CDCL SAT Solving

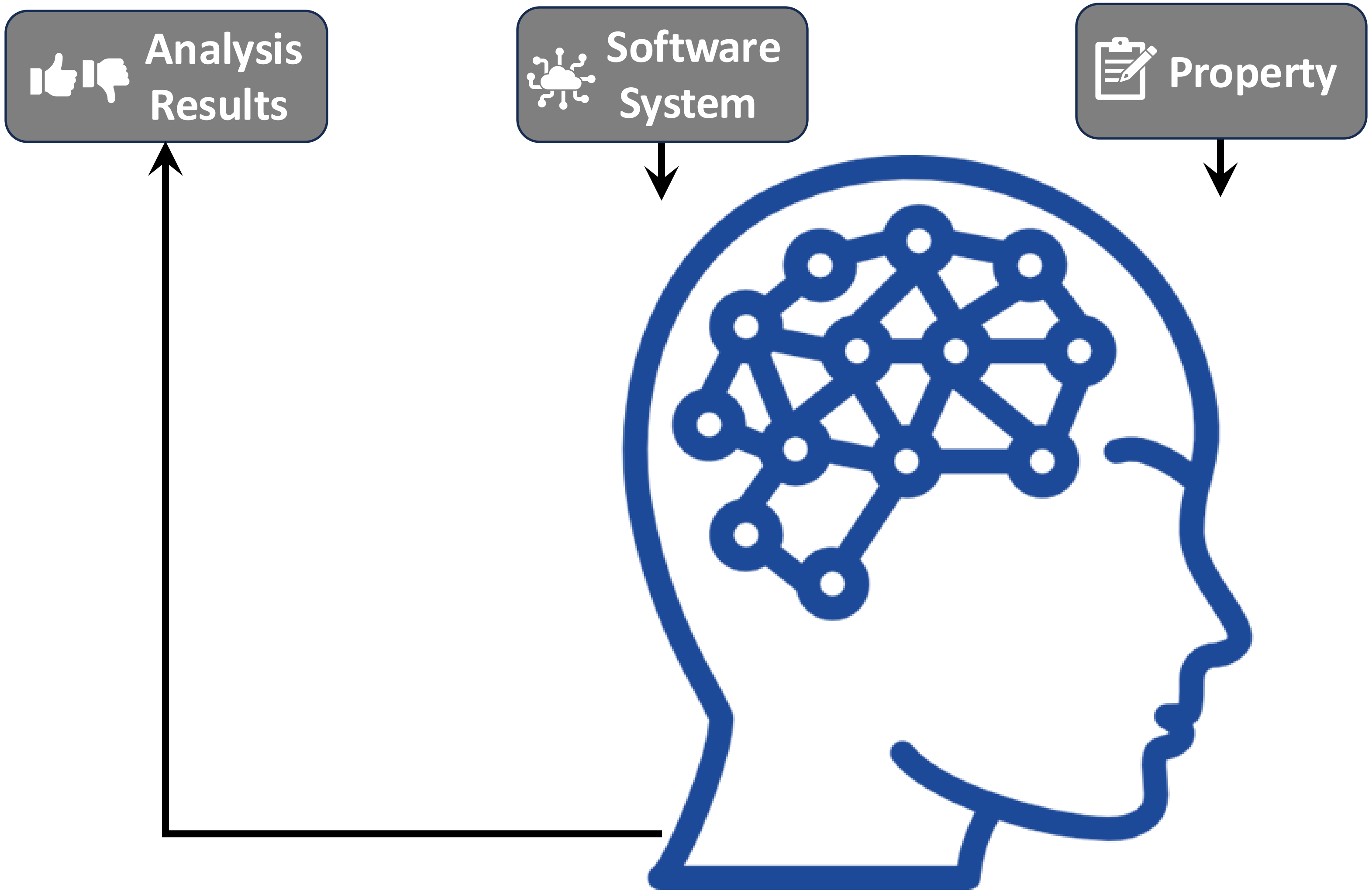
Wenxi Wang

University of Virginia

wenxiw@virginia.edu

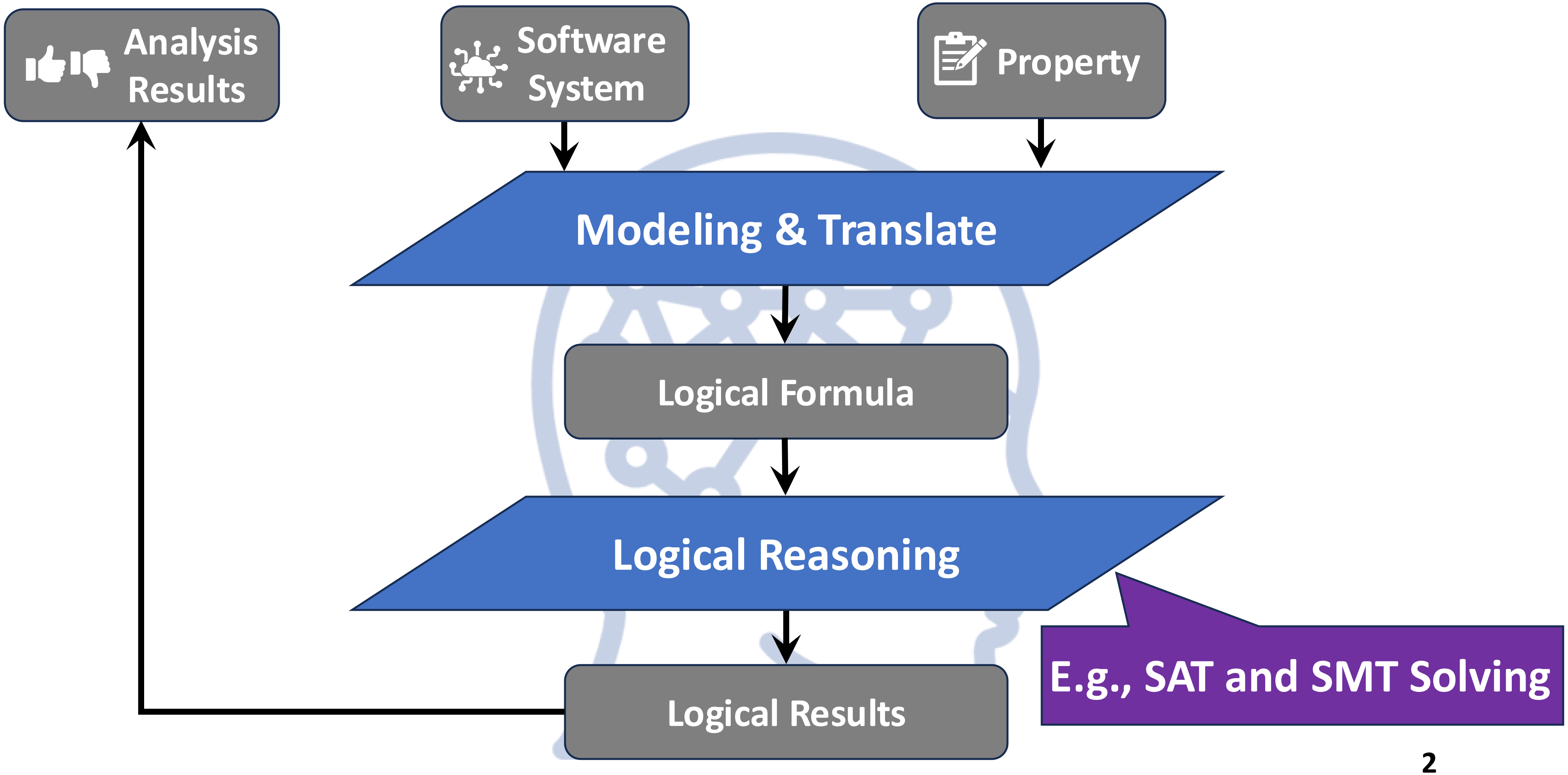
Recap Direction 1: Software Verification

Systematically and logically analyze software systems with **properties**



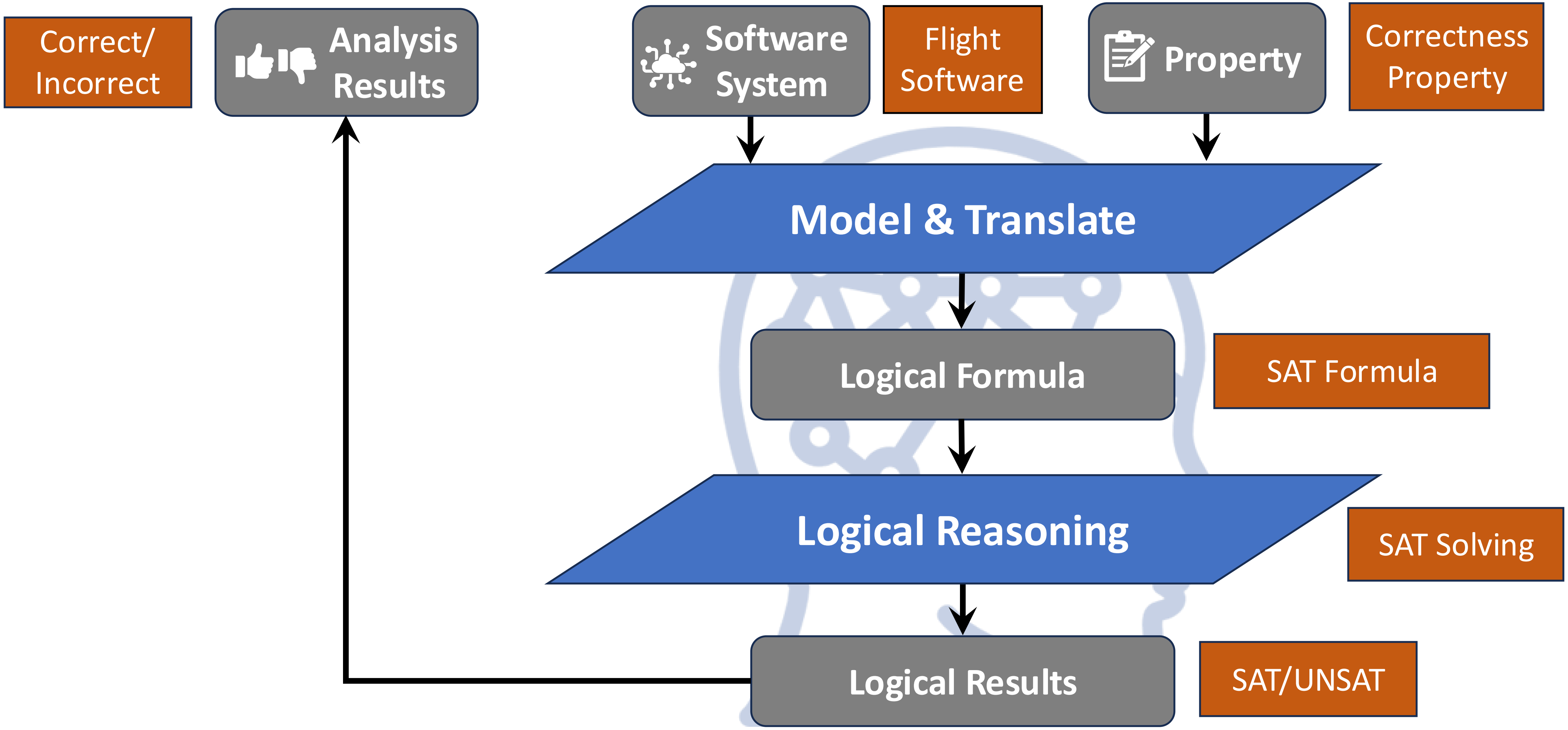
Recap Direction 1: Software Verification

Typically models software problems into logical formulas



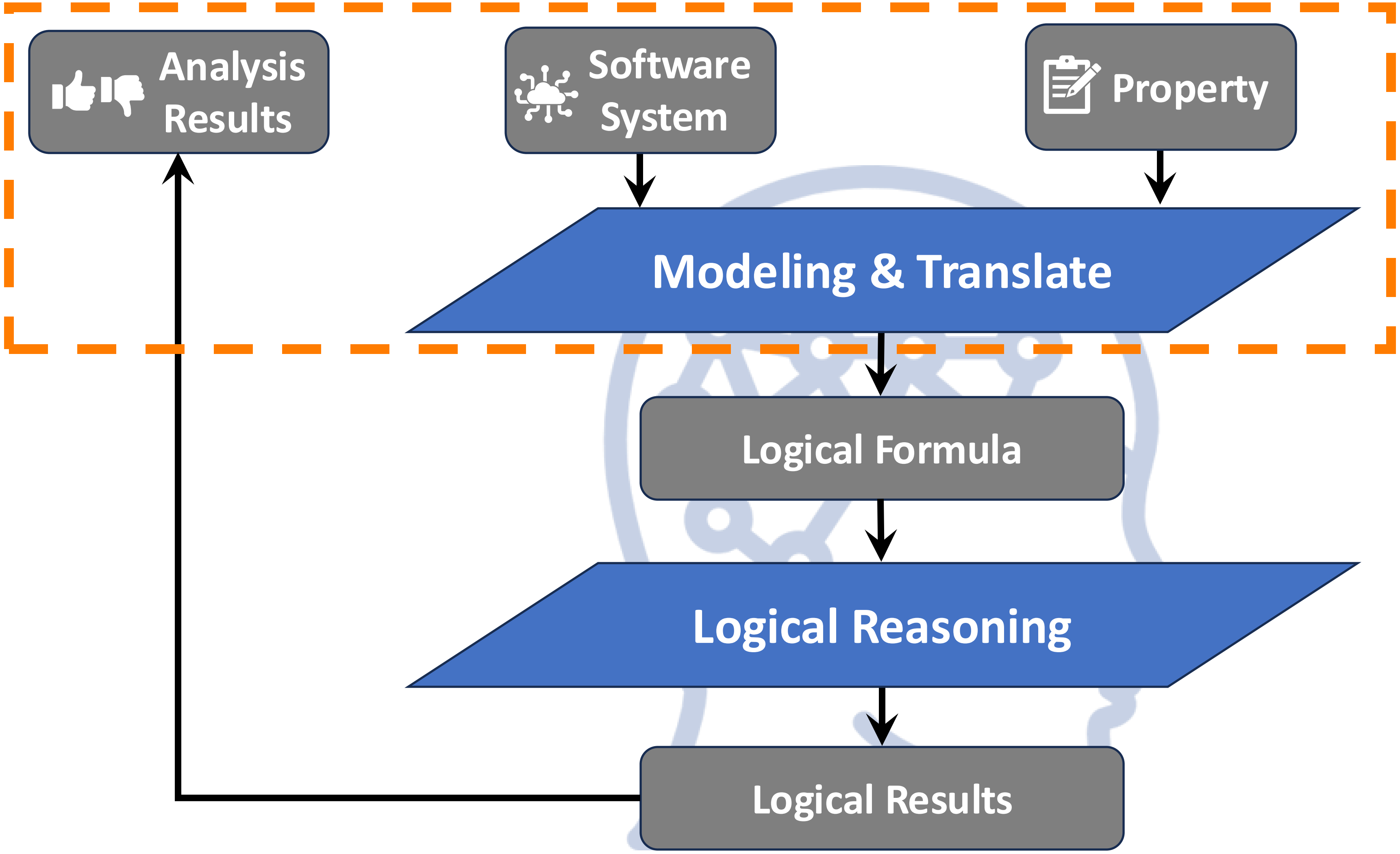
Recap — Formal Reasoning for Software Systems

For example: Flight software verification in NASA



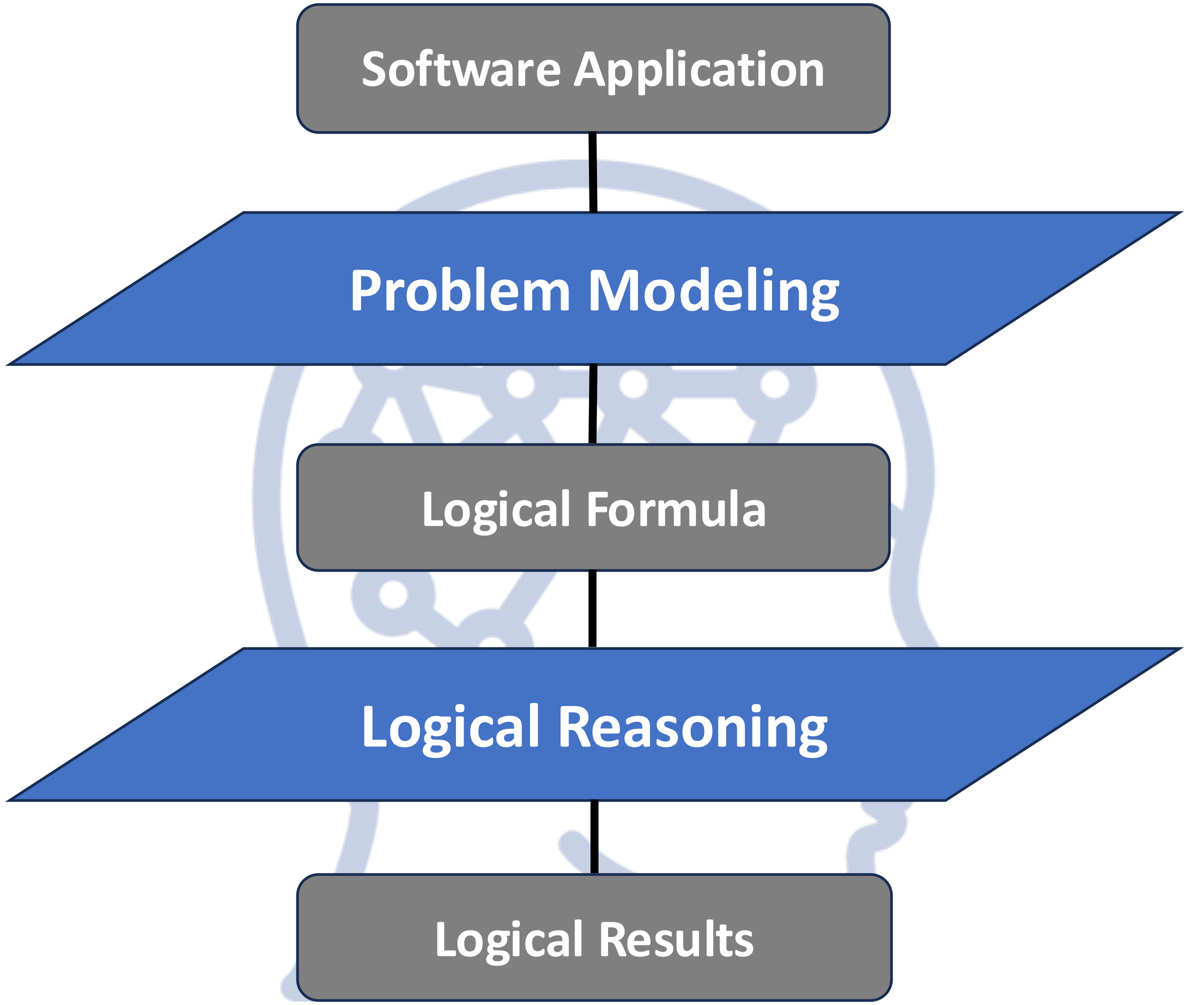
Recap Direction 1: Software Verification

Typically models software problems into logical formulas



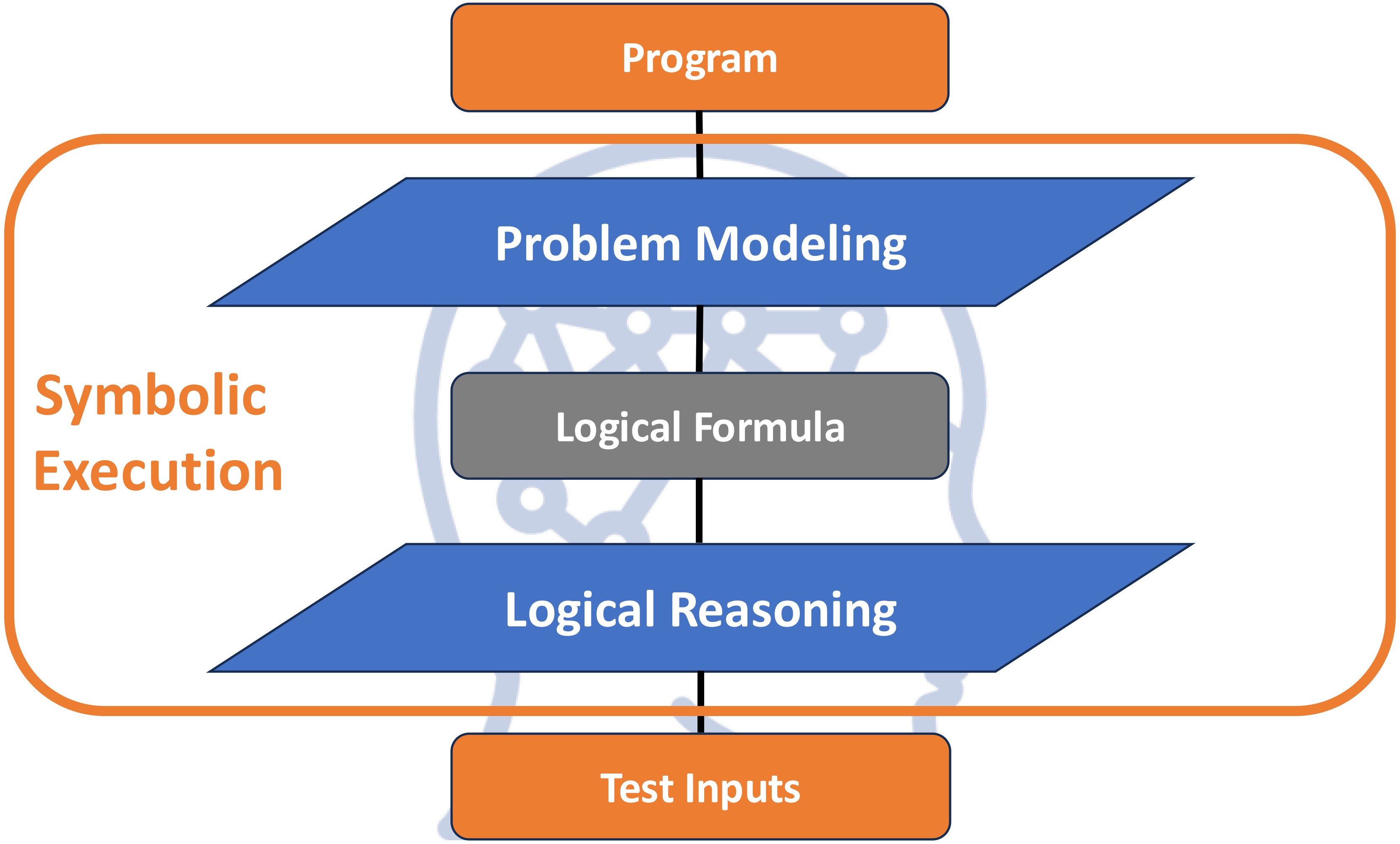
Recap Direction 1: Software Verification

Simplified view: we focus on both analysis layers

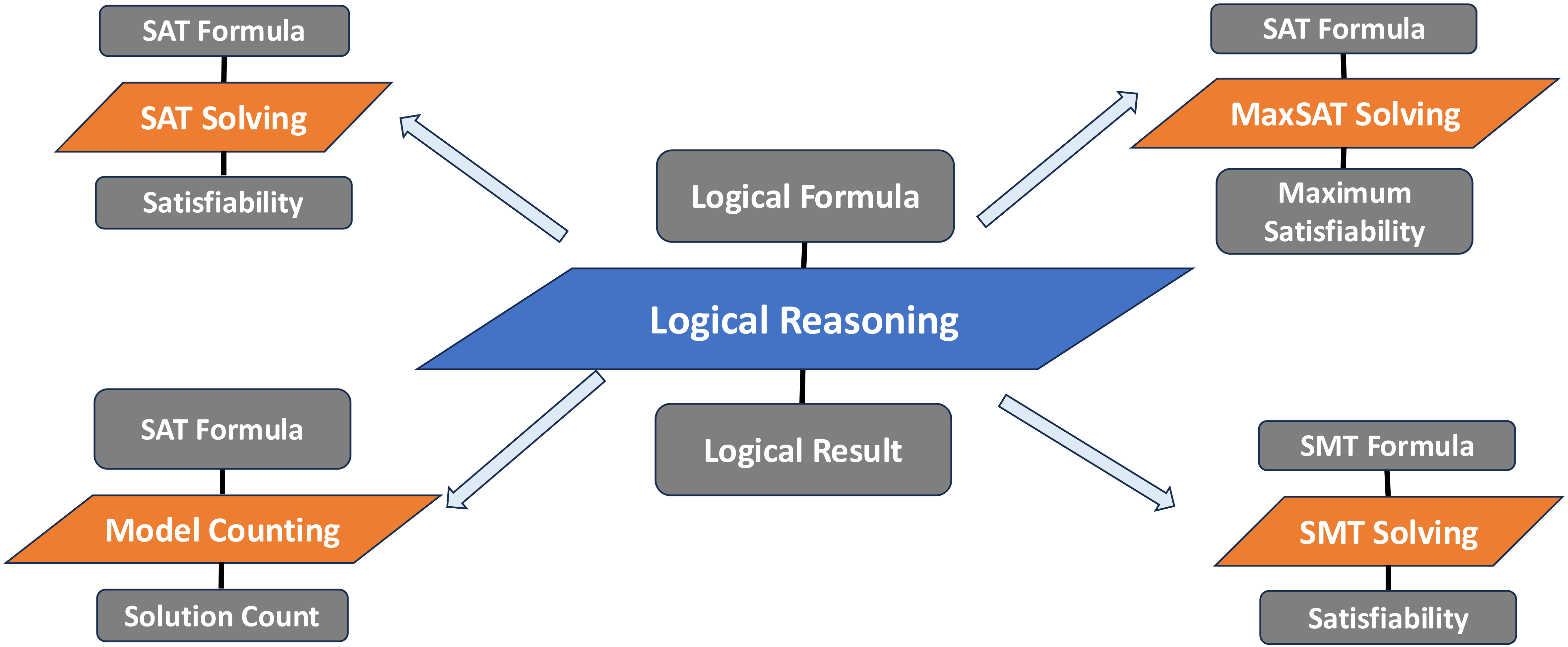


Recap Direction 1: Software Verification

Symbolic Execution

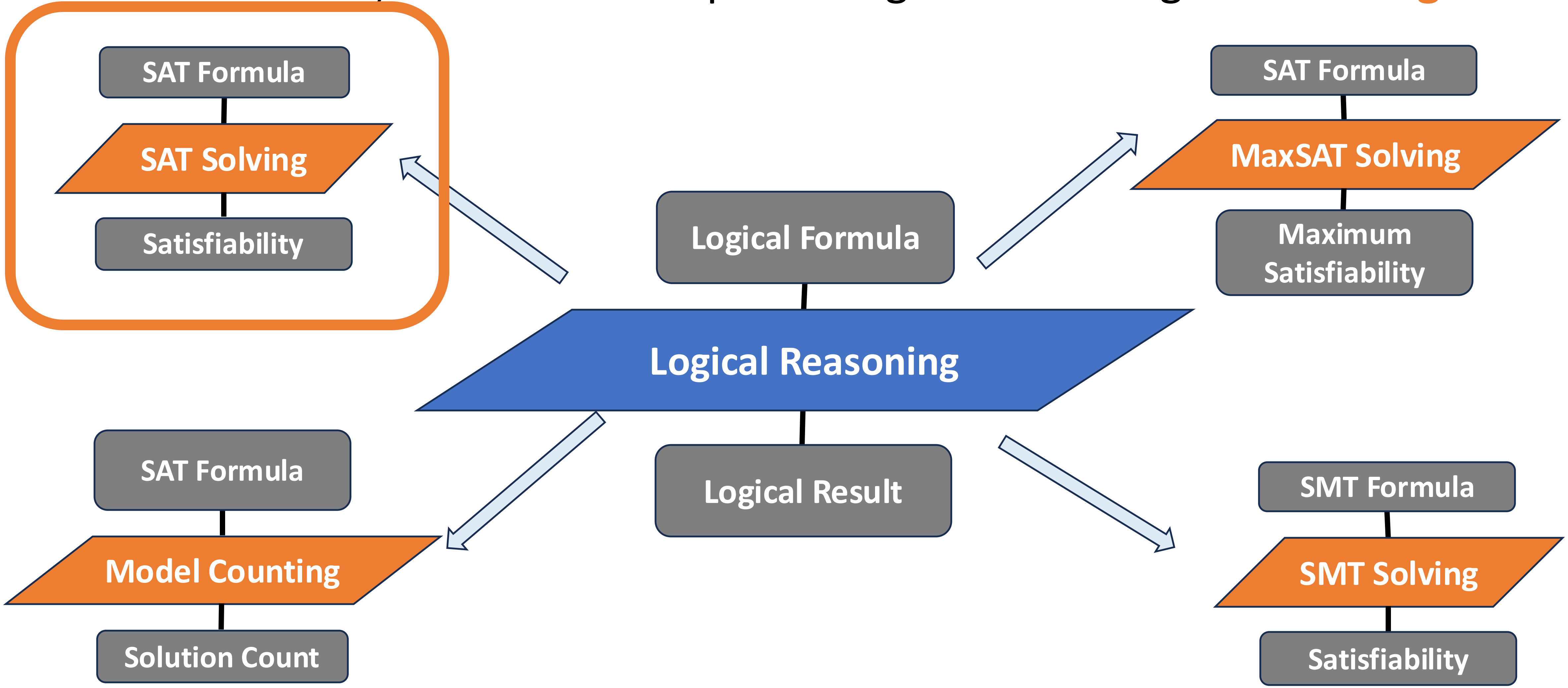


Logical Reasoning



Logical Reasoning

In this lecture, we focus on a specific logical reasoning- **SAT solving**



SAT Solving

One of the most fundamental problems in computer science

The first problem proven to be NP-complete

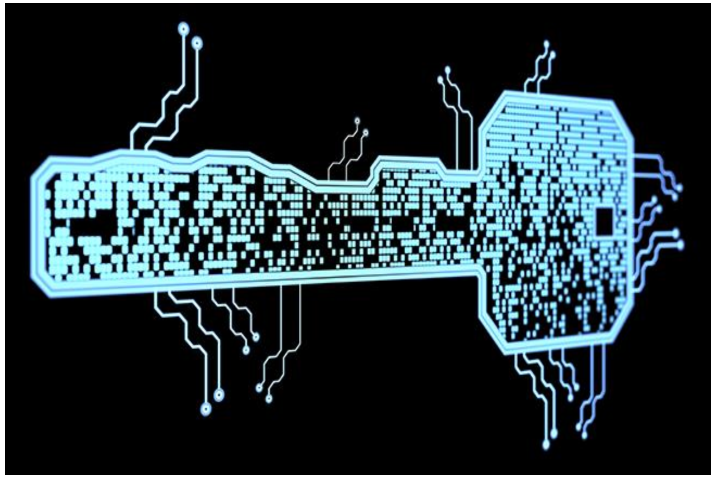
↓

Many problems in CS can be reduced to SAT

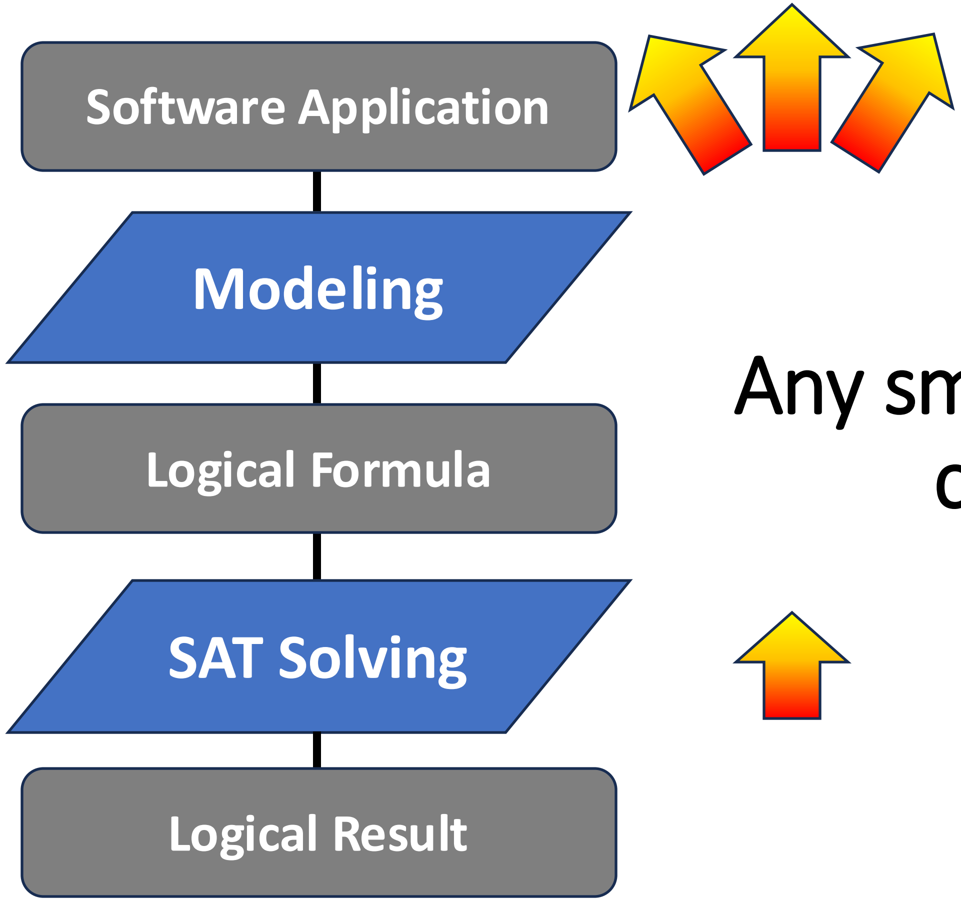
Including software and security problems

SAT Applications

Many software and security problems can be reduced to SAT



Why Improving SAT Solving is important



Any small improvement can make an essential contribution to many applications!

Input SAT formula: Boolean formula

CNF formula:

$$\phi = \underbrace{(\neg v_1 \vee \neg v_2)}_{c_1} \wedge \underbrace{(v_2 \vee v_3)}_{c_2} \wedge \underbrace{v_2}_{c_3}$$

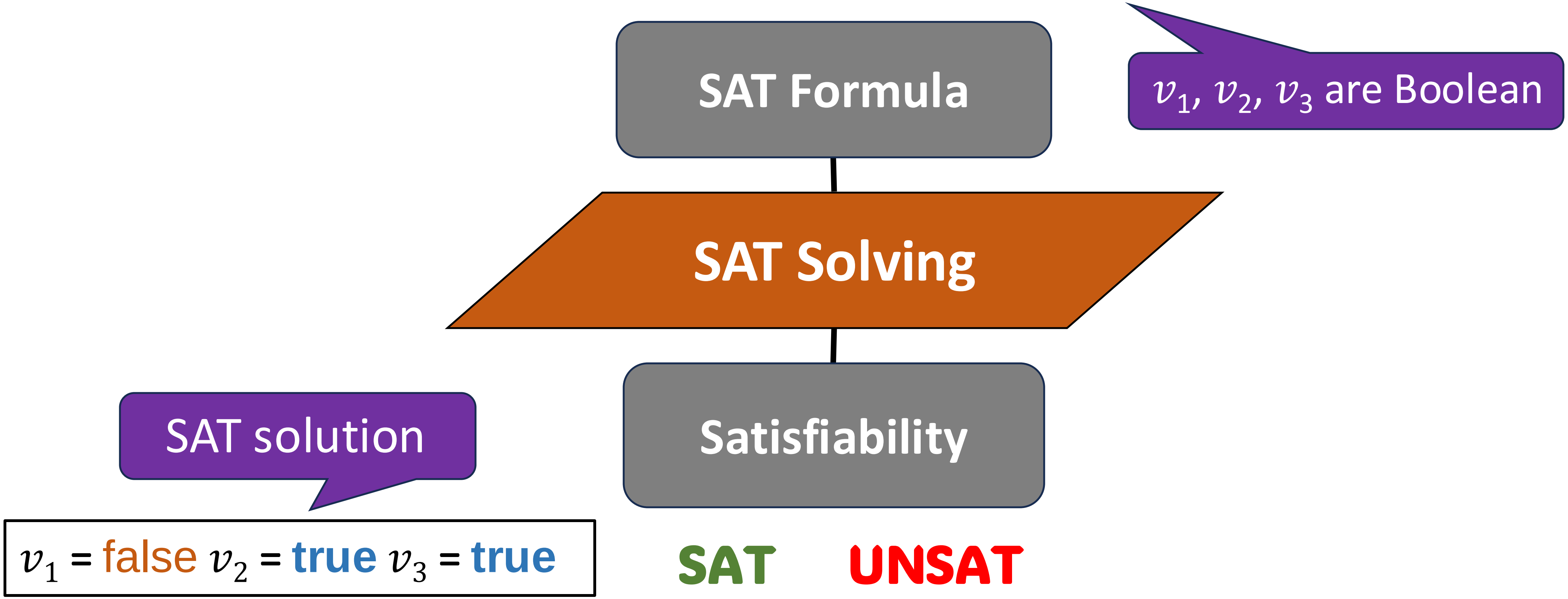
Clauses: c_1, c_2, c_3

Literals: $\neg v_1$, v_2, $\neg v_2$, v_3

Boolean variables: v_1, v_2, v_3

SAT Solving

$$\phi = (\neg v_1 \vee \neg v_2) \wedge (v_2 \vee v_3) \wedge v_2$$



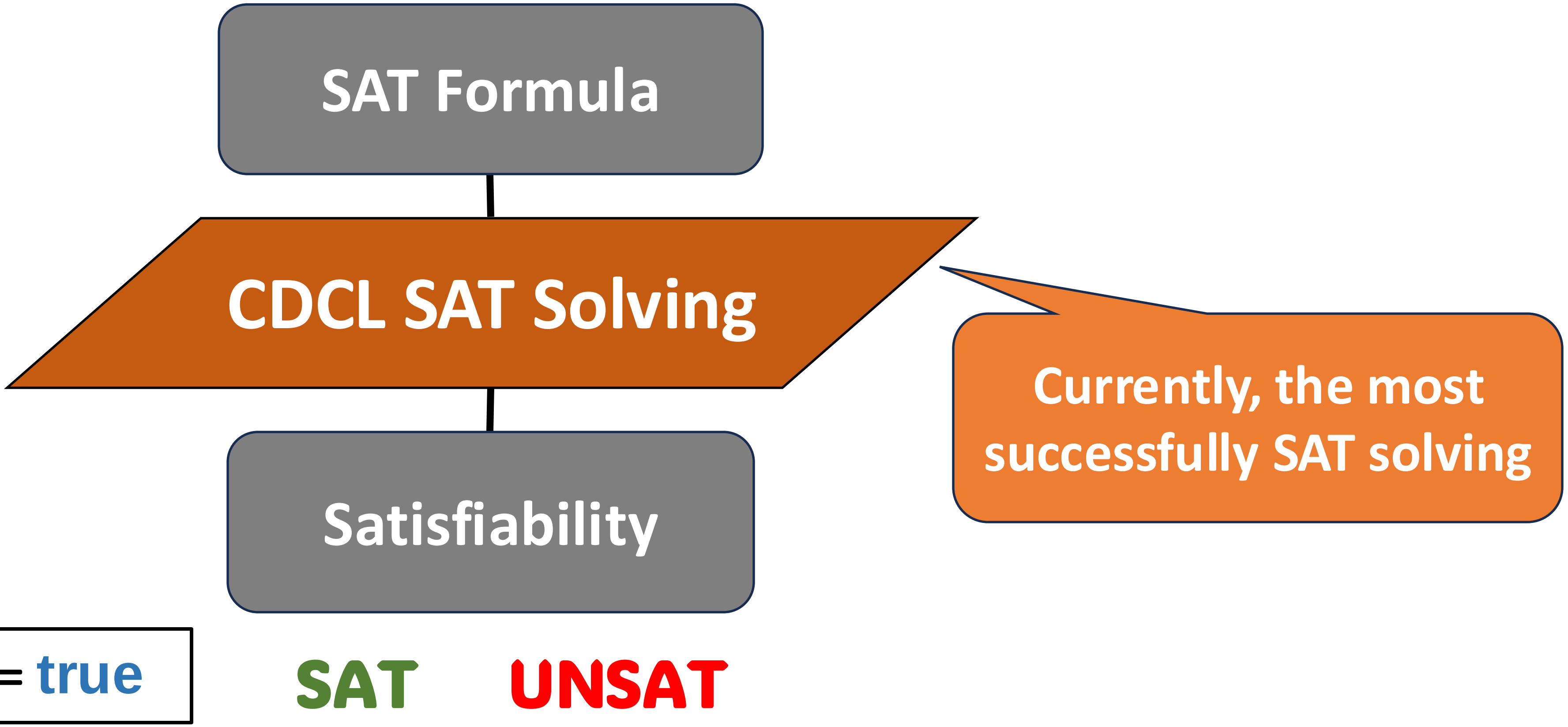
SAT Solving

Does there exist an assignment satisfying all clauses?

$$
\begin{aligned}
&(x5 \lor \neg x8 \lor x2) \land (x2 \lor x1 \lor x3) \land (x8 \lor x3 \lor x7) \land (x5 \lor x3 \lor x8) \land \\
&(x6 \lor x1 \lor \neg x5) \land (x8 \lor x9 \lor x3) \land (x2 \lor \neg x1 \lor x3) \land (x1 \lor \neg x8 \lor x4) \land \\
&(x9 \lor x6 \lor x8) \land (x8 \lor x3 \lor x9) \land (x9 \lor x3 \lor x8) \land (x6 \lor x9 \lor x5) \land \\
&(x2 \lor x3 \lor x8) \land (x8 \lor x6 \lor x3) \land (x8 \lor \neg x3 \lor x1) \land (x8 \lor x6 \lor x2) \land \\
&(x7 \lor x9 \lor \neg x2) \land (x8 \lor x9 \lor x2) \land (x1 \lor x9 \lor x4) \land (x8 \lor \neg x1 \lor x2) \land \\
&(x3 \lor \neg x4 \lor x6) \land (x1 \lor x7 \lor x5) \land (x7 \lor x1 \lor x6) \land (x5 \lor x4 \lor x6) \land \\
&(x4 \lor x9 \lor x8) \land (x2 \lor \neg x9 \lor x1) \land (x5 \lor \neg x7 \lor x1) \land (x7 \lor x9 \lor x6) \land \\
&(x2 \lor x5 \lor x4) \land (x8 \lor x4 \lor x5) \land (x5 \lor x9 \lor x3) \land (x5 \lor x7 \lor x9) \land \\
&(x2 \lor \neg x8 \lor x1) \land (x7 \lor \neg x1 \lor x5) \land (x1 \lor x4 \lor x3) \land (x1 \lor x9 \lor x4) \land \\
&(x3 \lor x5 \lor x6) \land (x6 \lor x3 \lor x9) \land (x7 \lor \neg x5 \lor x9) \land (x7 \lor \neg x5 \lor x2) \land \\
&(x4 \lor \neg x7 \lor x3) \land (x4 \lor \neg x9 \lor x7) \land (x5 \lor x1 \lor x7) \land (x5 \lor x1 \lor x7) \land \\
&(x6 \lor x7 \lor x3) \land (x8 \lor x6 \lor x7) \land (x6 \lor x2 \lor x3) \land (x8 \lor x2 \lor x5)
\end{aligned}
$$

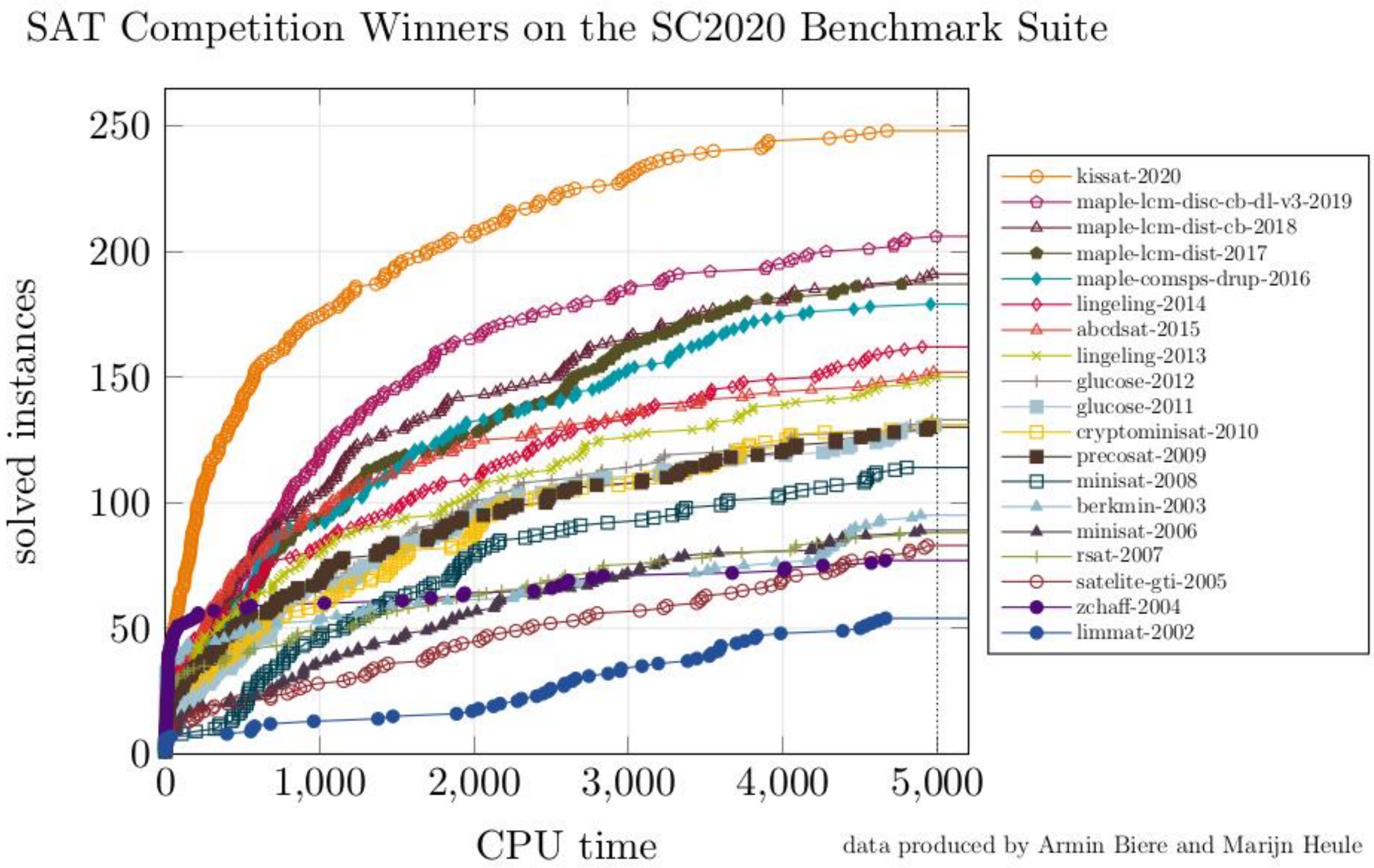
CDCL SAT solving

$$\phi = (\neg v_1 \vee \neg v_2) \wedge (v_2 \vee v_3) \wedge v_2$$



v_1 = false v_2 = true v_3 = true

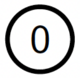
CDCL SAT solving



CDCL SAT solving

CDCL: Conflict Driven Clause Learning

$$(x_1 \vee x_4) \wedge$$
$$(x_3 \vee \overline{x}_4 \vee \overline{x}_5) \wedge$$
$$(\overline{x}_3 \vee \overline{x}_2 \vee \overline{x}_4) \wedge$$
$$\mathcal{F}_{\text{extra}}$$

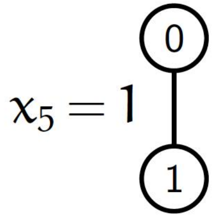


This slide is adapted from Marijn J.H. Heule's slides (https://www.cs.cmu.edu/~mheule/15816-f20/slides/CDCL.pdf).

CDCL SAT solving

CDCL: Conflict Driven Clause Learning

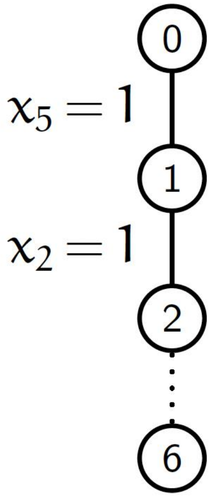
$$
\begin{aligned}
&(x_1 \lor x_4) \land \\
&(x_3 \lor \overline{x}_4 \lor \overline{x}_5) \land \\
&(\overline{x}_3 \lor \overline{x}_2 \lor \overline{x}_4) \land \\
&\mathcal{F}_{\text{extra}}
\end{aligned}
$$



CDCL SAT solving

CDCL: Conflict Driven Clause Learning

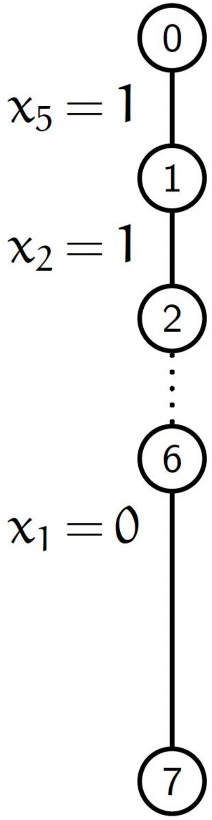
$$(x_1 \vee x_4) \wedge$$
$$(x_3 \vee \overline{x}_4 \vee \overline{x}_5) \wedge$$
$$(\overline{x}_3 \vee \overline{x}_2 \vee \overline{x}_4) \wedge$$
$$\mathcal{F}_{\text{extra}}$$



CDCL SAT solving

CDCL: Conflict Driven Clause Learning

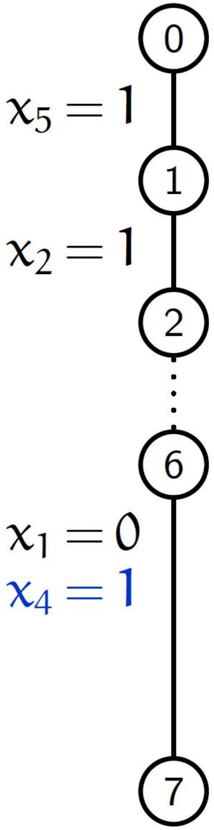
$$(x_1 \lor x_4) \land$$
$$(x_3 \lor \overline{x}_4 \lor \overline{x}_5) \land$$
$$(\overline{x}_3 \lor \overline{x}_2 \lor \overline{x}_4) \land$$
$$\mathcal{F}_{\text{extra}}$$



CDCL SAT solving

CDCL: Conflict Driven Clause Learning

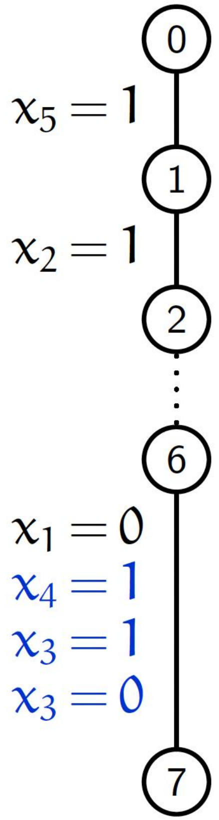
$$
\begin{aligned}
&(x_1 \lor x_4) \land \\
&(x_3 \lor \overline{x}_4 \lor \overline{x}_5) \land \\
&(\overline{x}_3 \lor \overline{x}_2 \lor \overline{x}_4) \land \\
&\mathcal{F}_{\text{extra}}
\end{aligned}
$$



CDCL SAT solving

CDCL: Conflict Driven Clause Learning

$$(x_1 \lor x_4) \land$$
$$(x_3 \lor \overline{x}_4 \lor \overline{x}_5) \land$$
$$(\overline{x}_3 \lor \overline{x}_2 \lor \overline{x}_4) \land$$
$$\mathcal{F}_{\text{extra}}$$

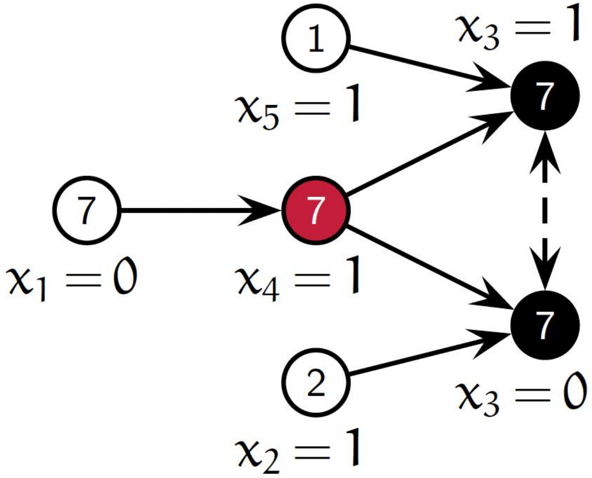


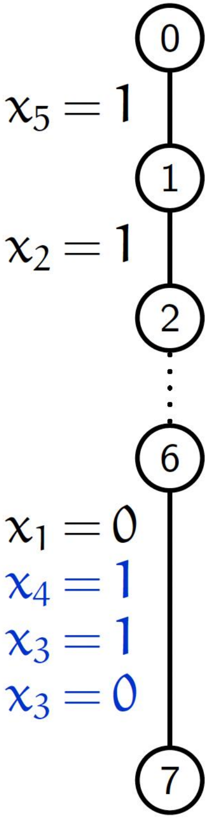
This slide is adapted from Marijn J.H. Heule's slides (https://www.cs.cmu.edu/~mheule/15816-f20/slides/CDCL.pdf).

CDCL SAT solving

CDCL: Conflict Driven Clause Learning

$(x_1 \lor x_4) \land$
$(x_3 \lor \overline{x}_4 \lor \overline{x}_5) \land$
$(\overline{x}_3 \lor \overline{x}_2 \lor \overline{x}_4) \land$
$\mathcal{F}_{\text{extra}}$

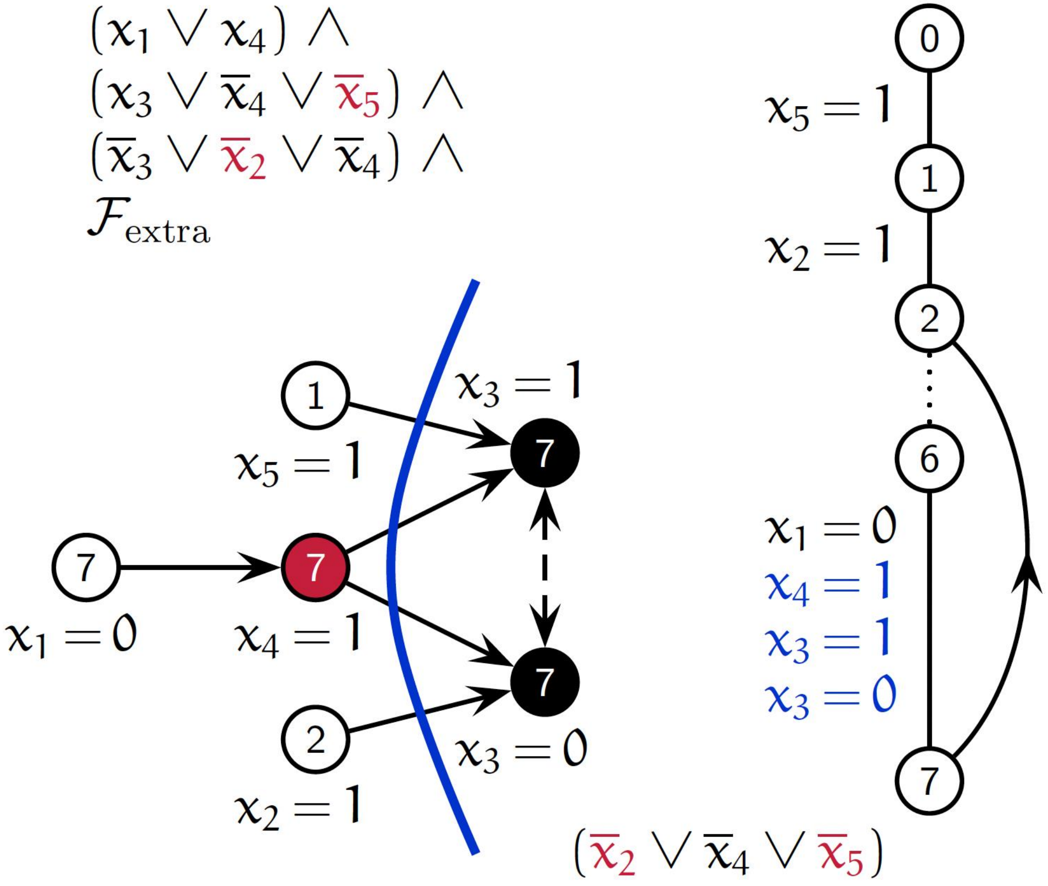




CDCL SAT solving

CDCL: Conflict Driven Clause Learning

$(x_1 \vee x_4) \wedge$
$(x_3 \vee \overline{x}_4 \vee \overline{x}_5) \wedge$
$(\overline{x}_3 \vee \overline{x}_2 \vee \overline{x}_4) \wedge$
$\mathcal{F}_{\text{extra}}$

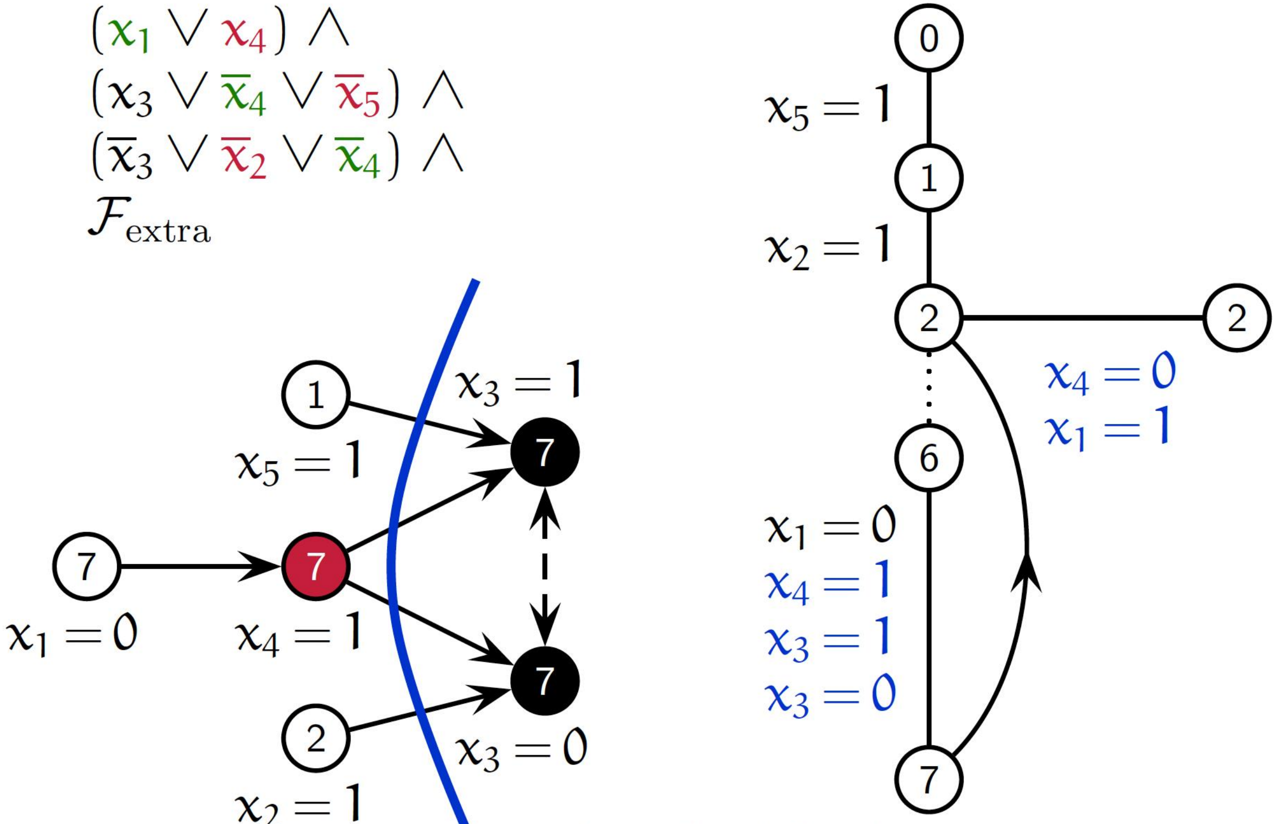


This slide is adapted from Marijn J.H. Heule's slides (https://www.cs.cmu.edu/~mheule/15816-f20/slides/CDCL.pdf).

CDCL SAT solving

CDCL: Conflict Driven Clause Learning

$(x_1 \vee x_4) \wedge$
$(x_3 \vee \overline{x}_4 \vee \overline{x}_5) \wedge$
$(\overline{x}_3 \vee \overline{x}_2 \vee \overline{x}_4) \wedge$
$\mathcal{F}_{\text{extra}}$

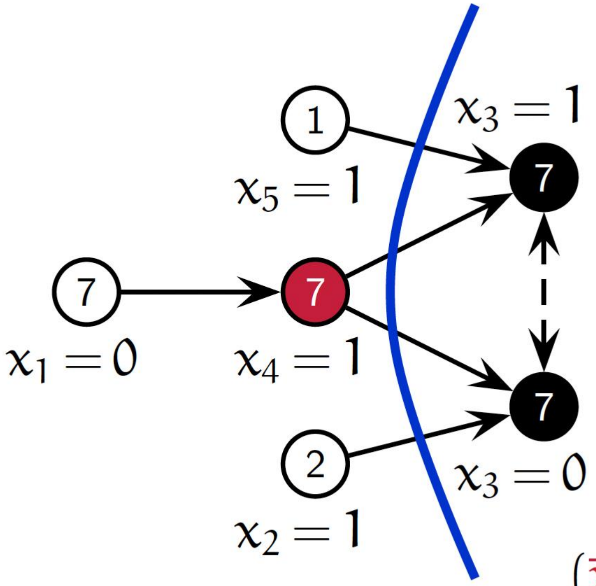


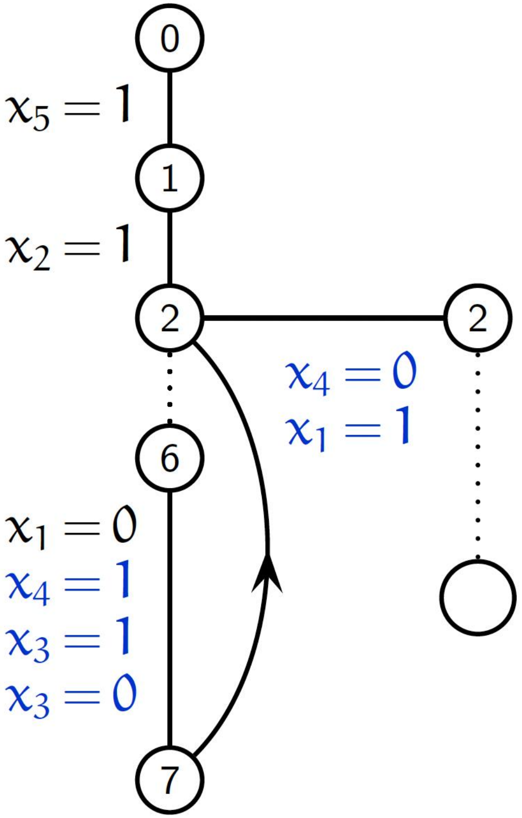
$(\overline{x}_2 \vee \overline{x}_4 \vee \overline{x}_5)$

CDCL SAT solving

CDCL: Conflict Driven Clause Learning

$$(x_1 \lor x_4) \land$$
$$(x_3 \lor \bar{x}_4 \lor \bar{x}_5) \land$$
$$(\bar{x}_3 \lor \bar{x}_2 \lor \bar{x}_4) \land$$
$$\mathcal{F}_{\text{extra}}$$





$$(\bar{x}_2 \lor \bar{x}_4 \lor \bar{x}_5)$$

CDCL SAT solving

General Algorithm

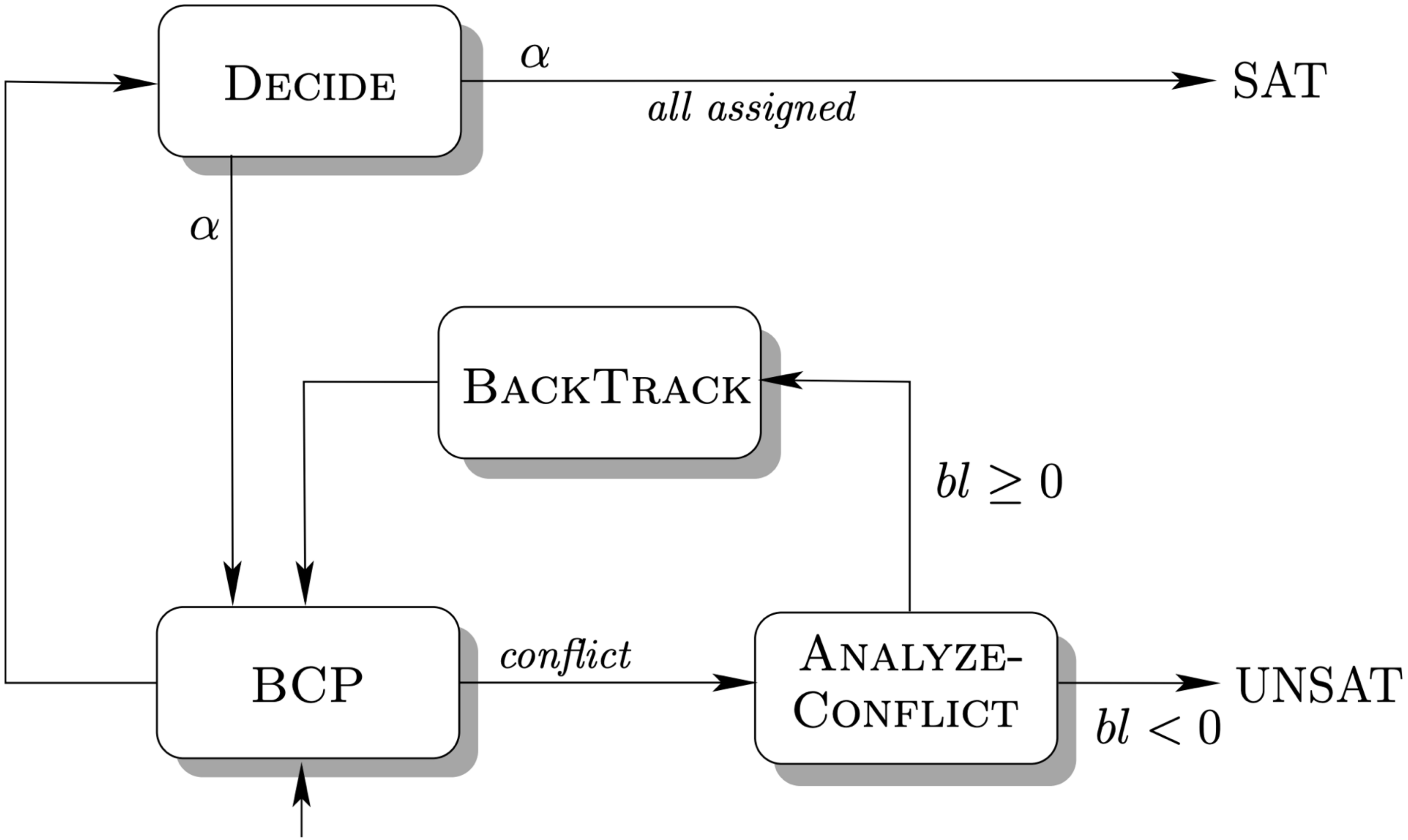
Input: A propositional CNF formula $\mathcal{B}$

Output: "Satisfiable" if the formula is satisfiable and "Unsatisfiable" otherwise

```
function CDCL
    while (TRUE) do
        while (BCP() = "conflict") do
            backtrack-level := ANALYZE-CONFLICT();
            if backtrack-level < 0 then return "Unsatisfiable";
            BackTrack(backtrack-level);
        if ¬DECIDE() then return "Satisfiable";
```

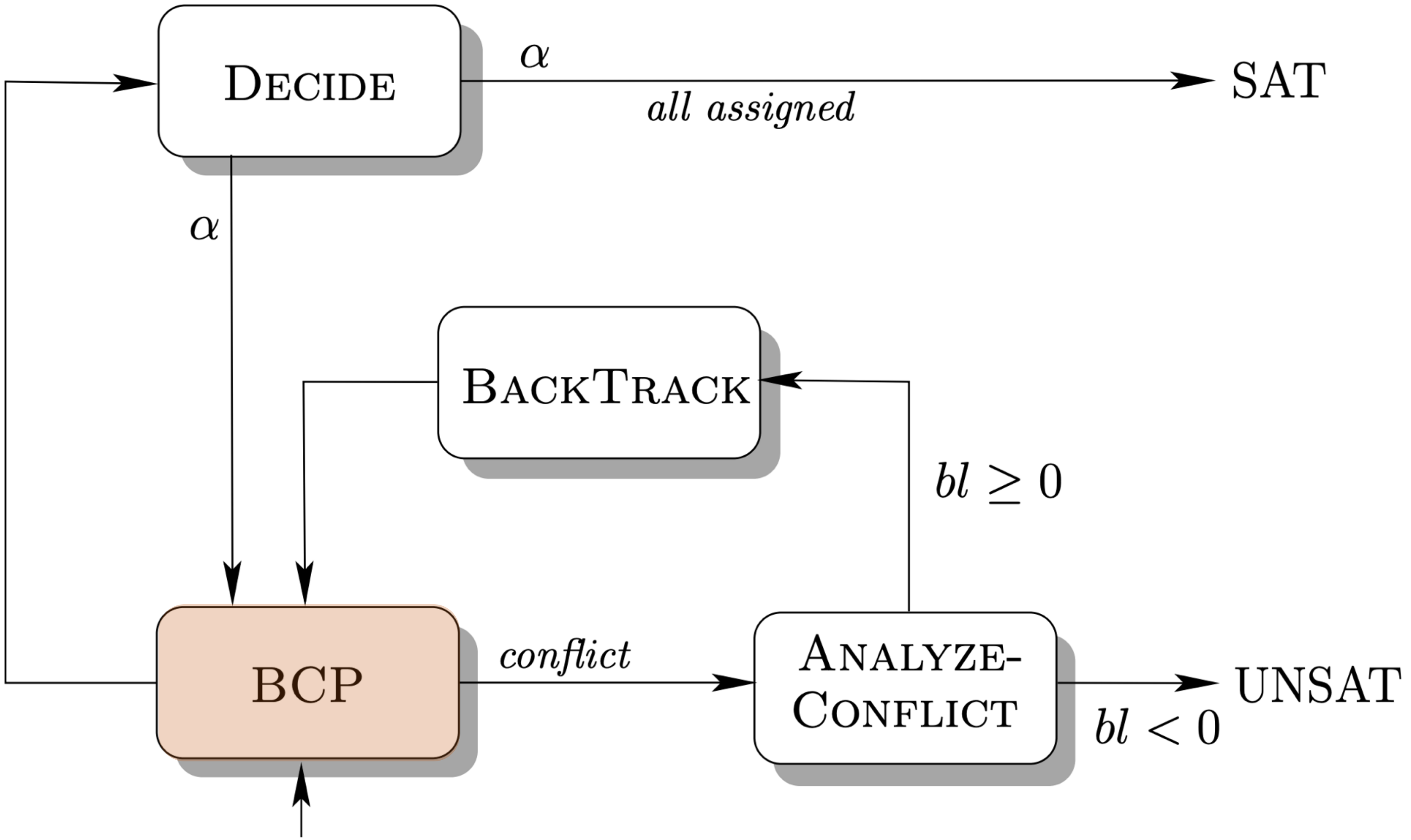
CDCL SAT solving

General Workflow



CDCL SAT solving

General Workflow



BCP: Boolean Constraint Propagation

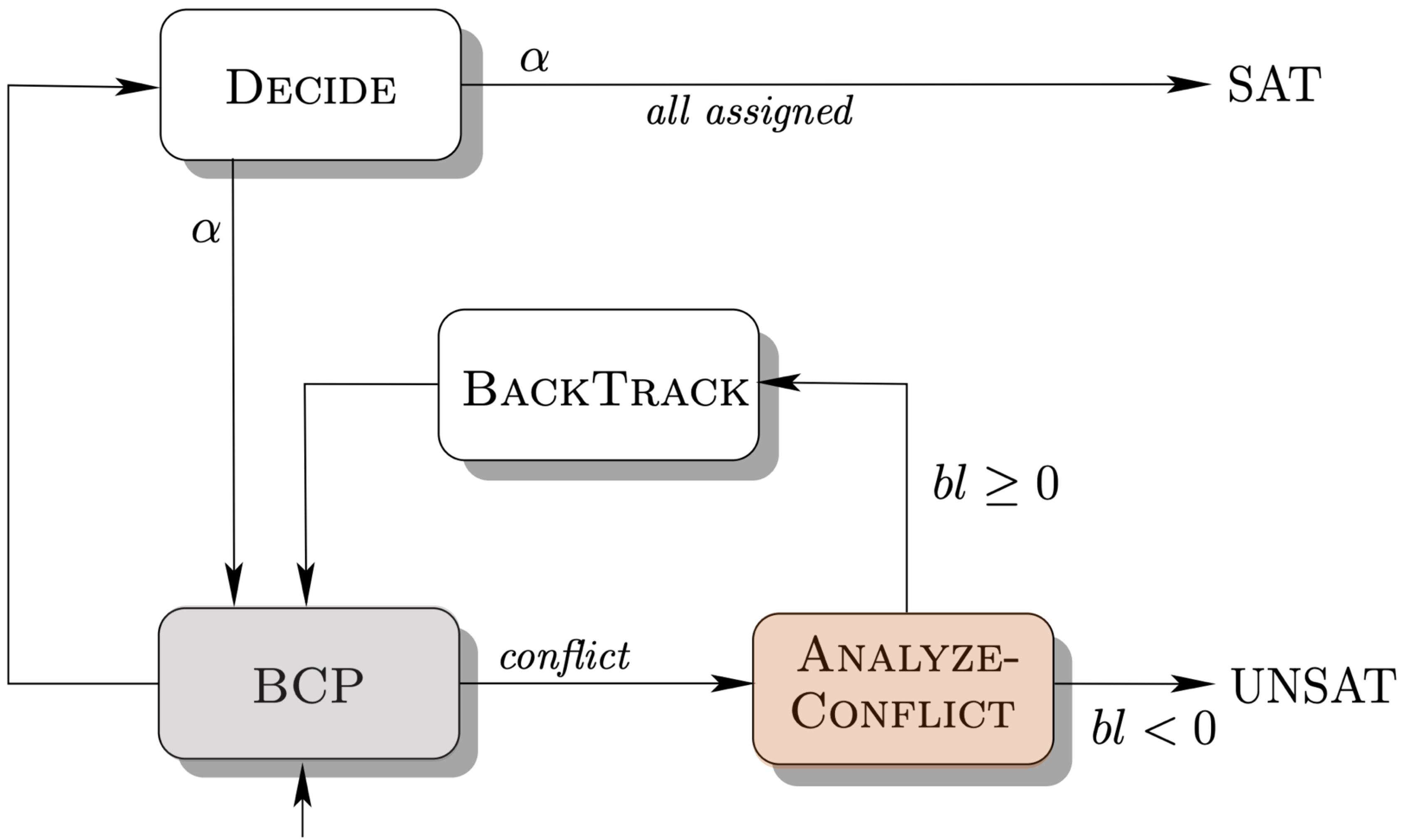
Unit Propagation

Unit Clause: x1 V ¬x2 V x3 V x4 V ... V xn

Clause: x1 V ¬x2 V x3 V x4 V ... V **xn**

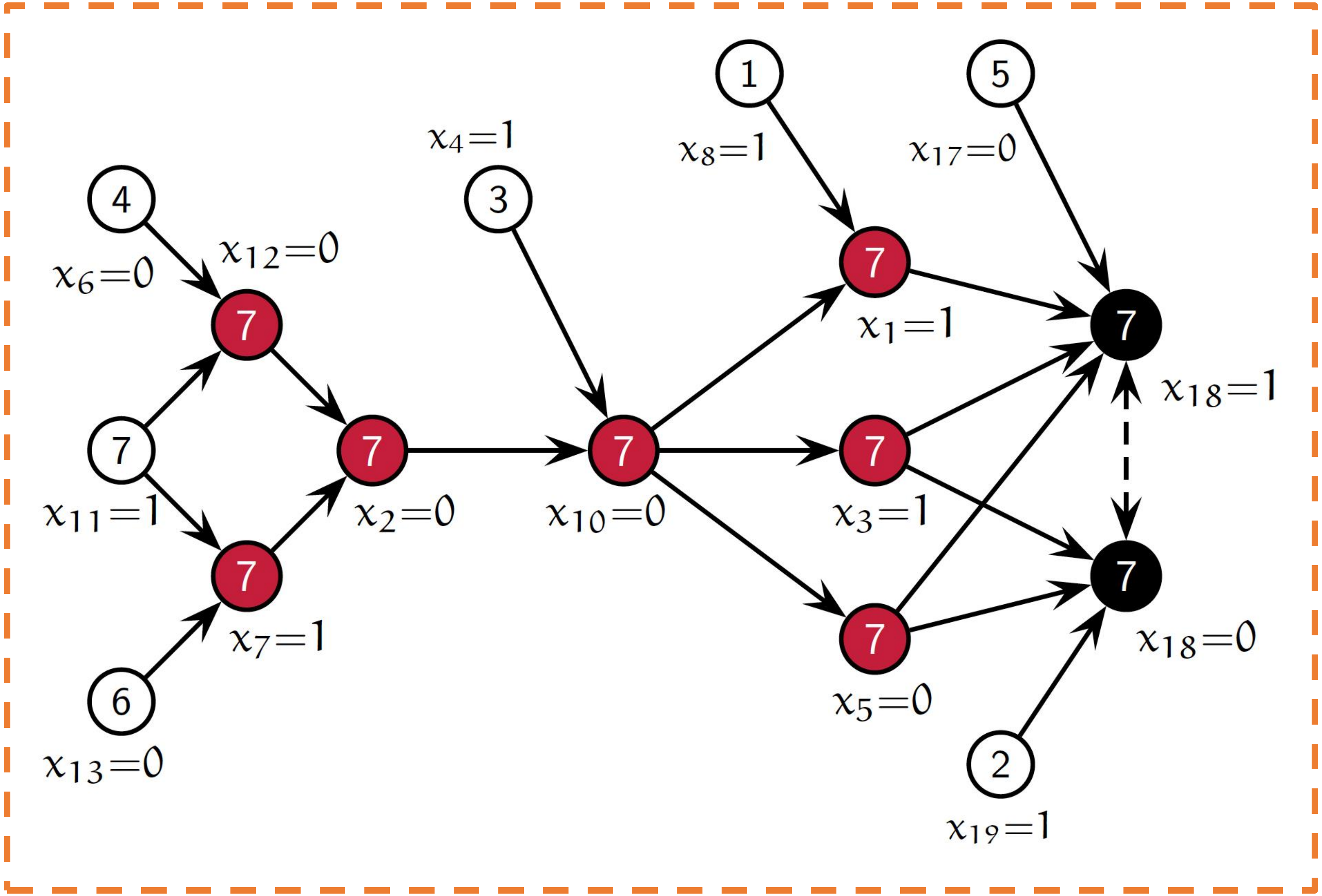
CDCL SAT solving

General Workflow



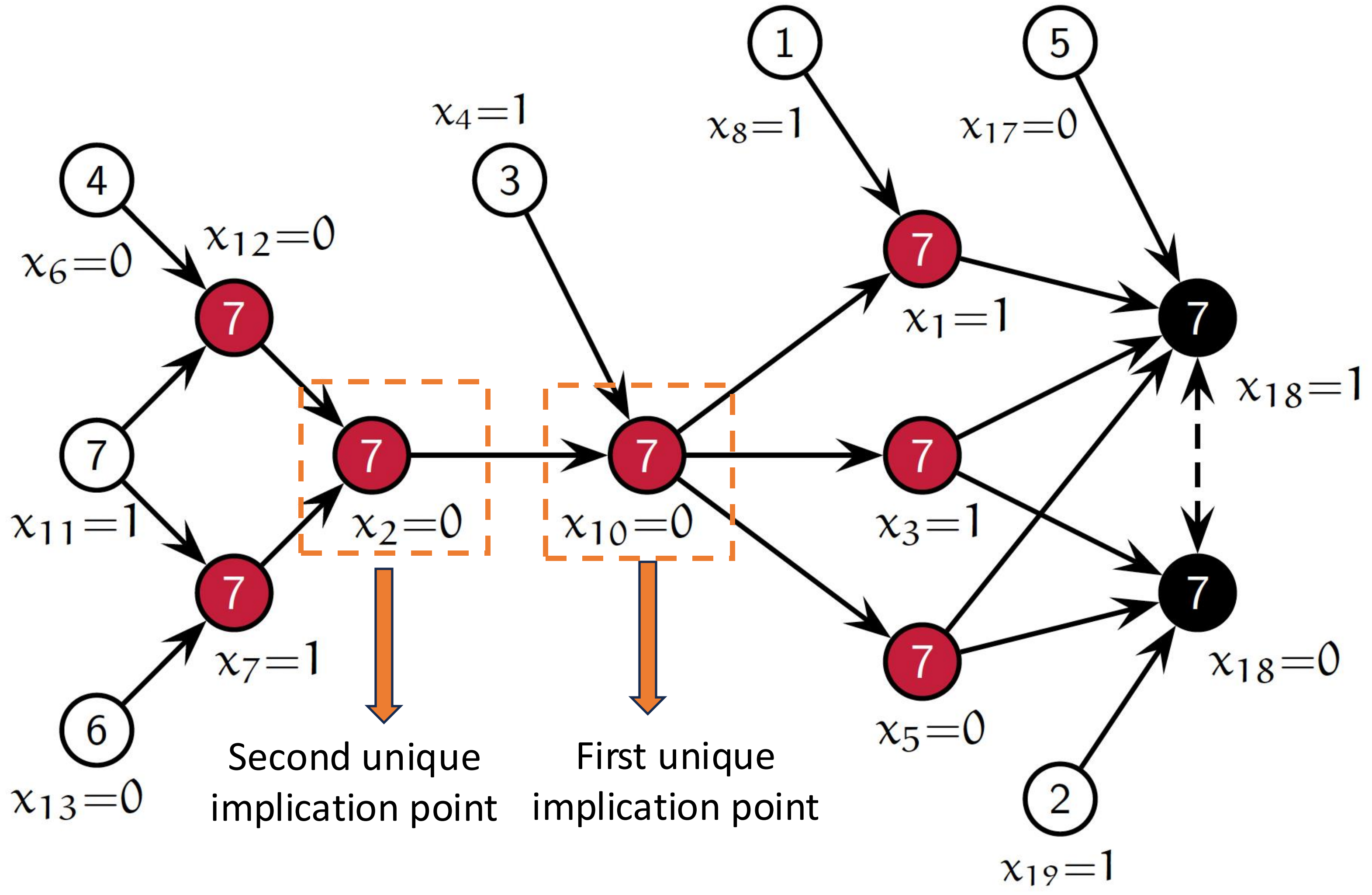
Conflict Analysis-learning a conflict clause

Implication Graph



This slide is adapted from Marijn J.H. Heule's slides (https://www.cs.cmu.edu/~mheule/15816-f20/slides/CDCL.pdf).

Conflict Analysis-learning a conflict clause

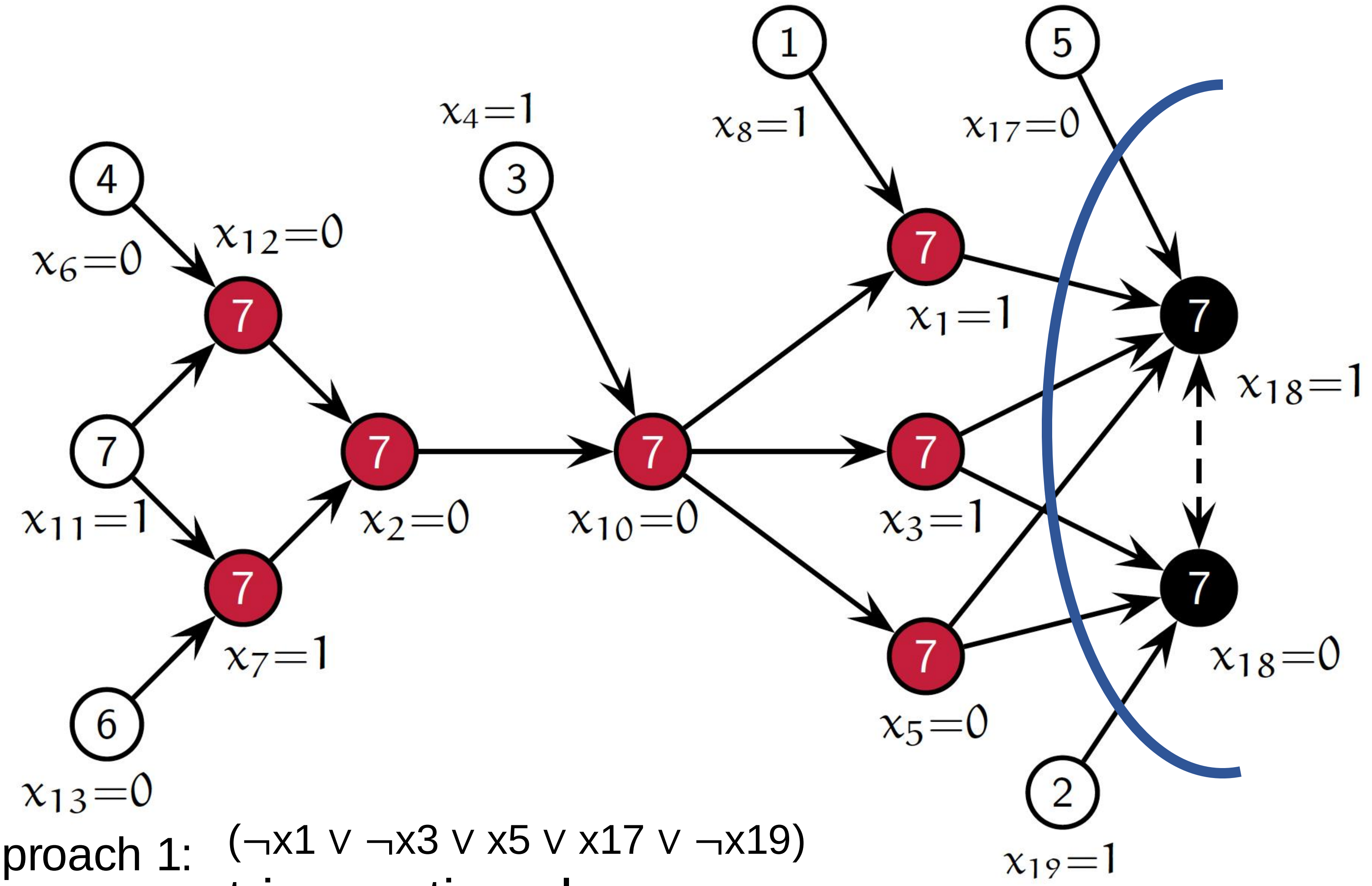


UIP: any node other than the conflict node that is on all paths from the decision node to the conflict node

Dominate the conflict nodes

This slide is adapted from Marijn J.H. Heule's slides (https://www.cs.cmu.edu/~mheule/15816-f20/slides/CDCL.pdf).

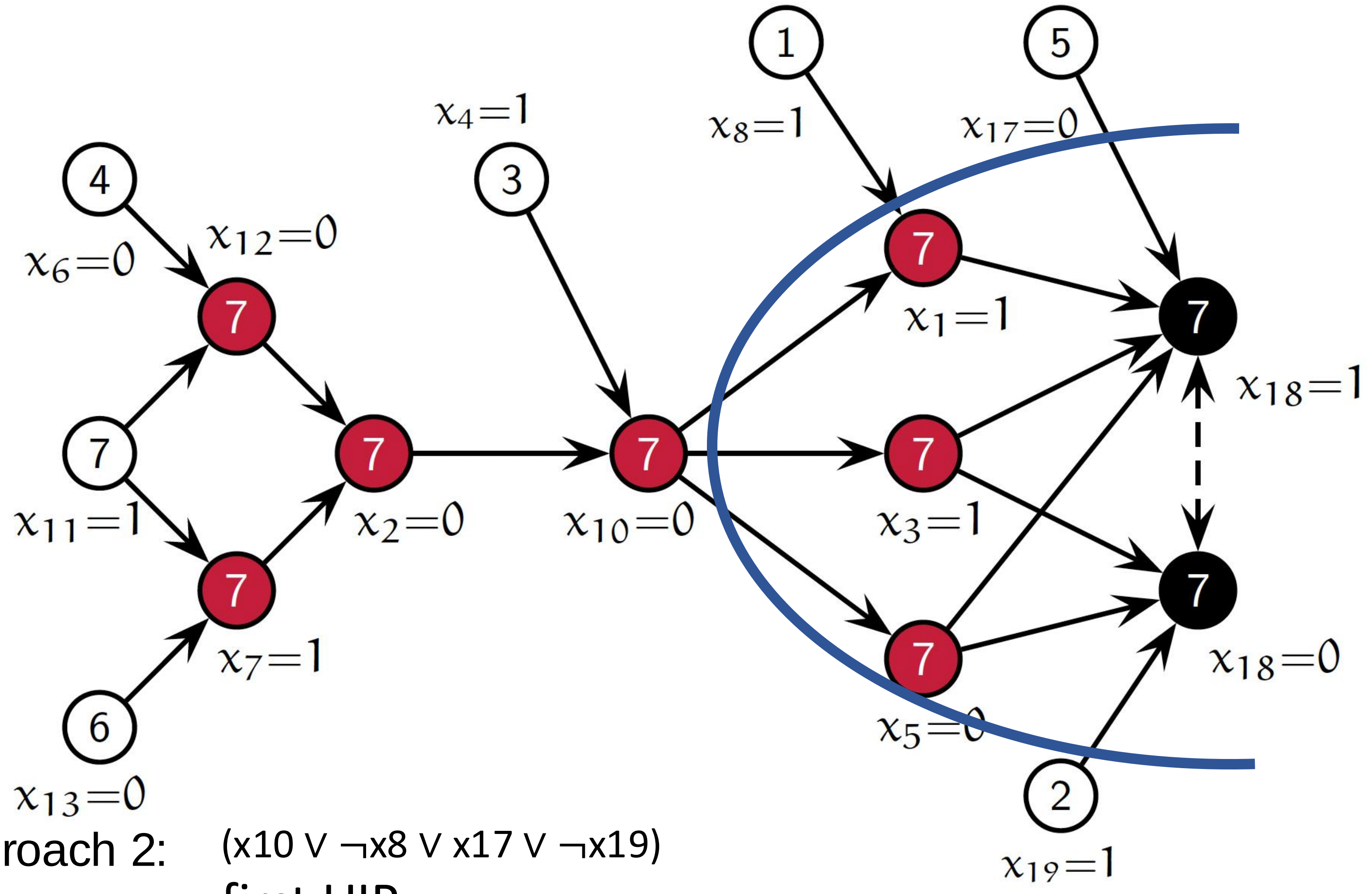
Conflict Analysis-learning a conflict clause



Approach 1: (¬x1 ∨ ¬x3 ∨ x5 ∨ x17 ∨ ¬x19)
tri-asserting clause

This slide is adapted from Marijn J.H. Heule's slides (https://www.cs.cmu.edu/~mheule/15816-f20/slides/CDCL.pdf).

Conflict Analysis-learning a conflict clause

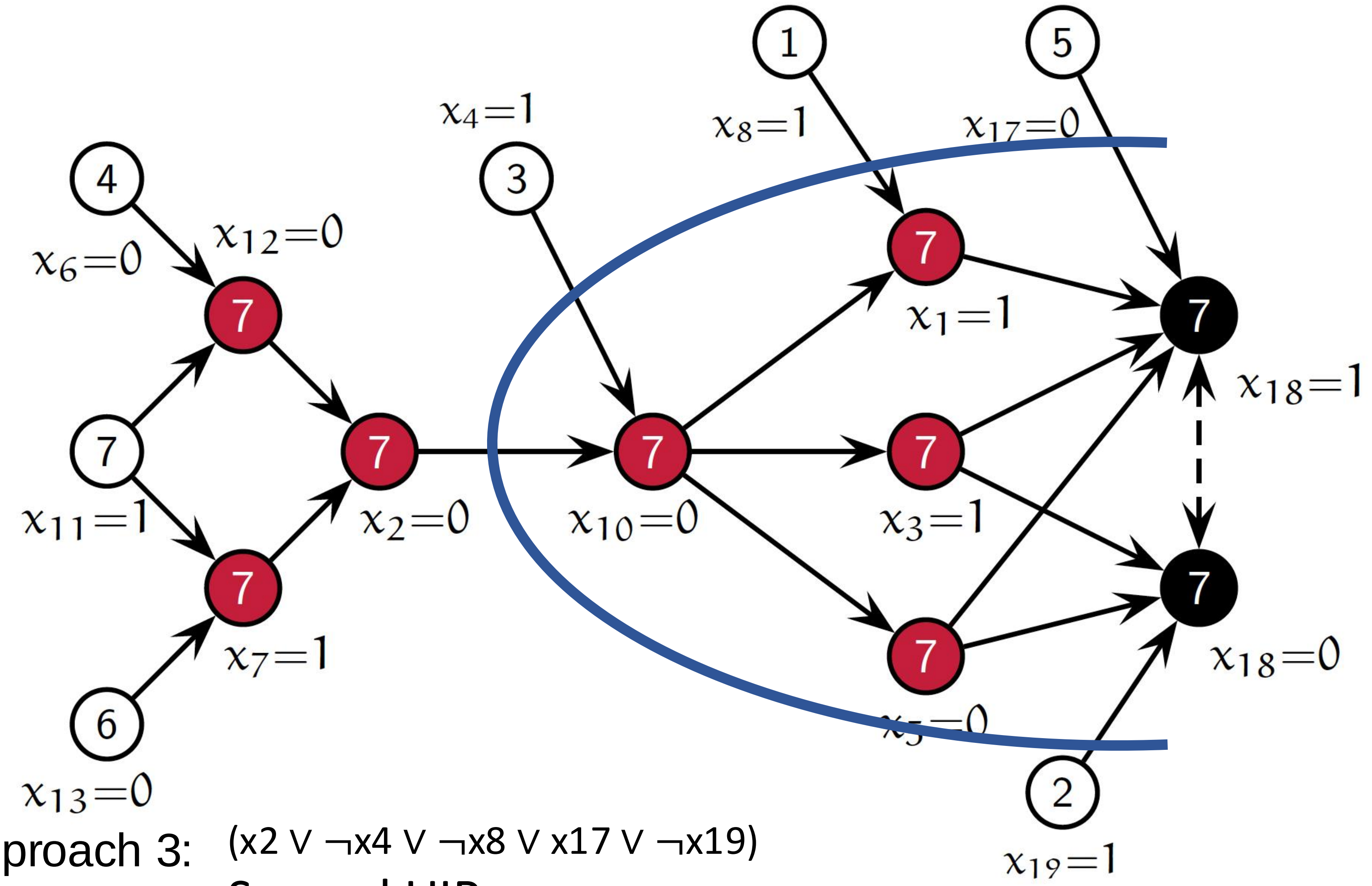


Approach 2: $(x10 \vee \neg x8 \vee x17 \vee \neg x19)$

first UIP

This slide is adapted from Marijn J.H. Heule's slides (https://www.cs.cmu.edu/~mheule/15816-f20/slides/CDCL.pdf).

Conflict Analysis-learning a conflict clause

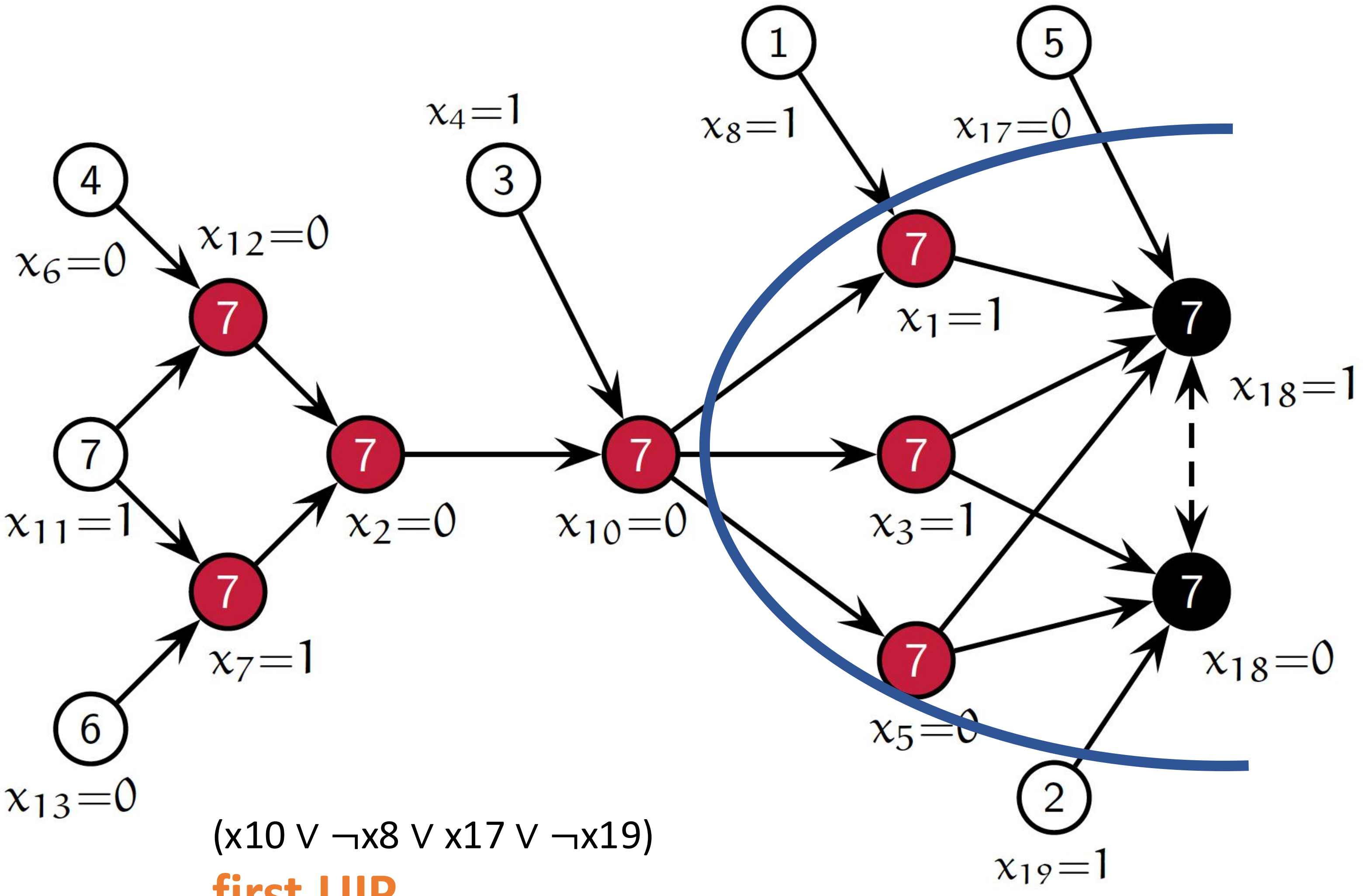


Approach 3: (x2 ∨ ¬x4 ∨ ¬x8 ∨ x17 ∨ ¬x19)

Second UIP

This slide is adapted from Marijn J.H. Heule's slides (https://www.cs.cmu.edu/~mheule/15816-f20/slides/CDCL.pdf).

Conflict Analysis-learning a conflict clause



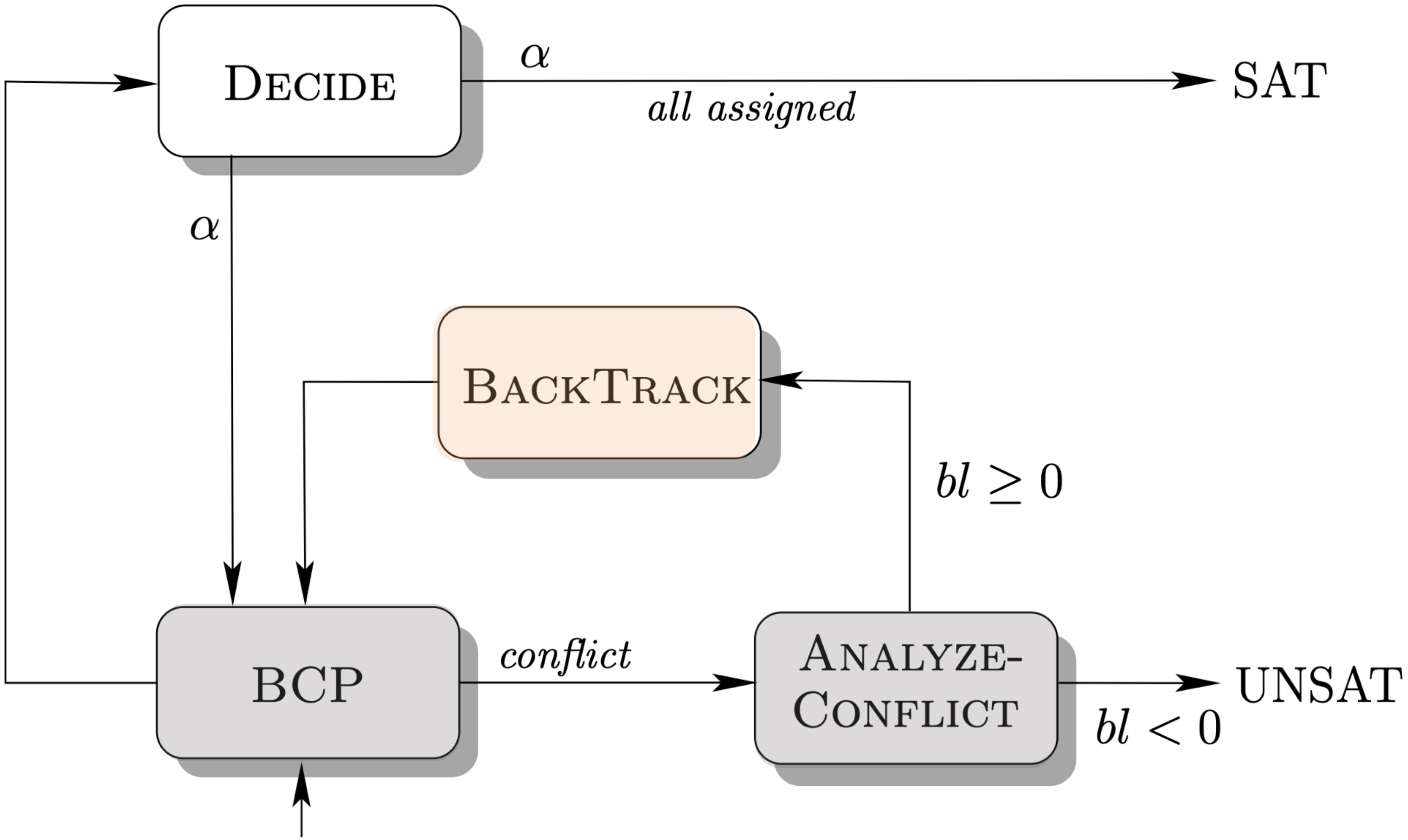
(x10 V ¬x8 V x17 V ¬x19)

first UIP

1. **Low computational cost (nearest to the conflict node)**
2. **Backtrack to the lowest decision level**

CDCL SAT solving

General Workflow

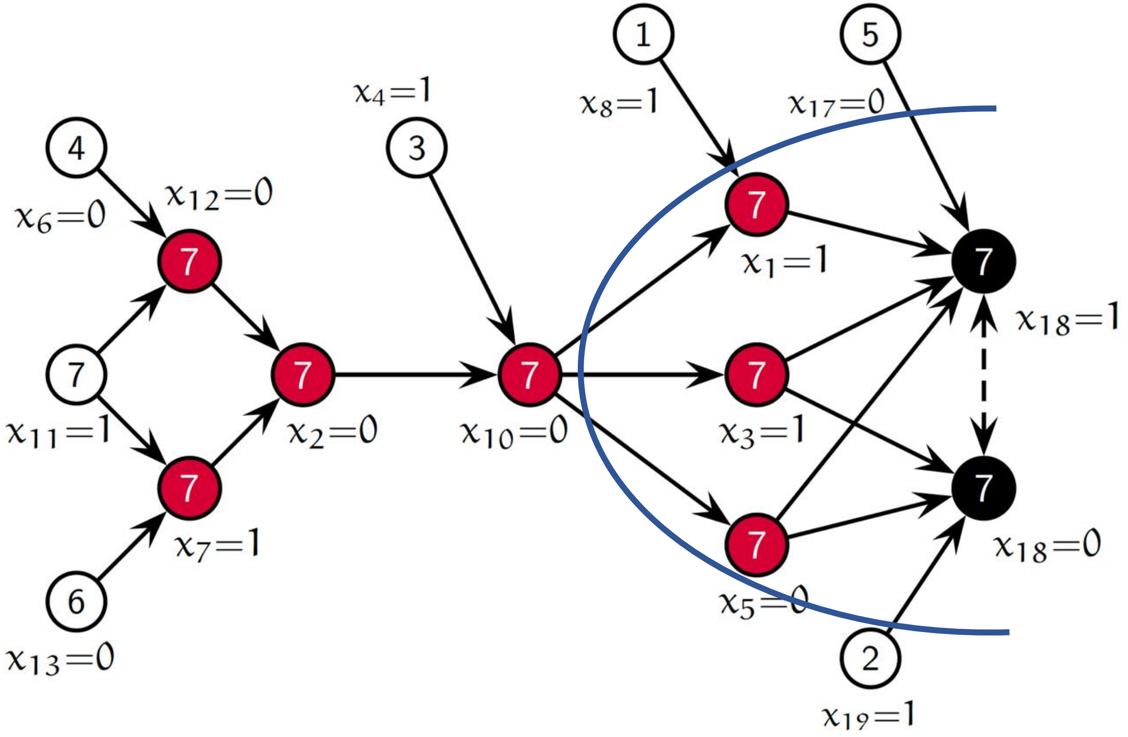


Backtrack using the learned conflict clause

Conflict clause: first_UIP ∨ l1 ∨ l2 ∨ … ∨ ln

Maximum decision level

Backtrack level

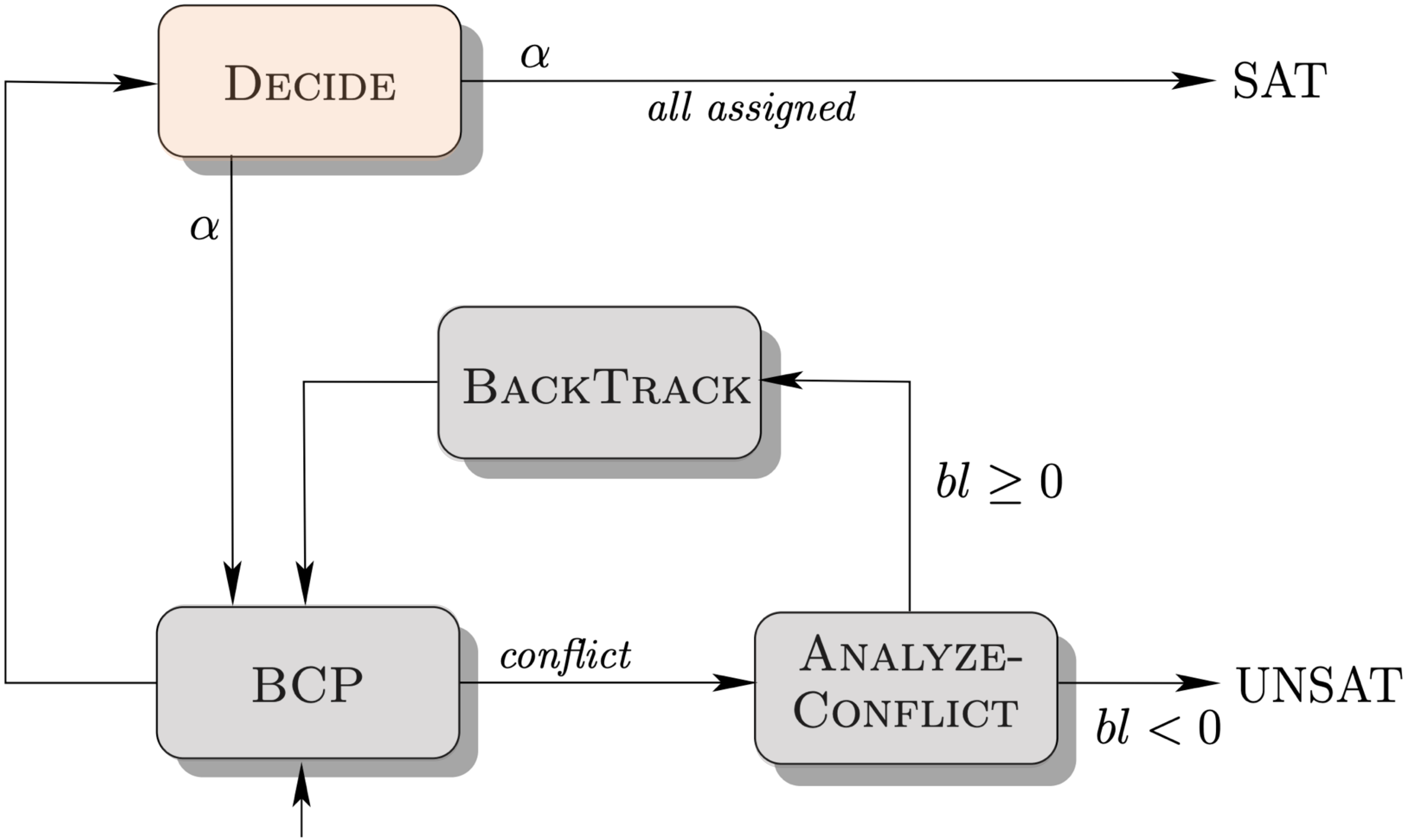


(x10 ∨ ¬x8 ∨ x17 ∨ ¬x19)

Backtrack level: 5

CDCL SAT solving

General Workflow



Decision Heuristics

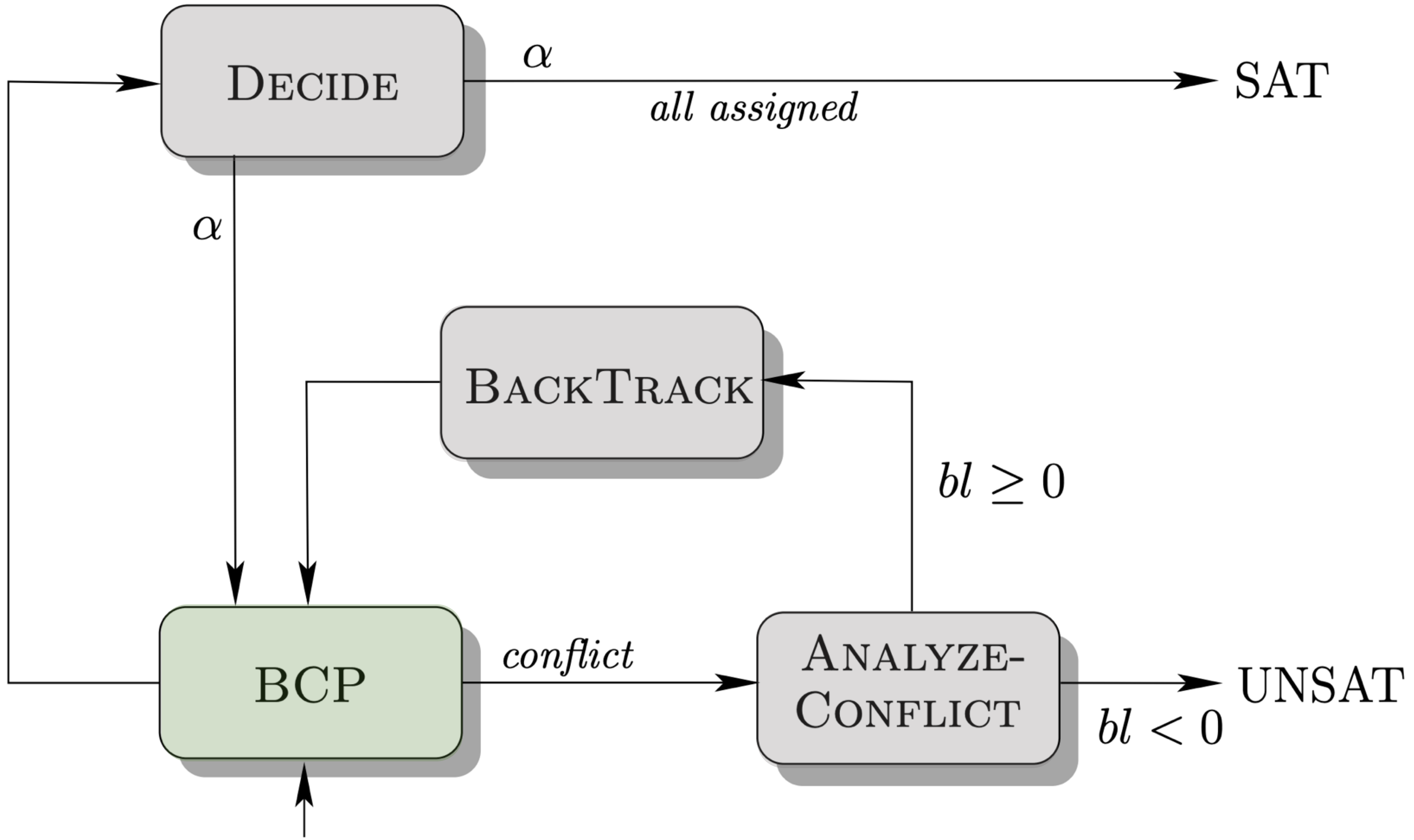
1. Variable selection heuristics
   aim: minimize the search space
   plus: could compensate a bad value selection

2. Value selection heuristics
   aim: guide search towards a solution or conflict
   plus: could compensate a bad variable selection, cache solutions of subproblems [PipatsrisawatDarwiche'07]

This slide is adapted from Marijn J.H. Heule's slides (https://www.cs.cmu.edu/~mheule/15816-f20/slides/CDCL.pdf).

CDCL SAT solving

Implementation?



Implementation: Two watched literal Scheme

Introduced by the SAT solver Chaff [1]

- Remember: Unit propagation fires when all but one literal is assigned false
- Idea: If **two** variables are either unassigned or assigned true, no need to do anything.
- So just find two variables which satisfy this condition.
- If can’t find two, do the unit propagate or a conflict is found

[1] Chaff: Engineering an Efficient SAT Solver by Moskewicz, Madigan, Zhao, Zhang, Malik, DAC 2001.

Implementation: Two watched literal Scheme

Introduced by the SAT solver Chaff [1]

- Remember: Unit propagation fires when all but one literal is assigned false
- Idea: If **two** variables are either unassigned or assigned true, no need to do anything.
- So just find two variables which satisfy this condition.
- If can't find two, do the unit propagate or a conflict is found

[1] Chaff: Engineering an Efficient SAT Solver by Moskewicz, Madigan, Zhao, Zhang, Malik, DAC 2001.

Implementation: Two watched literal Scheme

Propagation Example

0/1	0/1	0/1	0/1
a	b	c	d

Triggers:

- a ∨ b ∨ c ∨ d

Implementation: Two watched literal Scheme

Propagation Example

0	0/1	0/1	0/1
a	b	c	d

Triggers:

- *a* assigned false.
- Update pointer.

This Slide is adapted from https://school.a4cp.org/summer2011/slides/Gent/SATCP3.pdf

Implementation: Two watched literal Scheme

Propagation Example

0	0/1	0/1	0/1
a	b	c	d

Triggers: ↑ (b) ↑ (c)

- *a* assigned false.
- Update pointer.

Implementation: Two watched literal Scheme

Propagation Example

0/1	0/1	0/1	0/1
a	b	c	d

Triggers:

- Backtrack. *a* unassigned.
- **Pointers do not move back**

This Slide is adapted from https://school.a4cp.org/summer2011/slides/Gent/SATCP3.pdf

Implementation: Two watched literal Scheme

Propagation Example

0/1	**1**	0/1	0/1
a	**b**	c	d

Triggers:

- If *b* is assigned true,
  pointer doesn't move.

Implementation: Two watched literal Scheme

Propagation Example

0	0/1	0/1	0
a	b	c	**d**

Triggers:

- If other variables assigned, nothing happens!
- Can’t emphasise enough ....

Implementation: Two watched literal Scheme

Propagation Example

0	0/1	0/1	**0**
a	b	c	**d**

Triggers:

- The unwatched literals a/d cause no work
- Not even checking there is nothing to do
  - because that would be O(1)

This Slide is adapted from https://school.a4cp.org/summer2011/slides/Gent/SATCP3.pdf

Implementation: Two watched literal Scheme

Propagation Example

0	**0**	0/1	0
a	**b**	c	d

Triggers:

- If we cannot find something new & unassigned to watch...
-

Implementation: Two watched literal Scheme

Propagation Example

0	0	**1**	0
a	b	**c**	d

Triggers:

- We can set the remaining literal
- i.e. do unit propagation since this clause is unit

This Slide is adapted from https://school.a4cp.org/summer2011/slides/Gent/SATCP3.pdf

Implementation: Two watched literal Scheme

Propagation Example

0	0	1	0
a	b	c	d

Triggers:

- Leave triggers where they are!

Implementation: Two watched literal Scheme

Propagation Example

0	0/1	0/1	0
a	b	c	d

Triggers:

- Triggers in the right place to continue after backtracking.

This Slide is adapted from https://school.a4cp.org/summer2011/slides/Gent/SATCP3.pdf

Implementation: Two watched literal Scheme

Advantages:

- **ZERO** cost if a literal not watched.
- **ZERO** cost on backtrack.

This Slide is adapted from https://school.a4cp.org/summer2011/slides/Gent/SATCP3.pdf

Implementation: Two watched literal Scheme

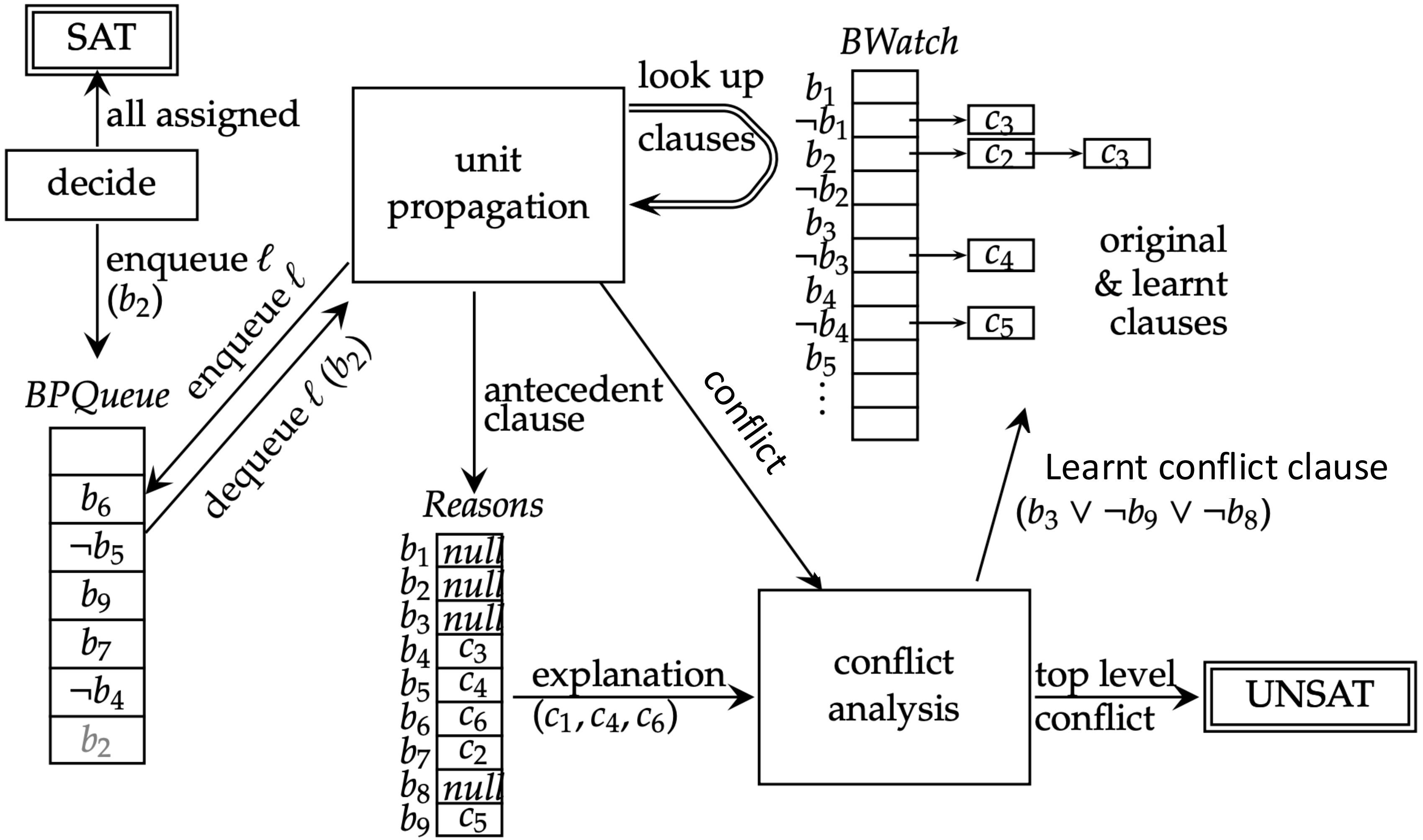
Discussions:

- Really come into their own on large clauses
    - probably not worthwhile on 3-SAT, for example
    - E.g. if there are 100 variables in clause
        - it still only needs to watch 2
        - and 98% of the time the solver will do no work
        - As if the problem was 98% smaller!
- We can handle problems with many large clauses
- benefits the conflict-driven learning
    - since the learned conflict clauses are often big

This Slide is adapted from https://school.a4cp.org/summer2011/slides/Gent/SATCP3.pdf

Implementation: Classic CDCL Solver MiniSat

Overall Architecture



This figure is adapted a figure from [Wang 2016 Dissertation]

Research in Machine Learning for SAT

One direction: Improving Decision Heuristics

1. Variable selection heuristics

    aim: minimize the search space

2. Value selection heuristics

    aim: guide search towards a solution or conflict