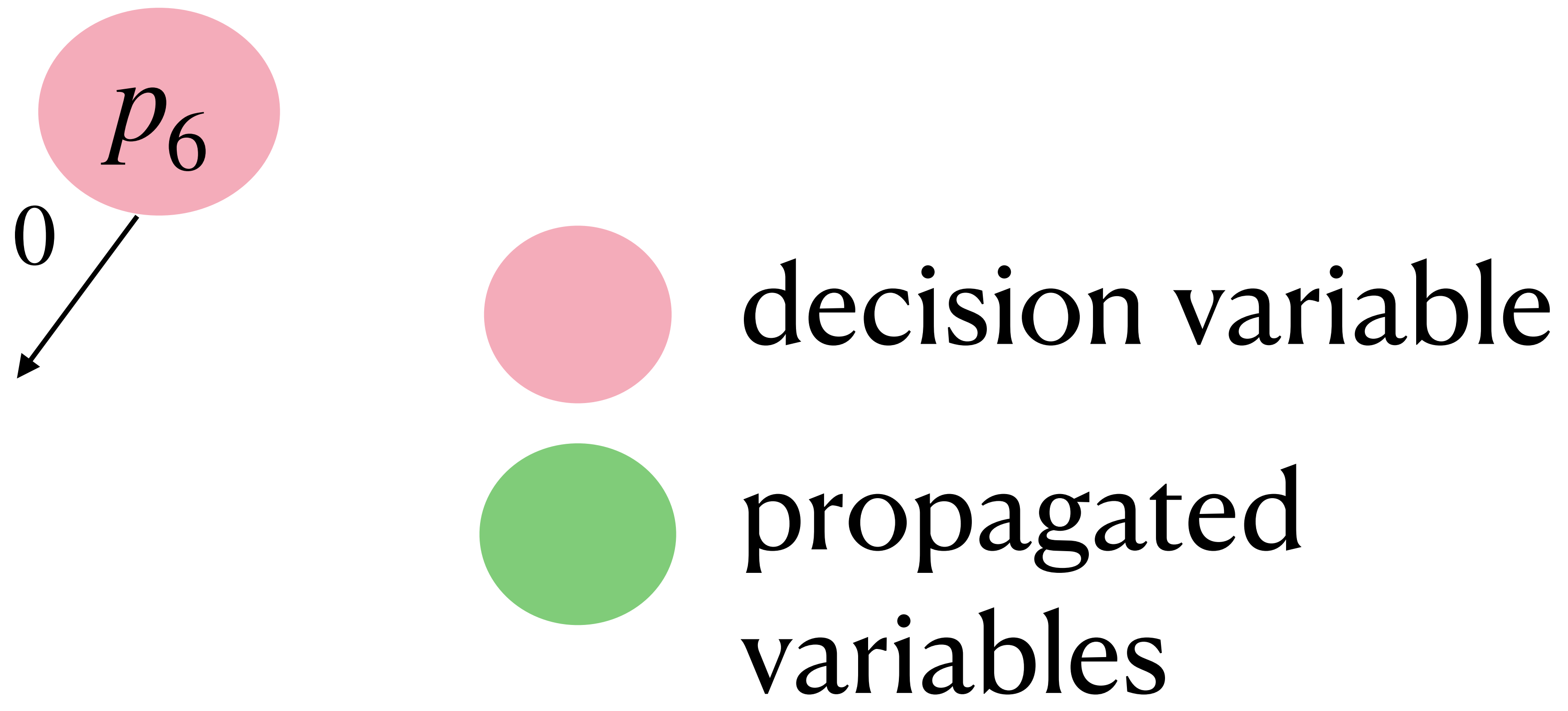
- Decision
- Unit propagation
- Backtracking

$$C_1 = (\neg p_1 \lor p_2)$$
$$C_2 = (\neg p_1 \lor p_3 \lor p_5)$$
$$C_3 = (\neg p_2 \lor p_4)$$
$$C_4 = (\neg p_3 \lor \neg p_4)$$
$$C_5 = (p_1 \lor p_5 \lor \neg p_2)$$
$$C_6 = (p_2 \lor p_3)$$
$$C_7 = (p_2 \lor \neg p_3 \lor p_7)$$
$$C_8 = (p_6 \lor \neg p_5)$$

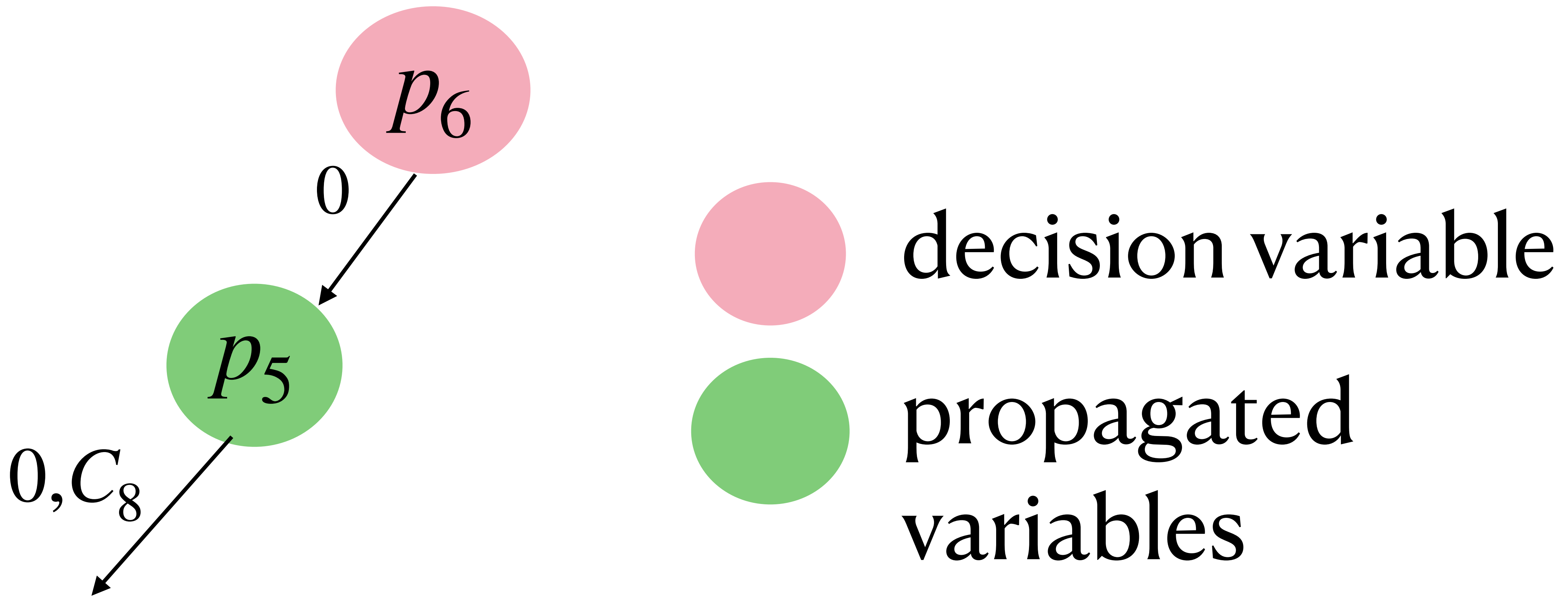
decision variable

propagated variables

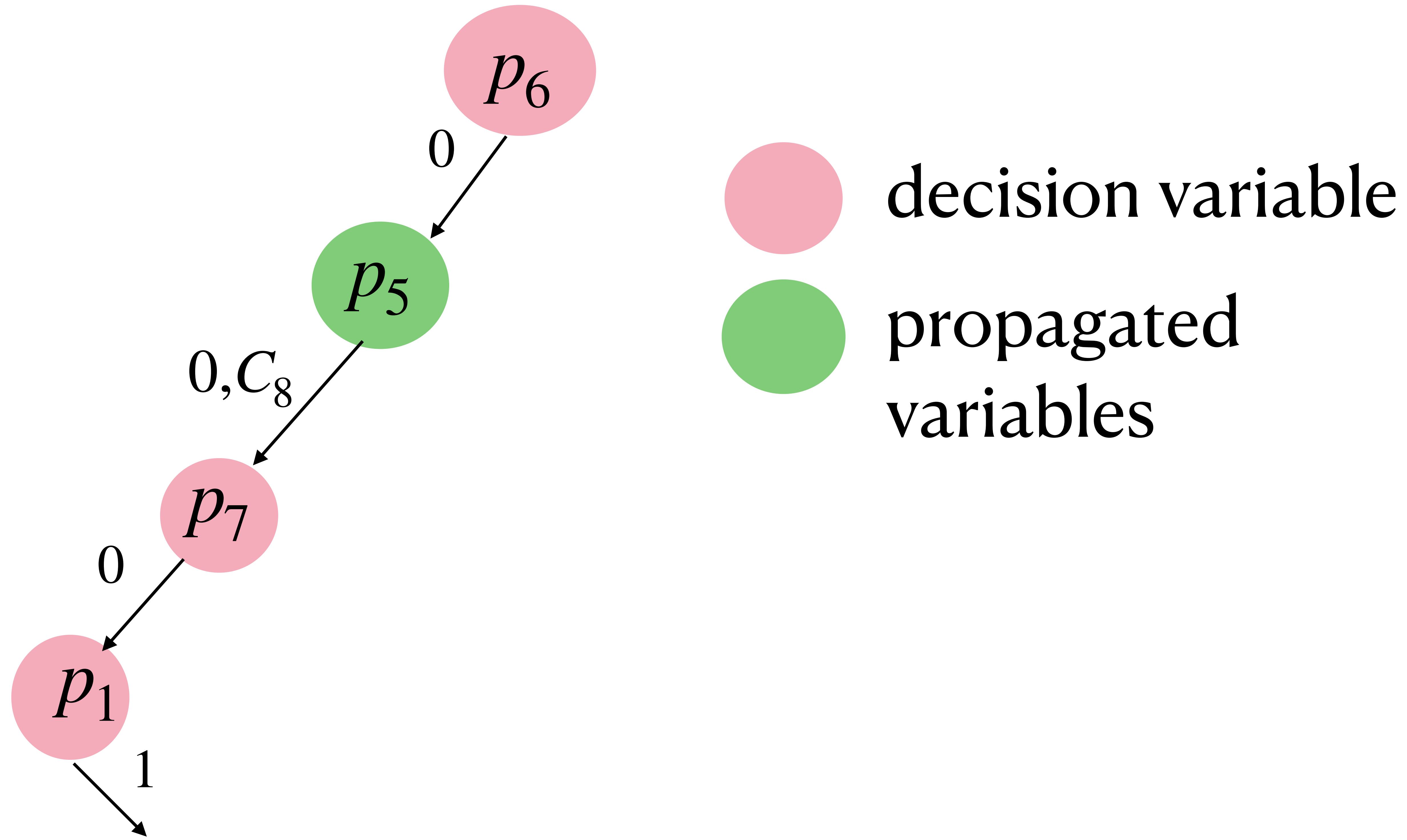
$$C_1 = (\neg p_1 \vee p_2)$$
$$C_2 = (\neg p_1 \vee p_3 \vee p_5)$$
$$C_3 = (\neg p_2 \vee p_4)$$
$$C_4 = (\neg p_3 \vee \neg p_4)$$
$$C_5 = (p_1 \vee p_5 \vee \neg p_2)$$
$$C_6 = (p_2 \vee p_3)$$
$$C_7 = (p_2 \vee \neg p_3 \vee p_7)$$
$$C_8 = (p_6 \vee \neg p_5)$$



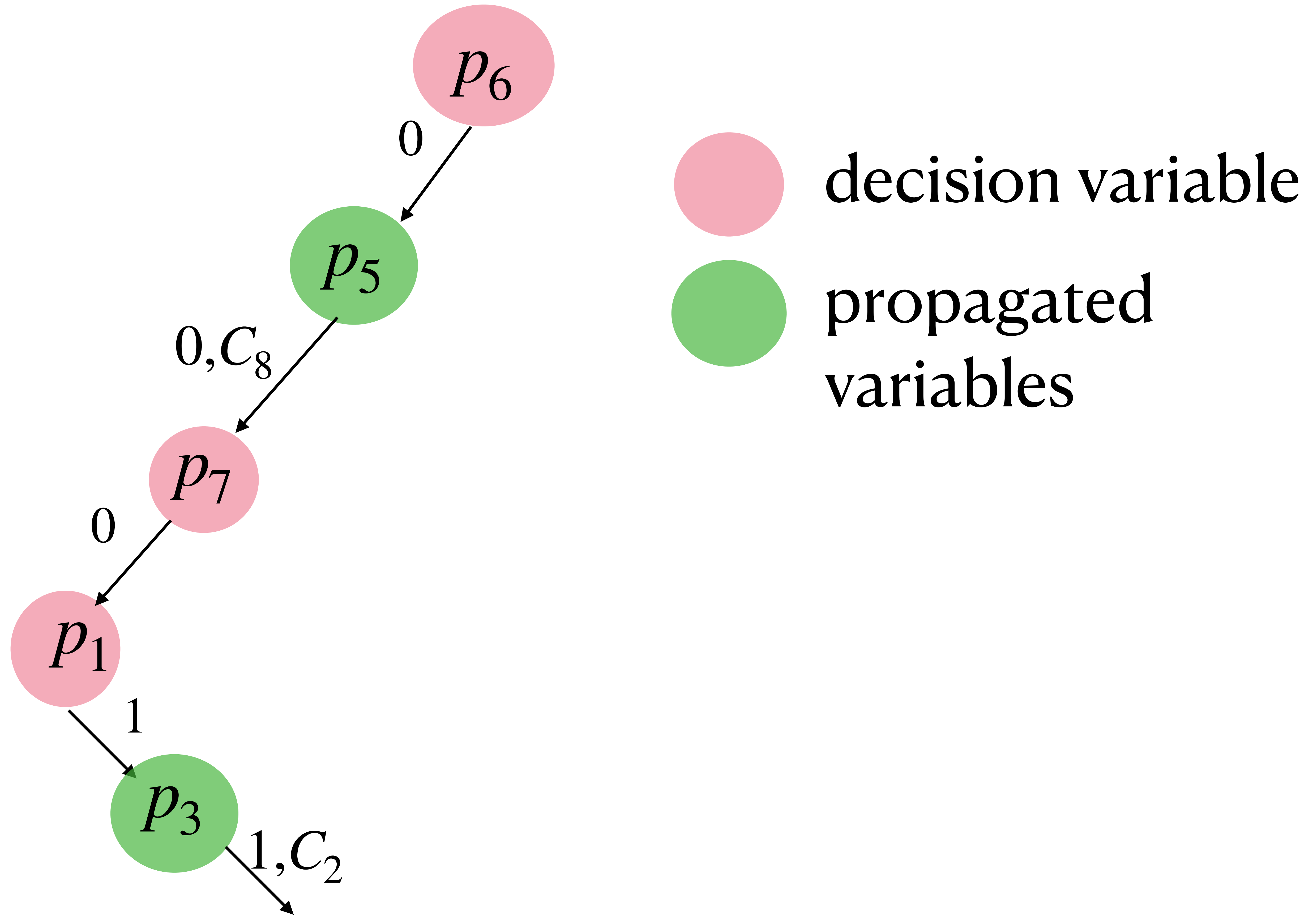
$$C_1 = (\neg p_1 \vee p_2)$$
$$C_2 = (\neg p_1 \vee p_3 \vee p_5)$$
$$C_3 = (\neg p_2 \vee p_4)$$
$$C_4 = (\neg p_3 \vee \neg p_4)$$
$$C_5 = (p_1 \vee p_5 \vee \neg p_2)$$
$$C_6 = (p_2 \vee p_3)$$
$$C_7 = (p_2 \vee \neg p_3 \vee p_7)$$
$$C_8 = (p_6 \vee \neg p_5)$$



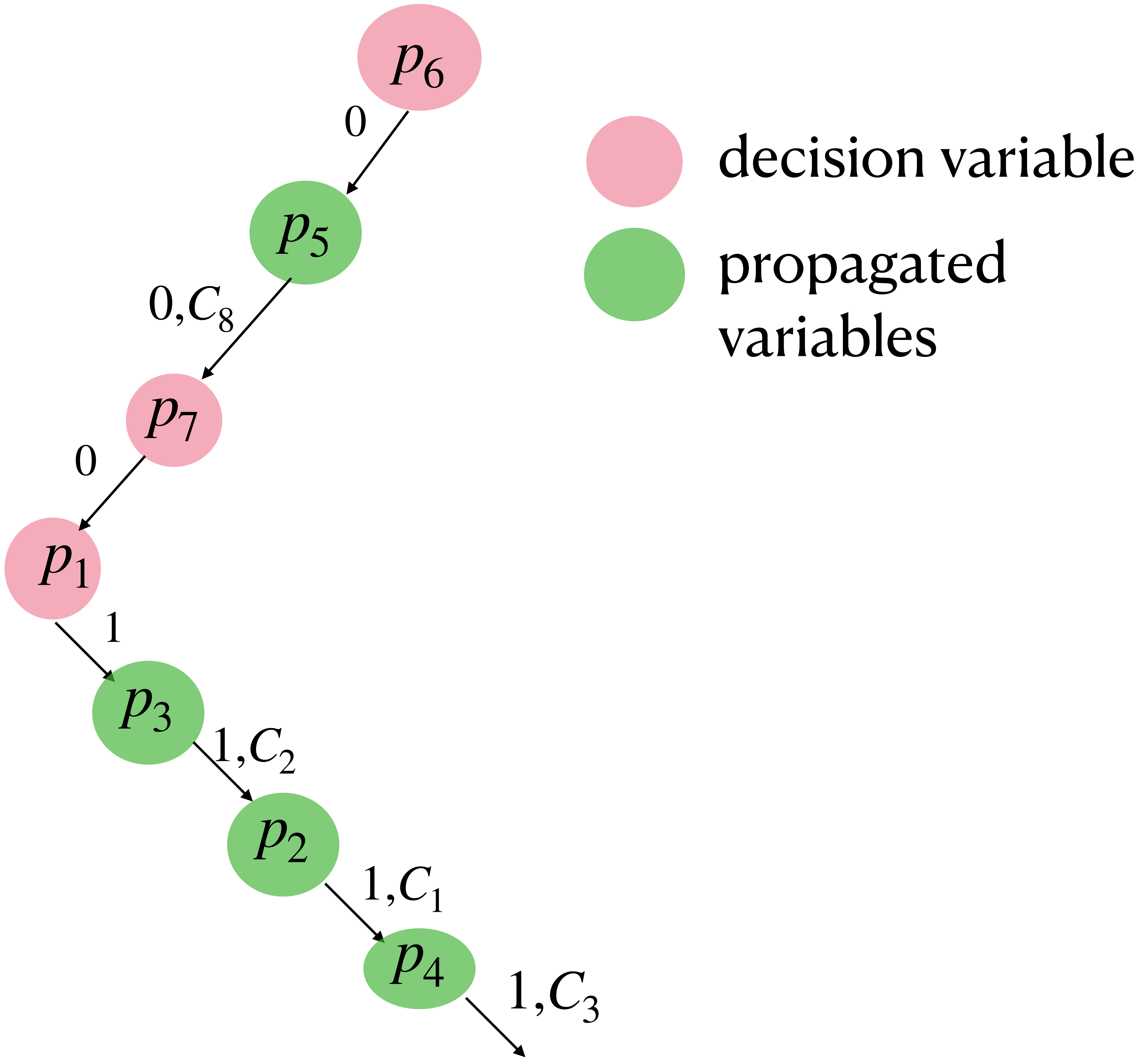
$C_1 = (\neg p_1 \vee p_2)$
$C_2 = (\neg p_1 \vee p_3 \vee p_5)$
$C_3 = (\neg p_2 \vee p_4)$
$C_4 = (\neg p_3 \vee \neg p_4)$
$C_5 = (p_1 \vee p_5 \vee \neg p_2)$
$C_6 = (p_2 \vee p_3)$
$C_7 = (p_2 \vee \neg p_3 \vee p_7)$
$C_8 = (p_6 \vee \neg p_5)$



$C_1 = (\neg p_1 \vee p_2)$
$C_2 = (\neg p_1 \vee p_3 \vee p_5)$
$C_3 = (\neg p_2 \vee p_4)$
$C_4 = (\neg p_3 \vee \neg p_4)$
$C_5 = (p_1 \vee p_5 \vee \neg p_2)$
$C_6 = (p_2 \vee p_3)$
$C_7 = (p_2 \vee \neg p_3 \vee p_7)$
$C_8 = (p_6 \vee \neg p_5)$



$$C_1 = (\neg p_1 \vee p_2)$$
$$C_2 = (\neg p_1 \vee p_3 \vee p_5)$$
$$C_3 = (\neg p_2 \vee p_4)$$
$$C_4 = (\neg p_3 \vee \neg p_4)$$
$$C_5 = (p_1 \vee p_5 \vee \neg p_2)$$
$$C_6 = (p_2 \vee p_3)$$
$$C_7 = (p_2 \vee \neg p_3 \vee p_7)$$
$$C_8 = (p_6 \vee \neg p_5)$$



$C_1 = (\neg p_1 \vee p_2)$

$C_2 = (\neg p_1 \vee p_3 \vee p_5)$

$C_3 = (\neg p_2 \vee p_4)$

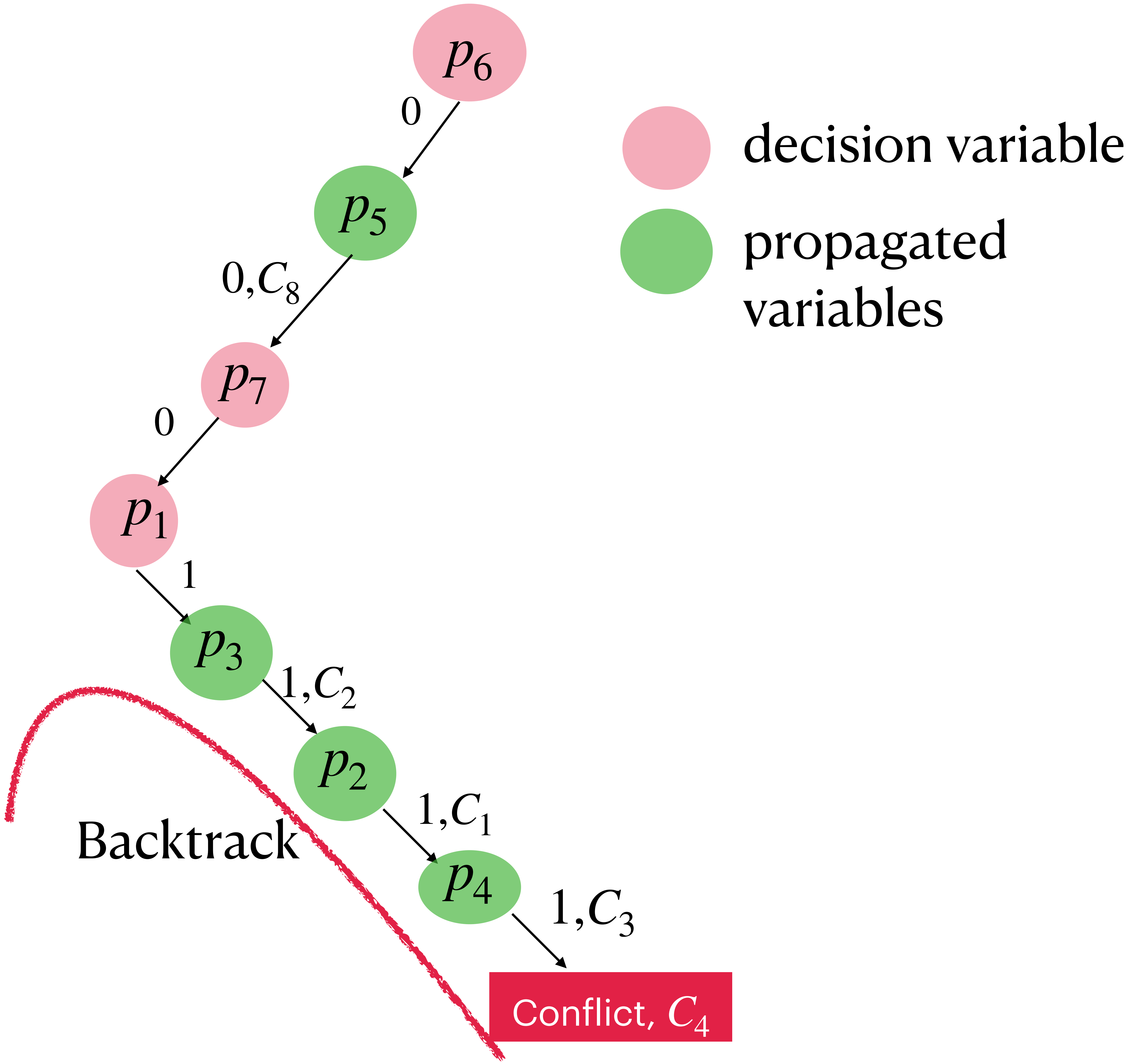
$C_4 = (\neg p_3 \vee \neg p_4)$

$C_5 = (p_1 \vee p_5 \vee \neg p_2)$

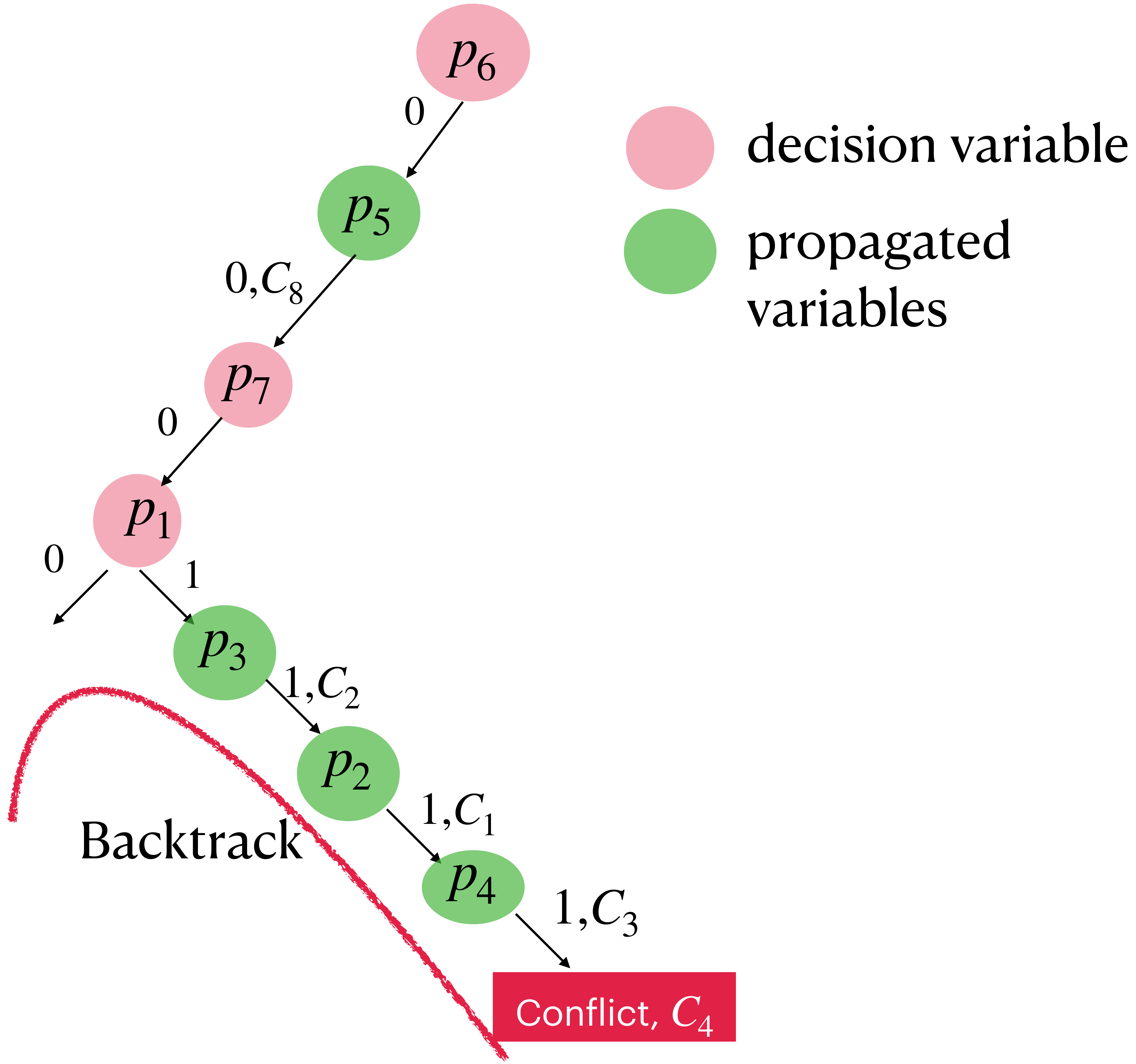
$C_6 = (p_2 \vee p_3)$

$C_7 = (p_2 \vee \neg p_3 \vee p_7)$

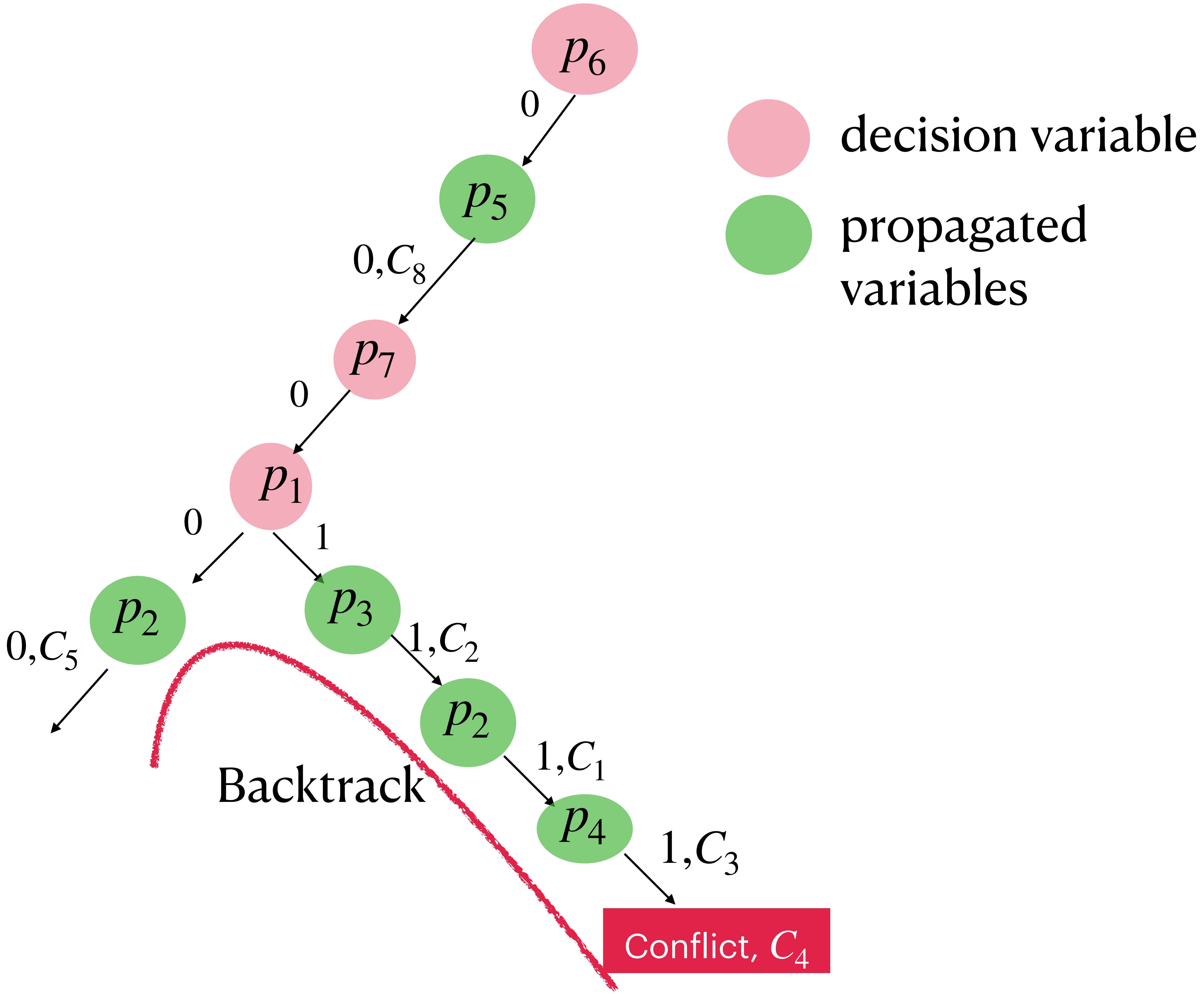
$C_8 = (p_6 \vee \neg p_5)$



$$C_1 = (\neg p_1 \lor p_2)$$
$$C_2 = (\neg p_1 \lor p_3 \lor p_5)$$
$$C_3 = (\neg p_2 \lor p_4)$$
$$C_4 = (\neg p_3 \lor \neg p_4)$$
$$C_5 = (p_1 \lor p_5 \lor \neg p_2)$$
$$C_6 = (p_2 \lor p_3)$$
$$C_7 = (p_2 \lor \neg p_3 \lor p_7)$$
$$C_8 = (p_6 \lor \neg p_5)$$



$$C_1 = (\neg p_1 \vee p_2)$$
$$C_2 = (\neg p_1 \vee p_3 \vee p_5)$$
$$C_3 = (\neg p_2 \vee p_4)$$
$$C_4 = (\neg p_3 \vee \neg p_4)$$
$$C_5 = (p_1 \vee p_5 \vee \neg p_2)$$
$$C_6 = (p_2 \vee p_3)$$
$$C_7 = (p_2 \vee \neg p_3 \vee p_7)$$
$$C_8 = (p_6 \vee \neg p_5)$$



$C_1 = (\neg p_1 \lor p_2)$

$C_2 = (\neg p_1 \lor p_3 \lor p_5)$

$C_3 = (\neg p_2 \lor p_4)$

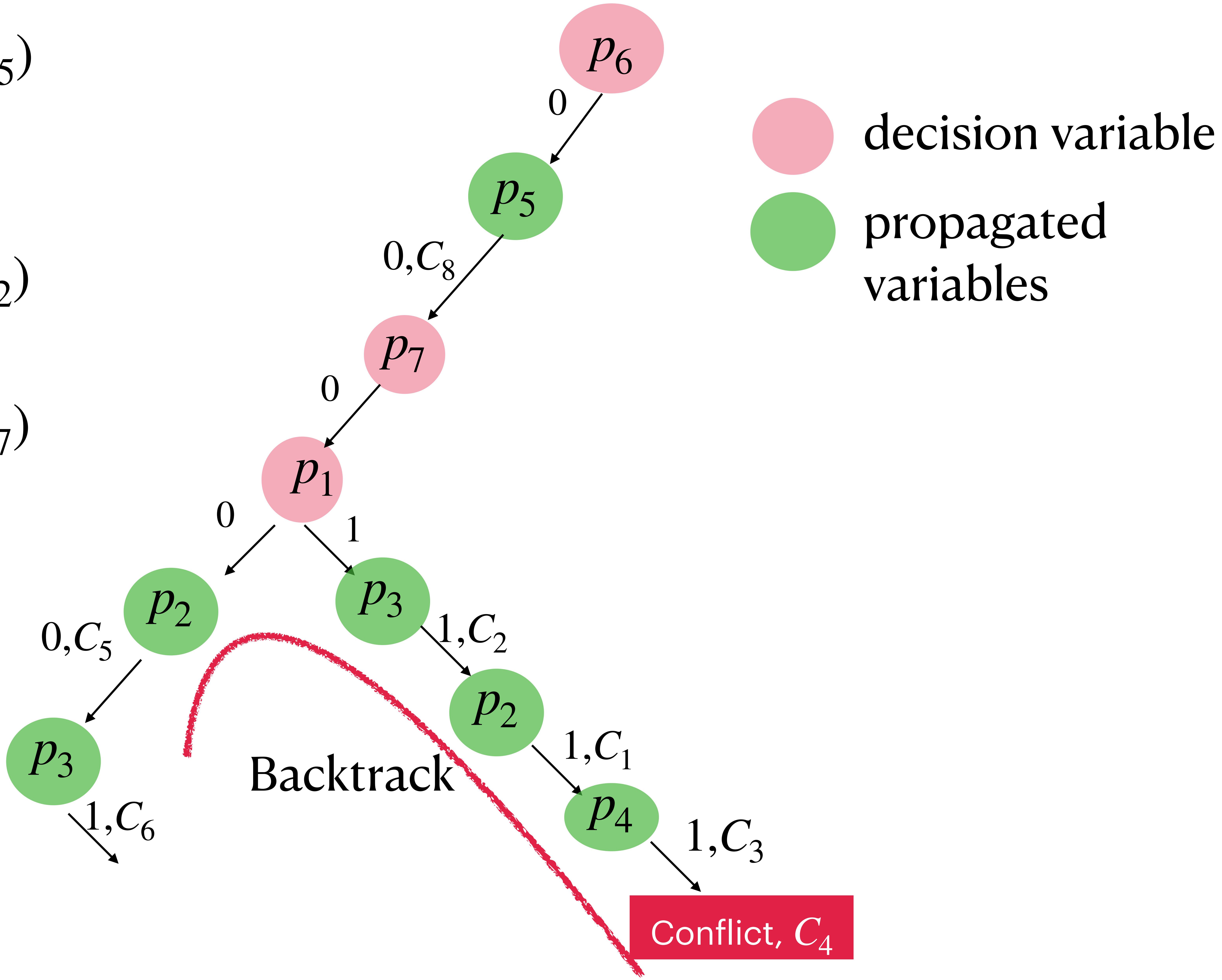
$C_4 = (\neg p_3 \lor \neg p_4)$

$C_5 = (p_1 \lor p_5 \lor \neg p_2)$

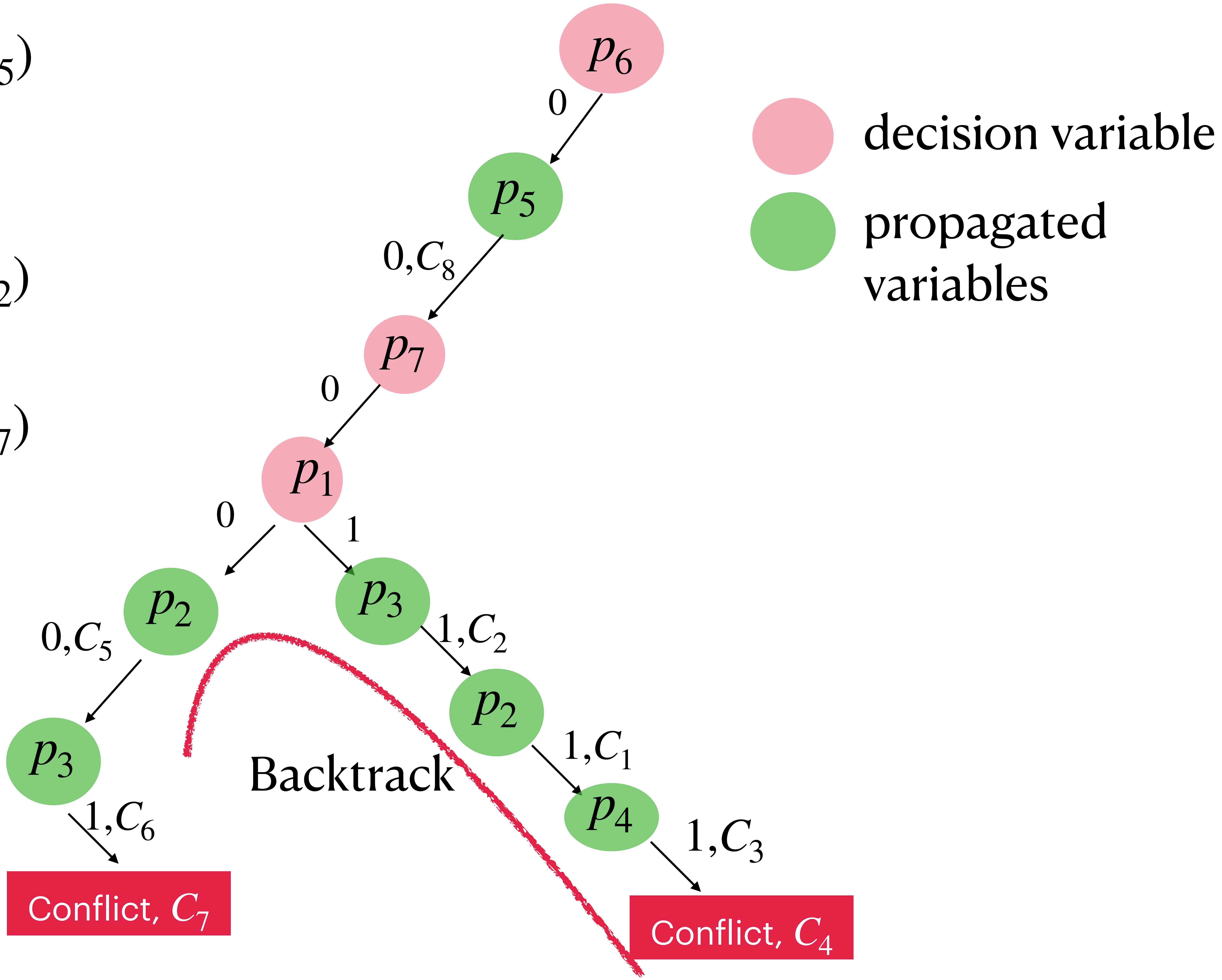
$C_6 = (p_2 \lor p_3)$

$C_7 = (p_2 \lor \neg p_3 \lor p_7)$

$C_8 = (p_6 \lor \neg p_5)$



$C_1 = (\neg p_1 \vee p_2)$
$C_2 = (\neg p_1 \vee p_3 \vee p_5)$
$C_3 = (\neg p_2 \vee p_4)$
$C_4 = (\neg p_3 \vee \neg p_4)$
$C_5 = (p_1 \vee p_5 \vee \neg p_2)$
$C_6 = (p_2 \vee p_3)$
$C_7 = (p_2 \vee \neg p_3 \vee p_7)$
$C_8 = (p_6 \vee \neg p_5)$



$C_1 = (\neg p_1 \vee p_2)$

$C_2 = (\neg p_1 \vee p_3 \vee p_5)$

$C_3 = (\neg p_2 \vee p_4)$

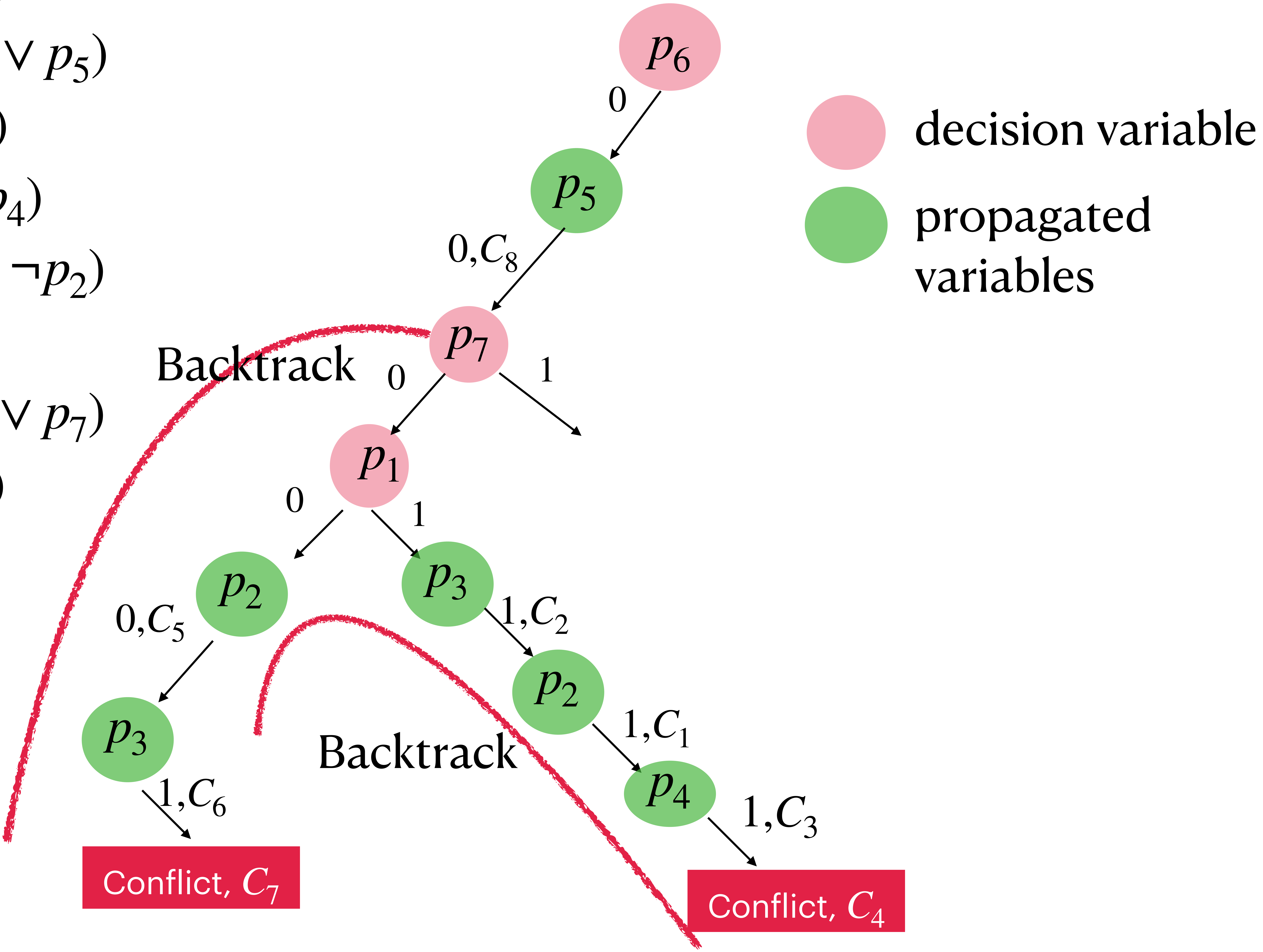
$C_4 = (\neg p_3 \vee \neg p_4)$

$C_5 = (p_1 \vee p_5 \vee \neg p_2)$

$C_6 = (p_2 \vee p_3)$

$C_7 = (p_2 \vee \neg p_3 \vee p_7)$

$C_8 = (p_6 \vee \neg p_5)$



DPLL

Complete and Sound algorithm.

Still the basis of SAT solver

zChaff Solver — efficient implementation of DPLL (2001)

Won test of time award at CAV.

DPLL

Complete and Sound algorithm.

Still the basis of SAT solver

zChaff Solver — efficient implementation of DPLL (2001)

Won test of time award at CAV.

An optimization of DPLL:

As we decide and propagate, we can observe the run, and avoid unnecessary backtracking.

DPLL

Complete and Sound algorithm.

Still the basis of SAT solver

zChaff Solver — efficient implementation of DPLL (2001)

Won test of time award at CAV.

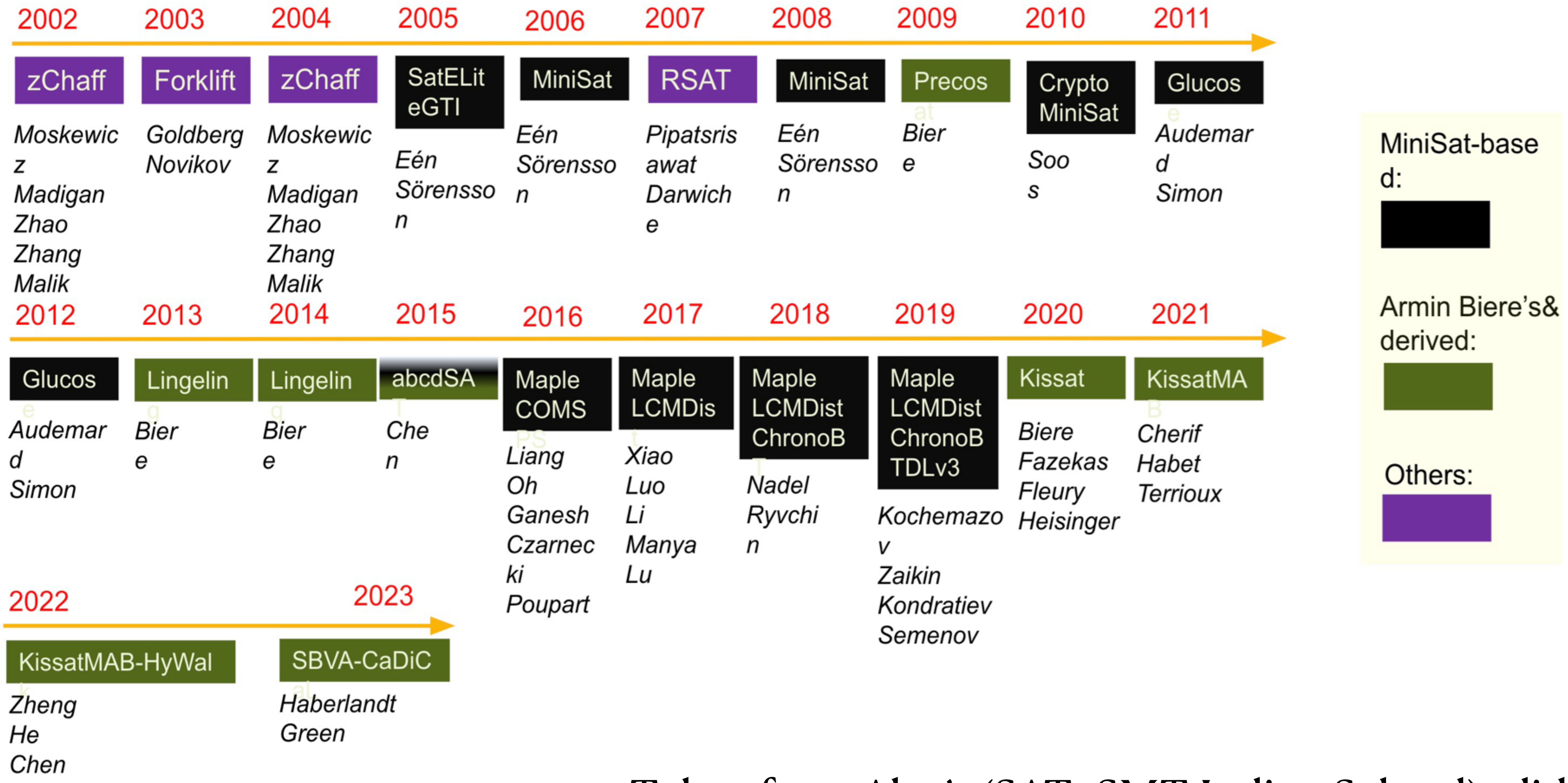
An optimization of DPLL:

As we decide and propagate, we can observe the run, and avoid unnecessary backtracking.

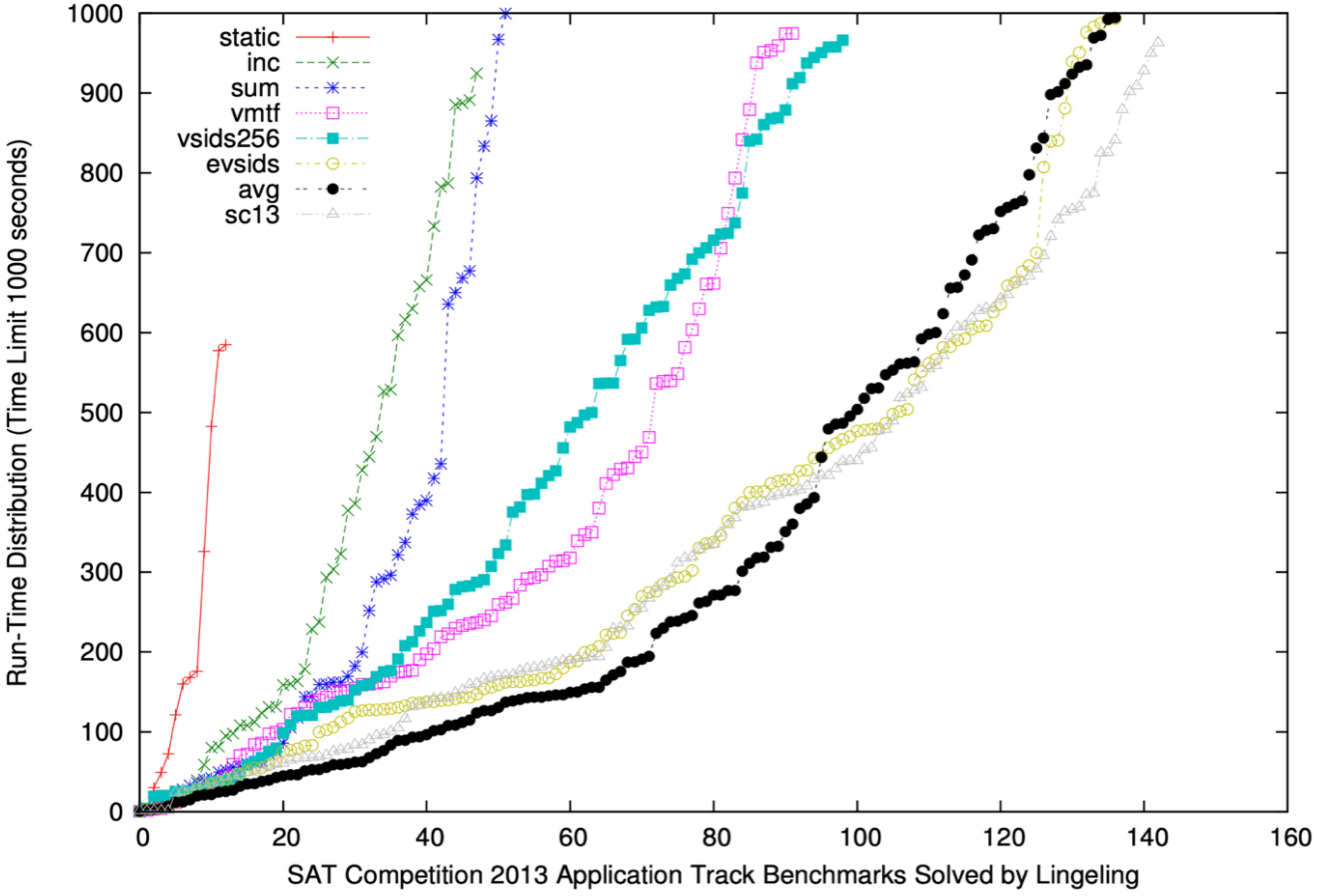
CDCL— Construct a data structure to avoid unnecessary backtracking.

Heuristics: which variables to pick, what value to assign?

SAT Competition & Race Winners (CNF & Appl. & Seq. & Non-incr. & All-inst.)



Taken from Alex's (SAT+SMT Indian School) slides.


Run-Time Distribution (Time Limit 1000 seconds)
SAT Competition 2013 Application Track Benchmarks Solved by Lingeling
static
inc
sum
vmtf
vsids256
evsids
avg
sc13
0
100
200
300
400
500
600
700
800
900
1000
0
20
40
60
80
100
120
140
160