COL:750/7250

Foundations of Automatic Verification

Instructor: Priyanka Golia

Course Webpage

https://priyanka-golia.github.io/teaching/COL-750-COL7250/index.html

Announcements

- Assignment 1 has been released. The deadline is 31st January, end of day (EoD).
- Please reach out to the TAs (Raj) if you have any questions related to the assignment.
- A Piazza class has been created. Please post all your doubts there.

DP algorithm for SAT Solving (Martin Davis - Hilary Putnam 1960)

1. Start with F_{CNF}
2. For every clause C in F_{CNF} that either contains both l and $\neg l$ or has pure literal do:
    1. $F_{CNF} \leftarrow remove\_from\_formula(C, F_{CNF})$
3. $F_{CNF} \leftarrow$ UnitPropagation(F_{CNF})
4. If F_{CNF} is empty
    1. Return SAT
5. If F_{CNF} has empty clause then
    1. Return UNSAT
6. Pick a literal l that occurs with both polarities in F_{CNF} .
    1. $F_{CNF} \leftarrow Resolution(C, l, F_{CNF})$
7. For every clause C that contains l or $\neg l$ do :
    1. $F_{CNF} \leftarrow remove\_from\_formula(C, F_{CNF})$

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

Complete and Sound algorithm & takes linear space in worst case.

Still the basis of SAT solver

zChaff Solver — efficient implementation of DPLL.

Won test of time award at CAV 2001.

Notations

Partial Model: subset of elements of Vars(F) maps to $\{0,1\}$

Under partial model m,

A literal l is True if $m(l)$= 1

A literal l is False if $m(l)$ = 0

Otherwise:

l is unassigned.

Example: $F = (x_1 \vee x_2 \vee \neg x_3); m = \{x_1 \mapsto 1, x_3 \mapsto 1\}$

x_1 is True, x_2 is unassigned, x_3 is False.

Notations

Partial Model: subset of elements of Vars(F) maps to $\{0,1\}$

Under partial model m,

Clause C is True if there is a $l \in C$, such that l is True.

Clause C is False if for each literal $l \in C$, l is False

Otherwise:

C is unassigned.

Example: $m = \{x_1 \mapsto 1, x_3 \mapsto 1\}$

$C = (x_1 \vee x_2 \vee x_3)$ — True

$C = (\neg x_1 \vee x_2 \vee \neg x_3)$ — Unassigned

$C = (\neg x_1 \vee \neg x_3)$ — False

Notations

Partial Model: subset of elements of Vars(F) maps to $\{0,1\}$

Under partial model m,

F_{CNF} is True if for each $C \in F_{CNF}$, C is True.

F_{CNF} is False if there is a $C \in F_{CNF}$ such that C is False

Otherwise:

F_{CNF} is unassigned.

Unit Clause (updated): C is a unit clause under partial model m if there is exactly one literal l in C which is unassigned, and rest all literals of C are False.

Example: $C = (x_1 \vee \neg x_3 \vee \neg x_2); m = \{x_1 \mapsto 0, x_2 \mapsto 1\}$ C is unit clause under m.

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 algorithm (Davis -Putnam-Logemann-Loveland 1960)

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

Initially m is $\varnothing$

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

$C_1 : (x_1 \vee \neg x_2)$ unassigned.

$C_2 : (\neg x_1 \vee x_2 \vee \neg x_3)$ unassigned.

Pick another variable, say x_1, and assign it a Boolean value, say 0.

Partial model $m = \{x_1 \mapsto 0, x_3 \mapsto 1\}$

$C_1 : (x_1 \vee \neg x_2)$ Unit clause, forced decision $(x_2 \mapsto 0)$

$C_2 : (\neg x_1 \vee x_2 \vee \neg x_3)$ True.

$m = \{x_1 \mapsto 0, x_2 \mapsto 0, x_3 \mapsto 1\}$ and $m \vDash F$

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

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

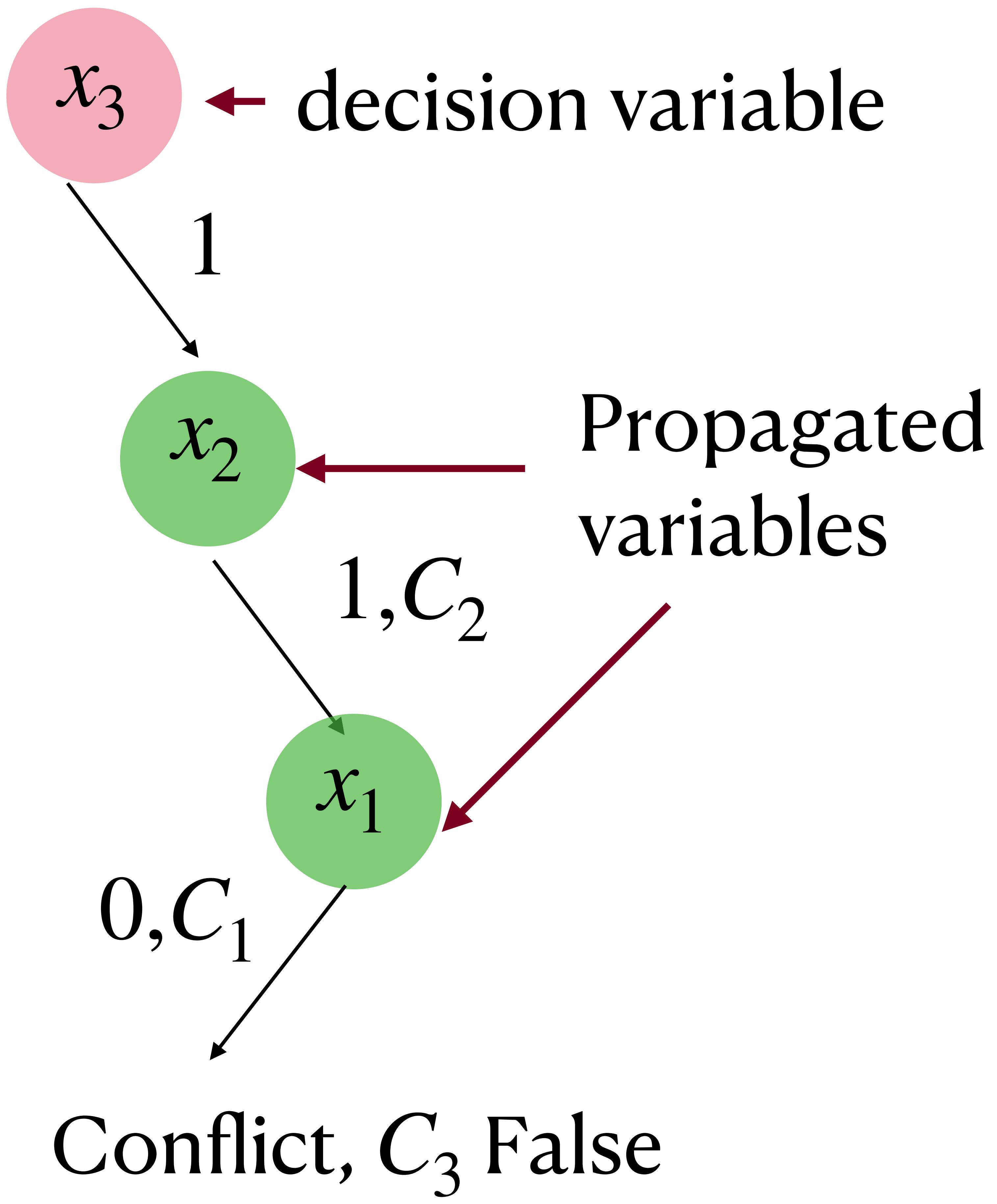
Backtracking

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

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



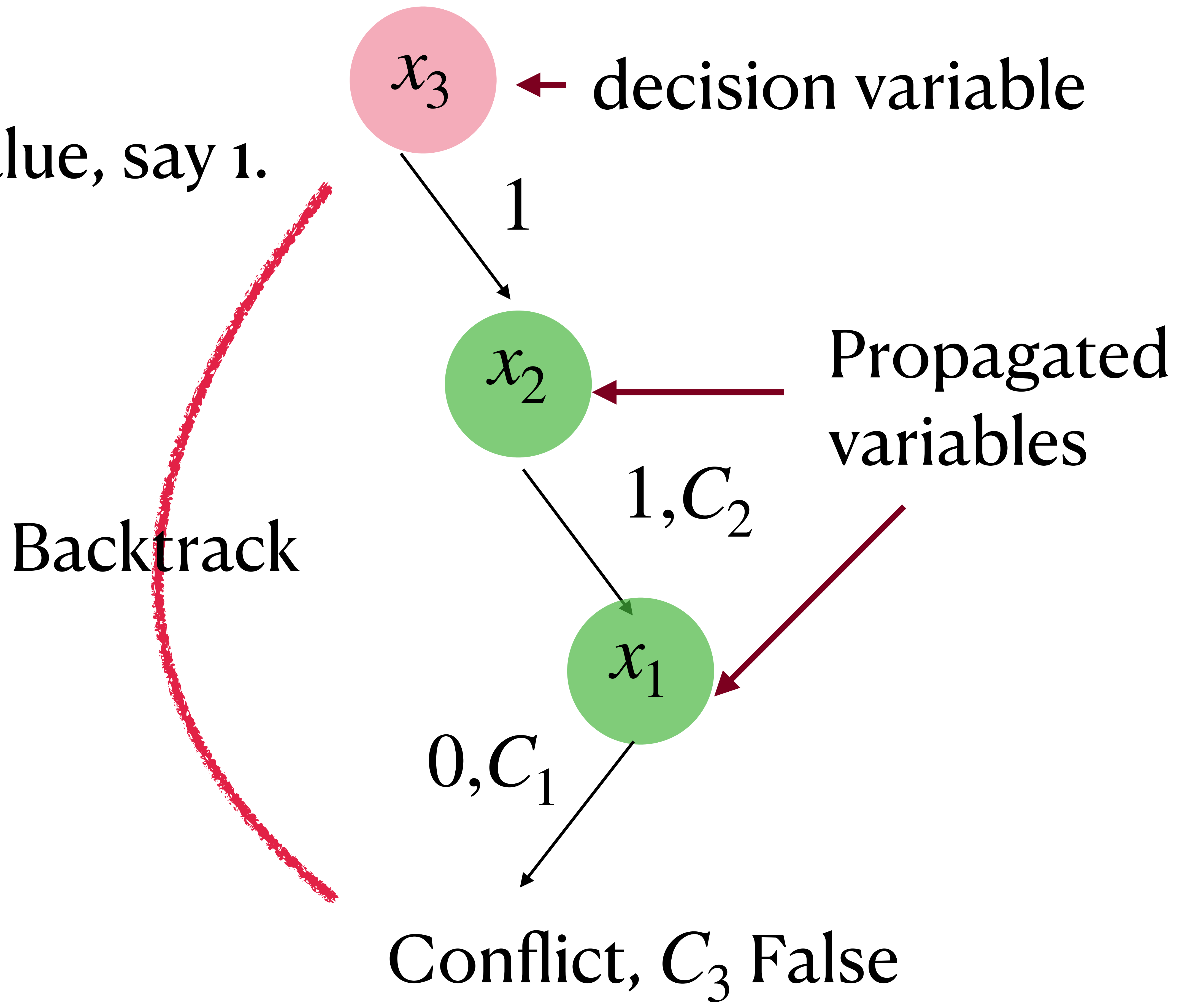
Backtracking

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

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

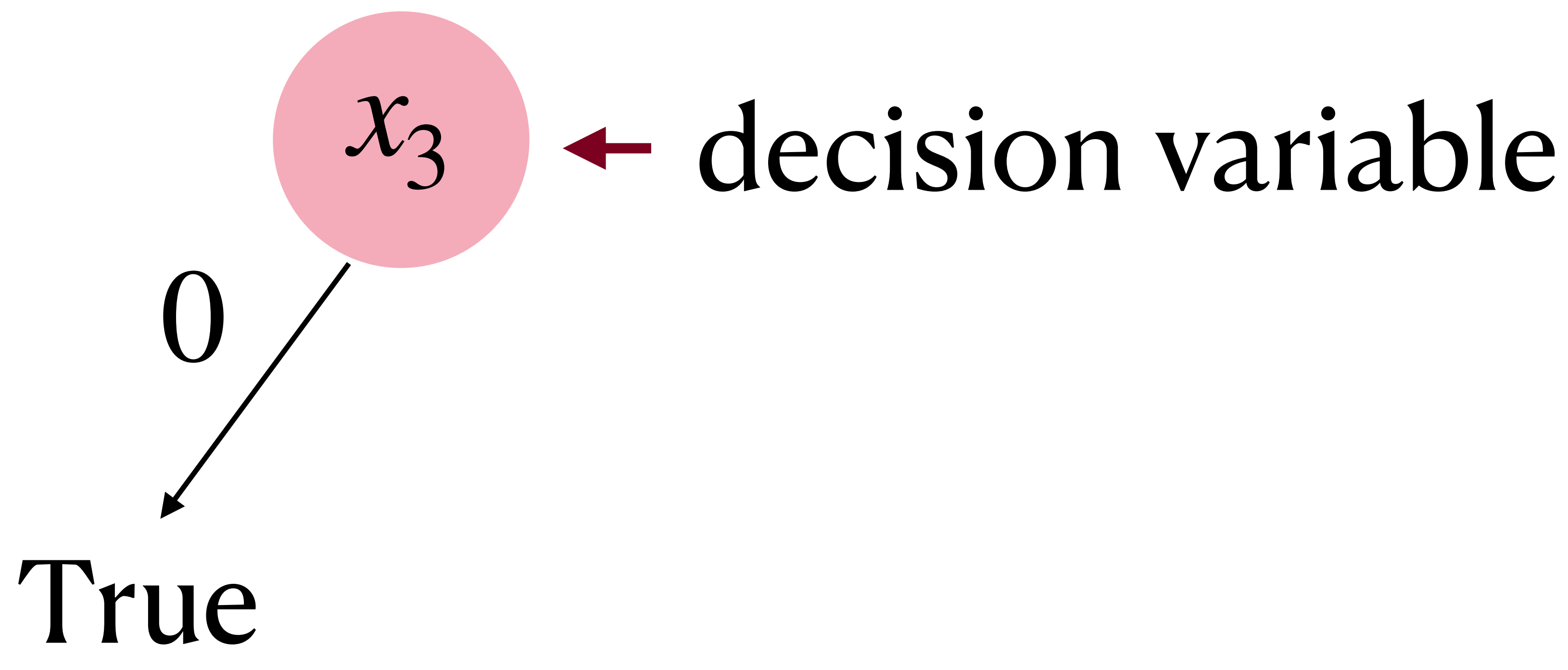


Backtracking

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

Backtrack to last decision, and change the polarity.

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



All clauses are True, hence F is True

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

$DPLL(F, m = \emptyset)\{$

Backtracking at conflict

Unit Propagation

1. If F is True under m then Return SAT
2. If F is False under m then Return UNSAT
3. If there is a unit literal l under m then Return $DPLL(F, m[l \mapsto 1])$
4. If there is a unit literal $\neg l$ under m then Return $DPLL(F, m[l \mapsto 0])$

   Choose an unassigned variable p, and random bit $b \in \{0,1\}$

5. If $DPLL(F, m[p \mapsto b]) == \text{SAT}$ then Return SAT

   Else Return $DPLL(F, m[p \mapsto 1 - b])$

$\}$

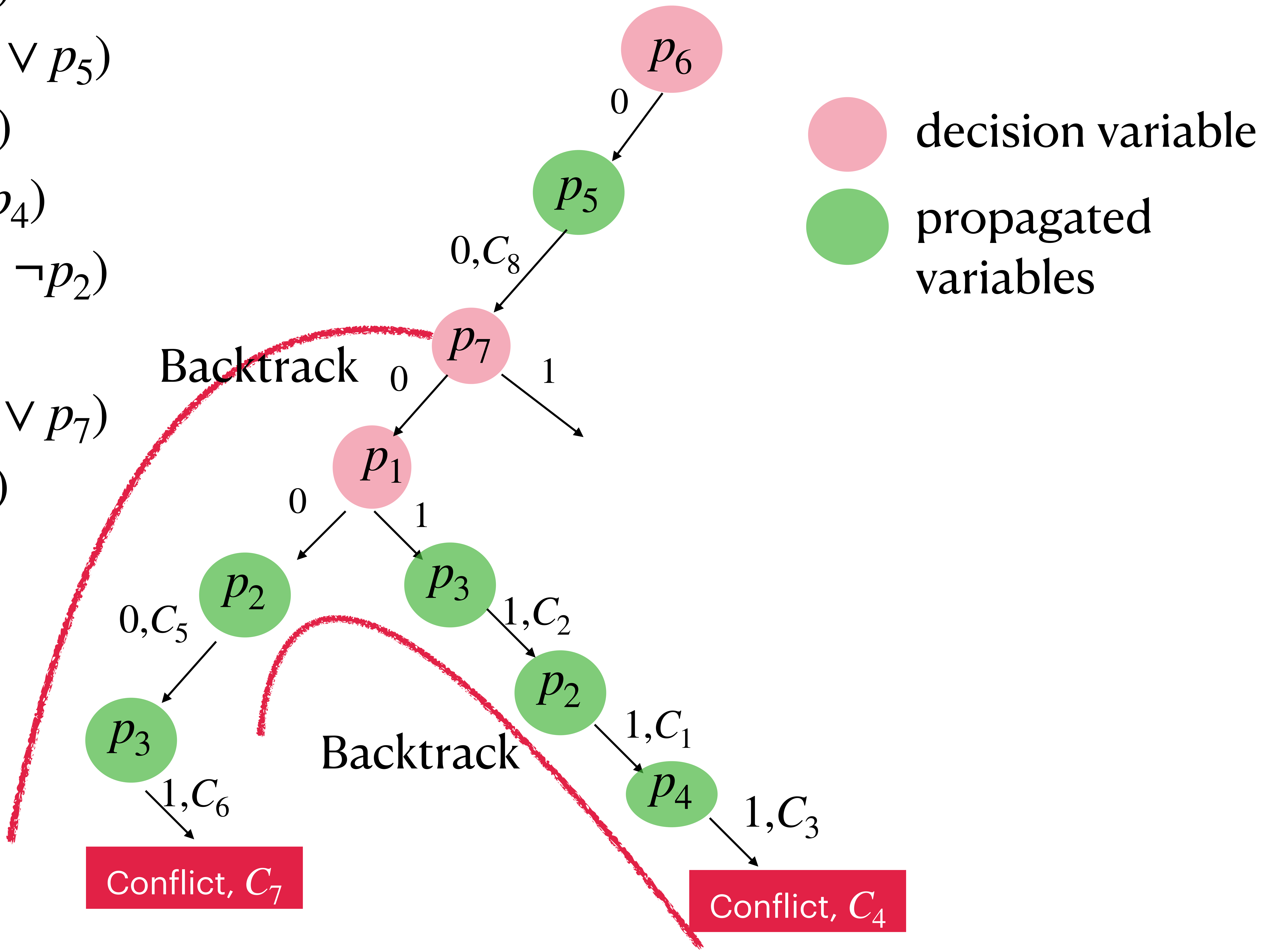
DPLL algorithm (Davis -Putnam-Logemann-Loveland 1960)

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

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



CDCL: Conflict Driven Clause Learning

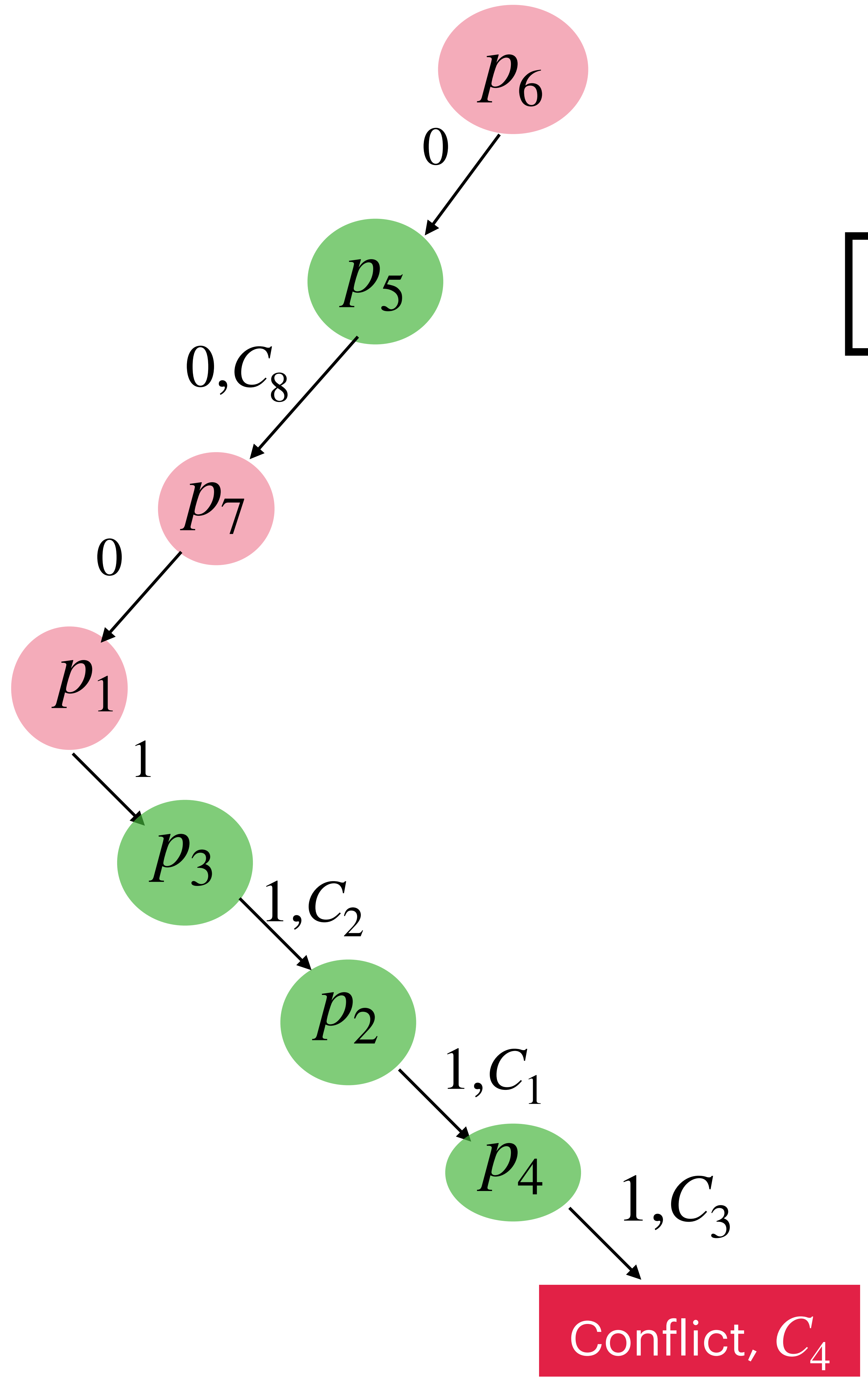
An optimization of DPLL:

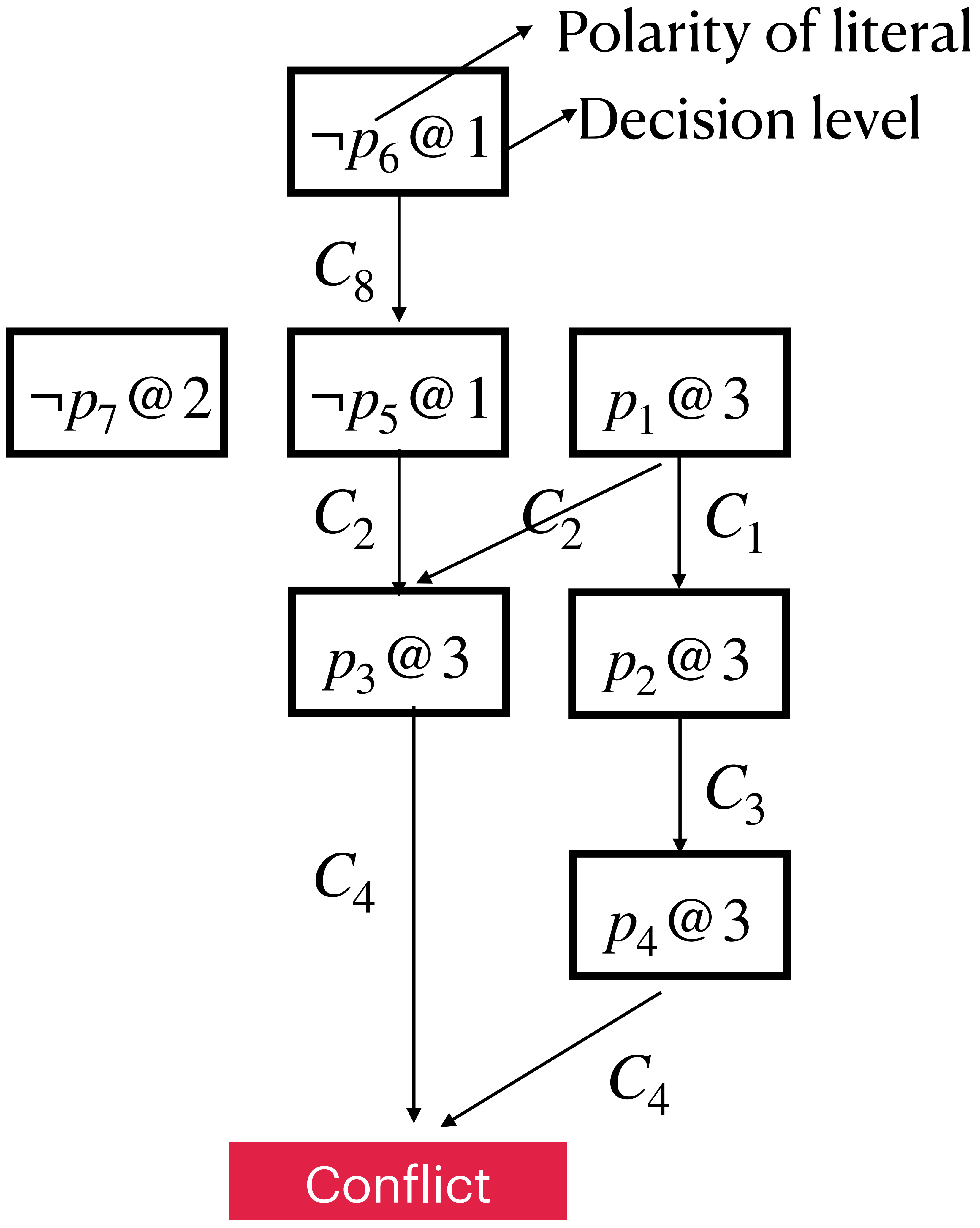
As we decide and propagate, we can observe the run, and avoid unnecessary backtracking.

Construct a data structure to avoid unnecessary backtracking.

Implication Graph.

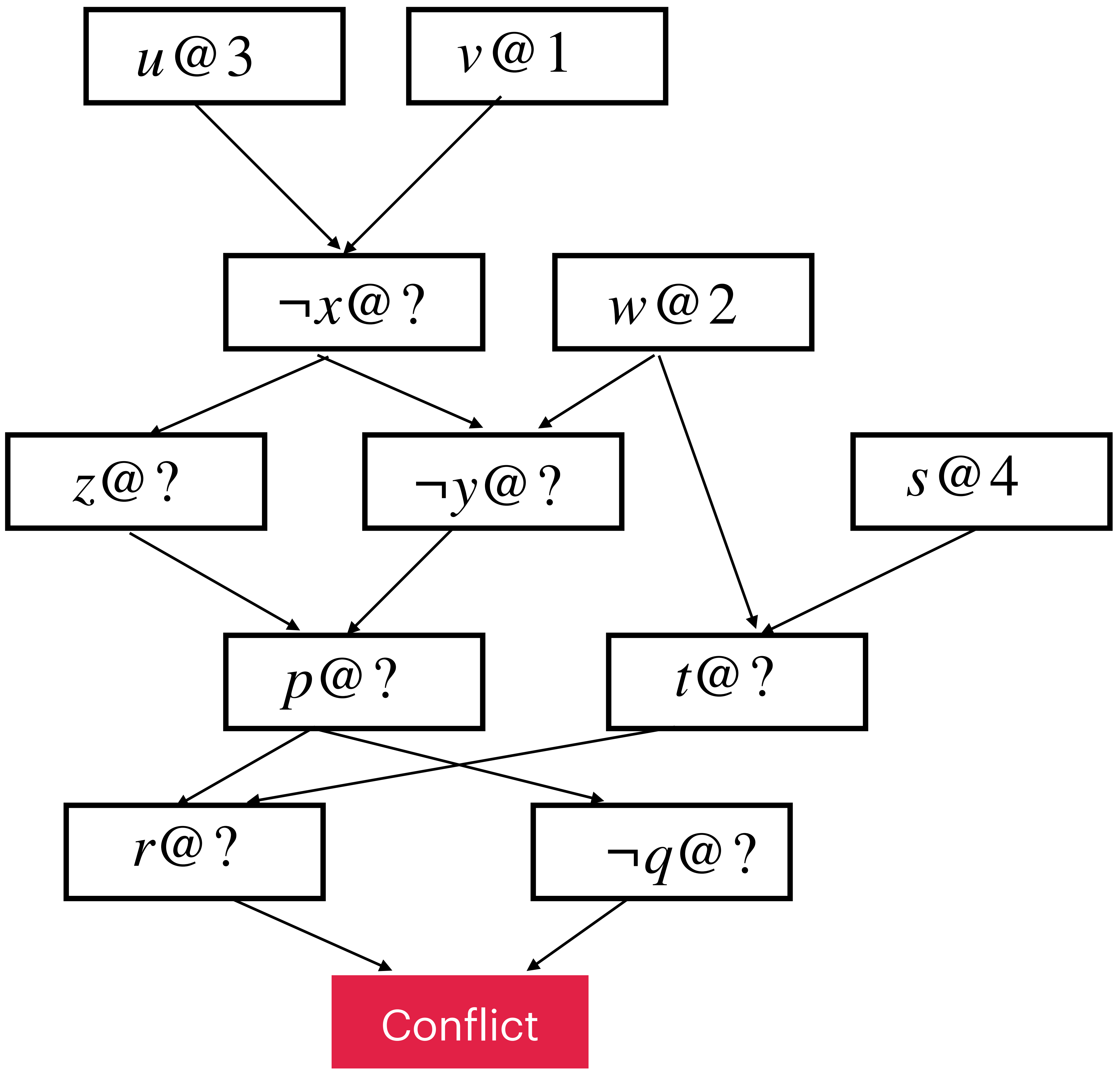
$$C_1 = (\neg p_1 \vee p_2)$$
$$C_2 = (\neg p_1 \vee p_3 \vee p_5)$$
$$C_3 = (\neg p_2 \vee p_4)$$
$$C_4 = (\neg p_3 \vee \neg p_4)$$
$$C_5 = (p_1 \vee p_5 \vee \neg p_2)$$
$$C_6 = (p_2 \vee p_3)$$
$$C_7 = (p_2 \vee \neg p_3 \vee p_7)$$
$$C_8 = (p_6 \vee \neg p_5)$$



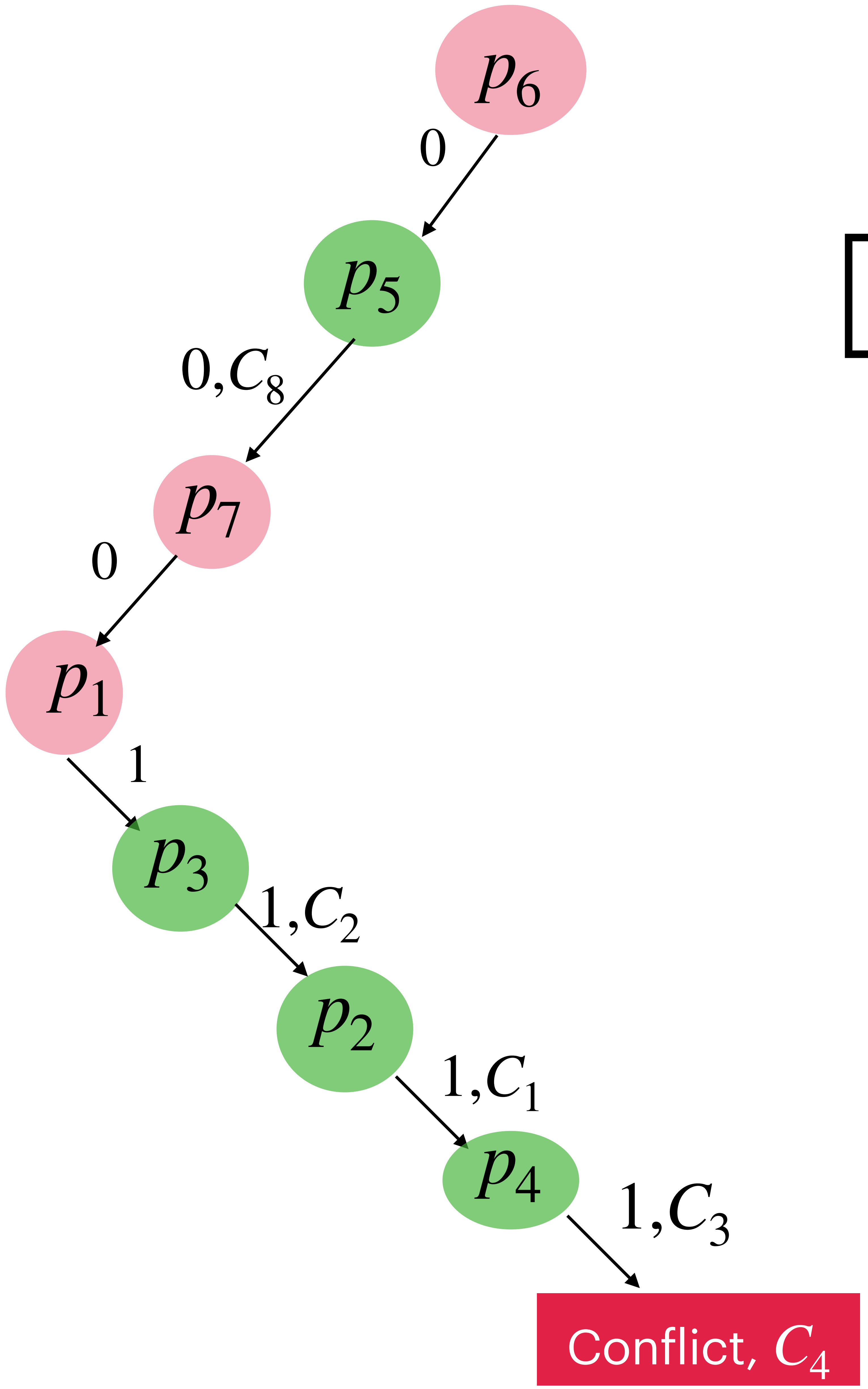


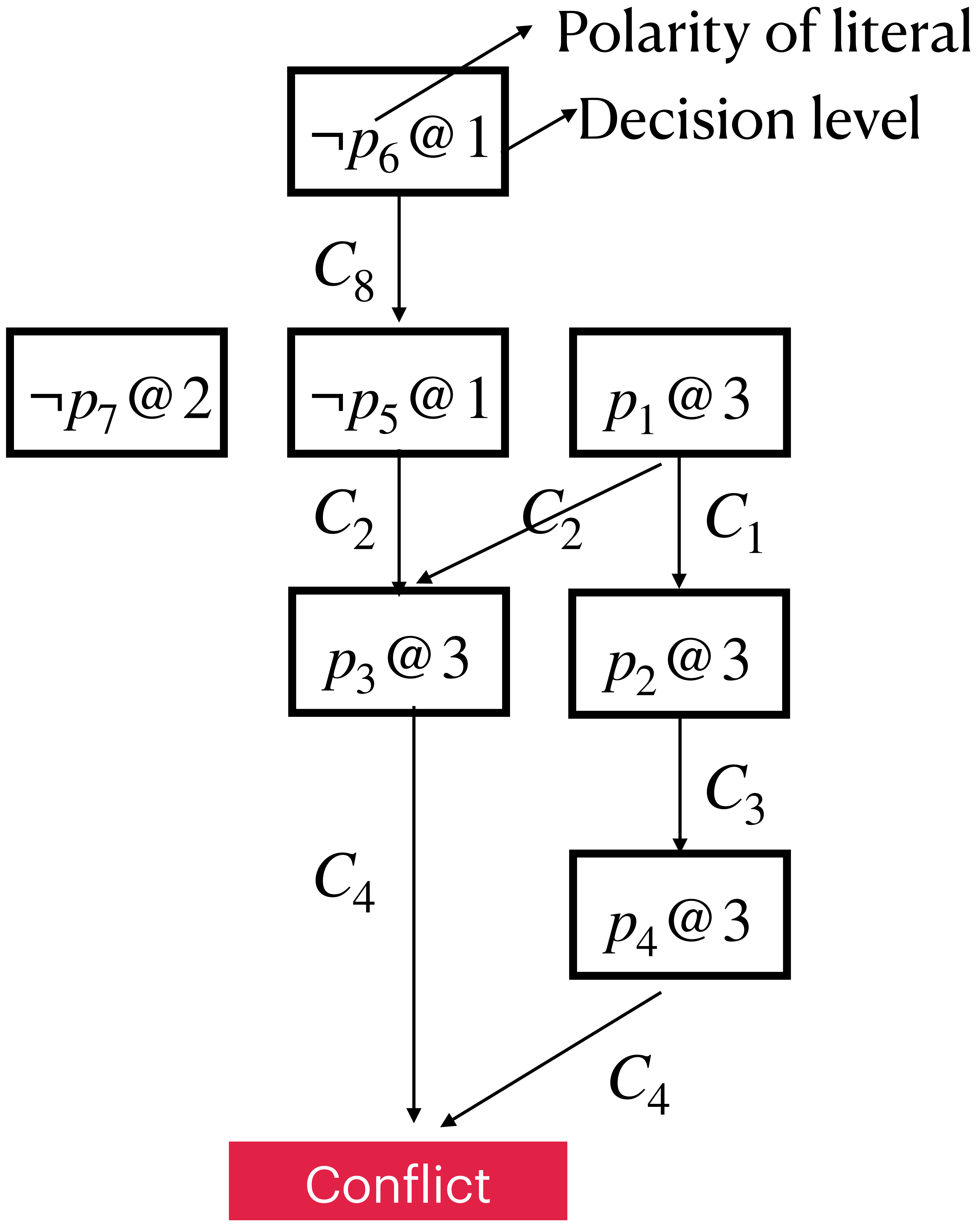
Implication Graph.

Assign decision level at every node in implication graph



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





Implication Graph.

Conflict Clause

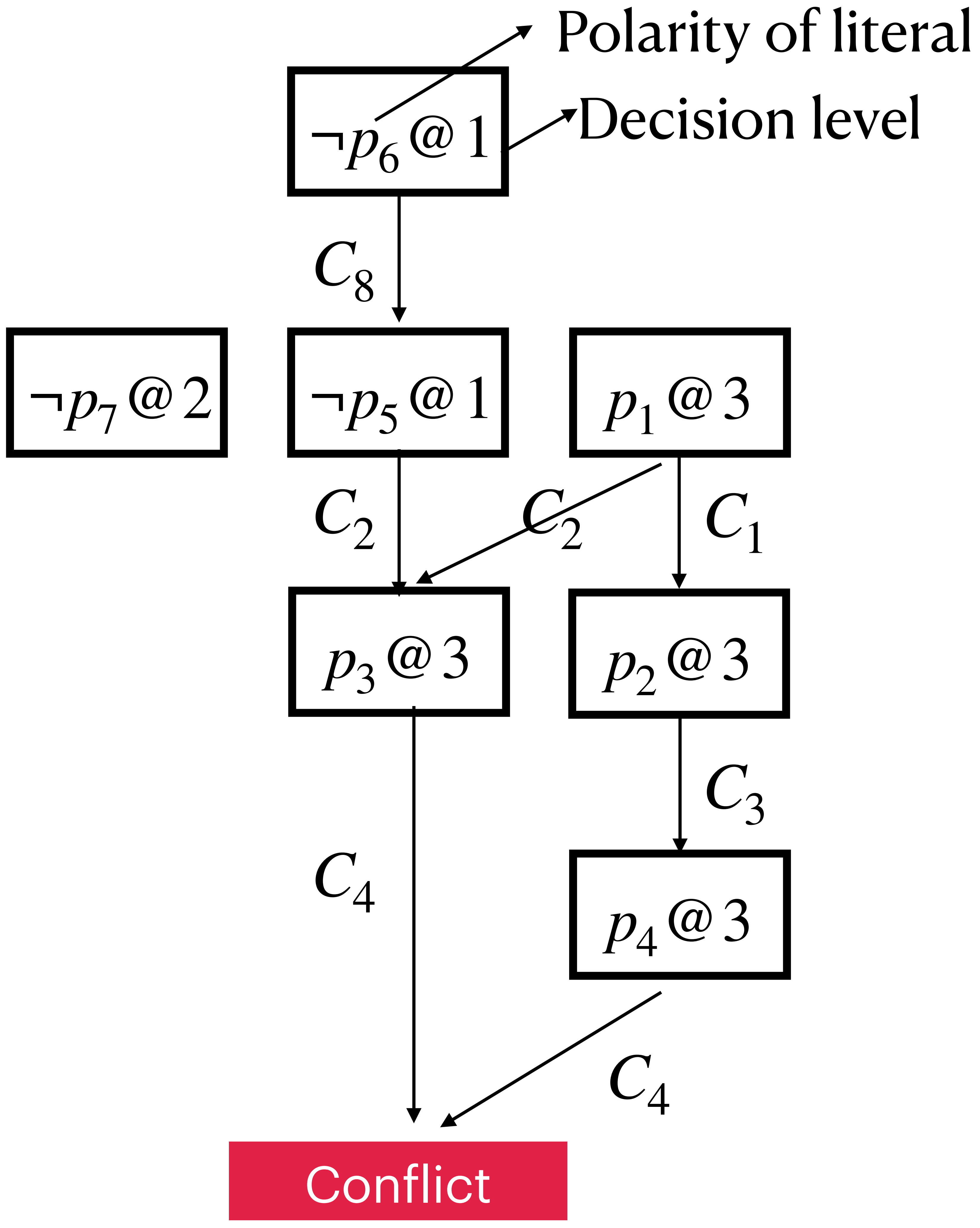
The clause of the negations of the causing decisions is called conflict clause.

Mistake: $p_6 = 0$ and $p_1 = 1$

Conflict clause : $\neg(\neg p_6 \wedge p_1) \equiv p_6 \vee \neg p_1$

$$\left.\begin{array}{l} m(p_6) = 0, m(p_7) = 1, m(p_1) = 1 \\ \\ m(p_6) = 0, m(p_1) = 1 \end{array}\right\}$$

This will never be tried again!



Implication Graph.

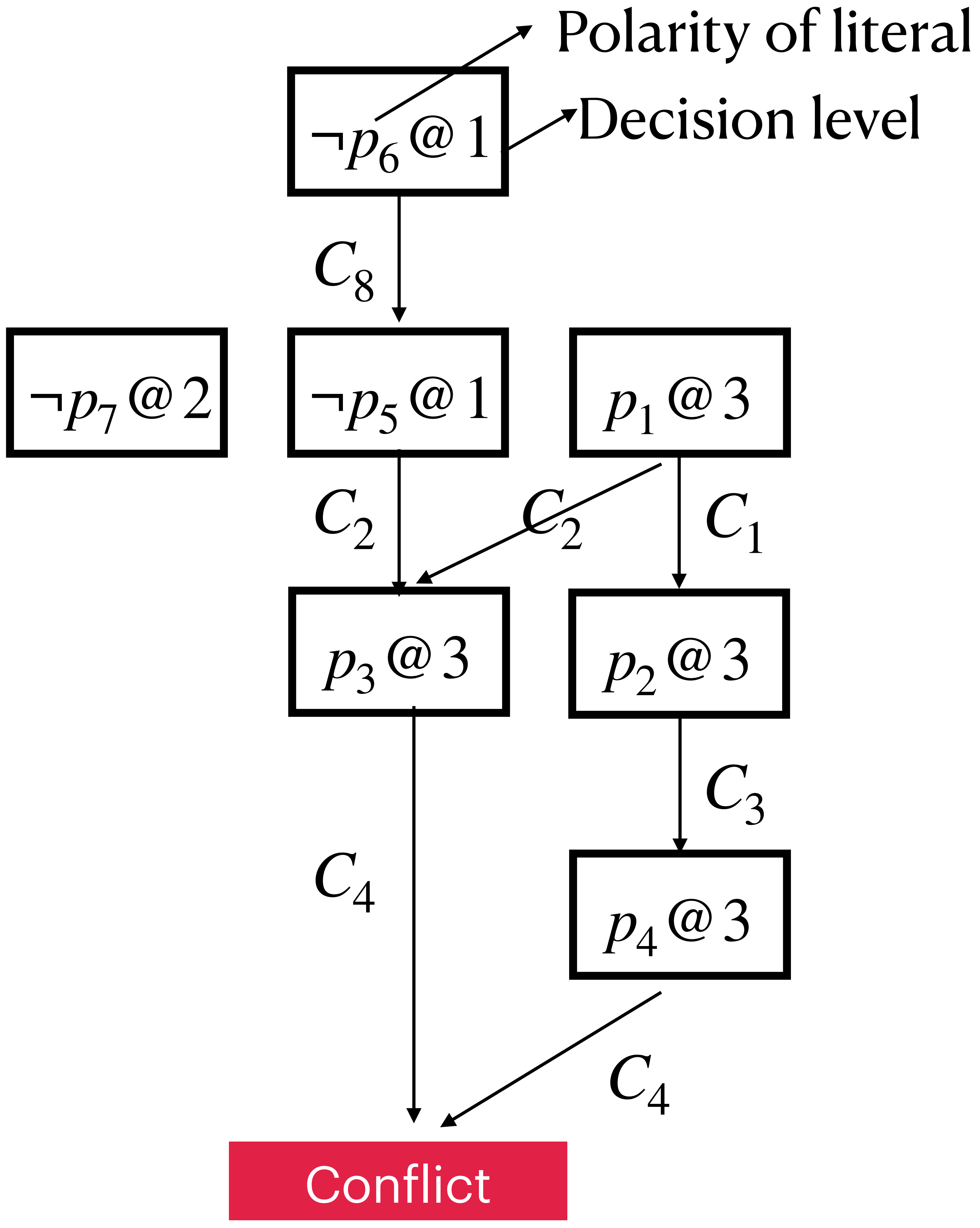
CDCL: Conflict Driven Clause Learning

$$C_1 = (\neg p_1 \vee p_2)$$
$$C_2 = (\neg p_1 \vee p_3 \vee p_5)$$
$$C_3 = (\neg p_2 \vee p_4)$$
$$C_4 = (\neg p_3 \vee \neg p_4)$$
$$C_5 = (p_1 \vee p_5 \vee \neg p_2)$$
$$C_6 = (p_2 \vee p_3)$$
$$C_7 = (p_2 \vee \neg p_3 \vee p_7)$$
$$C_8 = (p_6 \vee \neg p_5)$$
$$C_9 = (p_6 \vee \neg p_1)$$

Added a new clause!
Where should we backtrack?

Backtrack to second largest decision in the conflict clause.

Here we should backtrack to decision level 1.



Implication Graph.

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

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

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

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

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

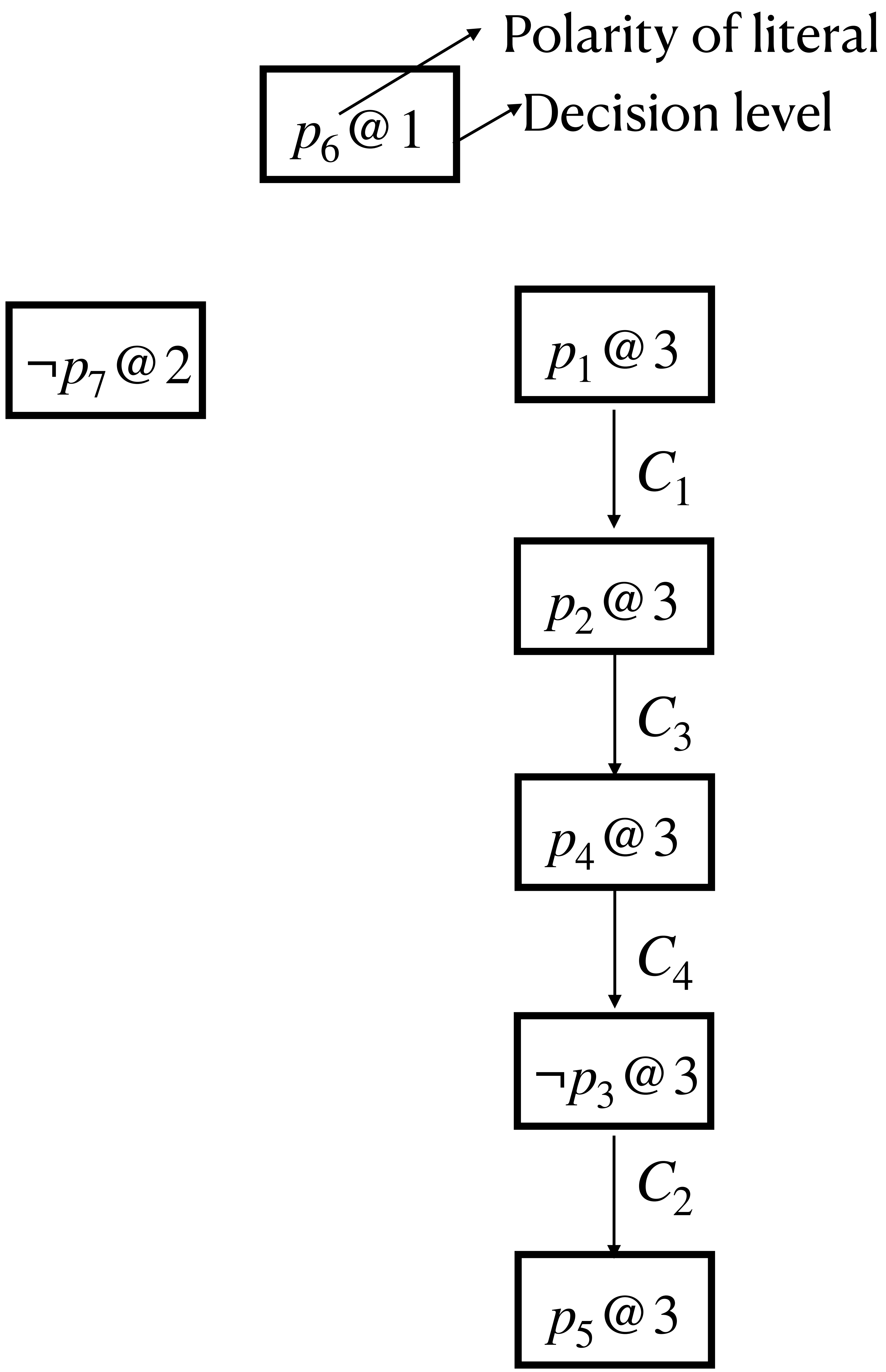
$C_6 = (p_2 \vee p_3)$

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

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

$C_9 = (p_6 \vee \neg p_1)$

Here we should backtrack to decision level 1.



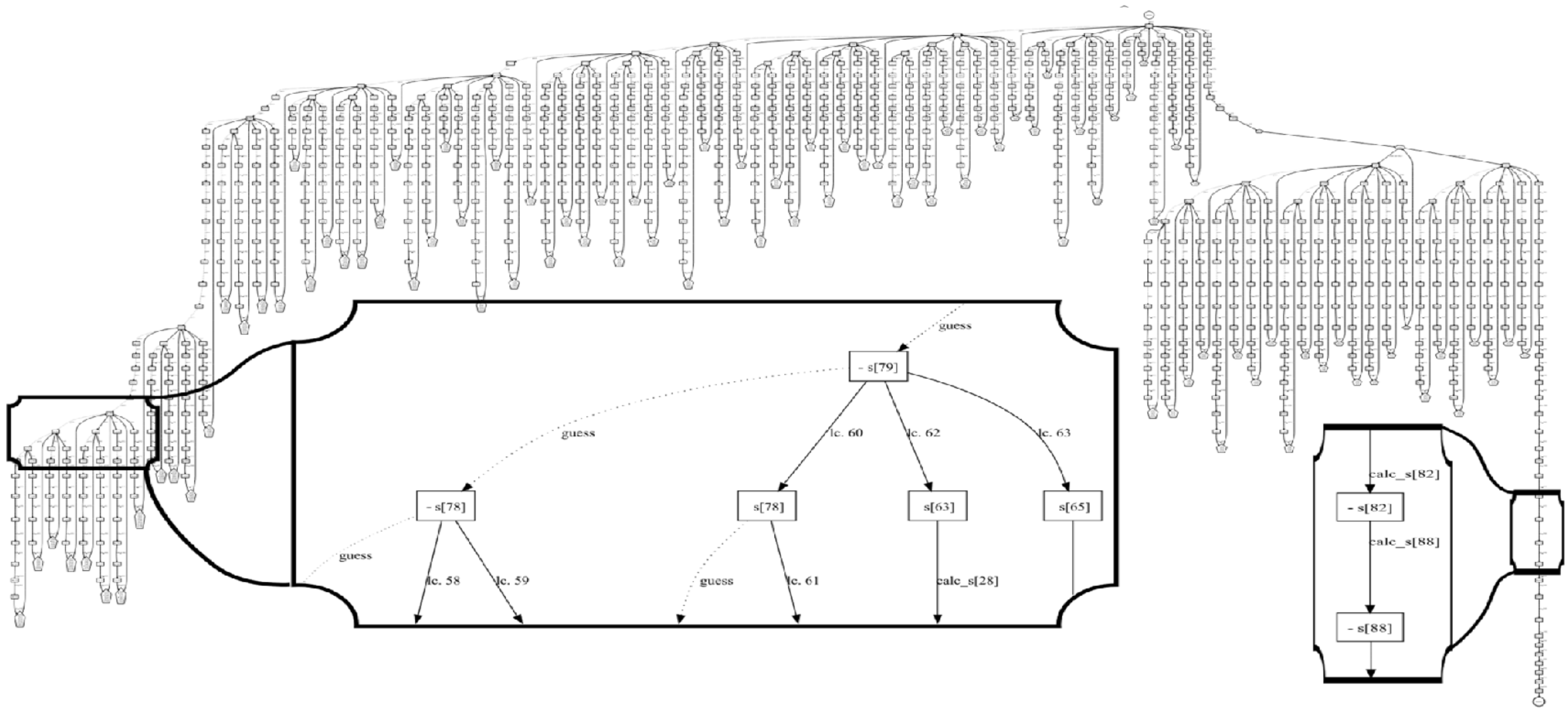
Implication Graph.

CDCL: Conflict Driven Clause Learning

1. UnitPropagation(m, F): applies unit propagation and extends m.

2. Decide(m, F): choose an unassigned variable in m and assign it a Boolean value.

3. AnalyzeConflict(m, F): returns a conflict clause learned using implication graph, and a decision level upto which the solver needs to backtrack.



Taken from Mate Soos 's slides.

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

2002	zChaff	Moskewicz, Madigan, Zhao, Zhang, Malik
2003	Forklift	Goldberg, Novikov
2004	zChaff	Moskewicz, Madigan, Zhao, Zhang, Malik
2005	SatELiteGTI	Eén, Sörensson
2006	MiniSat	Eén, Sörensson
2007	RSAT	Pipatsrisawat, Darwiche
2008	MiniSat	Eén, Sörensson
2009	Precosat	Biere
2010	CryptoMiniSat	Soos
2011	Glucose	Audemard, Simon
2012	Glucose	Audemard, Simon
2013	Lingeling	Biere
2014	Lingeling	Biere
2015	abcdSAT	Chen
2016	MapleCOMSPS	Liang, Oh, Ganesh, Czarnecki, Poupart
2017	MapleLCMDist	Xiao, Luo, Li, Manya, Lu
2018	MapleLCMDistChronoBT	Nadel, Ryvchin
2019	MapleLCMDistChronoBTDLv3	Kochemazov, Zaikin, Kondratiev, Semenov
2020	Kissat	Biere, Fazekas, Fleury, Heisinger
2021	KissatMAB	Cherif, Habet, Terrioux
2022	KissatMAB-HyWalk	Zheng, He, Chen
2023	SBVA-CaDiCal	Haberlandt, Green

MiniSat-based:

Armin Biere's& derived:

Others:

Taken from Alex's slides.

CDCL: Conflict Driven Clause Learning

1. UnitPropagation(m, F): applies unit propagation and extends m.

2. Decide(m, F): choose an unassigned variable in m and assign it a Boolean value.

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

3. ClauseLearning(m, F): returns a conflict clause learned using implication graph, and a decision level upto which the solver needs to backtrack.

Heuristics: how to learn a small conflict clause and unto which level to backtrack?

Heuristics: how to learn a small conflict clause and upto which level to backtrack?

AnalyzeConflict(m,F): some choices of clauses are found to be better than others.

Notations:

UIP (Unique Implication Point)

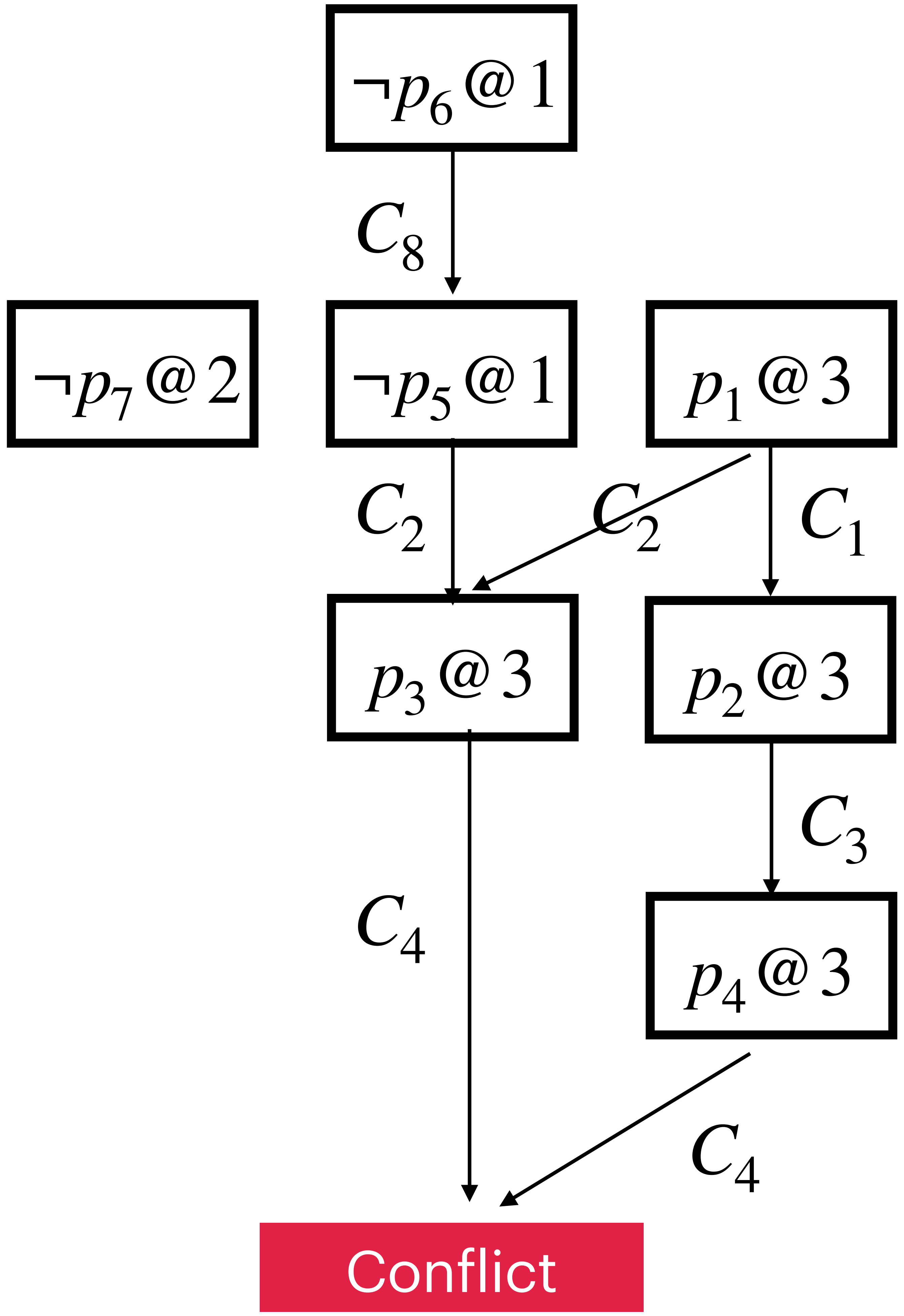
In an implication graph, node *"$l@d$"* is a UIP at decision level d if *"$l@d$"* occurs in each path from d^{th} decision literals to the conflict.

UIP points: In an implication graph, node "$l@d$" is a UIP at decision level d if "$l@d$" occurs in each path from d^{th} decision literals to the conflict.

UIP @ level 1:

UIP @ level 2:

UIP @ level 3:



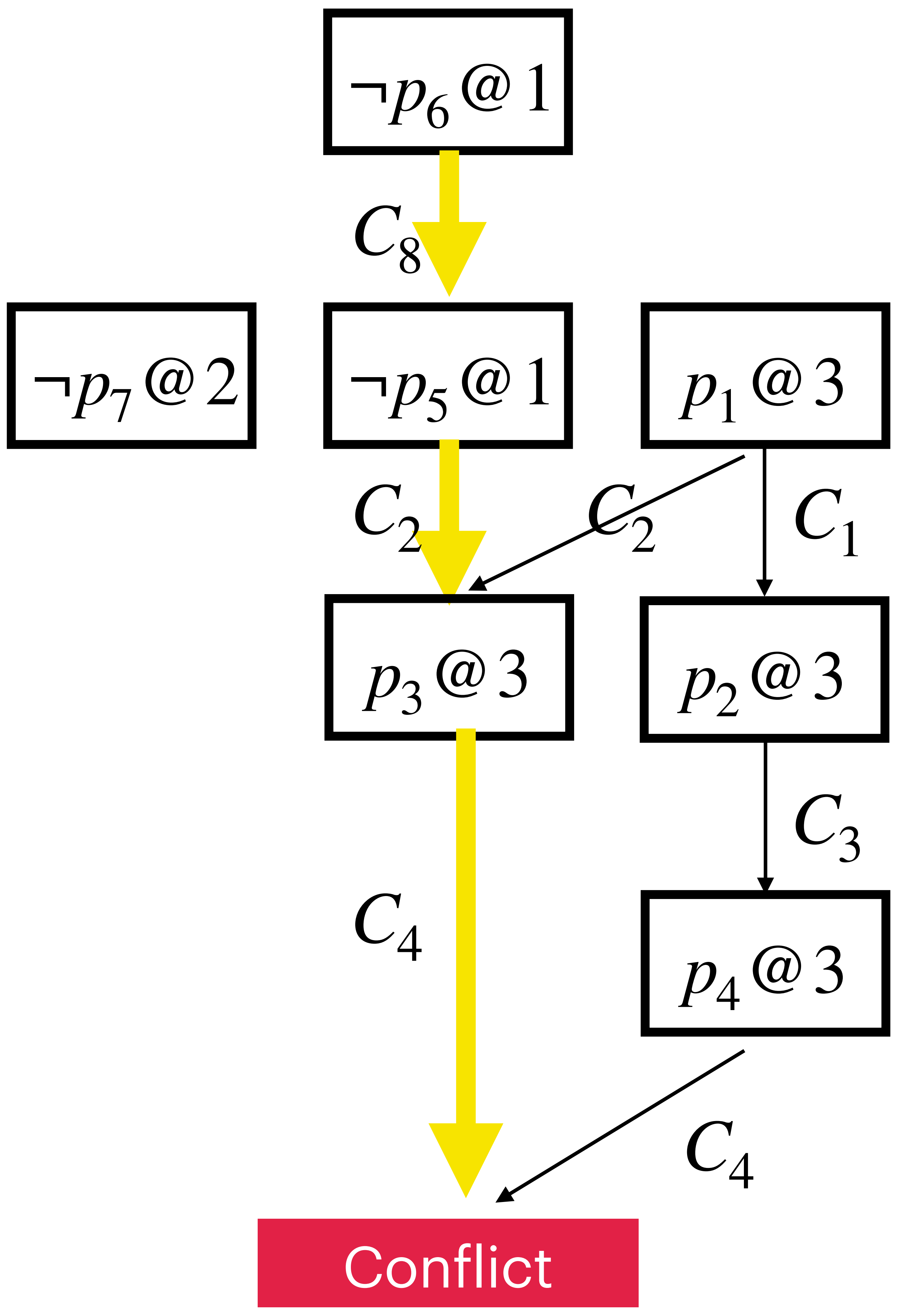
Implication Graph.

UIP points: In an implication graph, node "$l@d$" is a UIP at decision level d if "$l@d$" occurs in each path from d^{th} decision literals to the conflict.

UIP @ level 1: $\neg p_6 @ 1, \neg p_5 @ 1$

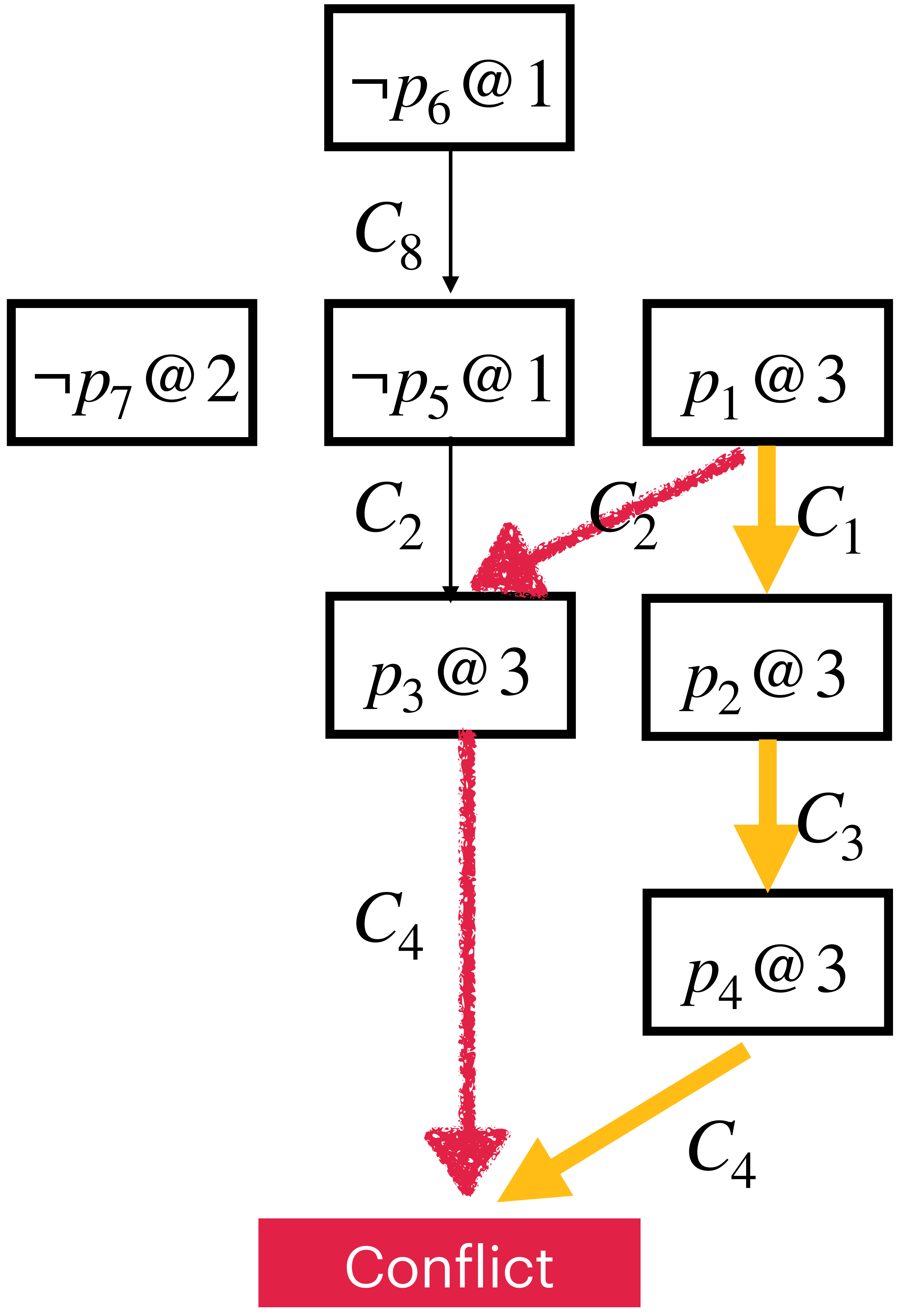
UIP @ level 2:

UIP @ level 3:



Implication Graph.

UIP points: In an implication graph, node *"$l@d$"* is a UIP at decision level d if *"$l@d$"* occurs in each path from d^{th} decision literals to the conflict.



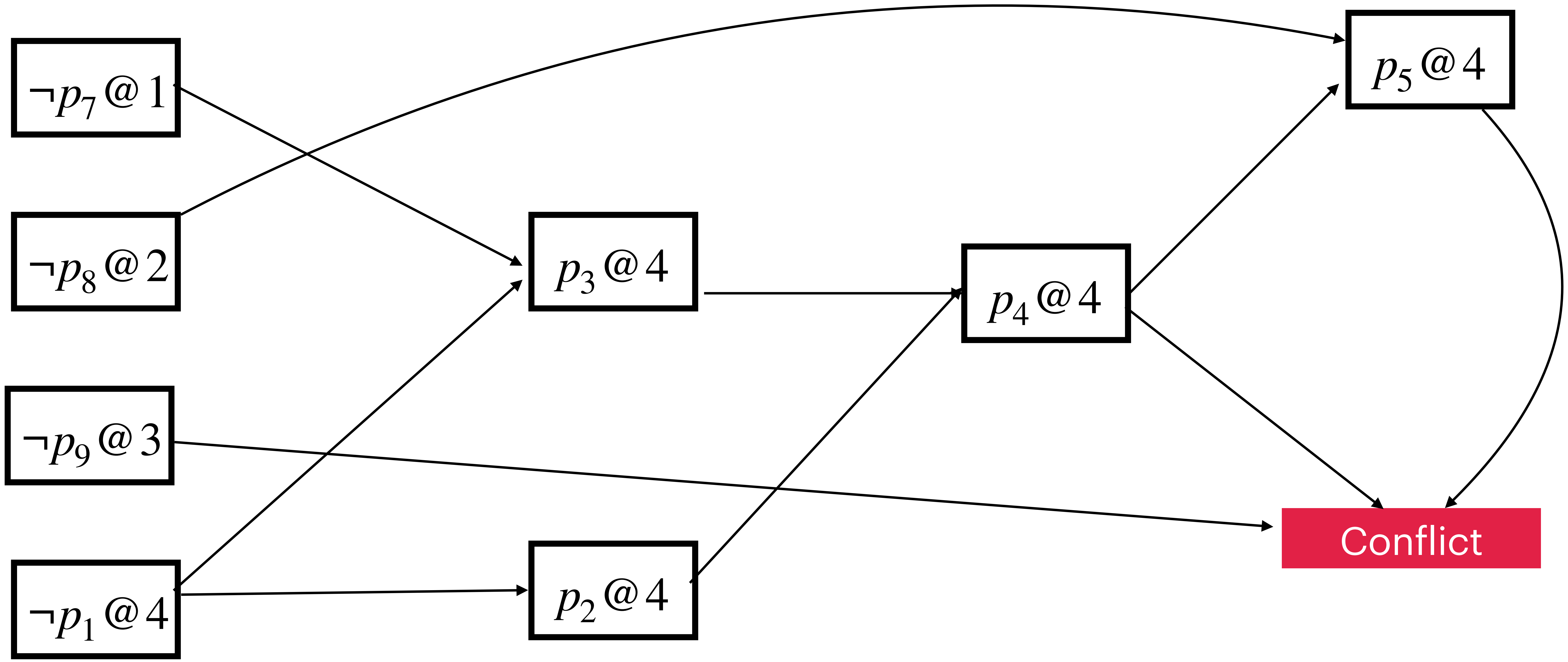
Implication Graph.

UIP @ level 1: $\neg p_6 @ 1, \neg p_5 @ 1$

UIP @ level 2:

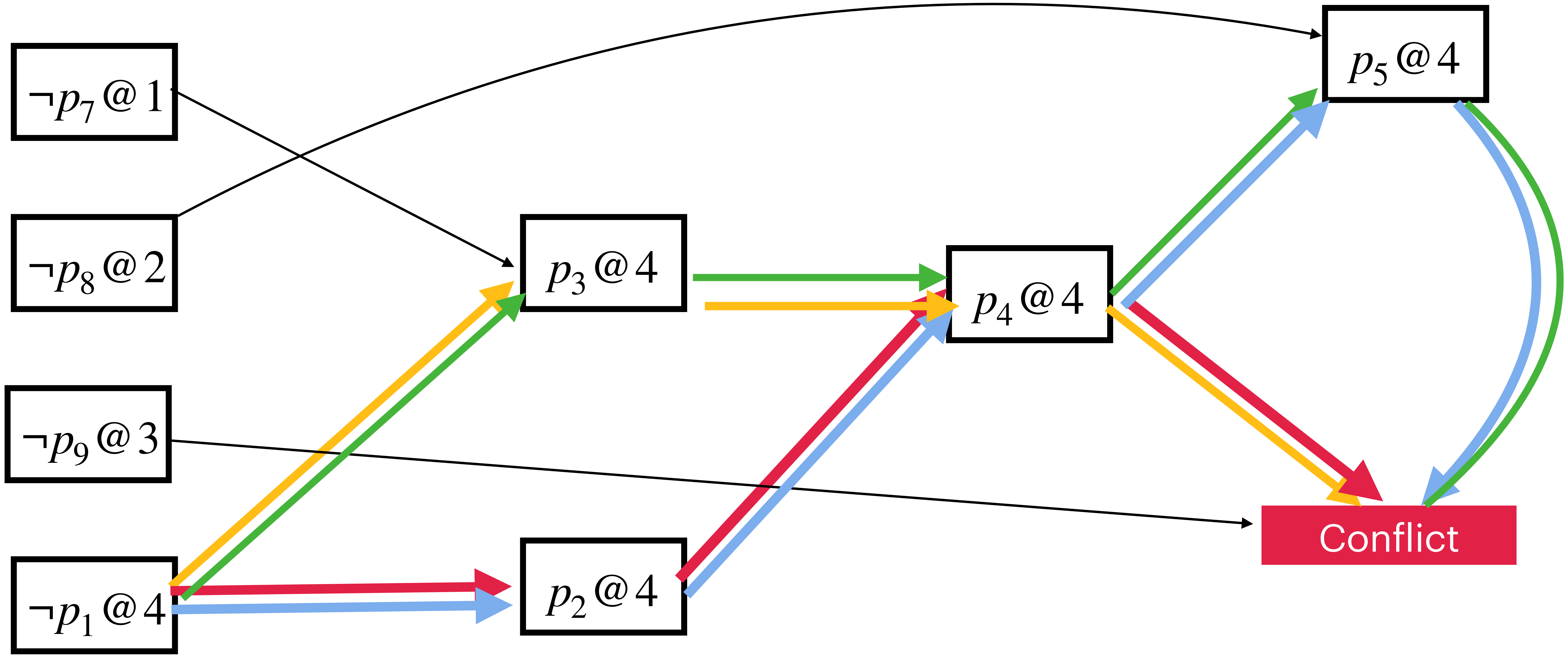
UIP @ level 3: $p_1 @ 3$

UIP points: In an implication graph, node "$l@d$" is a UIP at decision level d if "$l@d$" occurs in each path from d^{th} decision literals to the conflict.



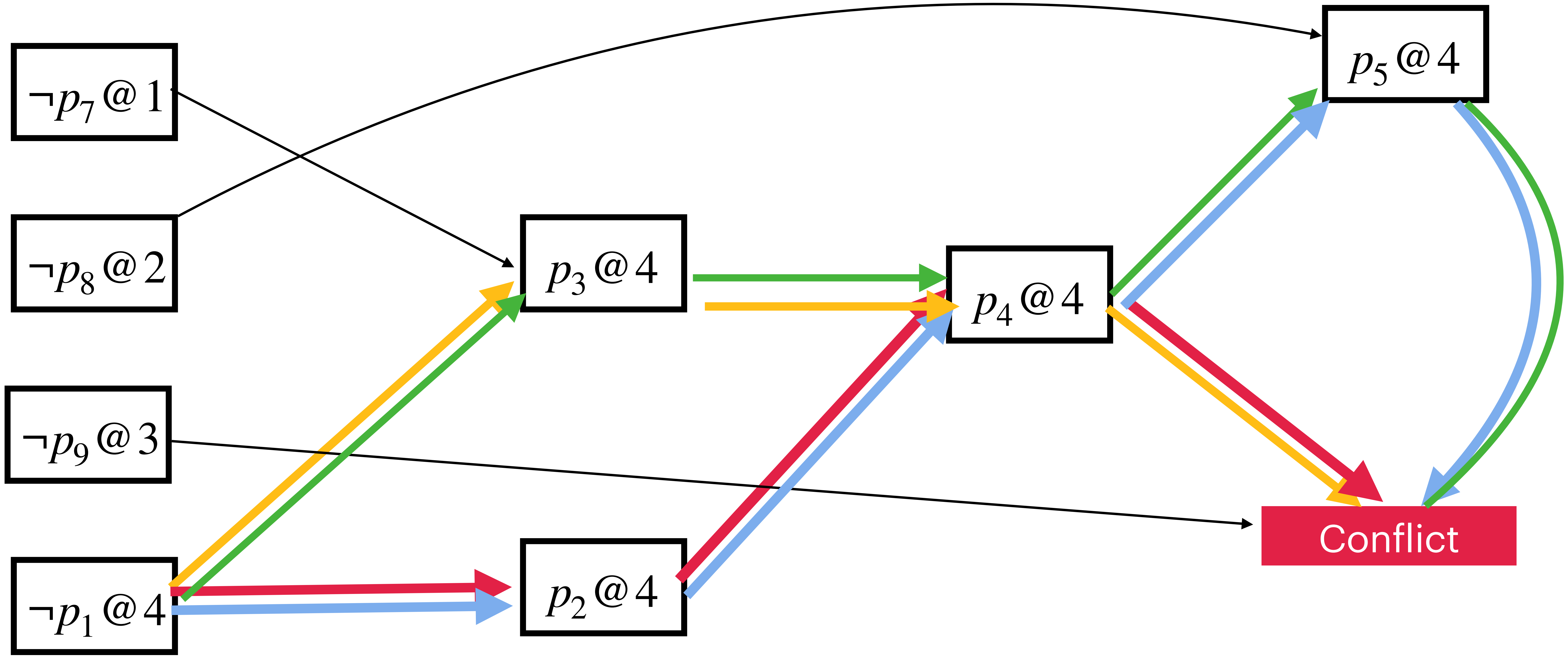
UIP $@4 = ???$

UIP points: In an implication graph, node "$l@d$" is a UIP at decision level d if "$l@d$" occurs in each path from d^{th} decision literals to the conflict.



UIP $@4 = ???$

UIP points: In an implication graph, node "$l@d$" is a UIP at decision level d if "$l@d$" occurs in each path from d^{th} decision literals to the conflict.



$$\text{UIP } @4 = \neg p_1 @ 4, p_4 @ 4$$

First UIP Point: $p_4 @ 4$

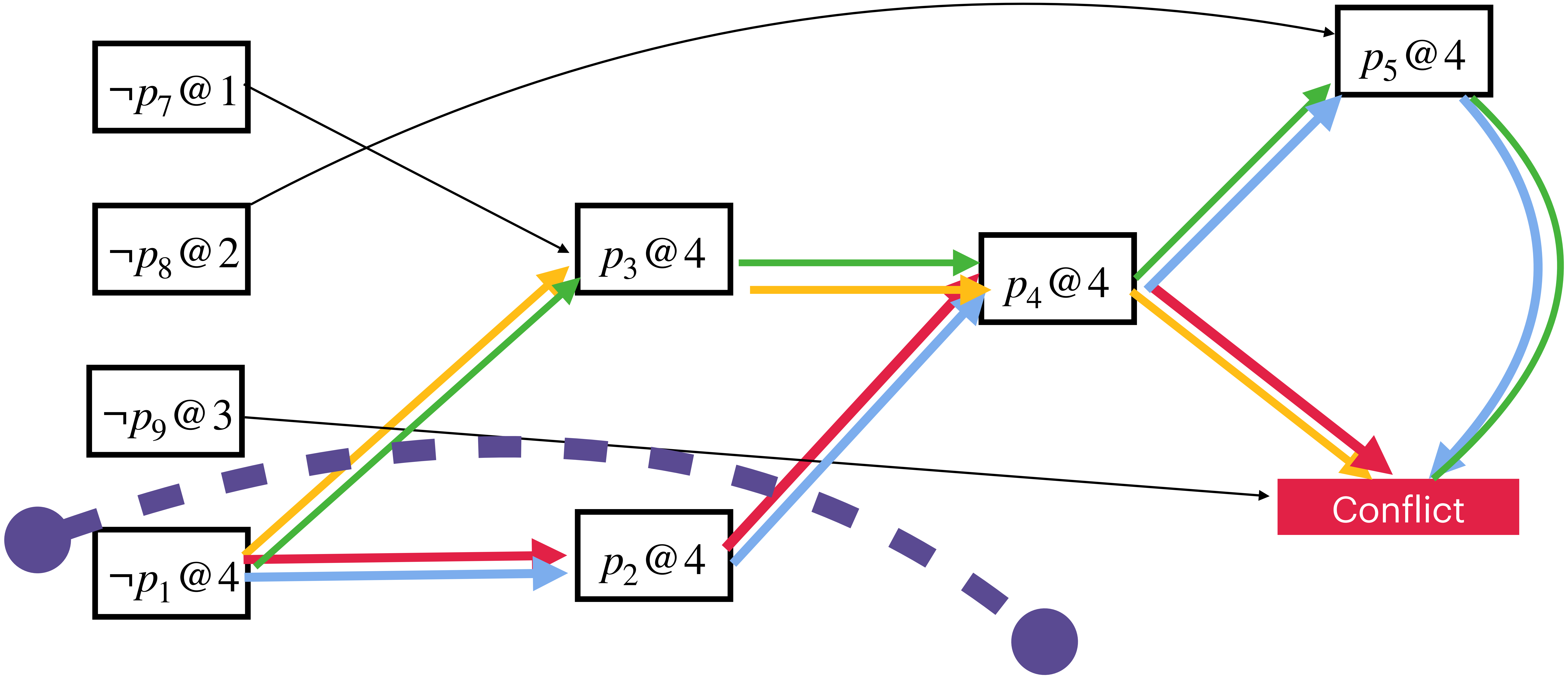
Last UIP Point: $\neg p_1 @ 4$

UIP cuts to analyze conflicts:

If l is UIP, then corresponding UIP cut is (A,B) of the implication graph.
Where,

B contains all the successors of l from which there is a path to conflict.
A contains the rest.

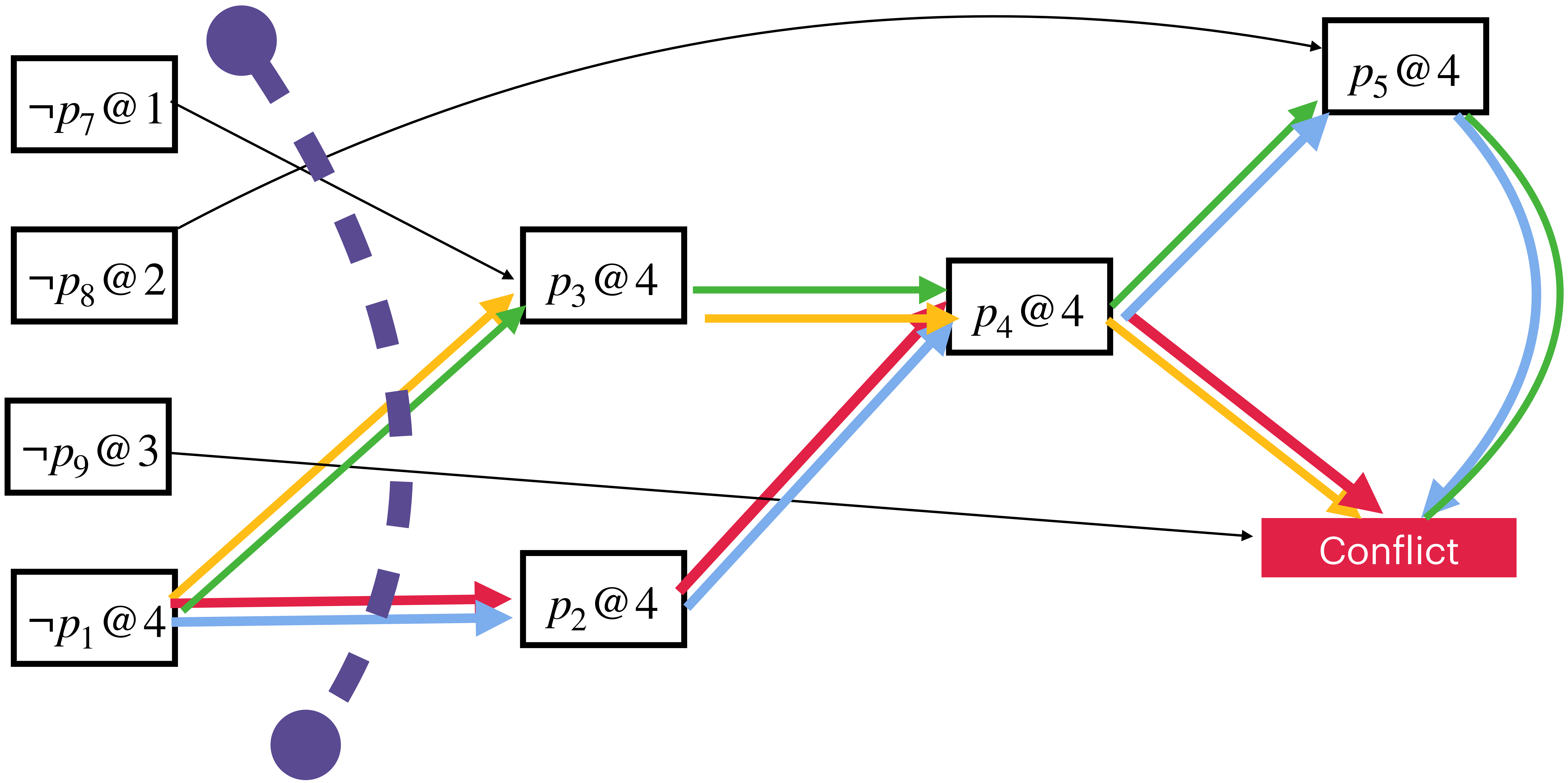
UIP cuts to analyze conflicts: If l is UIP, then corresponding UIP cut is (A,B) of the implication graph, where B contains all the successors of l from which there is a path to conflict, and A contains the rest.



UIP $@4 = \neg p_1 @4, p_4 @4$

Is it a UIP cut?

UIP cuts to analyze conflicts: If l is UIP, then corresponding UIP cut is (A,B) of the implication graph, where B contains all the successors of l from which there is a path to conflict, and A contains the rest.

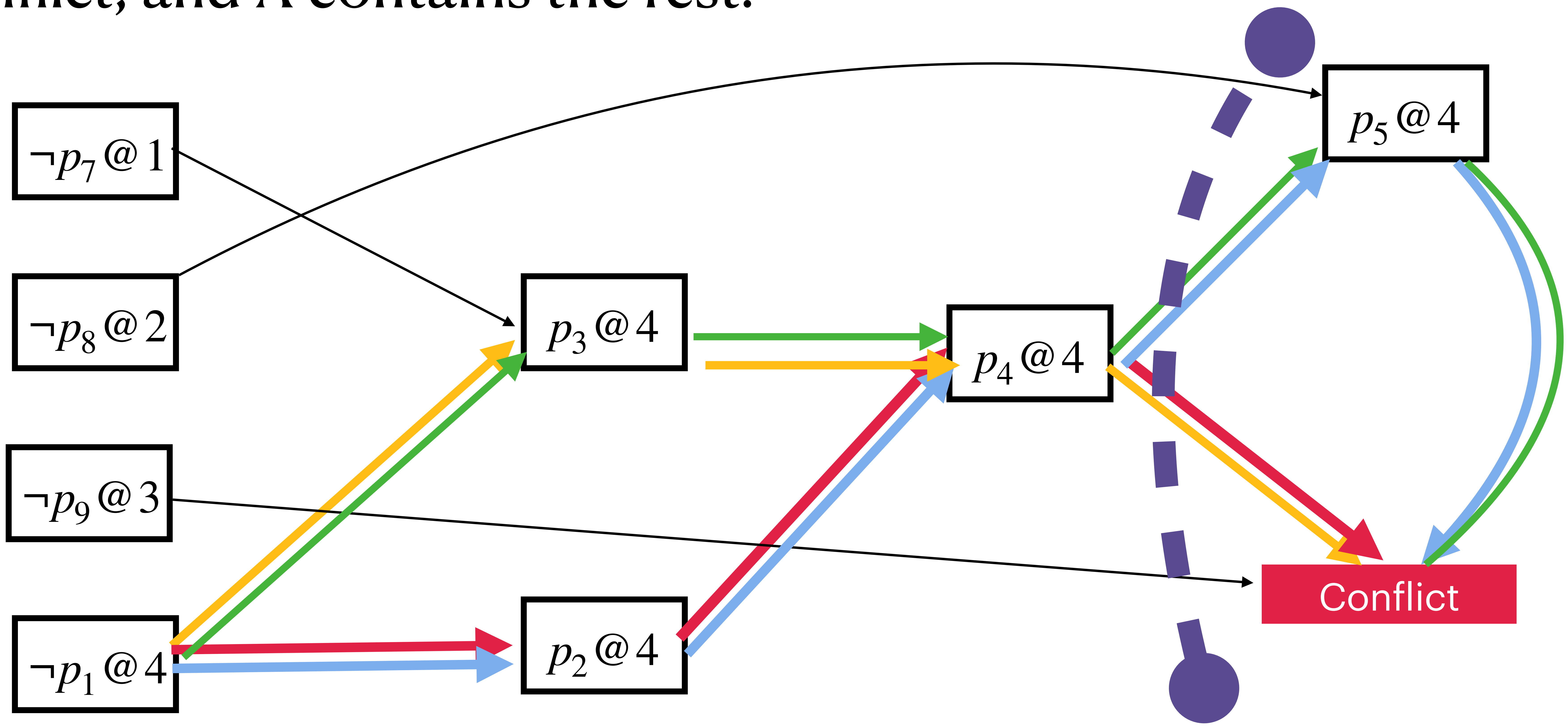


UIP $@4 = \neg p_1 @ 4, p_4 @ 4$

Is it a UIP cut?

Yes, with respect to $\neg p_1 @ 4$

UIP cuts to analyze conflicts: If l is UIP, then corresponding UIP cut is (A,B) of the implication graph, where B contains all the successors of l from which there is a path to conflict, and A contains the rest.



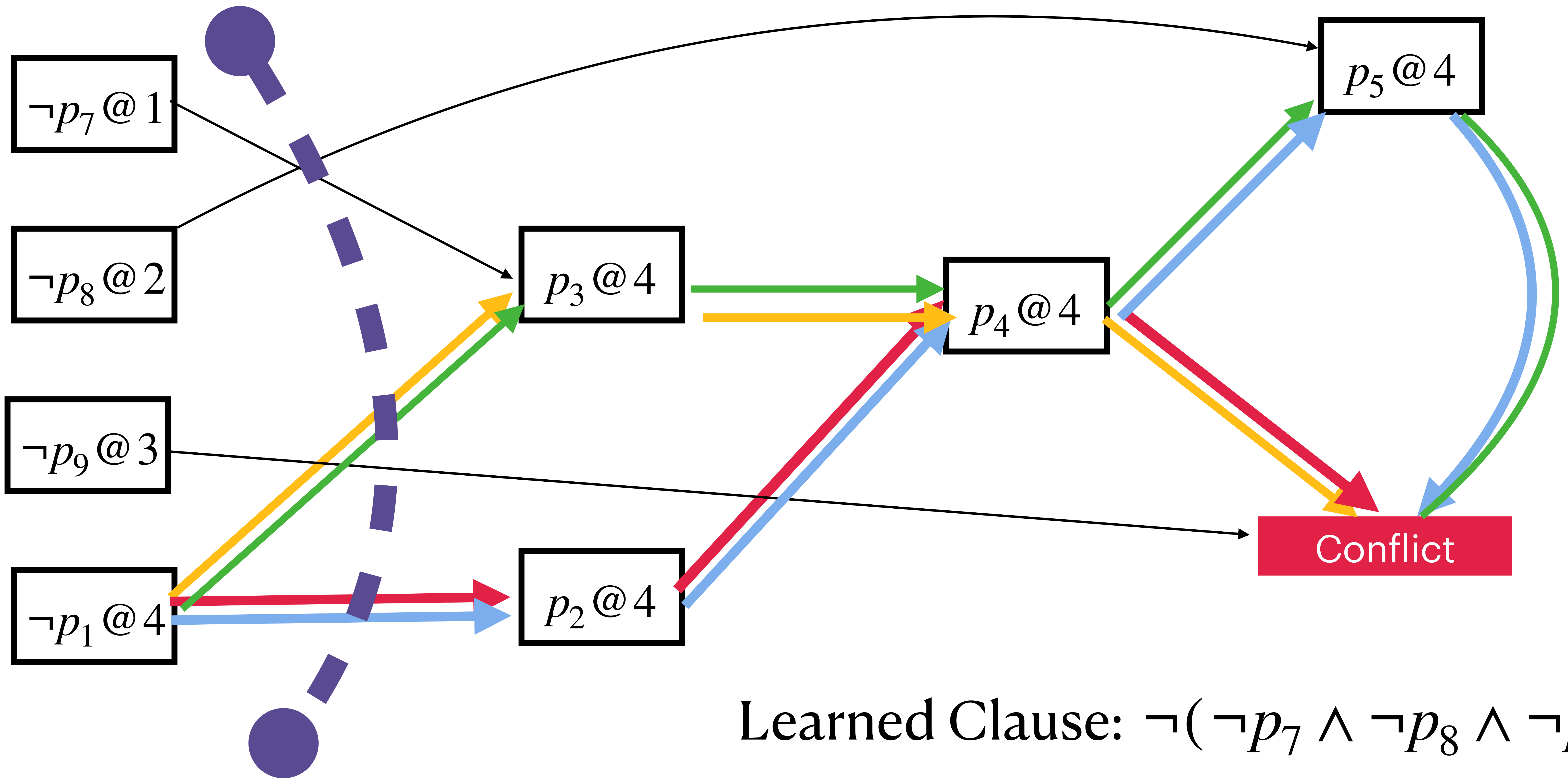
UIP $@4 = \neg p_1 @ 4, p_4 @ 4$

Is it a UIP cut?

Yes, with respect to $p_4 @ 4$

Learned Conflict Clause from UIP cut

The literals on the A side of the cut, which have an edge directed from A to B, form a clause. These literals are then negated and combined into a disjunction.

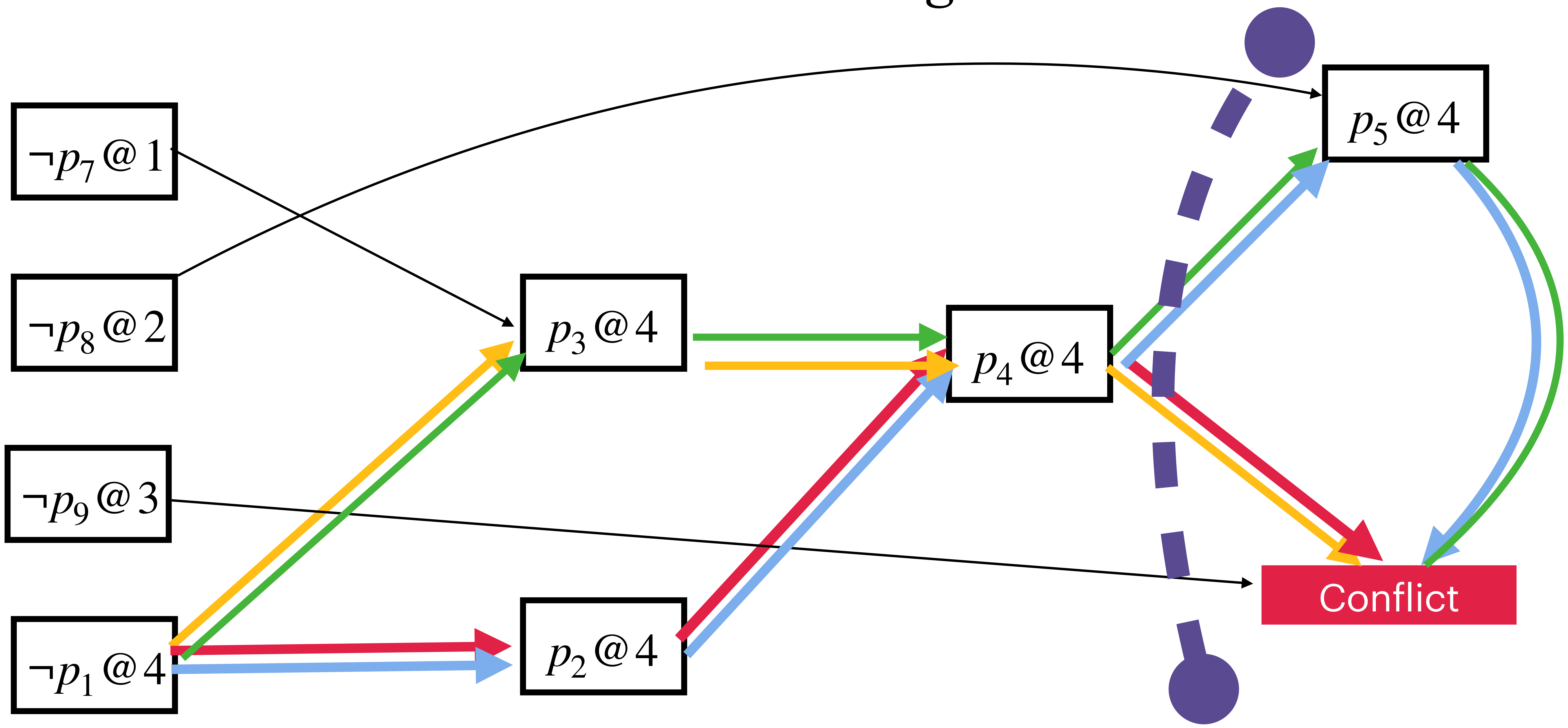


Learned Clause: $\neg(\neg p_7 \wedge \neg p_8 \wedge \neg p_9 \wedge \neg p_1)$

UIP $@4 = \neg p_1 @ 4, p_4 @ 4$

Learned Conflict Clause from UIP cut

The literals on the A side of the cut, which have an edge directed from A to B, form a clause. These literals are then negated and combined into a disjunction.



UIP $@4 = \neg p_1 @ 4, p_4 @ 4$ Learned Clause: $\neg(\neg p_8 \wedge p_4 \wedge \neg p_9)$