CDCL SAT Solving

Wenxi Wang

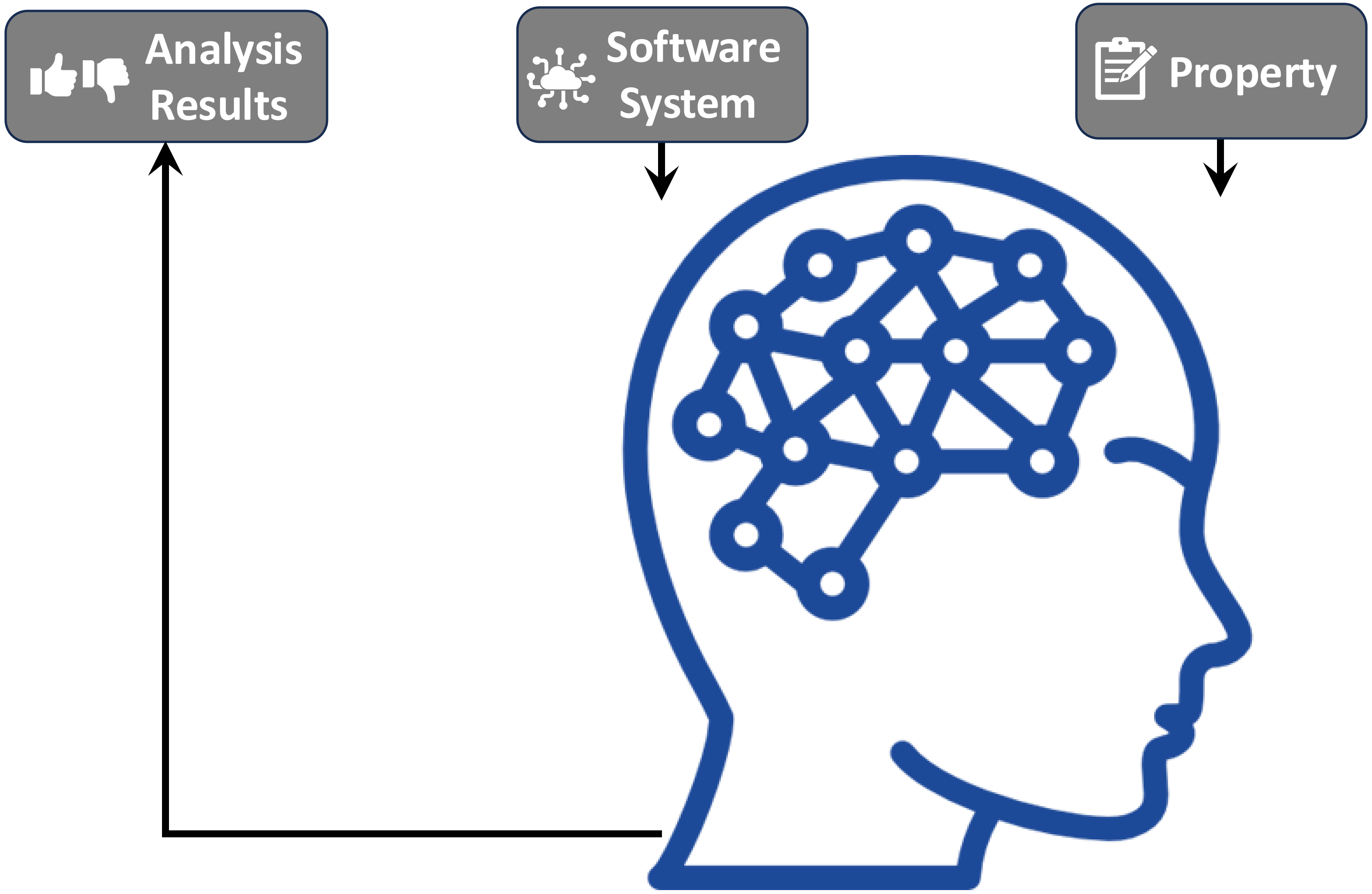
University of Virginia

wenxiw@virginia.edu

Recap

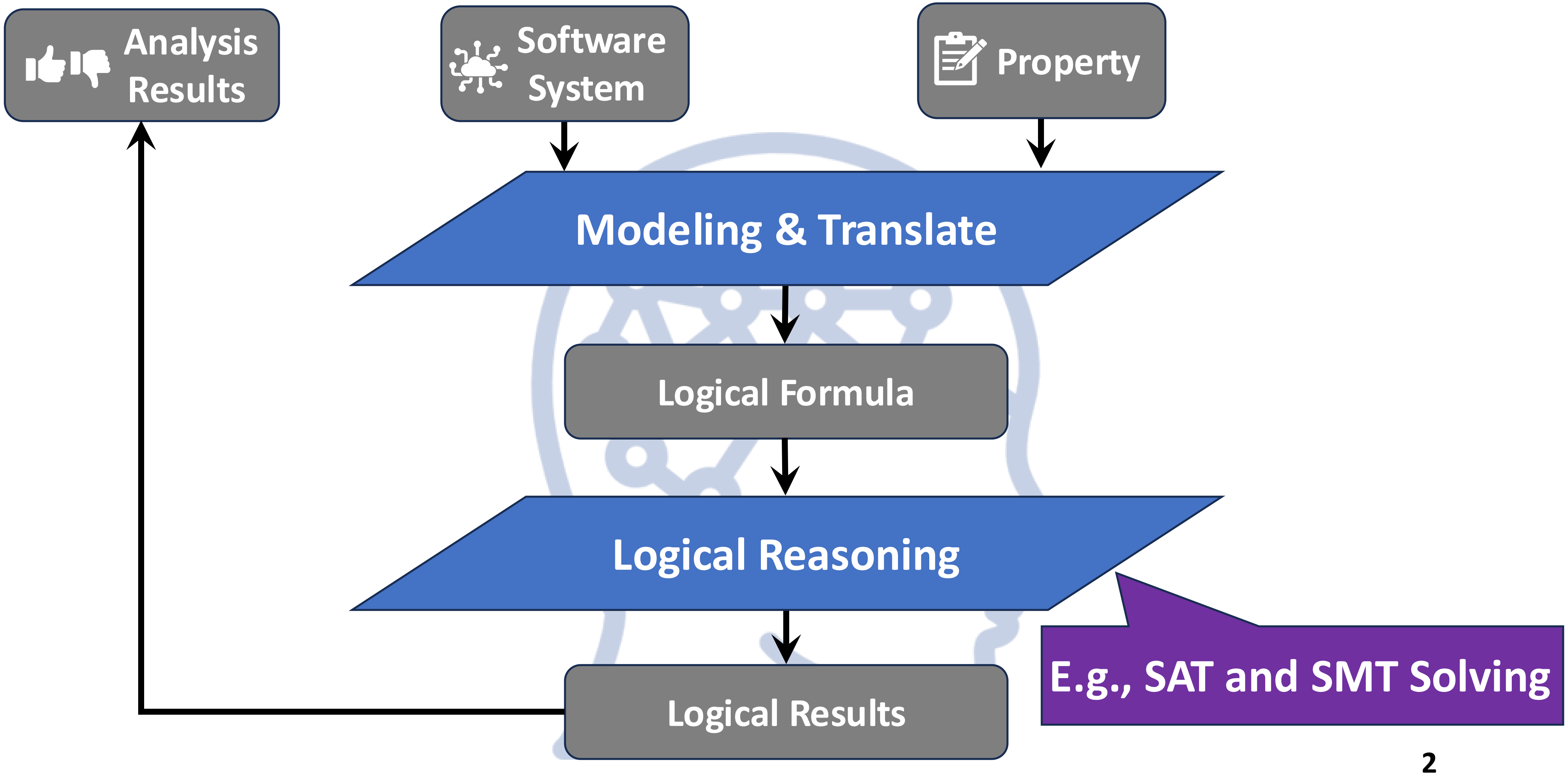
Direction 1: Software Verification

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



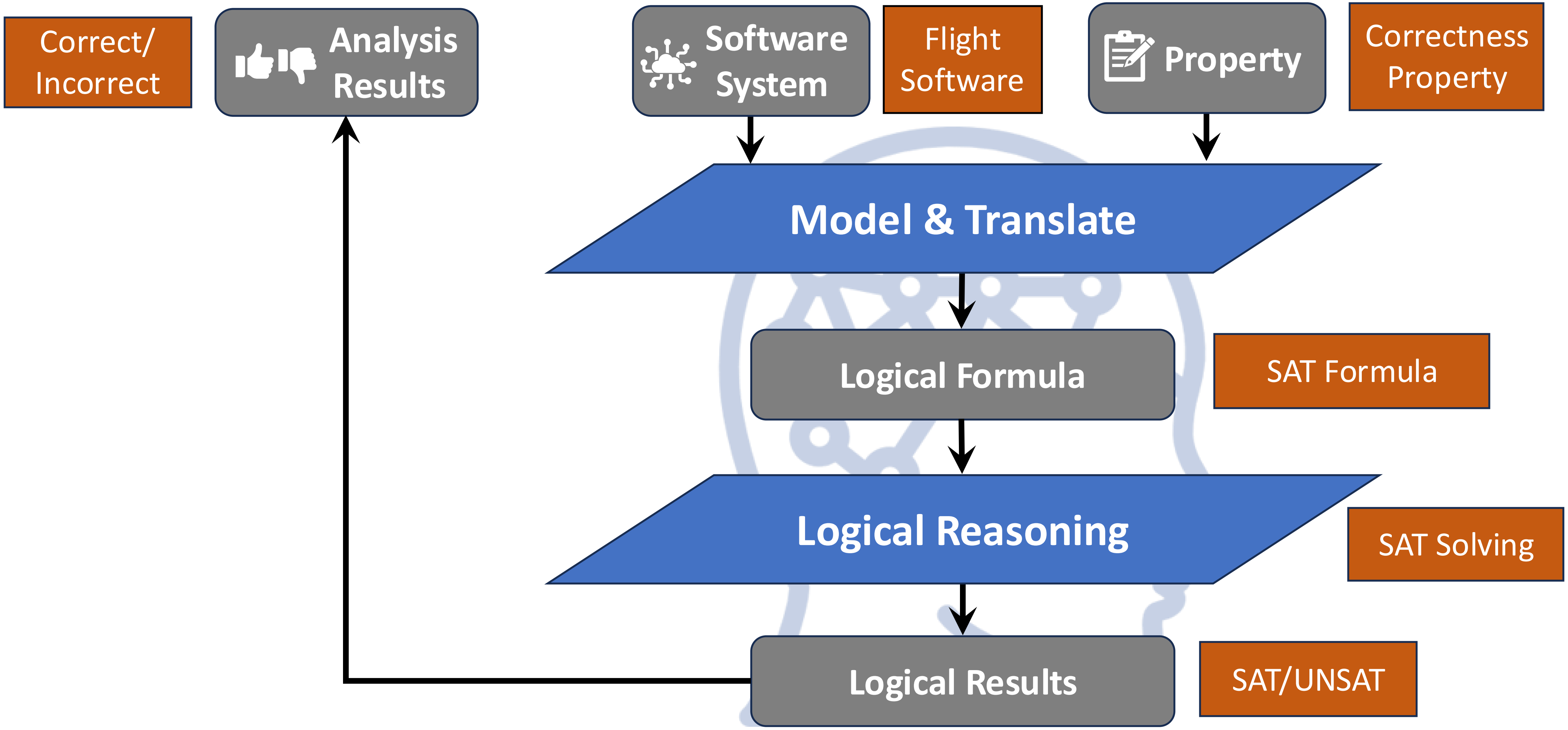
Recap Direction 1: Software Verification

Typically models software problems into logical formulas



Recap Formal Reasoning for Software Systems

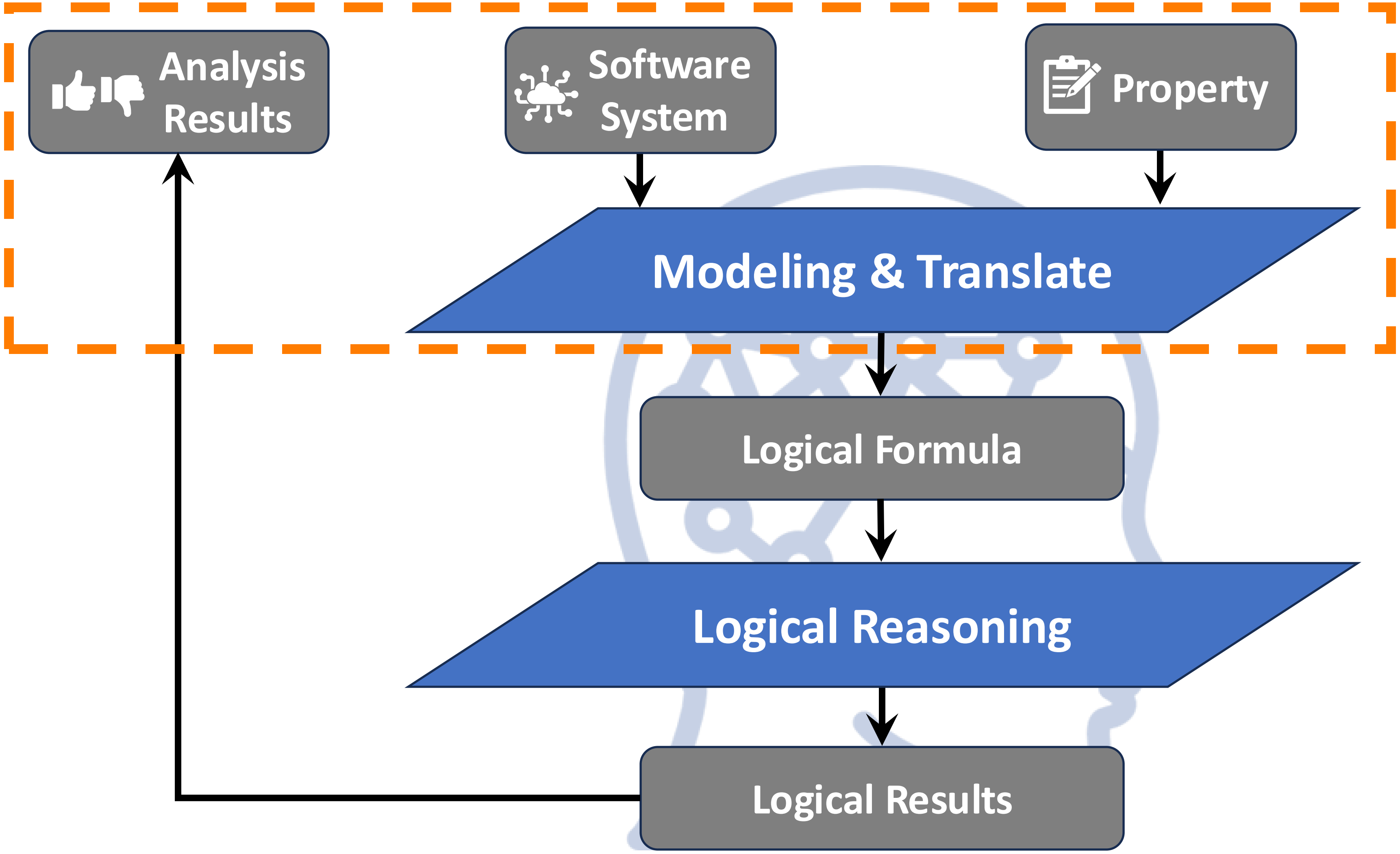
For example: Flight software verification in NASA



Recap

Direction 1: Software Verification

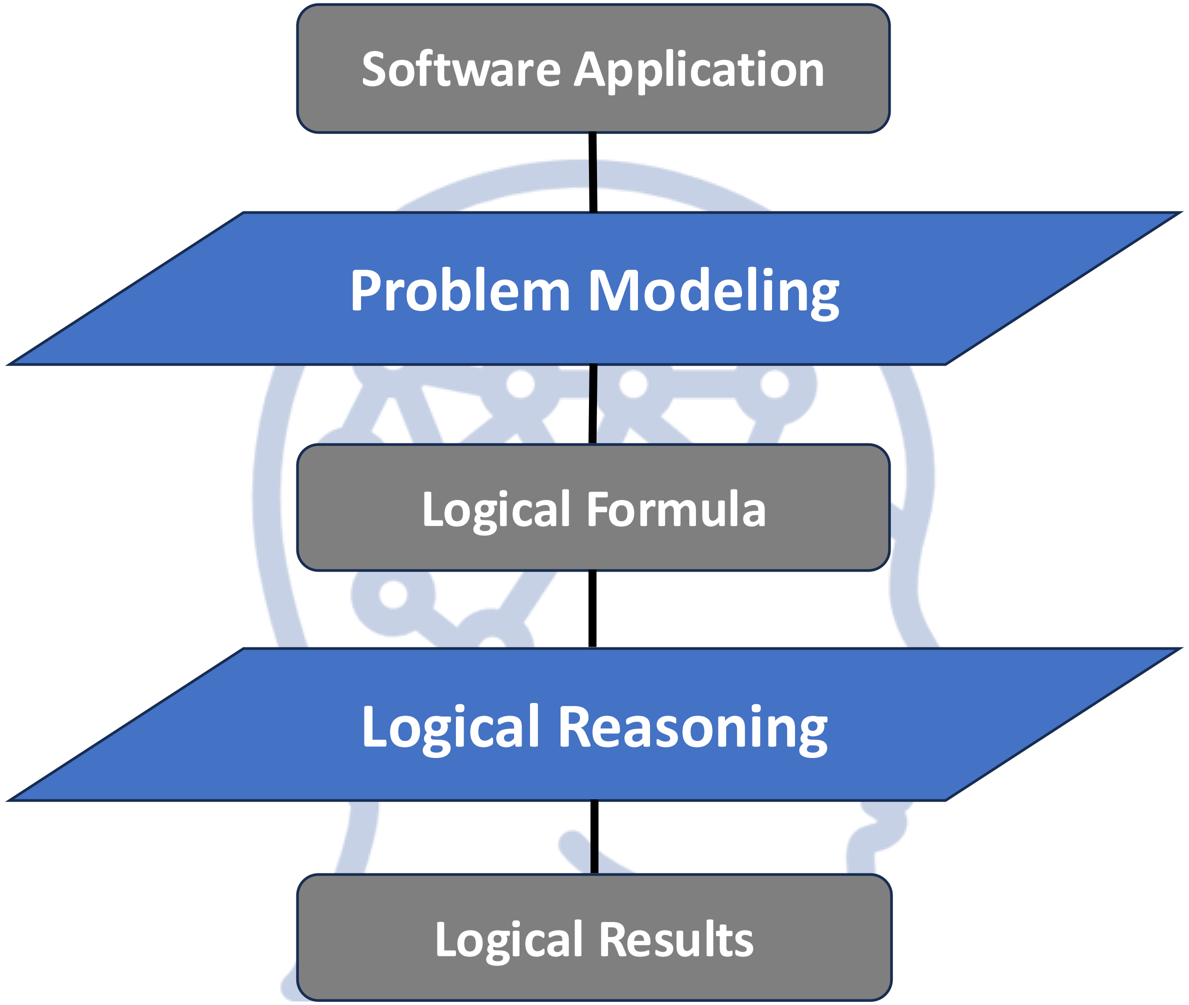
Typically models software problems into logical formulas



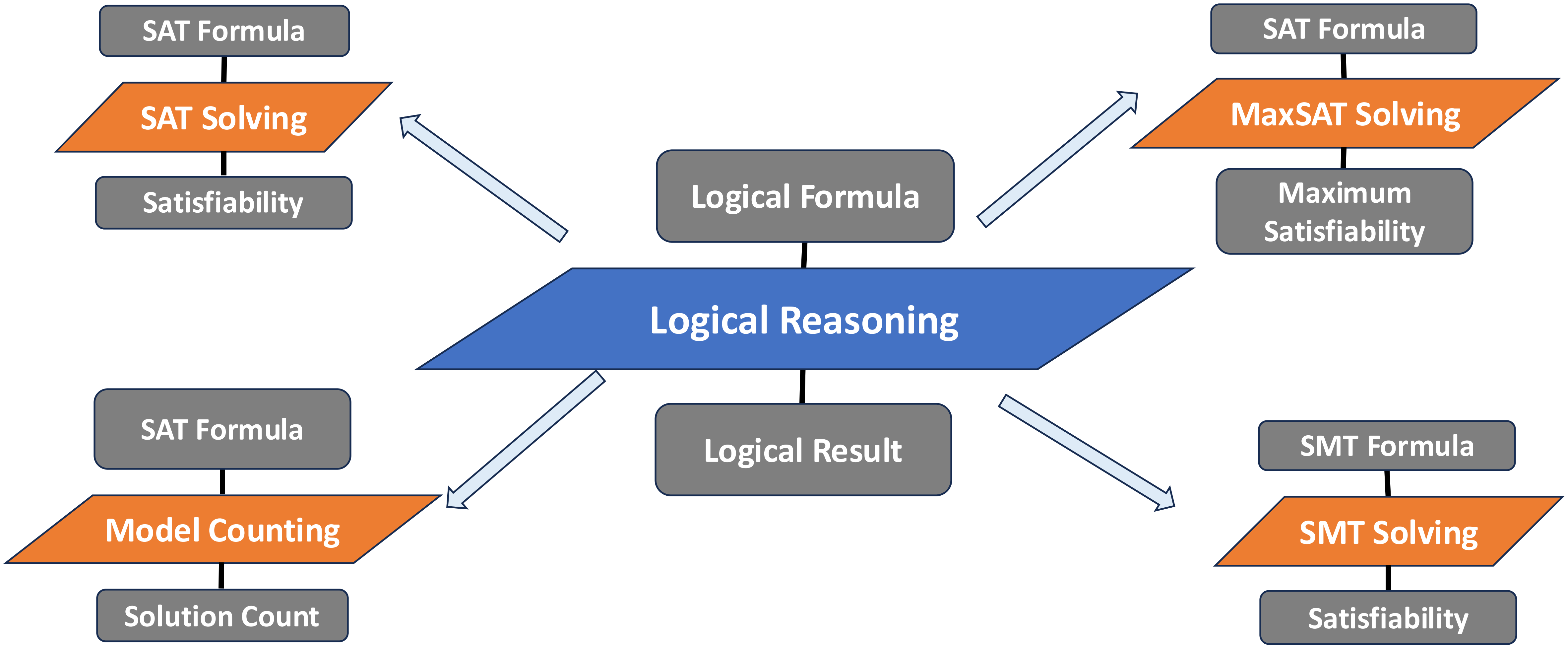
Recap

Direction 1: Software Verification

Simplified view: we focus on both analysis layers

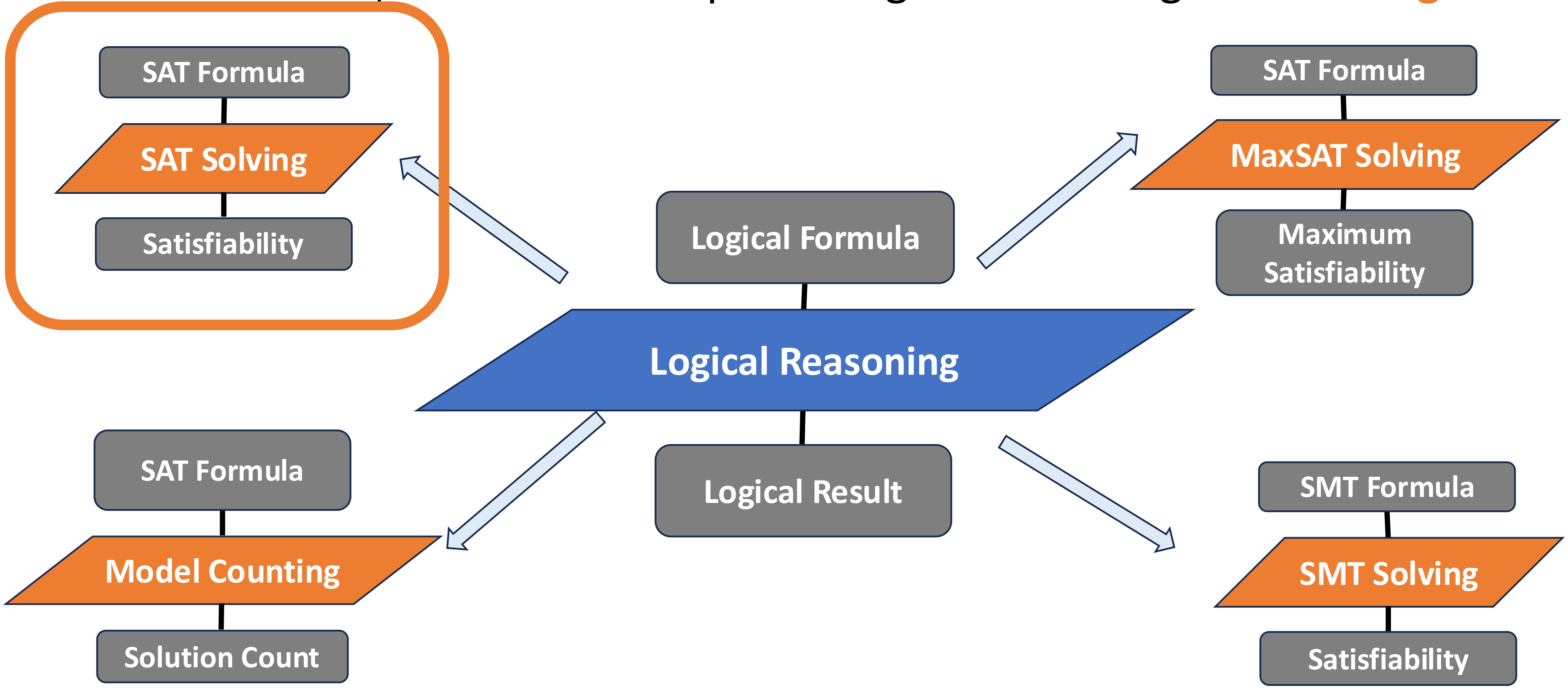


Logical Reasoning



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

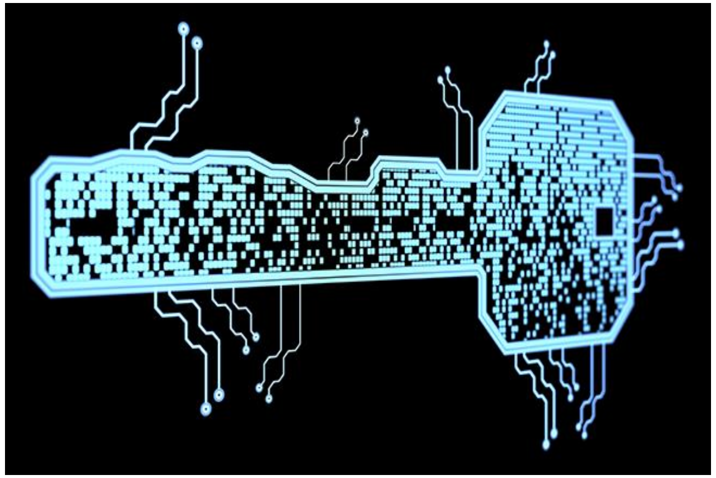
↓

Many problems in CS can be reduced to SAT

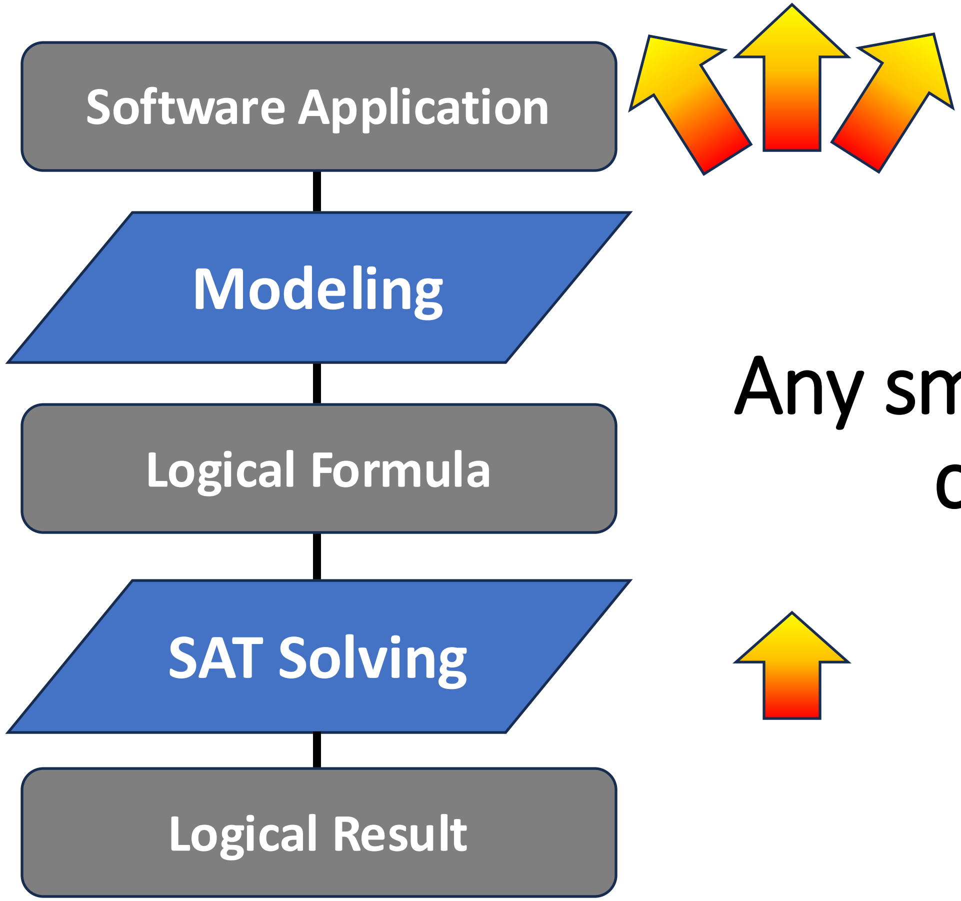
Including software and security problems

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:

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

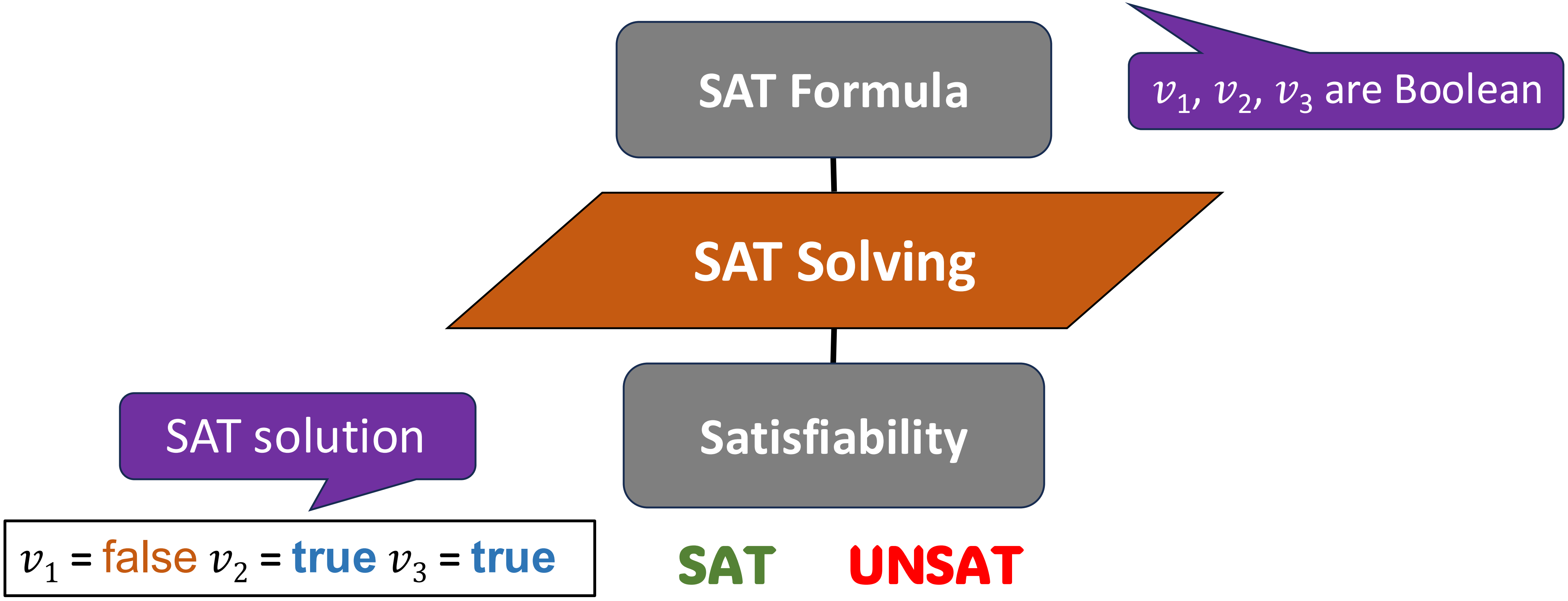
Clauses: c_1, c_2, c_3

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

Boolean variables: v_1, v_2, v_3

SAT Solving

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



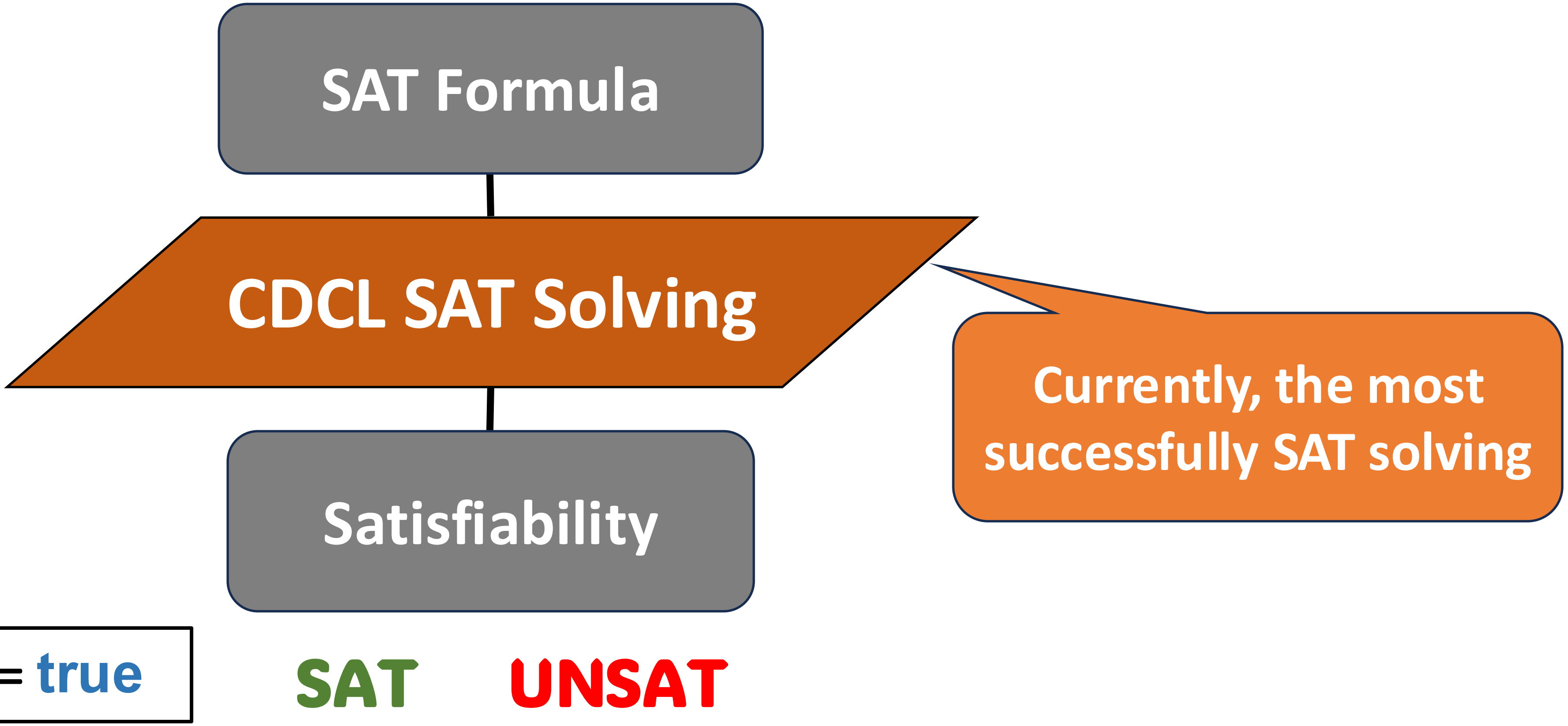
SAT Solving

Does there exist an assignment satisfying all clauses?

(x5 ∨ ¬x8 ∨ x2) ∧ (x2 ∨ x1 ∨ x3) ∧ (x8 ∨ x3 ∨ x7) ∧ (x5 ∨ x3 ∨ x8) ∧
(x6 ∨ x1 ∨ ¬x5) ∧ (x8 ∨ x9 ∨ x3) ∧ (x2 ∨ ¬x1 ∨ x3) ∧ (x1 ∨ ¬x8 ∨ x4) ∧
(x9 ∨ x6 ∨ x8) ∧ (x8 ∨ x3 ∨ x9) ∧ (x9 ∨ x3 ∨ x8) ∧ (x6 ∨ x9 ∨ x5) ∧
(x2 ∨ x3 ∨ x8) ∧ (x8 ∨ x6 ∨ x3) ∧ (x8 ∨ ¬x3 ∨ x1) ∧ (x8 ∨ x6 ∨ x2) ∧
(x7 ∨ x9 ∨ ¬x2) ∧ (x8 ∨ x9 ∨ x2) ∧ (x1 ∨ x9 ∨ x4) ∧ (x8 ∨ ¬x1 ∨ x2) ∧
(x3 ∨ ¬x4 ∨ x6) ∧ (x1 ∨ x7 ∨ x5) ∧ (x7 ∨ x1 ∨ x6) ∧ (x5 ∨ x4 ∨ x6) ∧
(x4 ∨ x9 ∨ x8) ∧ (x2 ∨ ¬x9 ∨ x1) ∧ (x5 ∨ ¬x7 ∨ x1) ∧ (x7 ∨ x9 ∨ x6) ∧
(x2 ∨ x5 ∨ x4) ∧ (x8 ∨ x4 ∨ x5) ∧ (x5 ∨ x9 ∨ x3) ∧ (x5 ∨ x7 ∨ x9) ∧
(x2 ∨ ¬ x8 ∨ x1) ∧ (x7 ∨ ¬x1 ∨ x5) ∧ (x1 ∨ x4 ∨ x3) ∧ (x1 ∨ x9 ∨ x4) ∧
(x3 ∨ x5 ∨ x6) ∧ (x6 ∨ x3 ∨ x9) ∧ (x7 ∨ ¬x5 ∨ x9) ∧ (x7 ∨ ¬x5 ∨ x2) ∧
(x4 ∨ ¬x7 ∨ x3) ∧ (x4 ∨ ¬x9 ∨ x7) ∧ (x5 ∨ x1 ∨ x7) ∧ (x5 ∨ x1 ∨ x7) ∧
(x6 ∨ x7 ∨ x3) ∧ (x8 ∨ x6 ∨ x7) ∧ (x6 ∨ x2 ∨ x3) ∧ (x8 ∨ x2 ∨ x5) ....

CDCL SAT solving

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

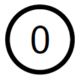


v_1 = false v_2 = **true** v_3 = **true**

CDCL SAT solving

CDCL: Conflict Driven Clause Learning

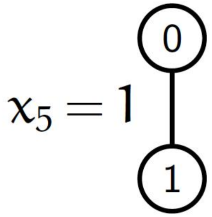
$$(x_1 \vee x_4) \wedge$$
$$(x_3 \vee \overline{x}_4 \vee \overline{x}_5) \wedge$$
$$(\overline{x}_3 \vee \overline{x}_2 \vee \overline{x}_4) \wedge$$
$$\mathcal{F}_{\text{extra}}$$



CDCL SAT solving

CDCL: Conflict Driven Clause Learning

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

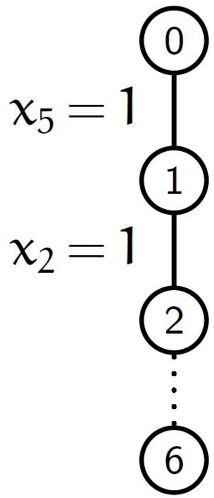


This slide is adapted from Marijn J.H. Heule's slides (https://www.cs.cmu.edu/~mheule/15816-f20/slides/CDCL.pdf).

CDCL SAT solving

CDCL: Conflict Driven Clause Learning

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

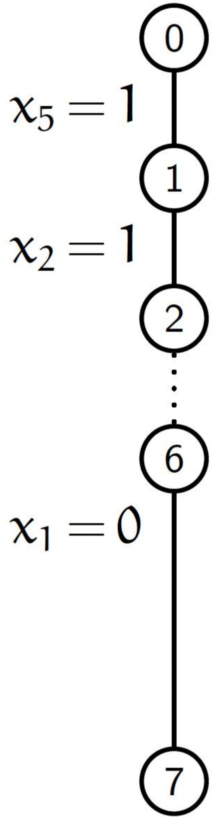


This slide is adapted from Marijn J.H. Heule's slides (https://www.cs.cmu.edu/~mheule/15816-f20/slides/CDCL.pdf).

CDCL SAT solving

CDCL: Conflict Driven Clause Learning

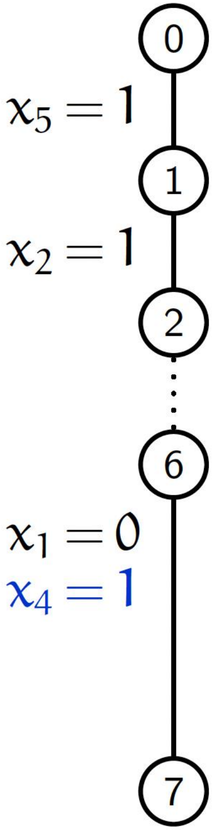
$(x_1 \lor x_4) \land$
$(x_3 \lor \overline{x}_4 \lor \overline{x}_5) \land$
$(\overline{x}_3 \lor \overline{x}_2 \lor \overline{x}_4) \land$
$\mathcal{F}_{\text{extra}}$



CDCL SAT solving

CDCL: Conflict Driven Clause Learning

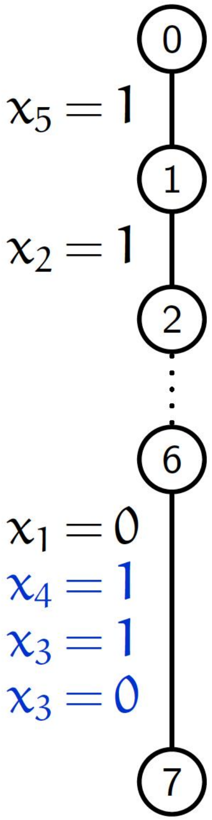
$$(x_1 \vee x_4) \wedge$$
$$(x_3 \vee \overline{x}_4 \vee \overline{x}_5) \wedge$$
$$(\overline{x}_3 \vee \overline{x}_2 \vee \overline{x}_4) \wedge$$
$$\mathcal{F}_{\text{extra}}$$



CDCL SAT solving

CDCL: Conflict Driven Clause Learning

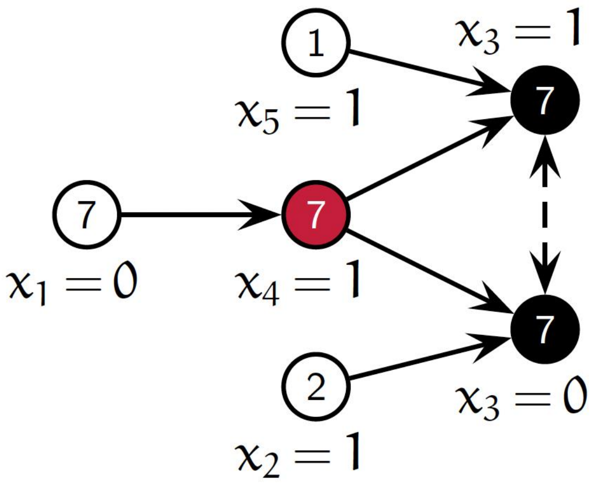
$(x_1 \vee x_4) \wedge$
$(x_3 \vee \overline{x}_4 \vee \overline{x}_5) \wedge$
$(\overline{x}_3 \vee \overline{x}_2 \vee \overline{x}_4) \wedge$
$\mathcal{F}_{\text{extra}}$

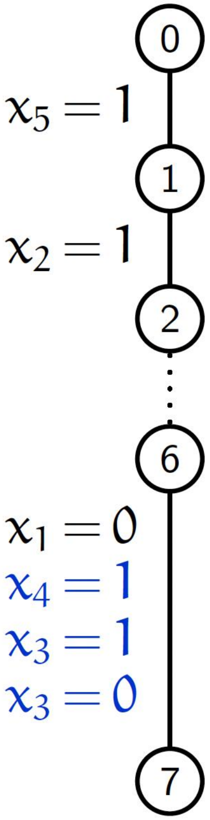


CDCL SAT solving

CDCL: Conflict Driven Clause Learning

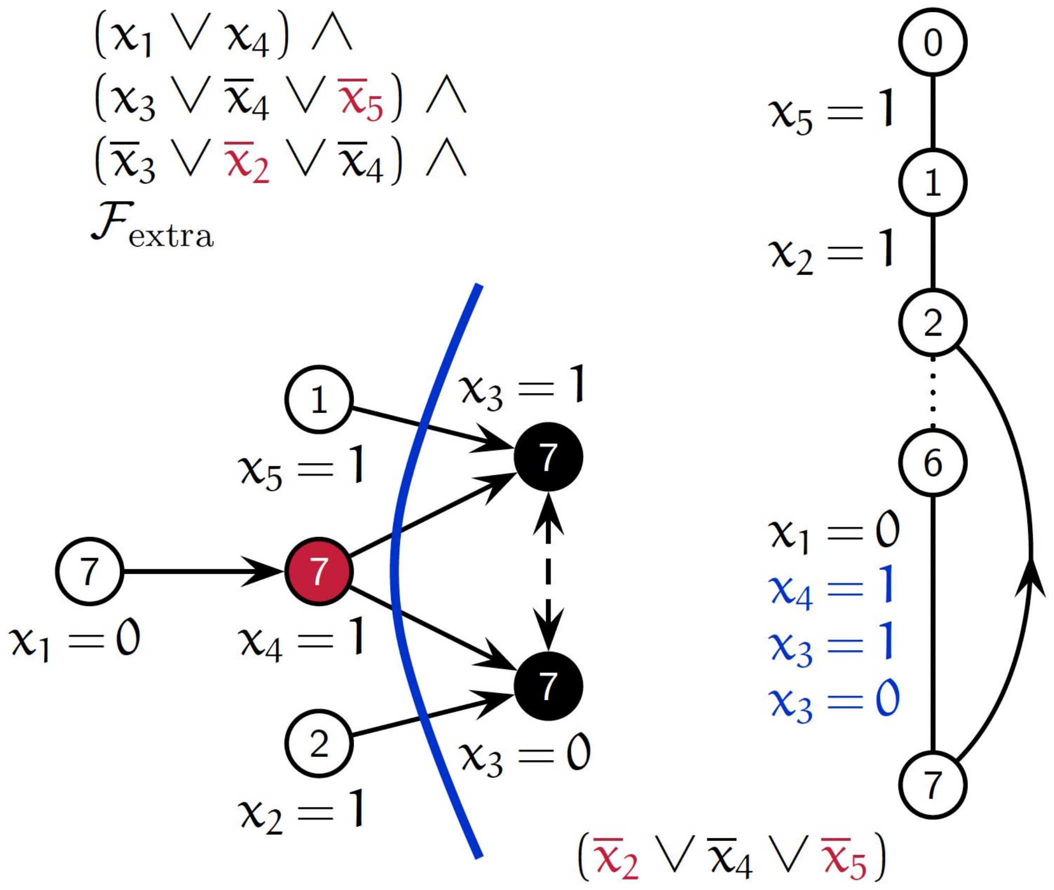
$$(x_1 \vee x_4) \wedge$$
$$(x_3 \vee \overline{x}_4 \vee \overline{x}_5) \wedge$$
$$(\overline{x}_3 \vee \overline{x}_2 \vee \overline{x}_4) \wedge$$
$$\mathcal{F}_{\text{extra}}$$





CDCL SAT solving

CDCL: Conflict Driven Clause Learning

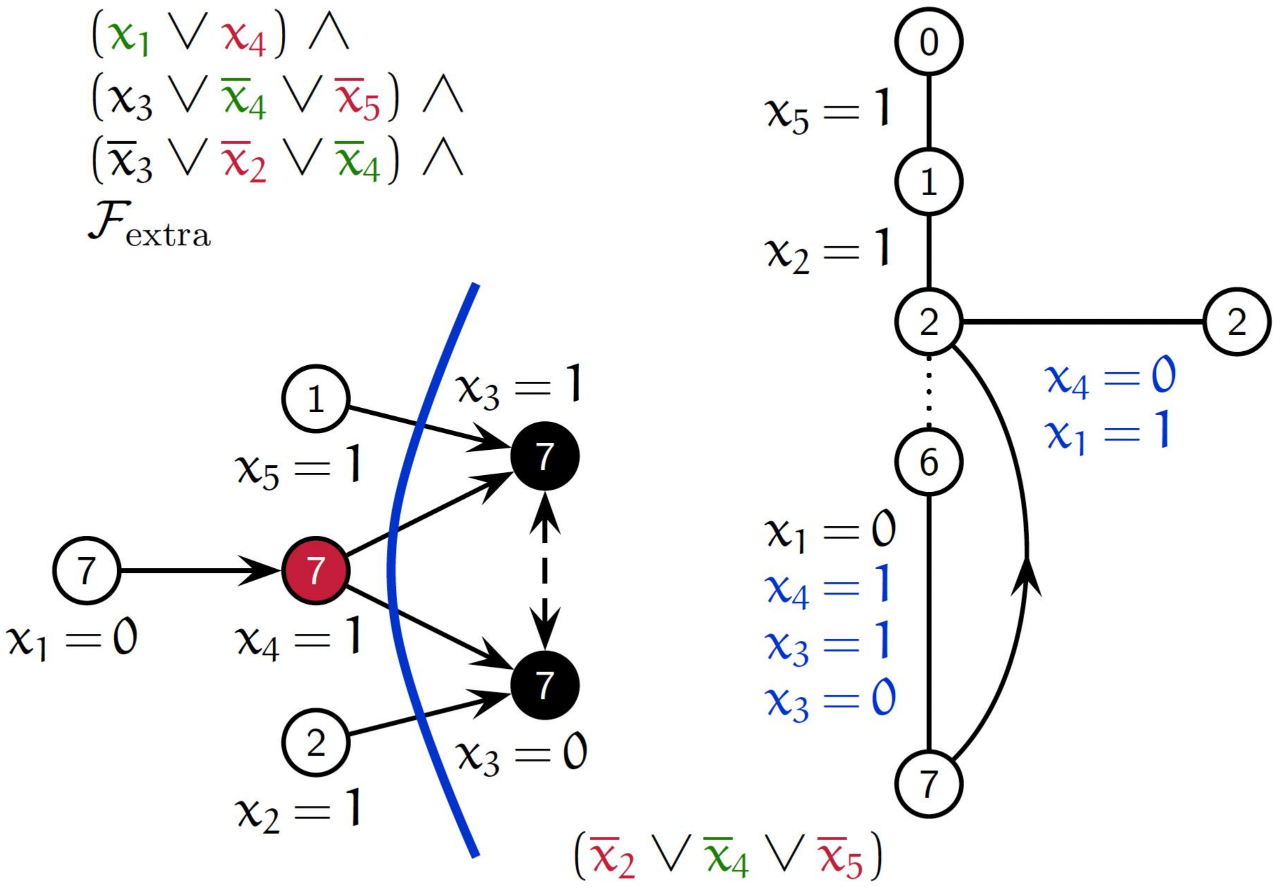


This slide is adapted from Marijn J.H. Heule's slides (https://www.cs.cmu.edu/~mheule/15816-f20/slides/CDCL.pdf).

CDCL SAT solving

CDCL: Conflict Driven Clause Learning

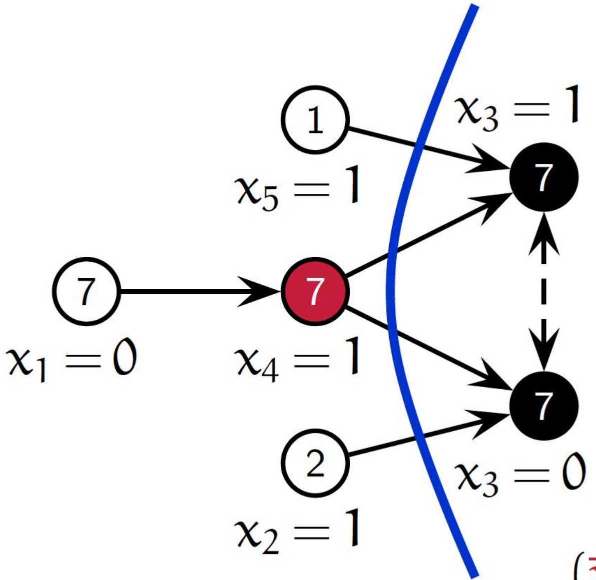
$(x_1 \vee x_4) \wedge$
$(x_3 \vee \overline{x}_4 \vee \overline{x}_5) \wedge$
$(\overline{x}_3 \vee \overline{x}_2 \vee \overline{x}_4) \wedge$
$\mathcal{F}_{\text{extra}}$

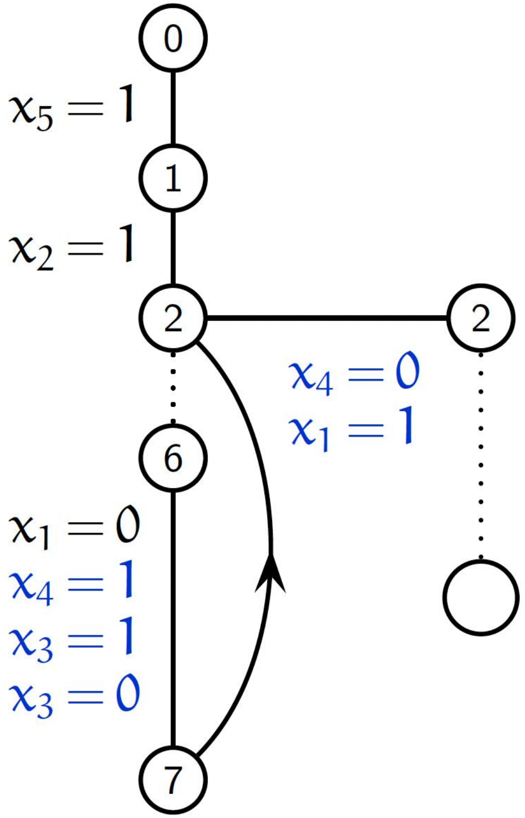


CDCL SAT solving

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

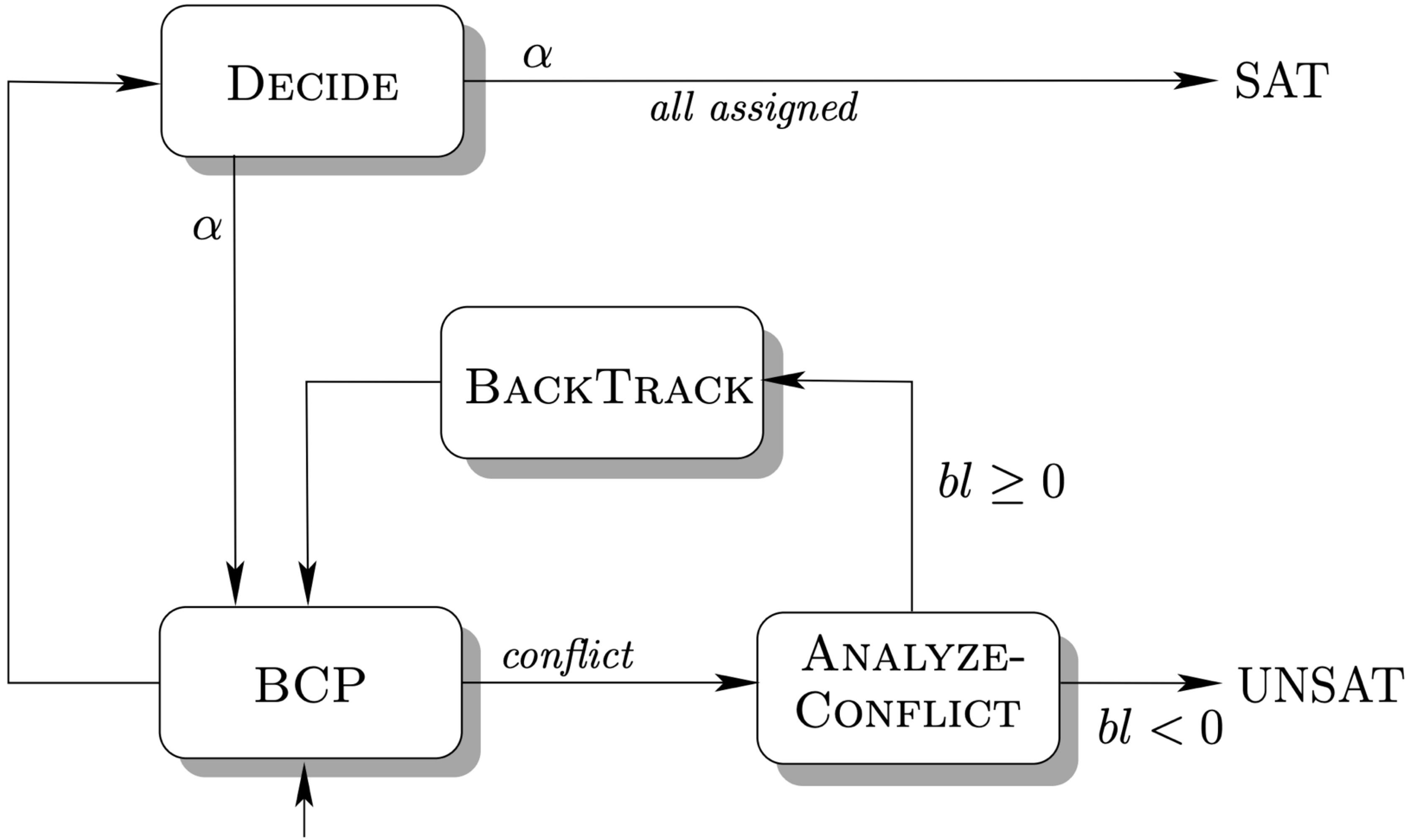
General Algorithm

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";
```

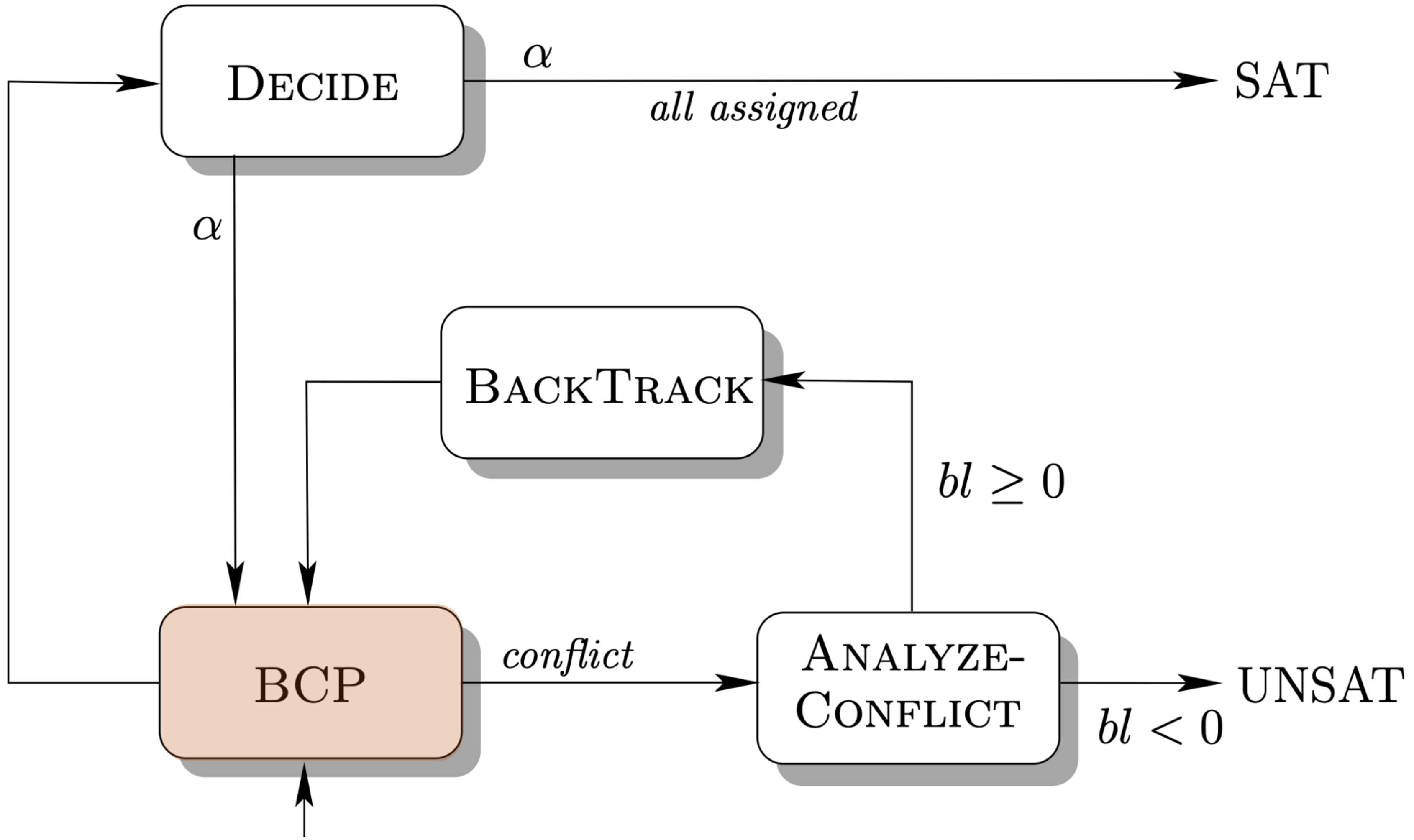
CDCL SAT solving

General Workflow



CDCL SAT solving

General Workflow



BCP: Boolean Constraint Propagation

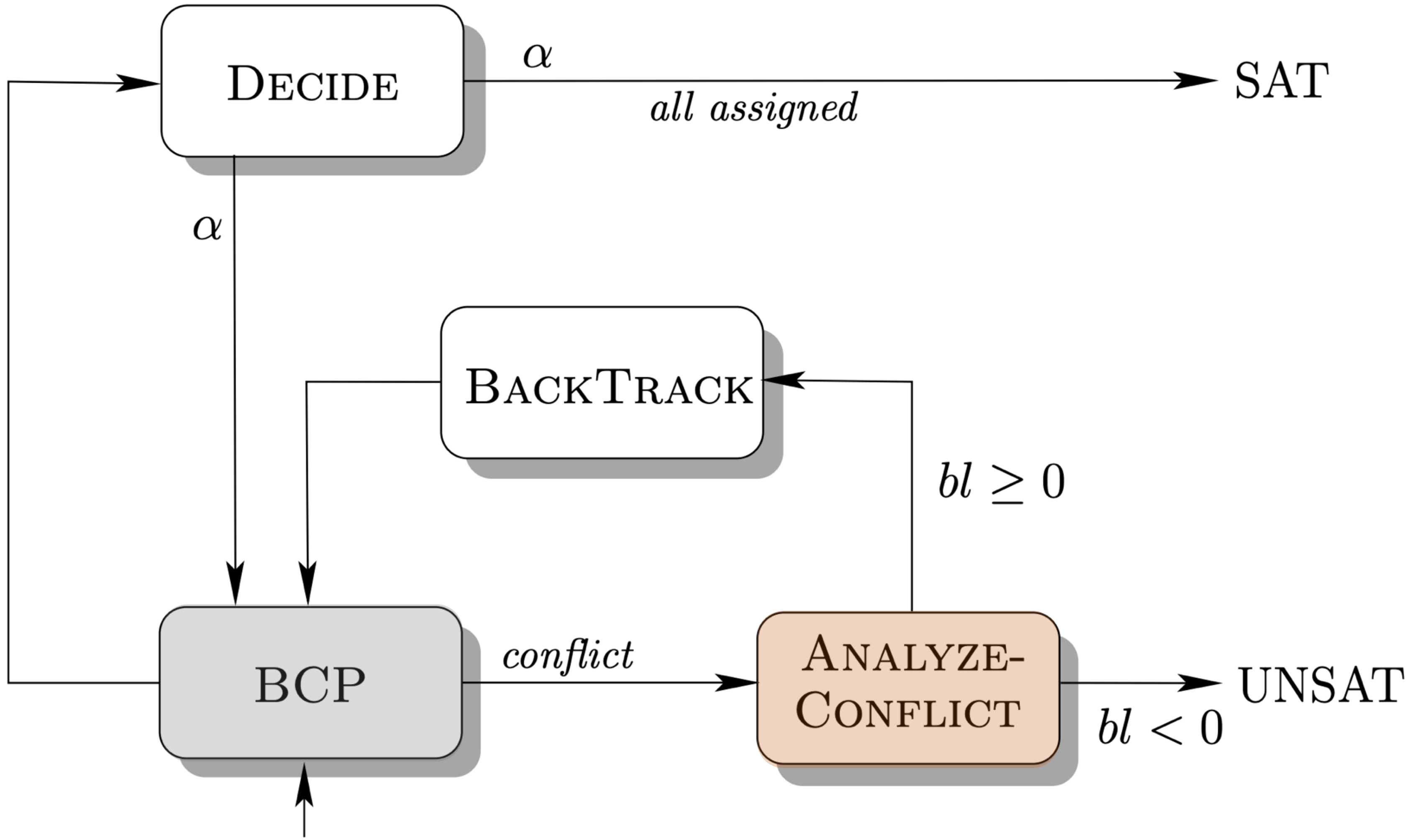
Unit Propagation

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

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

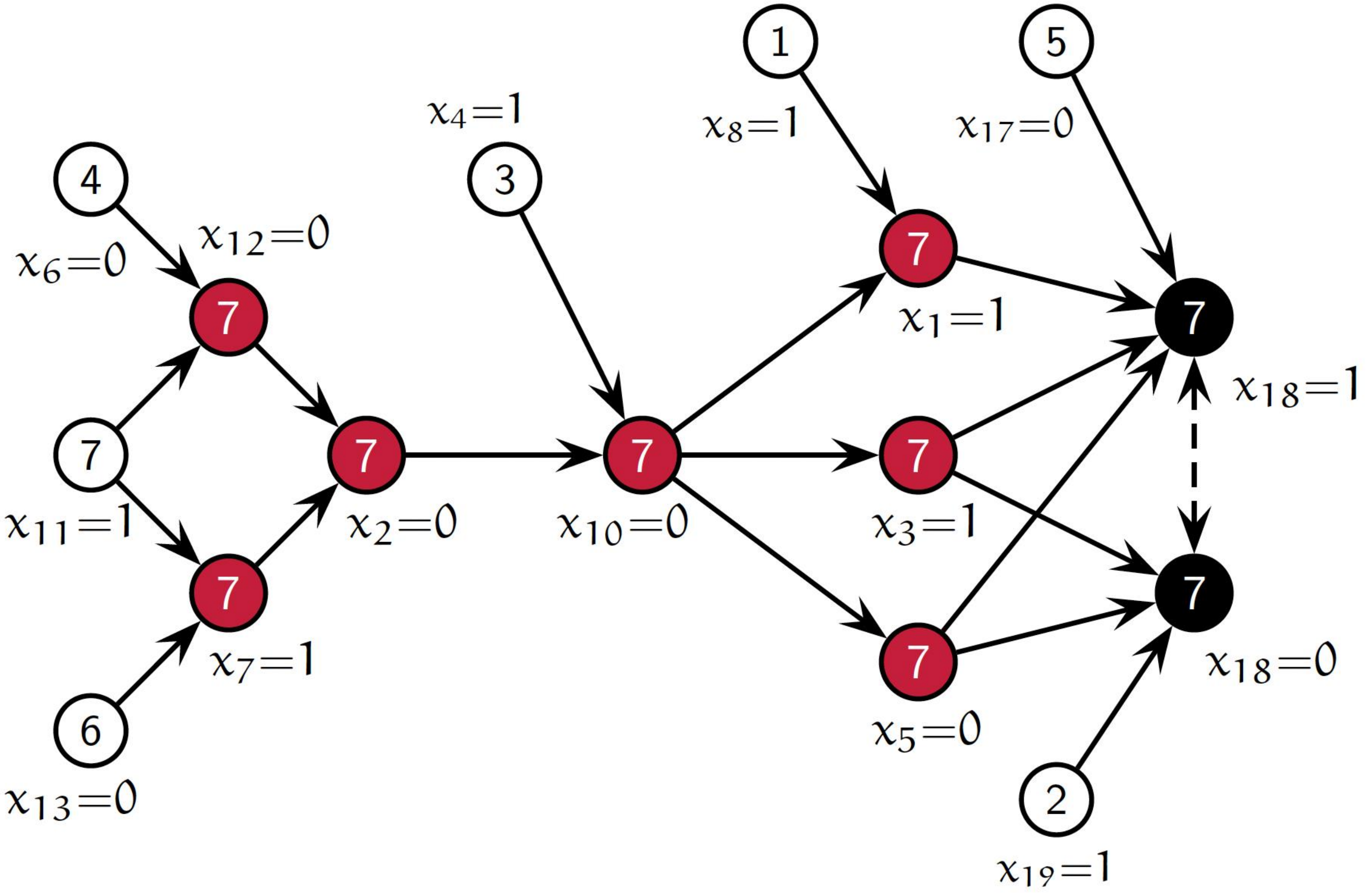
CDCL SAT solving

General Workflow

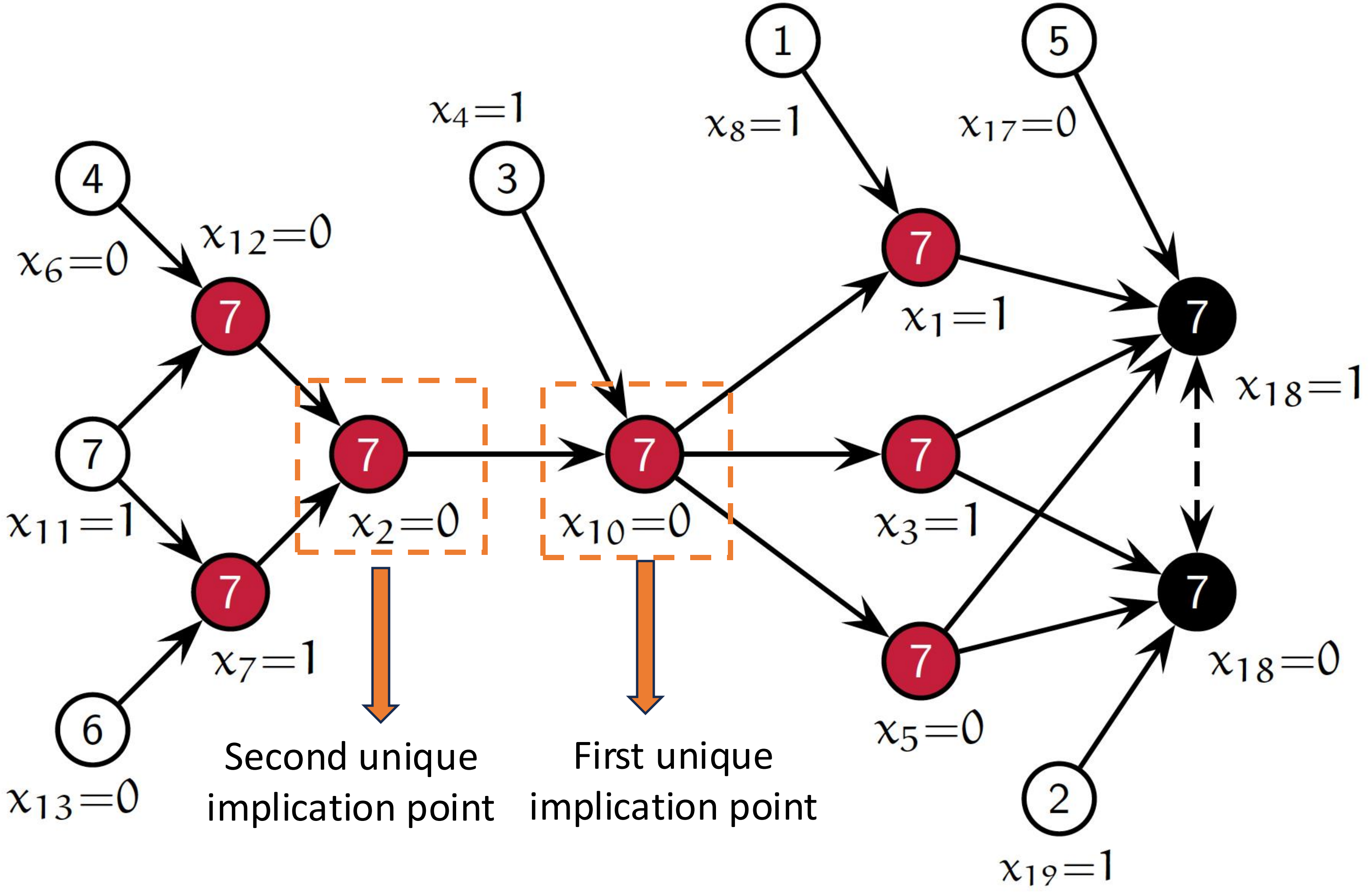


Conflict Analysis-learning a conflict clause

Implication Graph



Conflict Analysis-learning a conflict clause

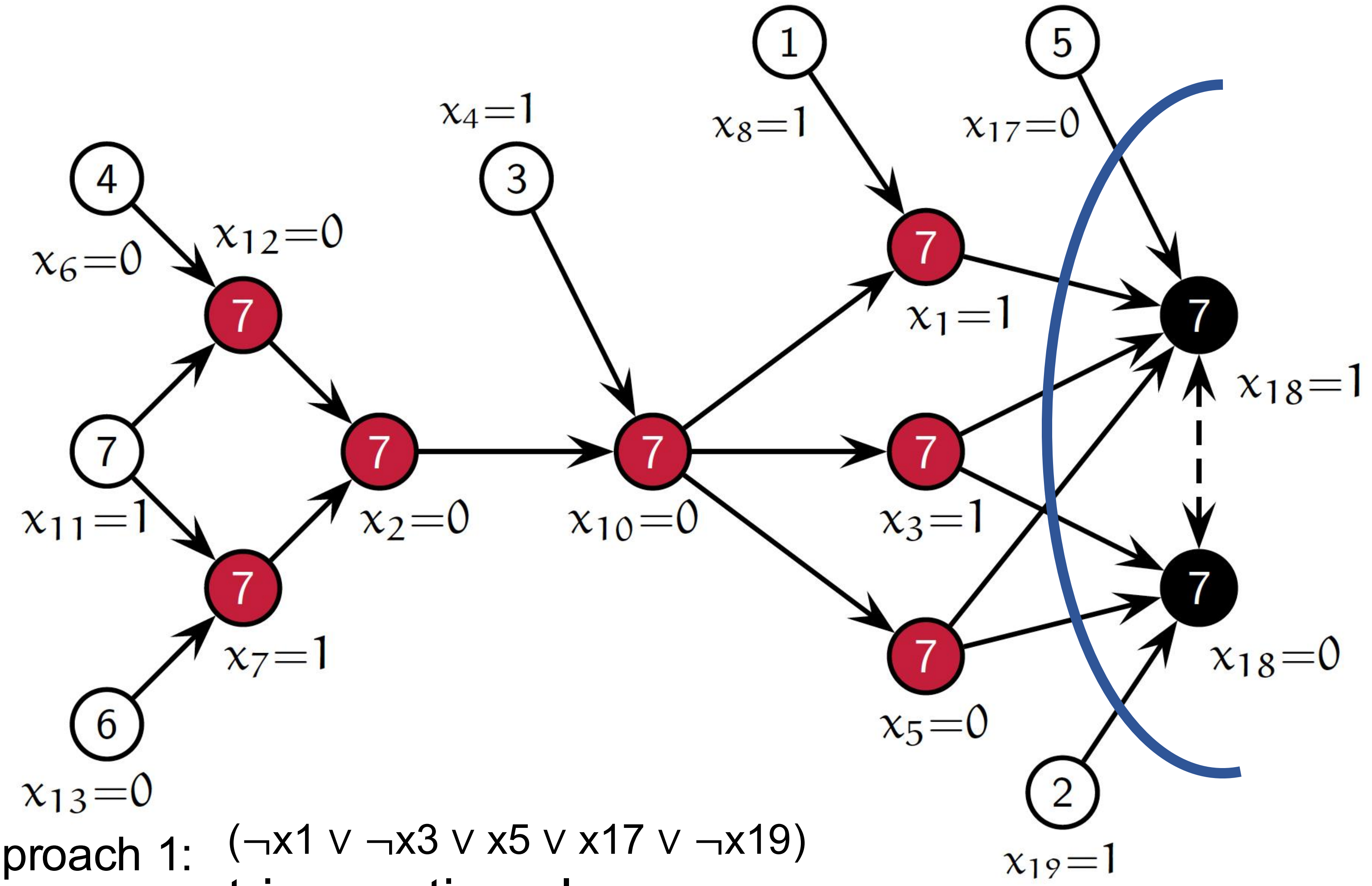


UIP: any node other than the conflict node that is on all paths from the decision node to the conflict node

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).

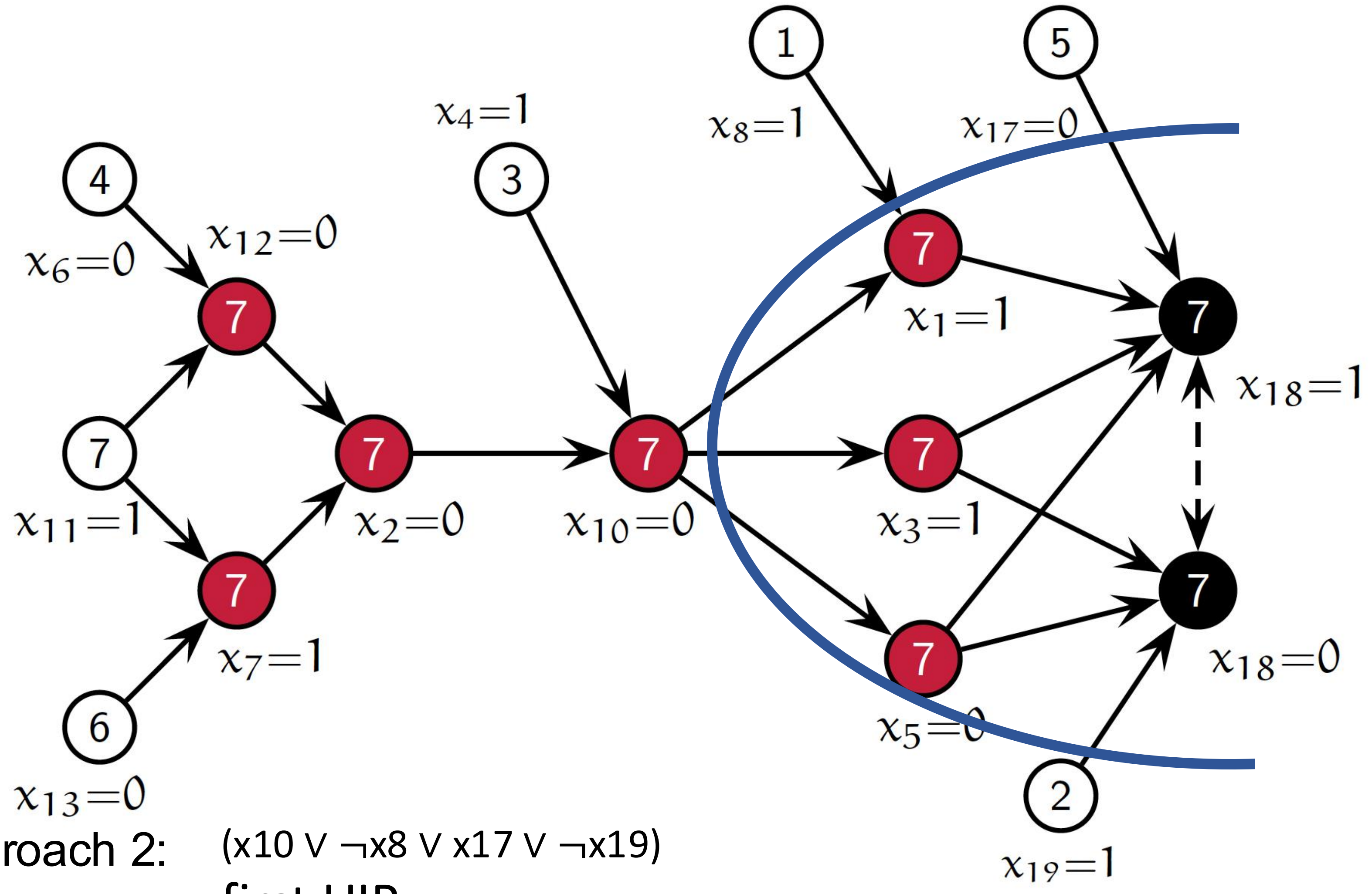
Conflict Analysis-learning a conflict clause



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).

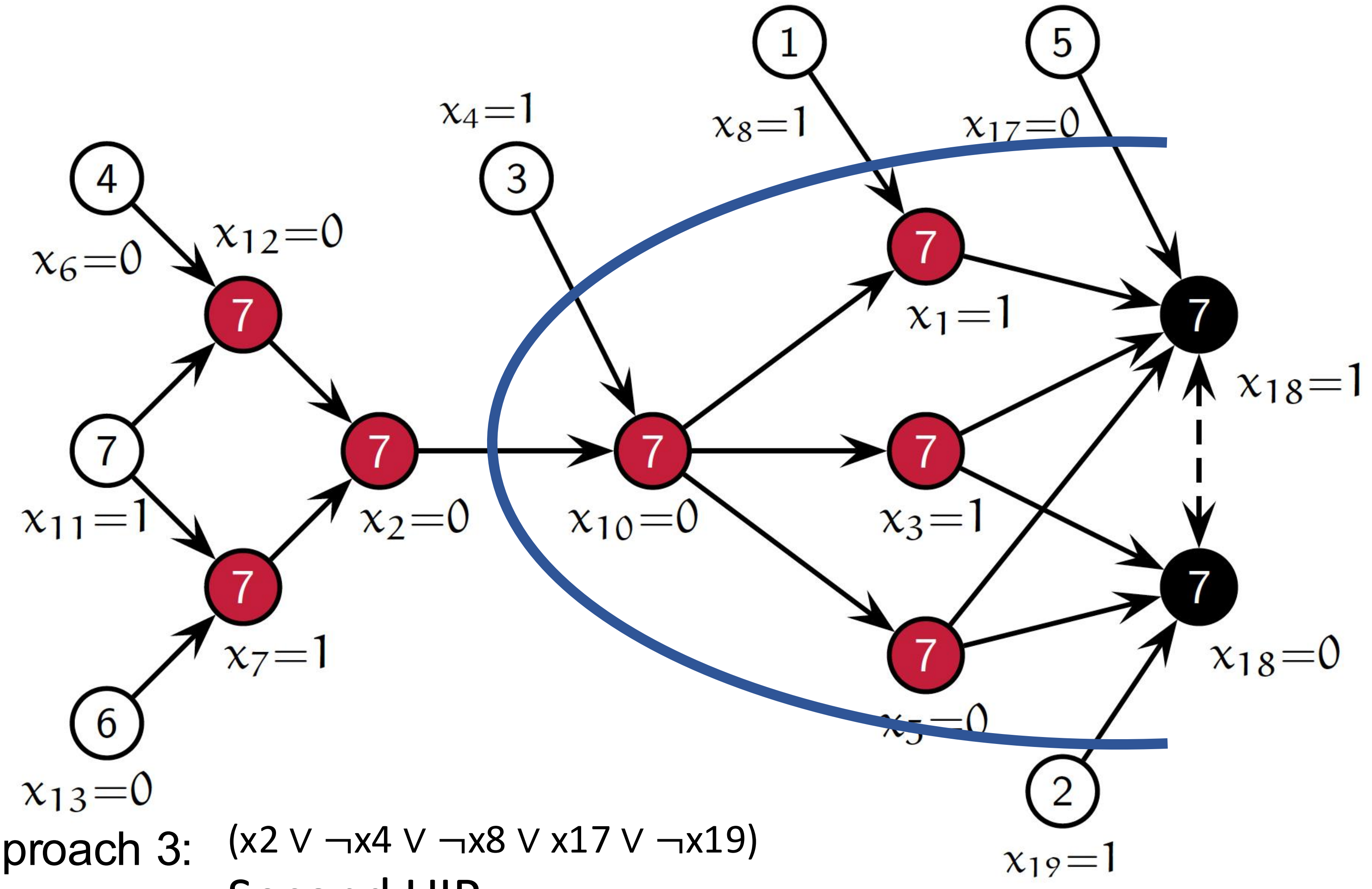
Conflict Analysis-learning a conflict clause



Approach 2: (x10 ∨ ¬x8 ∨ x17 ∨ ¬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

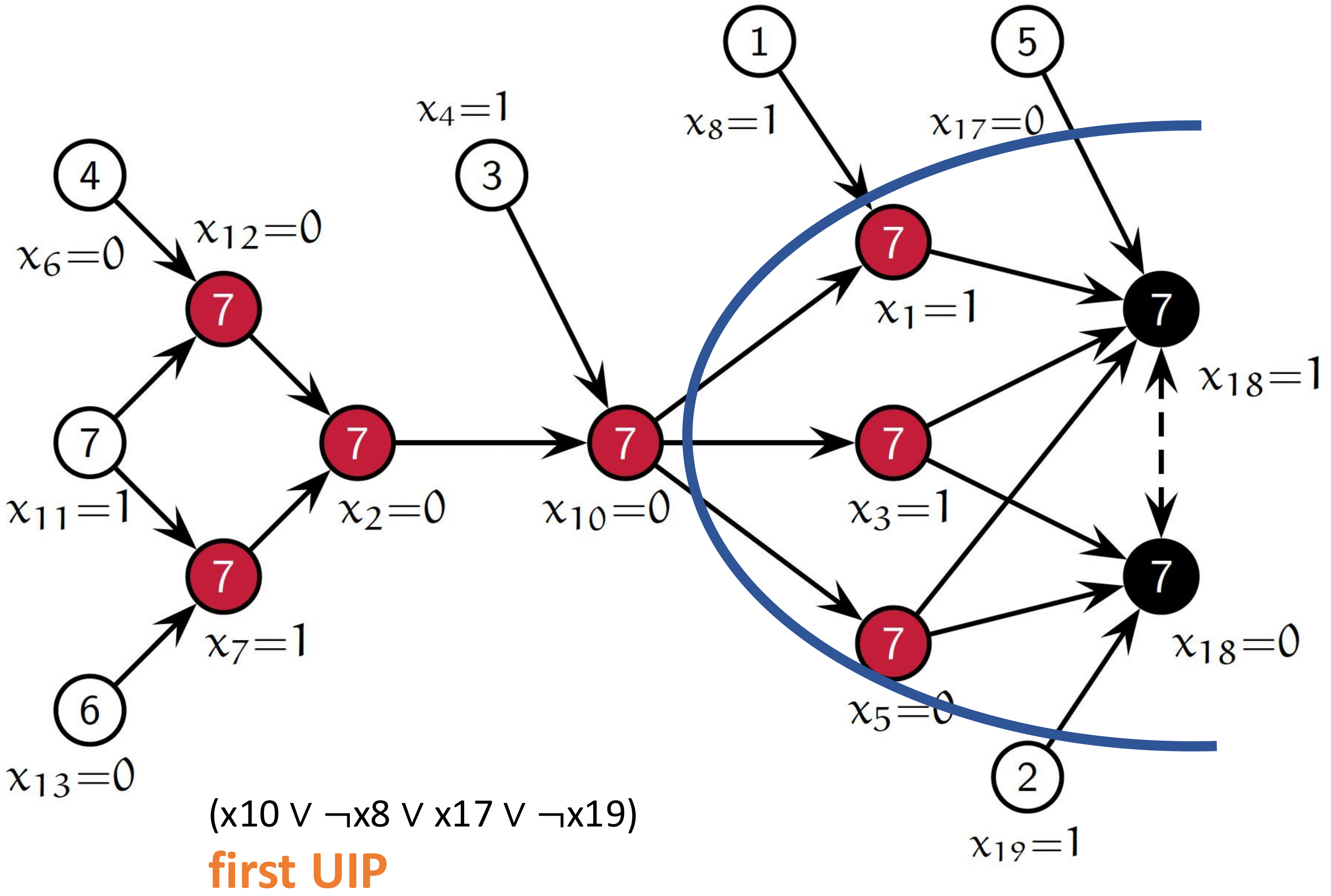


Approach 3: (x2 ∨ ¬x4 ∨ ¬x8 ∨ x17 ∨ ¬x19)

Second UIP

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

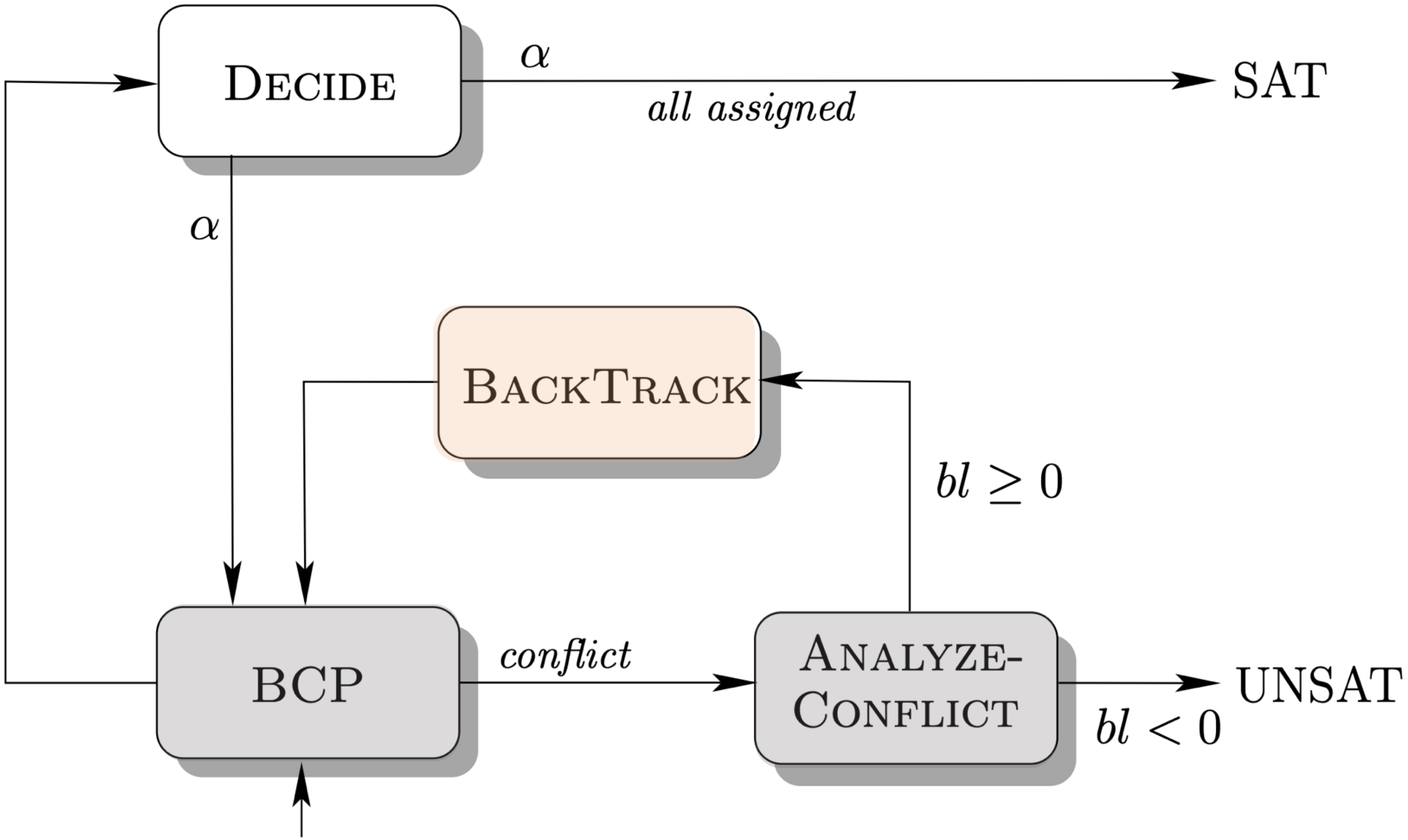
Conflict Analysis-learning a conflict clause



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

CDCL SAT solving

General Workflow

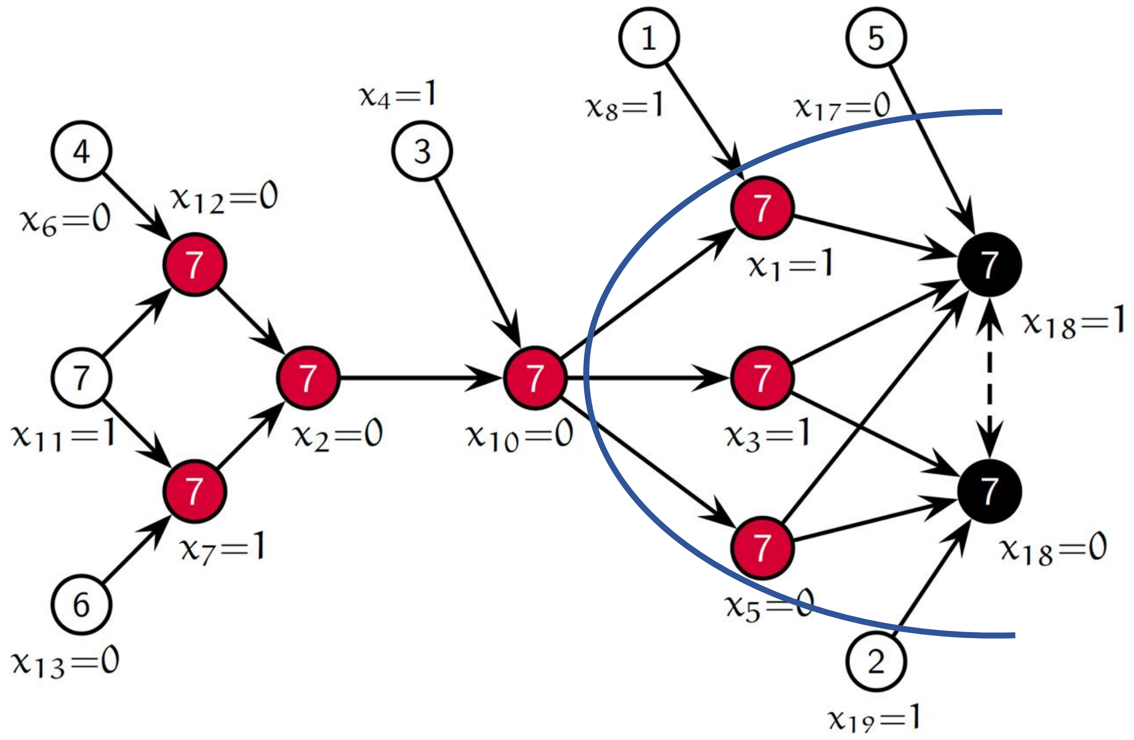


Backtrack using the learned conflict clause

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

Maximum decision level

Backtrack level



(x10 V ¬x8 V x17 V ¬x19)

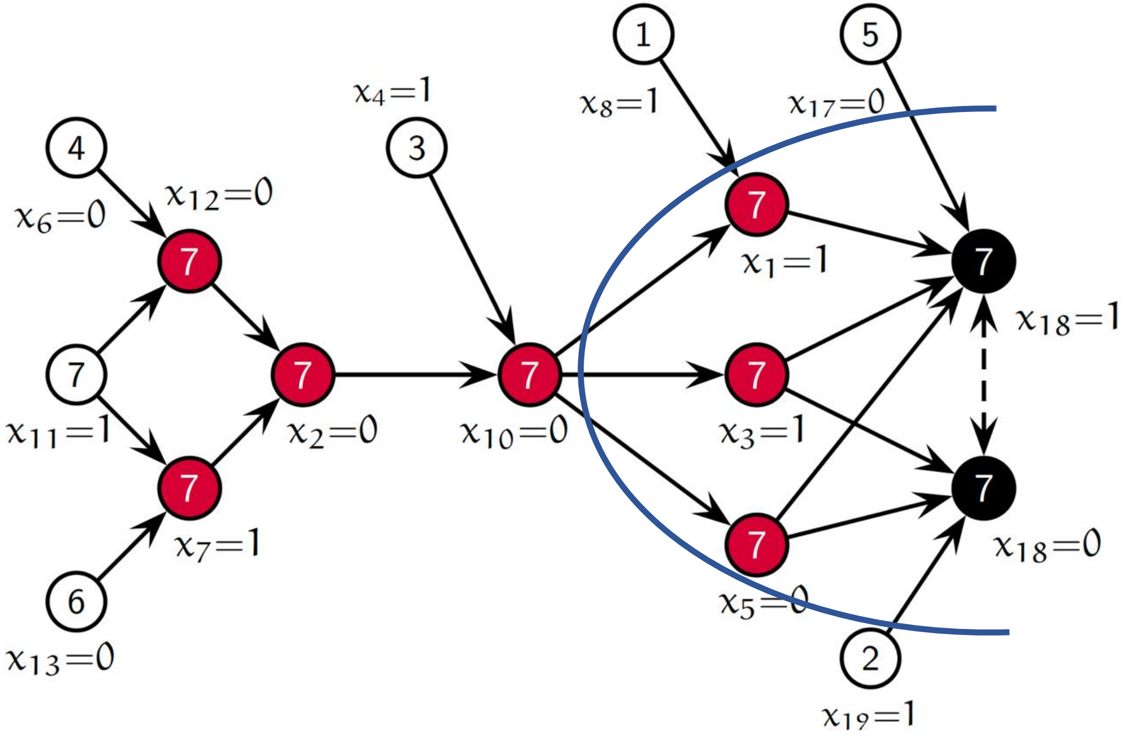
Backtrack level: ?

Backtrack using the learned conflict clause

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

Maximum decision level

Backtrack level



(x10 V ¬x8 V x17 V ¬x19)

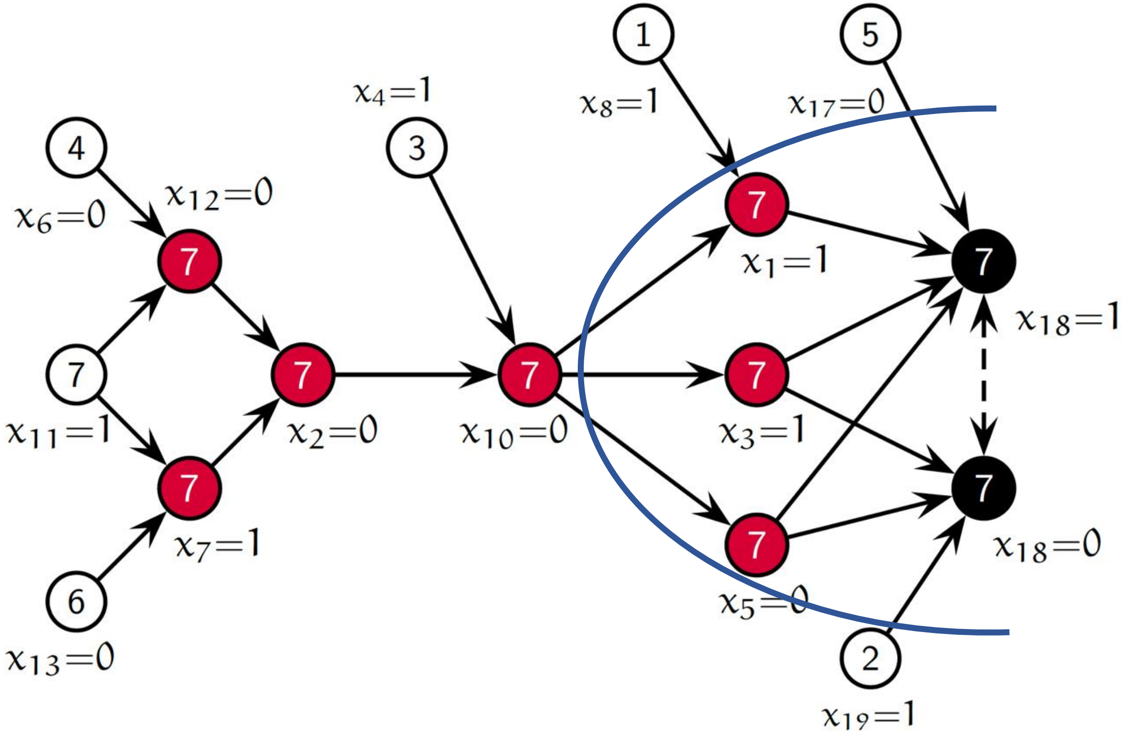
Backtrack level: 5

Backtrack using the learned conflict clause

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

Maximum decision level

Backtrack level



Why?

(x10 V ¬x8 V x17 V ¬x19)

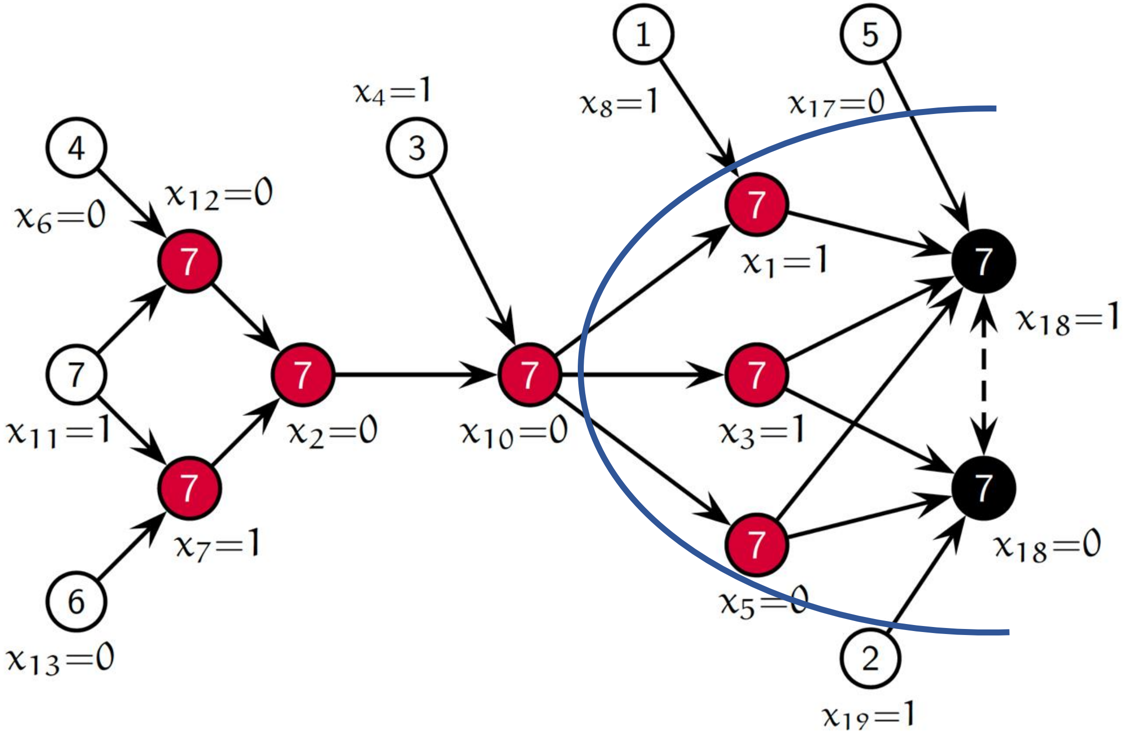
Backtrack level: 5

Backtrack using the learned conflict clause

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

Maximum decision level

Backtrack level



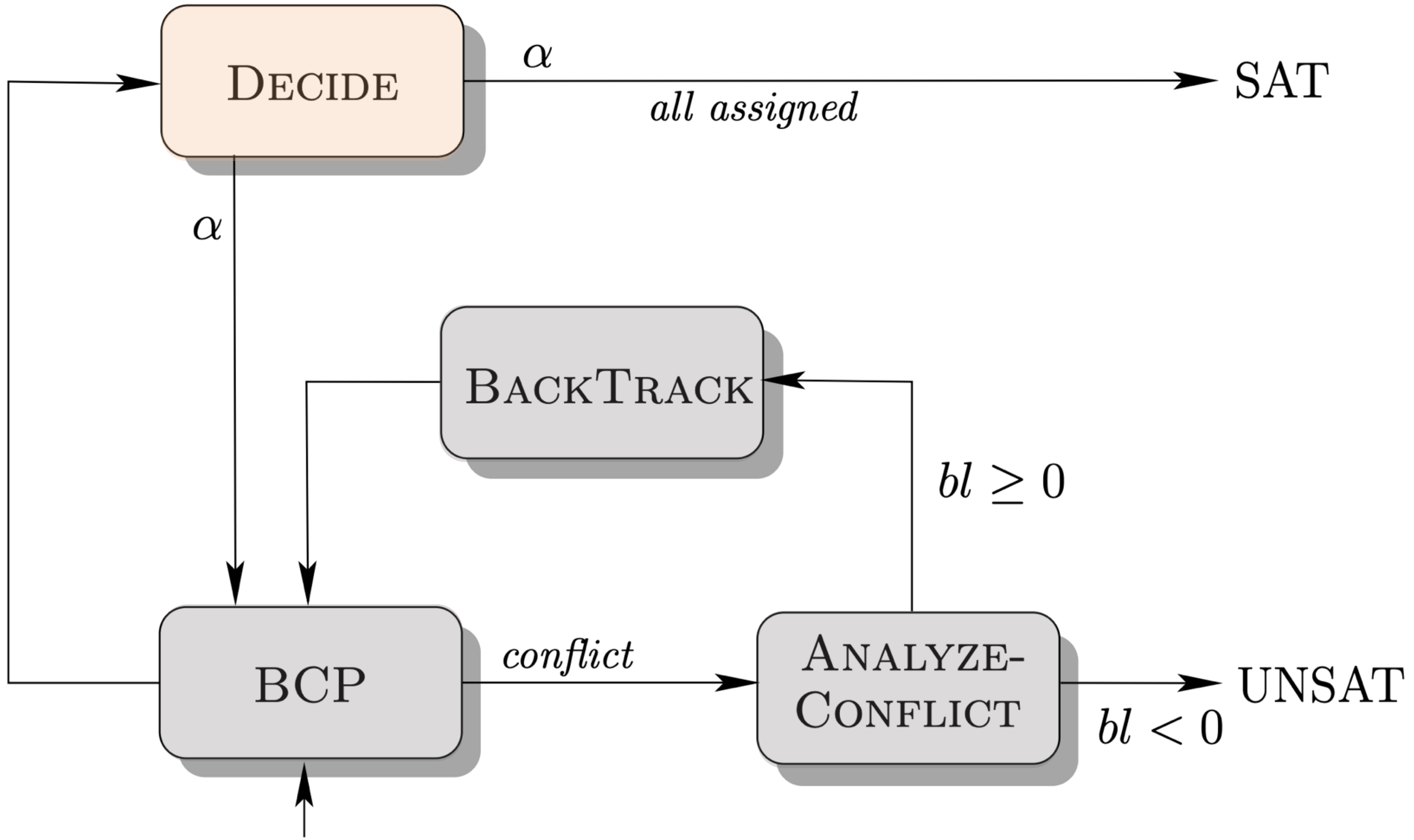
Because the conflict clause can become unit clause And we can flip the first UIP!

(x10 ∨ ¬x8 ∨ x17 ∨ ¬x19)

Backtrack level: 5

CDCL SAT solving

General Workflow



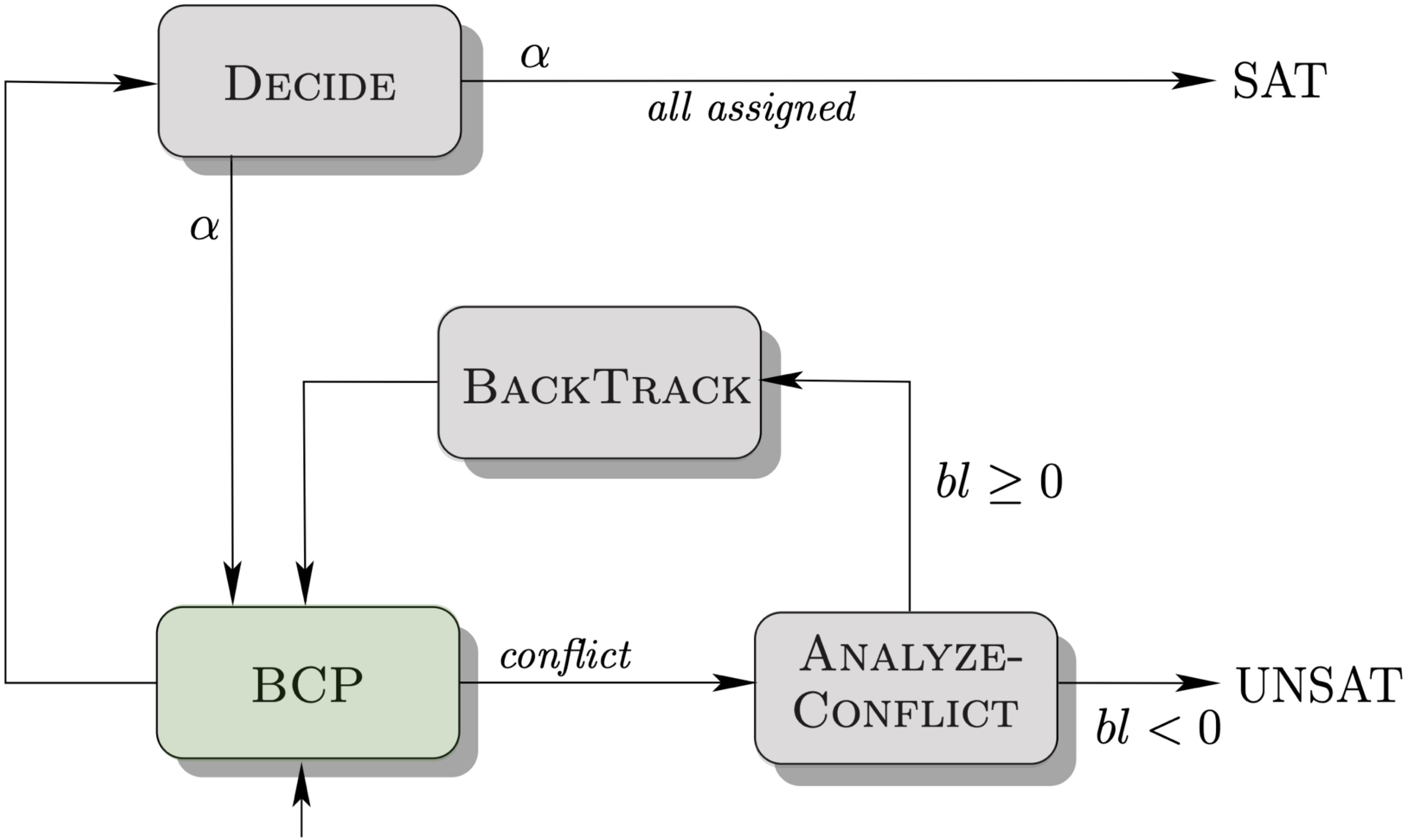
Decision Heuristics

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]

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

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

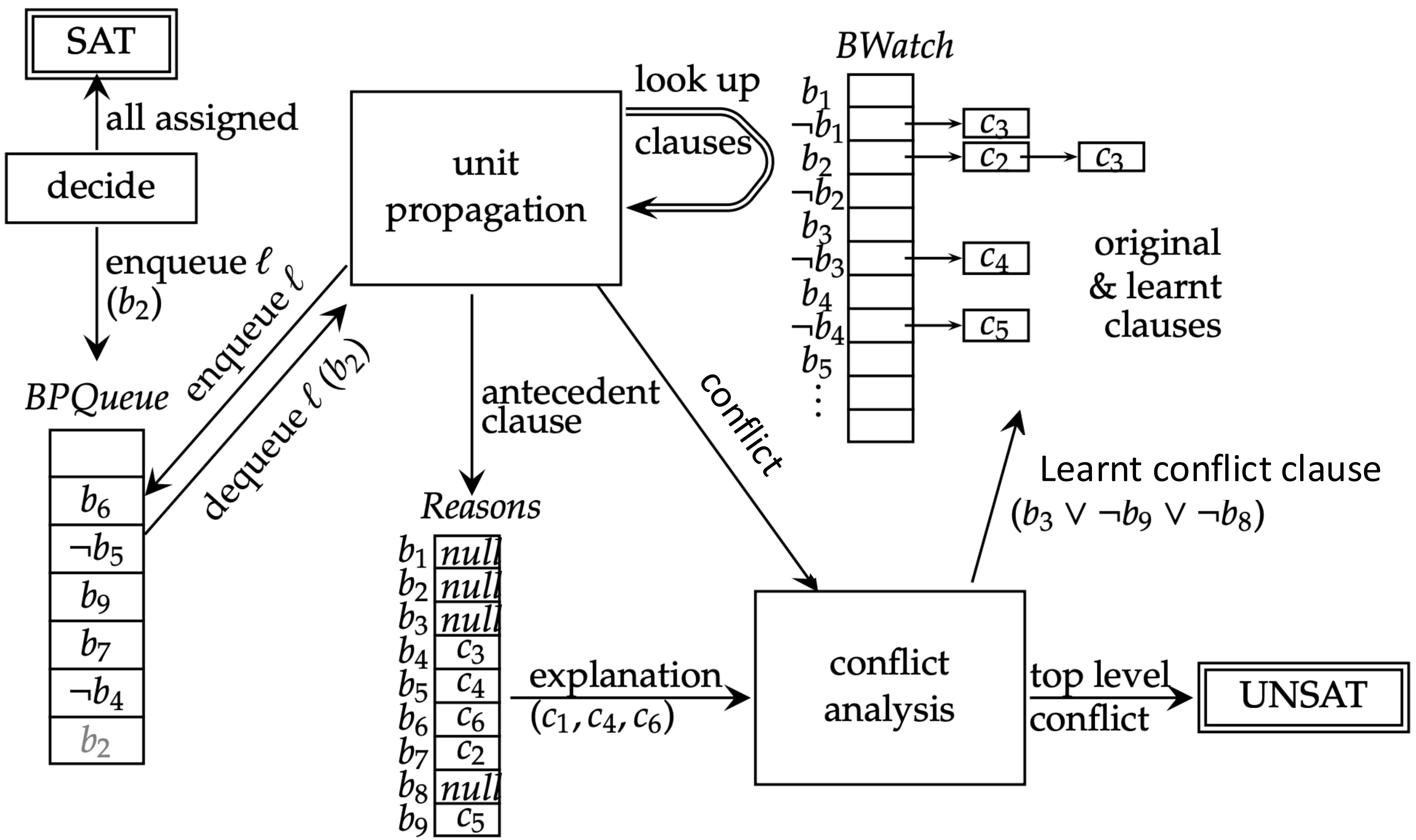
Discussions:

- 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

Improving CDCL SAT Solving using Graph Neural Networks

Wenxi Wang, Yang Hu, Mohit Tiwari, Sarfraz Khurshid, Kenneth McMillan, Risto Miikkulainen [ICLR'24]

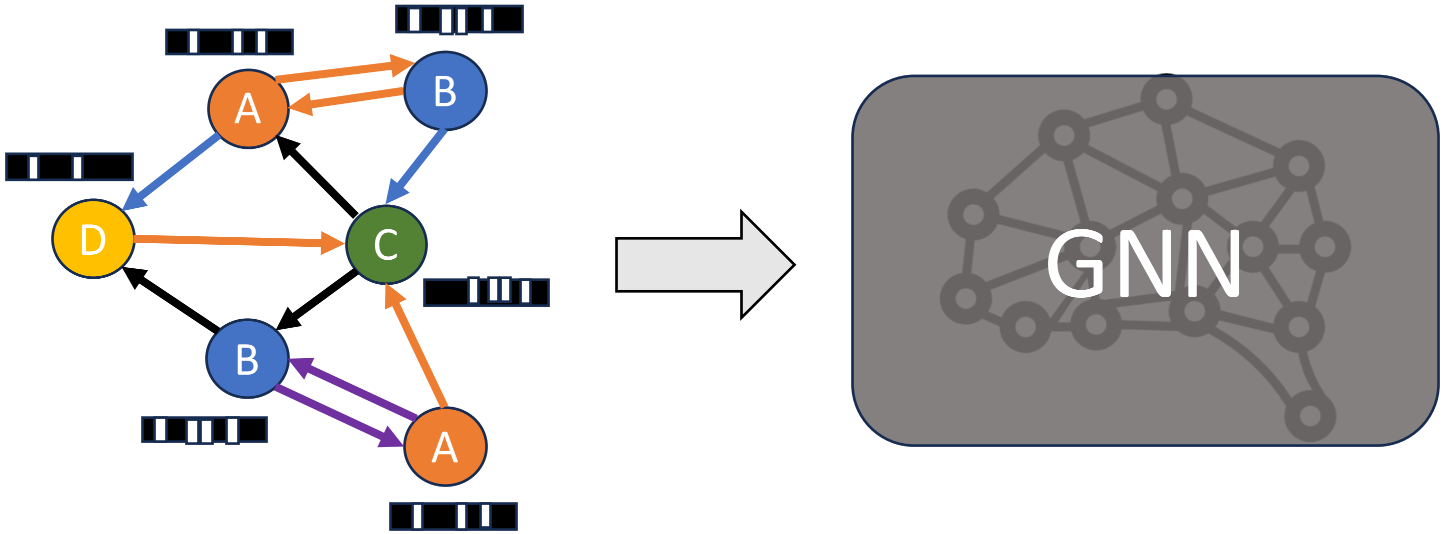
Armin Biere, Nils Froleyks, Wenxi Wang [SAT'23, Tool]

Background: GNN

A type of neural networks

Background: GNN

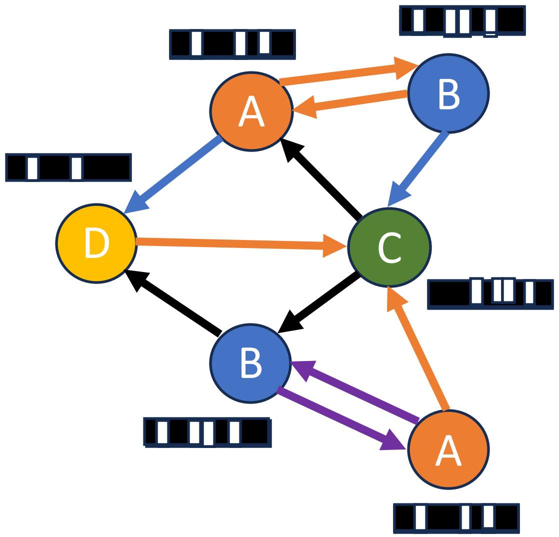
Operates on graph structured data



Initial node feature vectors

Background: GNN

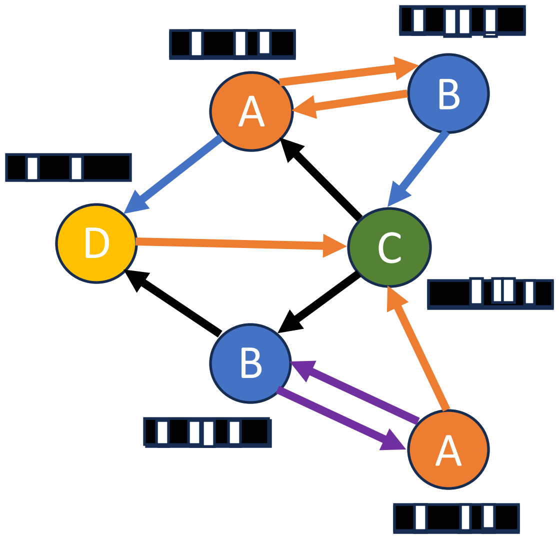
Message passing



Background: GNN

Message passing

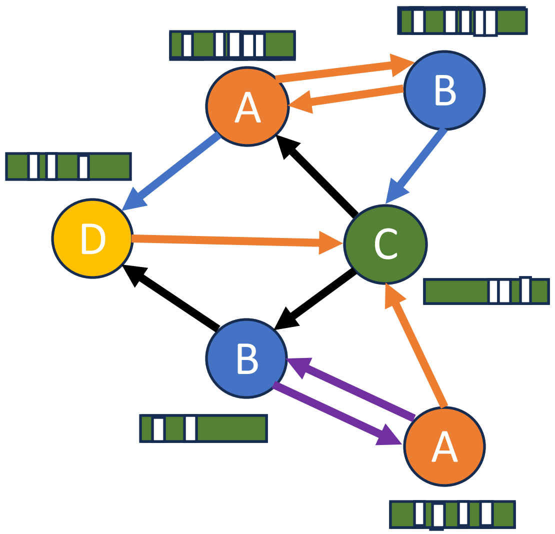
– aggregating and transforming node and edge information



Round 1

Background: GNN

Message passing
– aggregating and transforming node and edge information

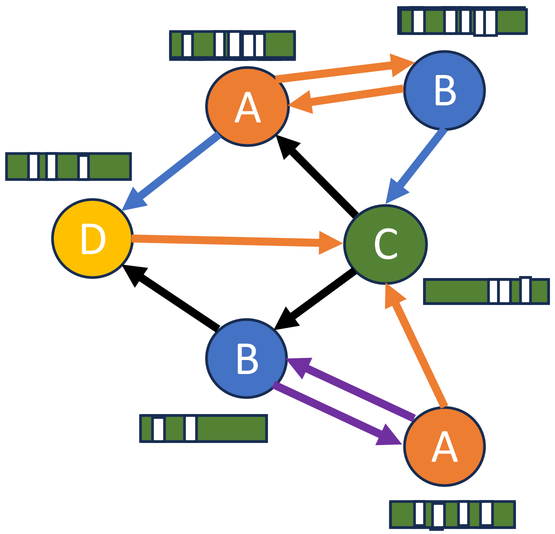


Round 1

Background: GNN

Message passing

– aggregating and transforming node and edge information

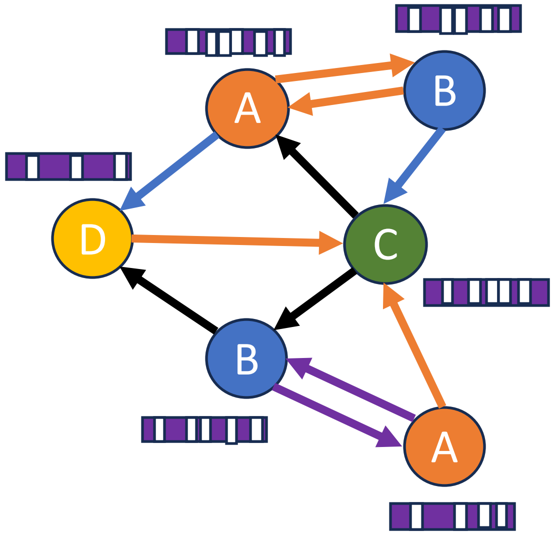


Round 2

Background: GNN

Message passing

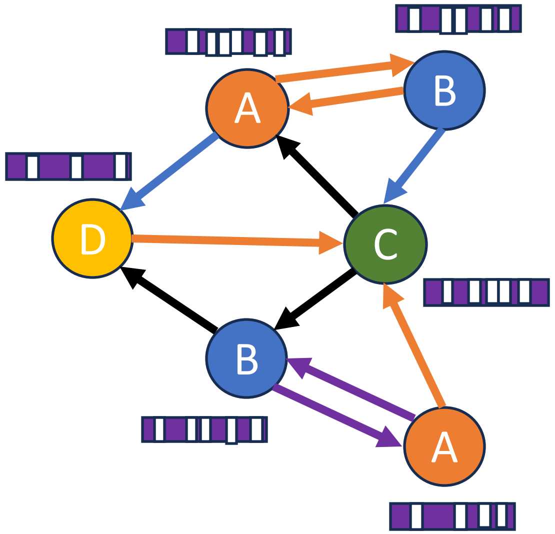
– aggregating and transforming node and edge information



Round 2

Background: GNN

Message passing
– aggregating and transforming node and edge information

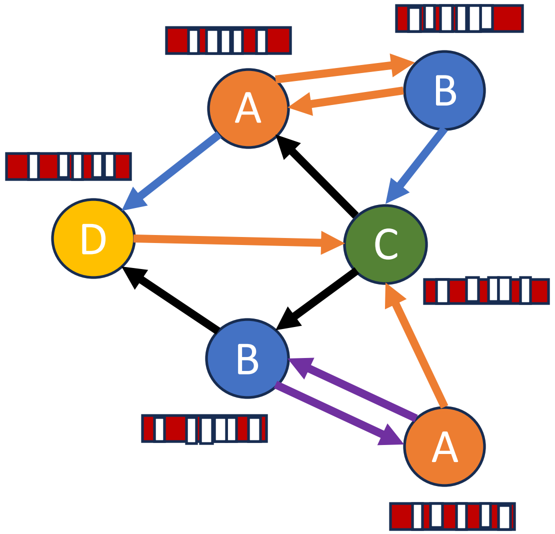


Round 3, 4, 5, …

Background: GNN

Message passing

– aggregating and transforming node and edge information

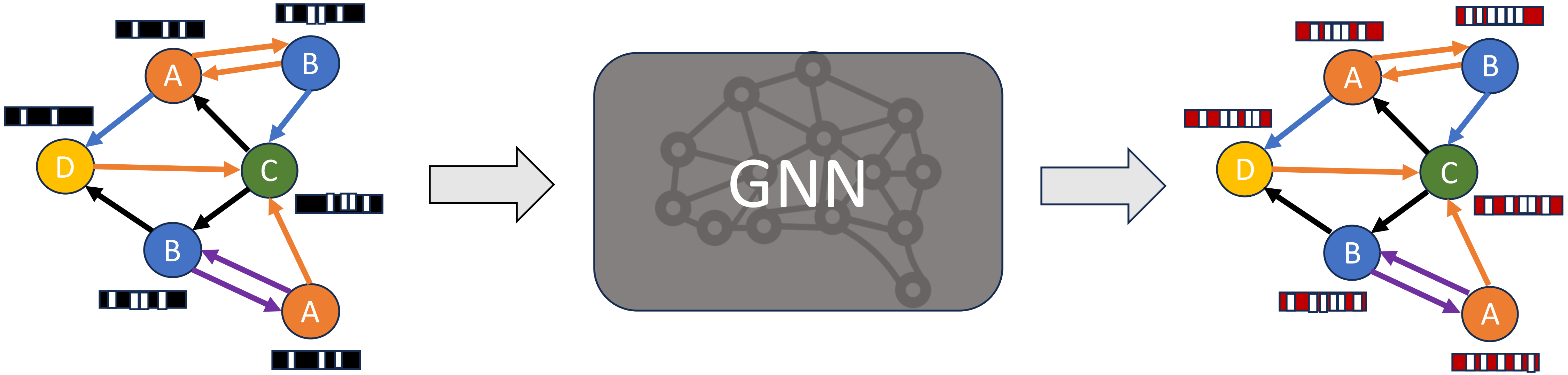


Round n

Background: GNN

Capture graph structures

- reason about complex relationships/dependencies



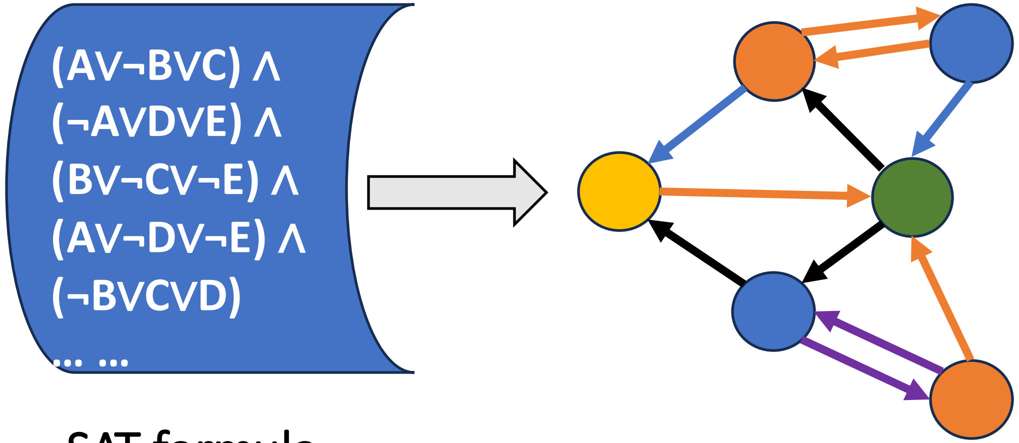
Initial node feature vectors

Updated node embeddings

Related: GNN for SAT

SAT formulas can be naturally converted into graphs

without information loss

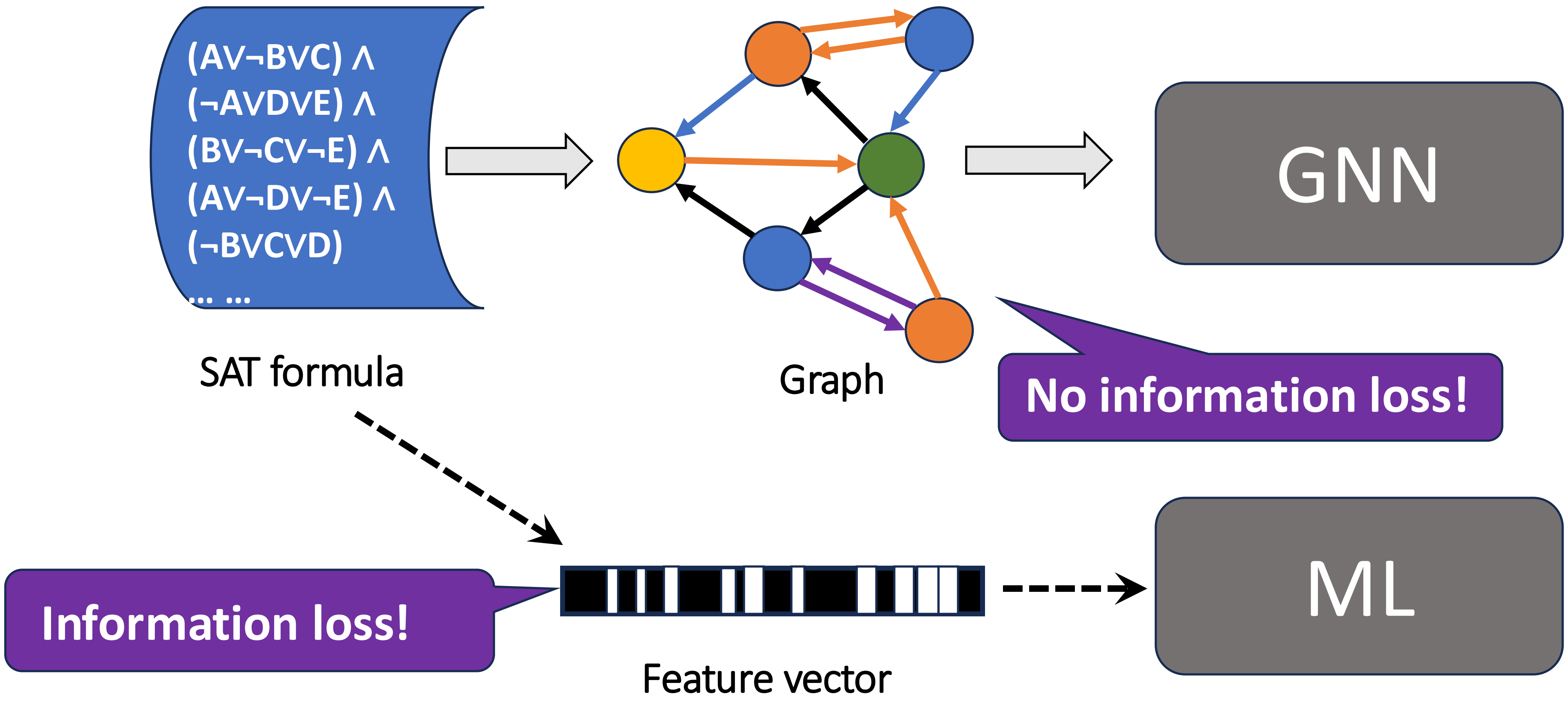


SAT formula

No information loss!

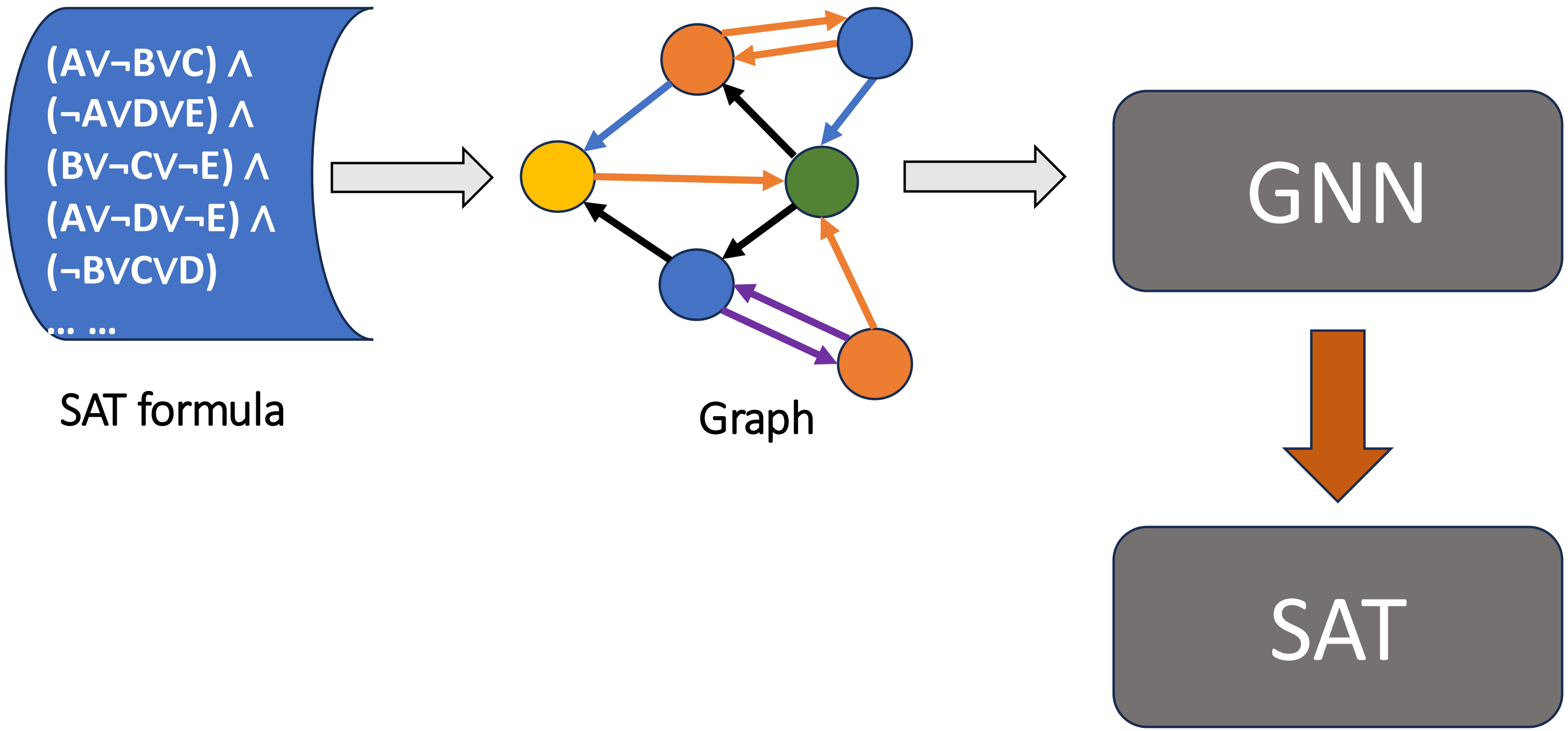
Related: GNN for SAT

GNN captures complex dependency information of SAT

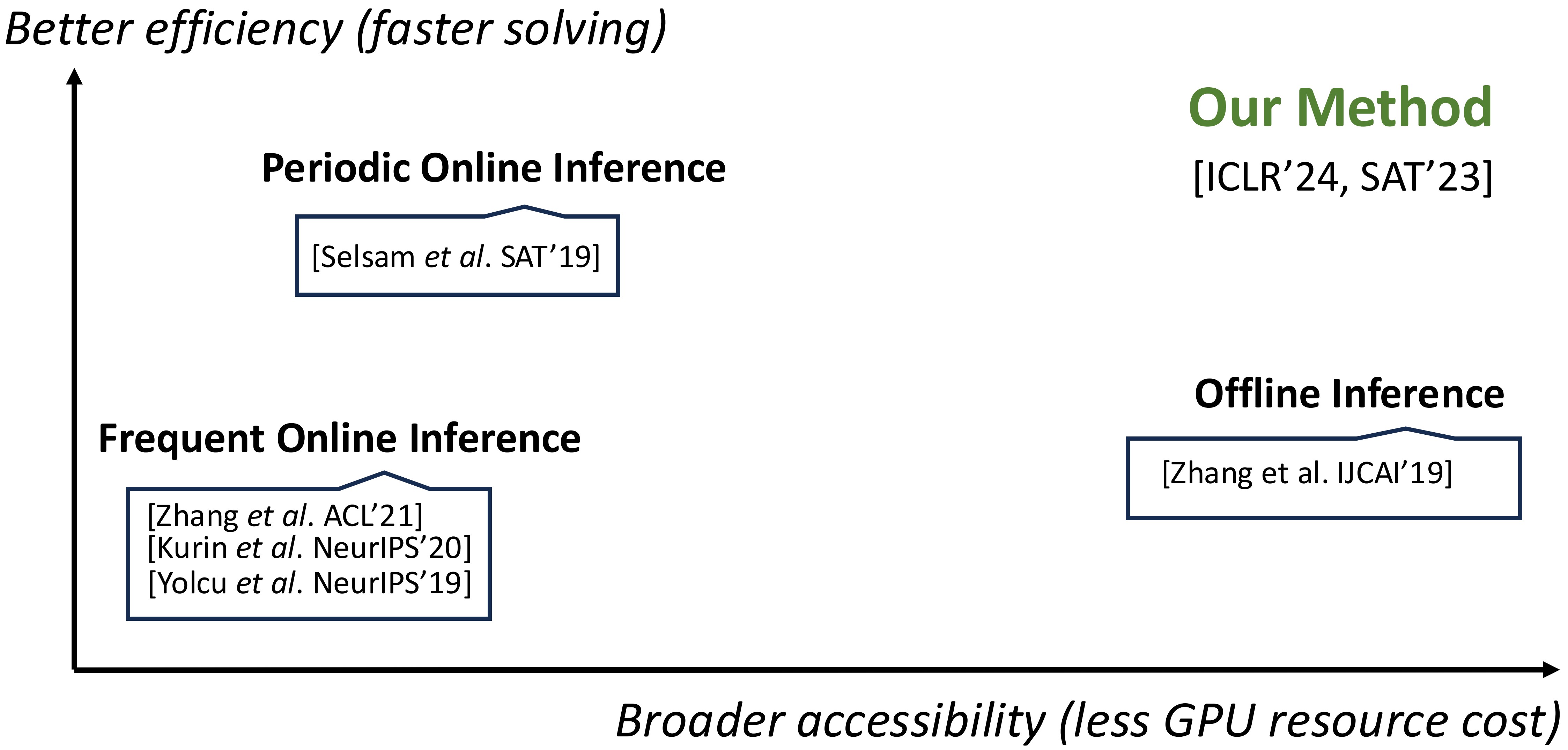


Related: GNN for SAT

Opens up deep learning for SAT field



Related: GNN for SAT



Our Insight

Using **offline** GNN inference to predict instructive static information

Values of backbone variables

Background: Backbone[Parkes, 1997]

Variables that have the same value across all possible solutions

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

All SAT solutions:

v_1 = false	v_2 = **true**	v_3 = **true**
v_1 = false	v_2 = **true**	v_3 = **false**

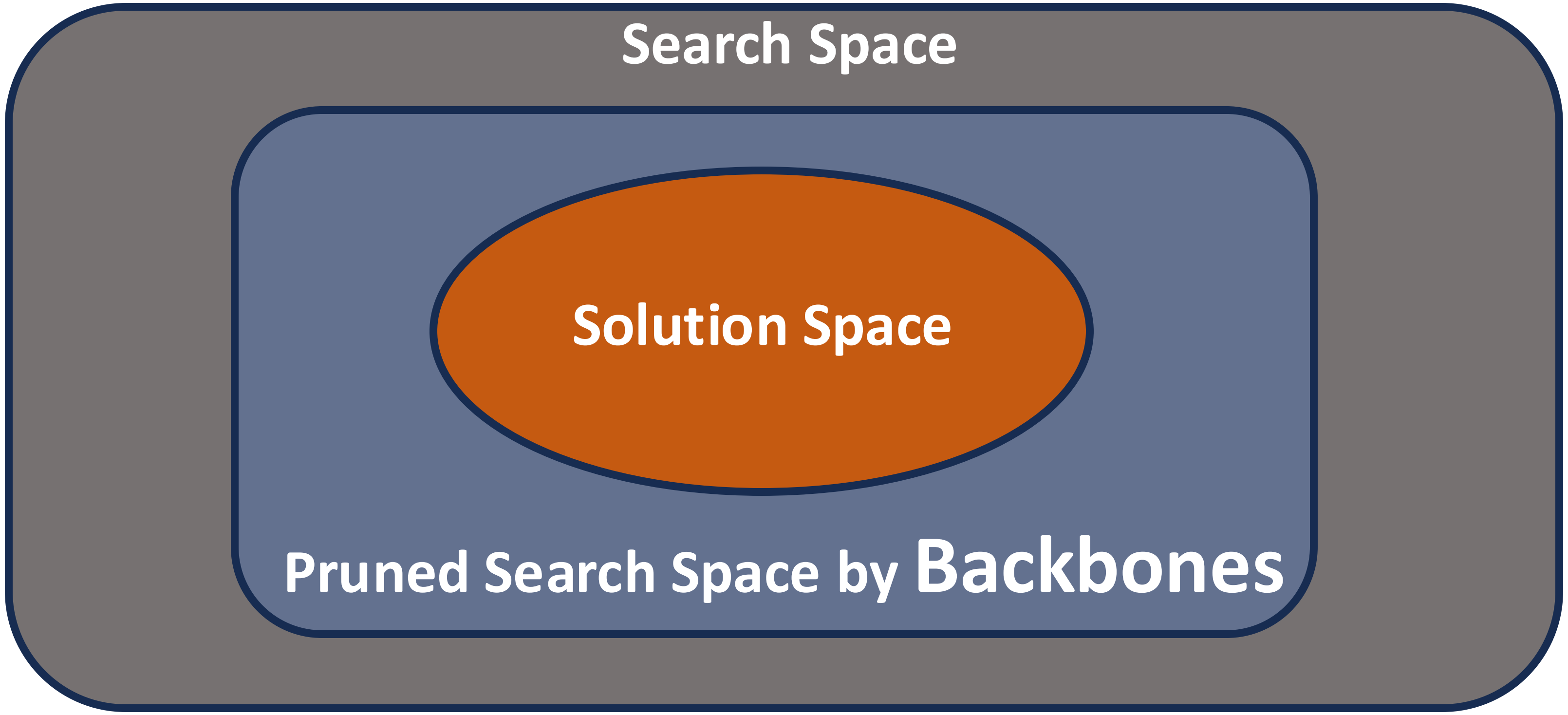
v_1 is the negative backbone

v_2 is the positive backbone

Background: Backbone[Parkes, 1997]

In theory, backbones can enhance SAT!

Satisfiable case: Increase solution-to-search space ratio



Challenge on Backbone Computation

In practice, hard to apply backbones to facilitate SAT!

Very expensive to compute backbones!

Our Idea

Very expensive to compute backbones

Using offline GNN inference to predict backbones!

Our Idea: Advantage

Using offline GNN inference to predict backbones

Much faster than computing backbones!

Our Idea: Challenges

Using offline GNN inference to predict backbones

1. How to make accurate predictions?

2. What if predictions contain a small fraction of errors?

Our Method: NeuroBack

Using offline GNN inference to predict backbones

1. How to make accurate predictions?

Training a robust GNN model

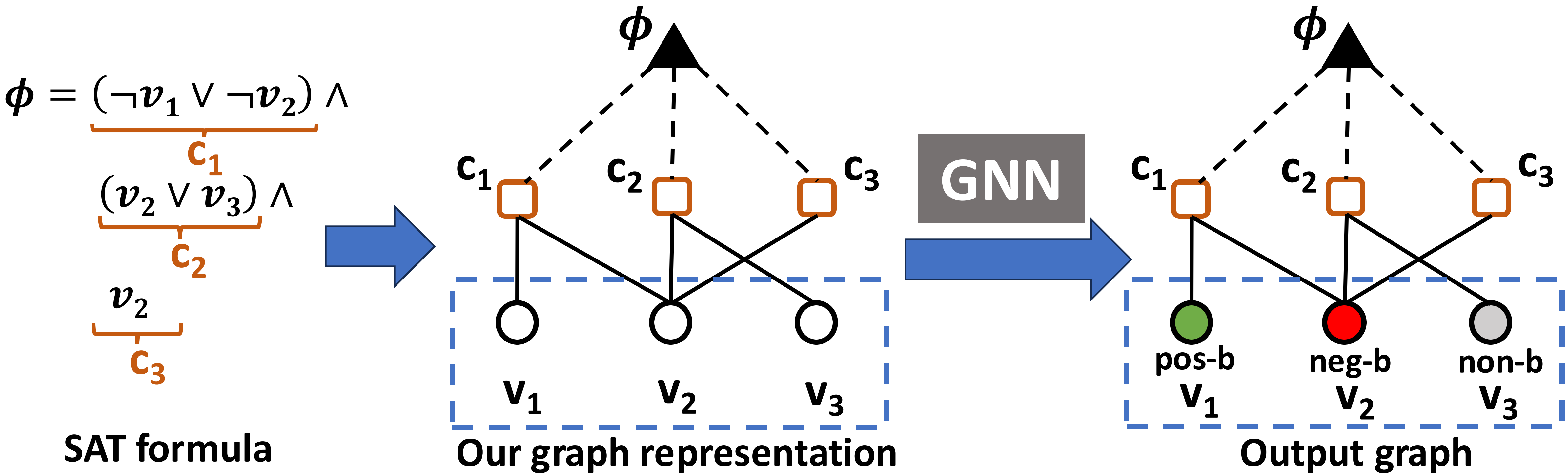
2. What if predictions contain a small fraction of errors?

Applying predictions cleverly

Our Method: NeuroBack

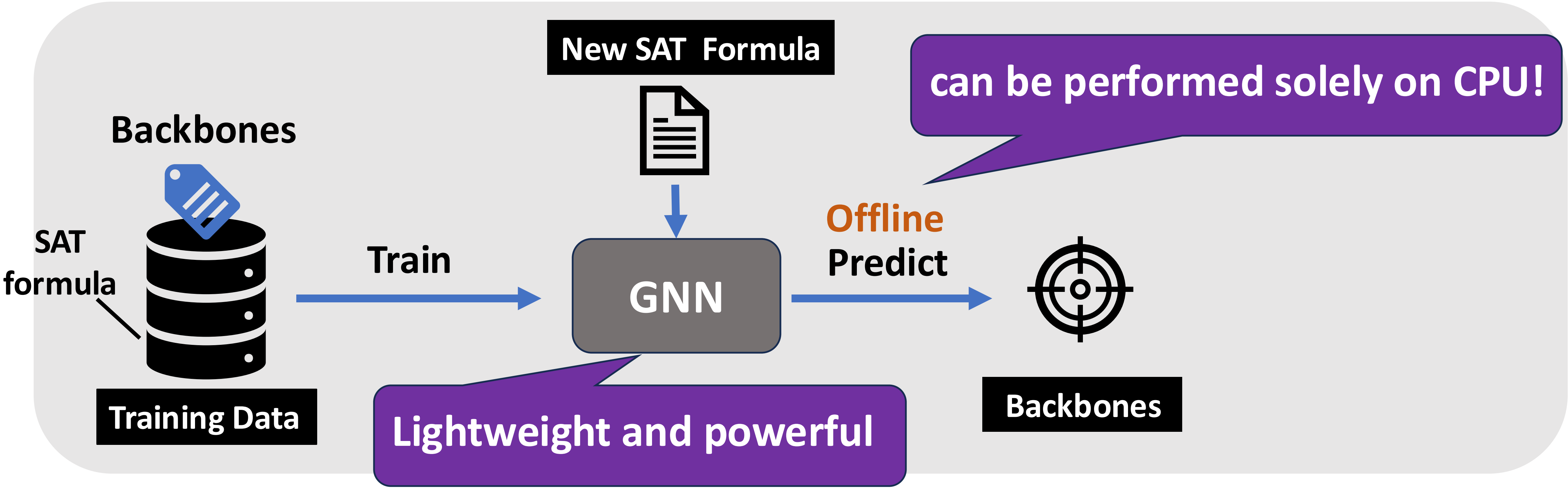
Train GNN to predict backbones

Node classification problem



Our Method: NeuroBack

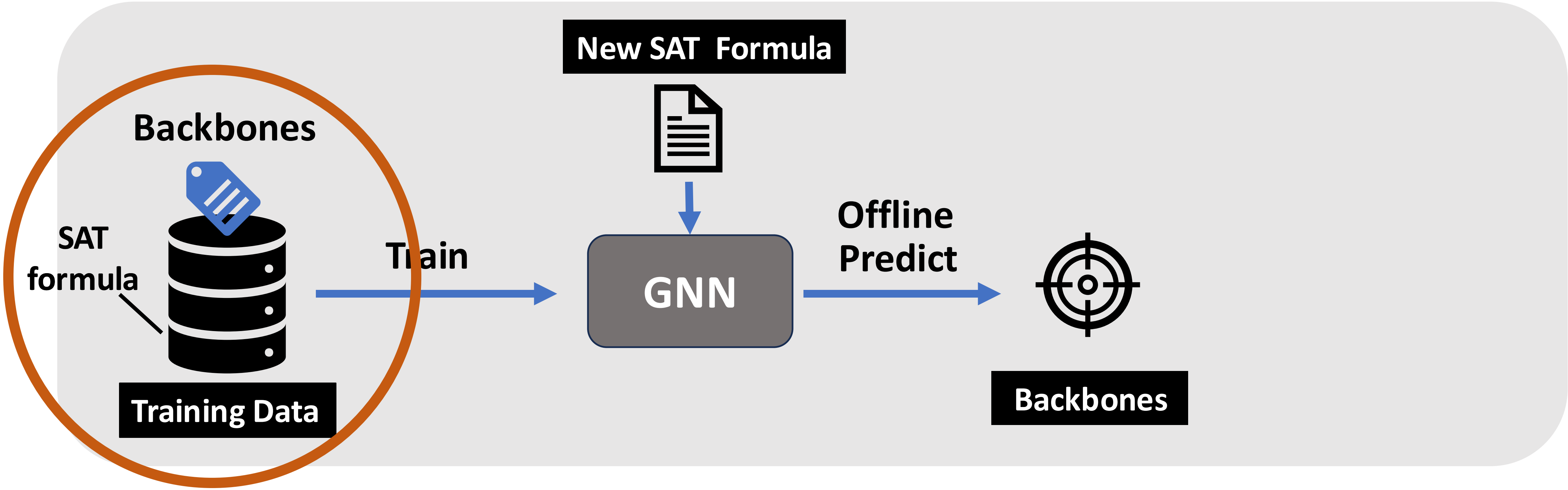
Train GNN to predict backbones **offline**



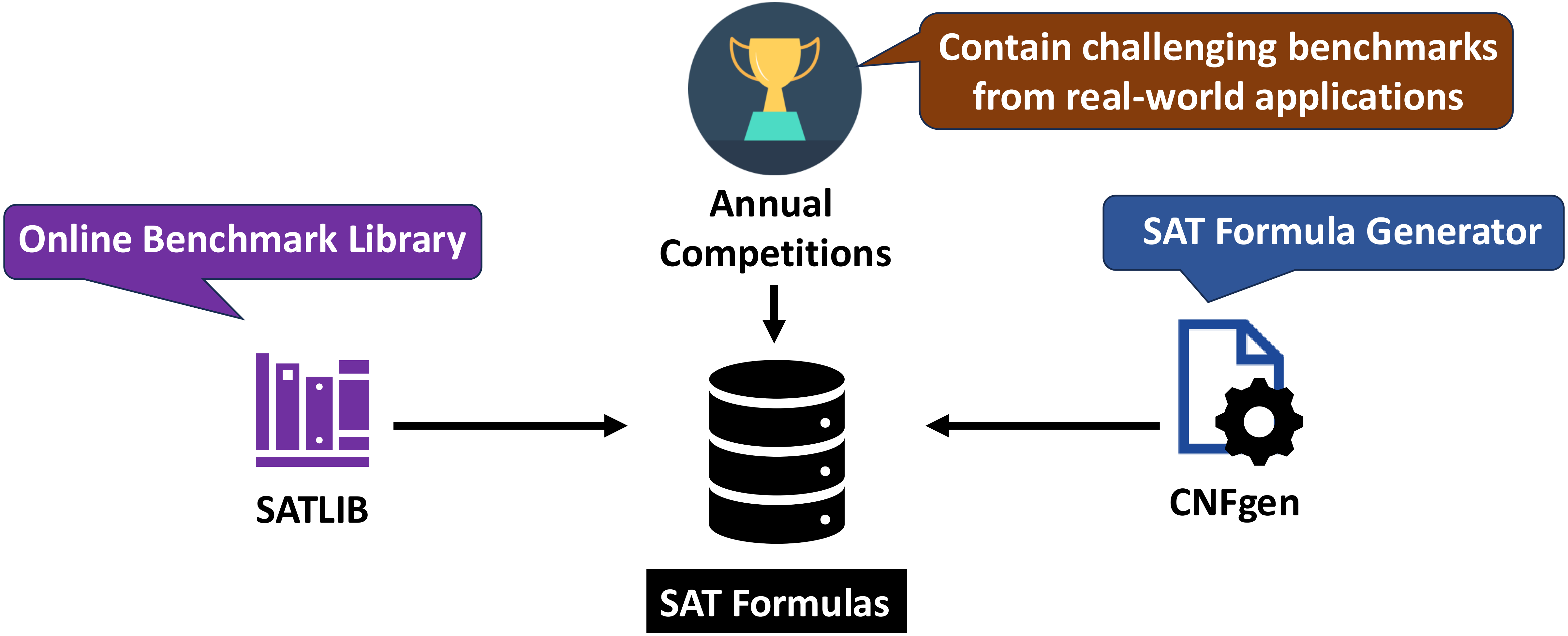
Our Method: NeuroBack

Train a **robust** GNN to predict backbones **accurately**

Training data is the key!

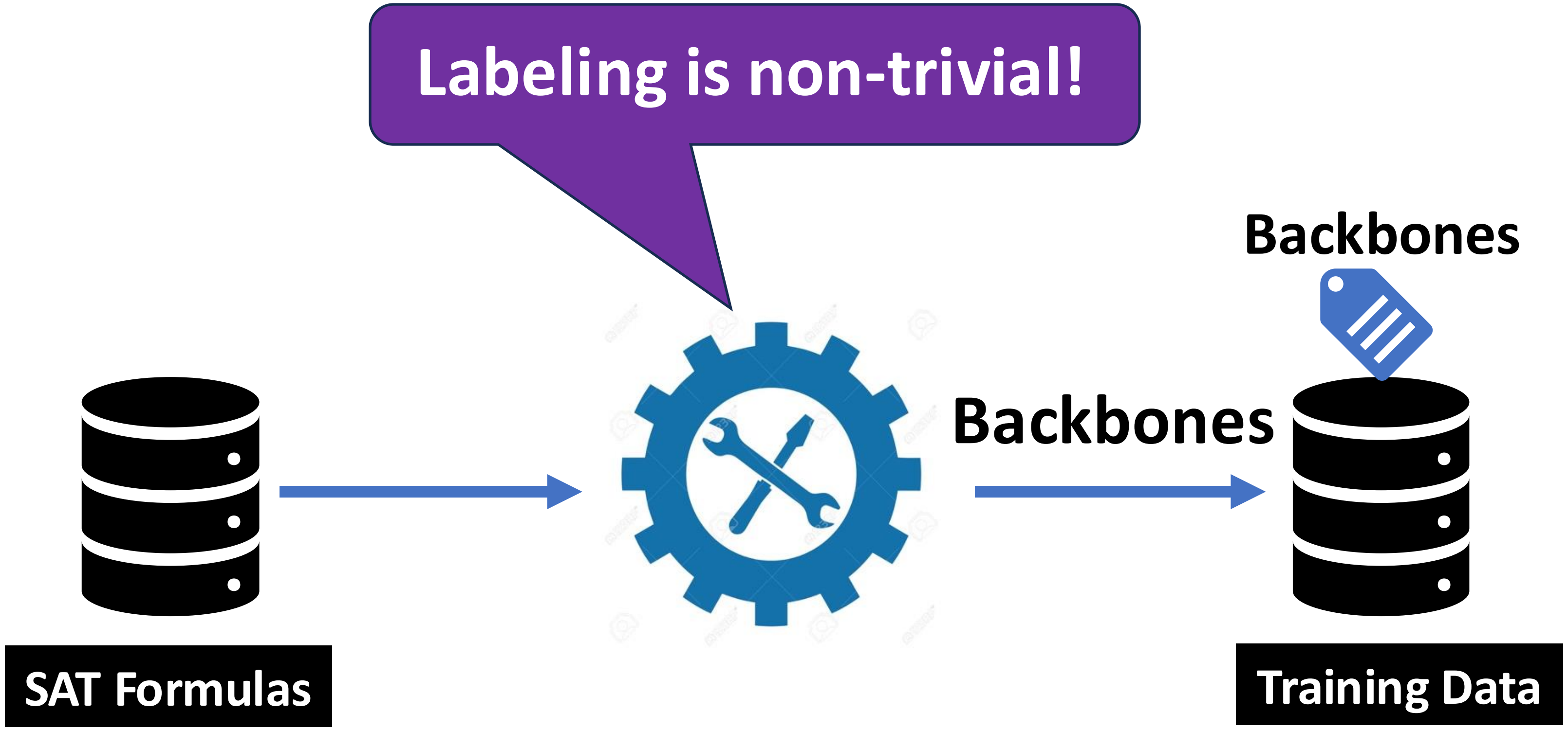


Data Collection



Data Labeling

Existing backbone computation tools are outdated and inefficient! ☹

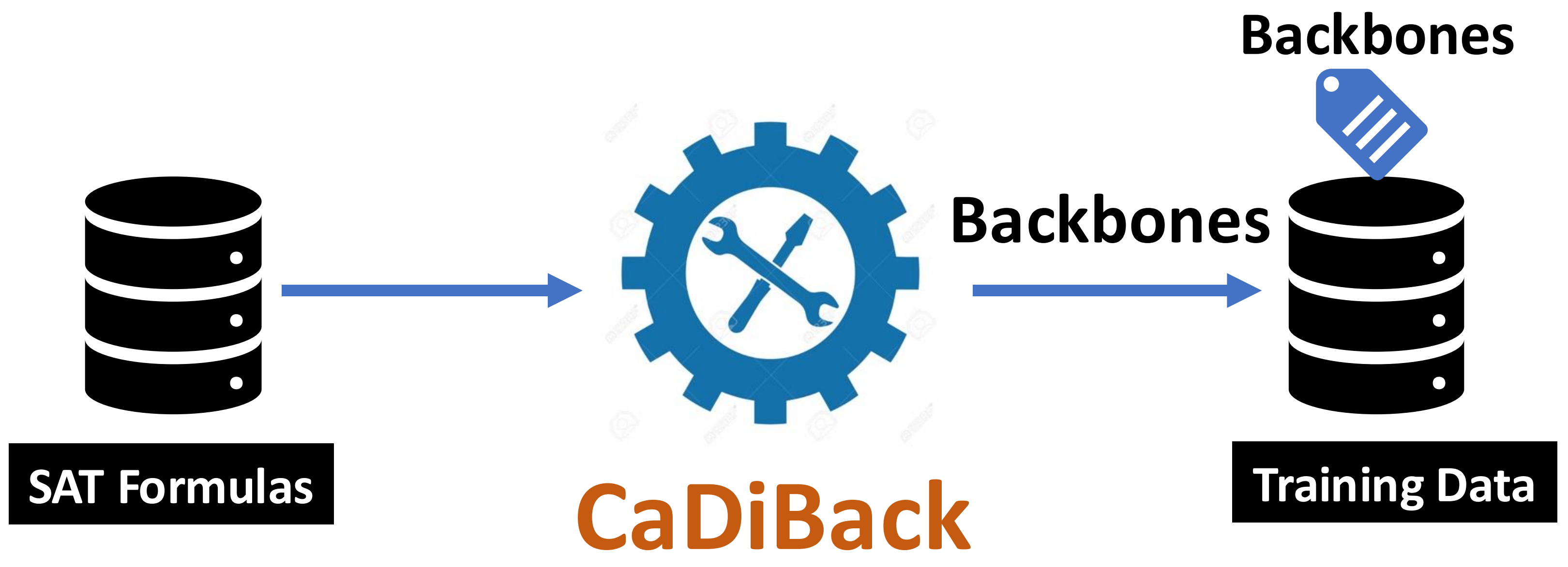


Data Labeling: CaDiBack

We developed **CaDiBack** on top of **CadiCaL** [Biere et al.]

State-of-the-art!
Extract backbones for 60% more problems from past 10 years of SAT competitions

Cutting edge SAT solver



Dataset: DataBack

First public large dataset in deep learning for SAT!

containing 120,286 data samples

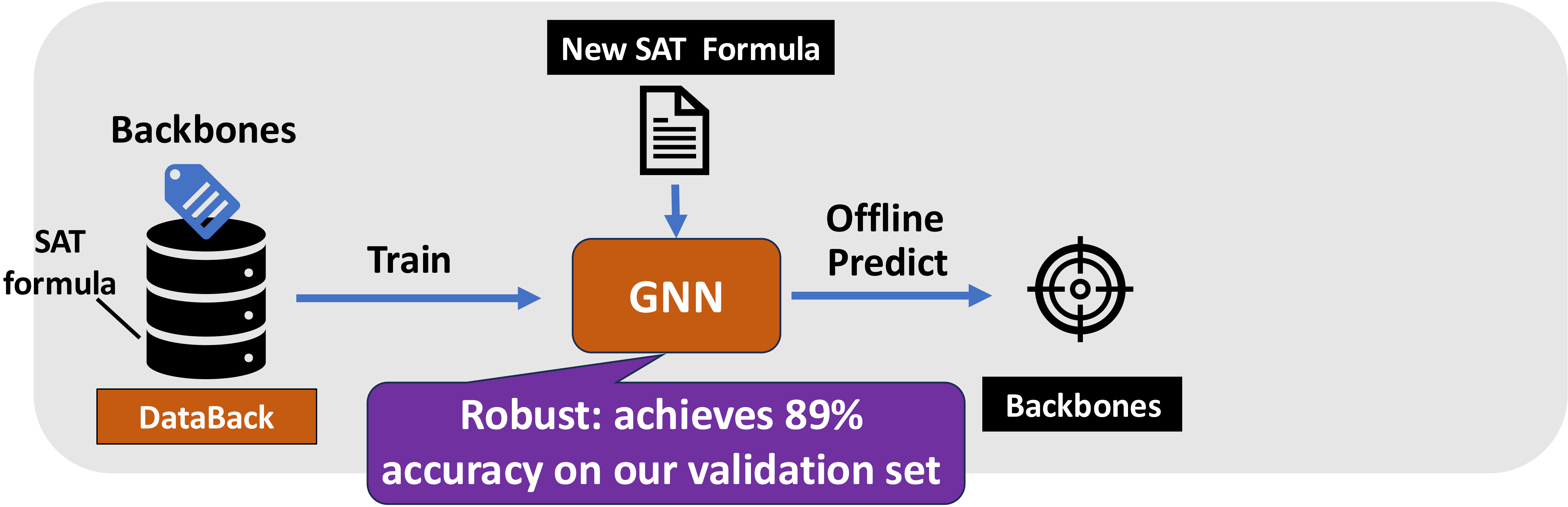
Backbones

DataBack

Available online: https://huggingface.co/datasets/neuroback/DataBack

Our Method: NeuroBack

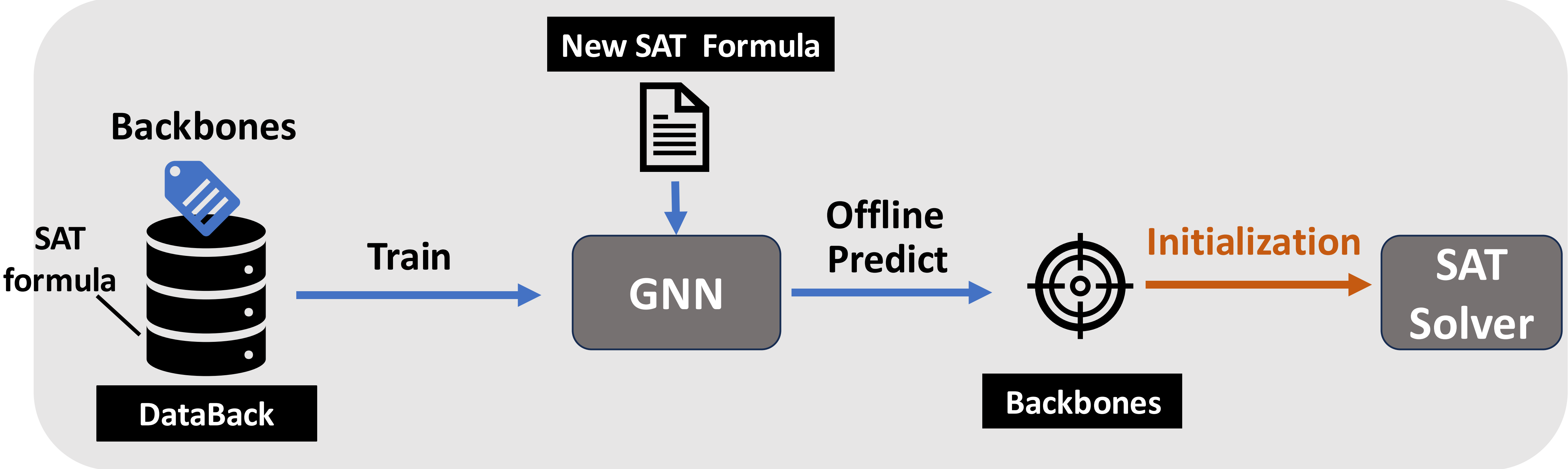
Train a **robust** GNN to predict backbones **accurately**



Our Method: NeuroBack

Apply backbone predictions **cleverly** to facilitate SAT

Enhance variable value selection heuristic in SAT



Our Method: NeuroBack

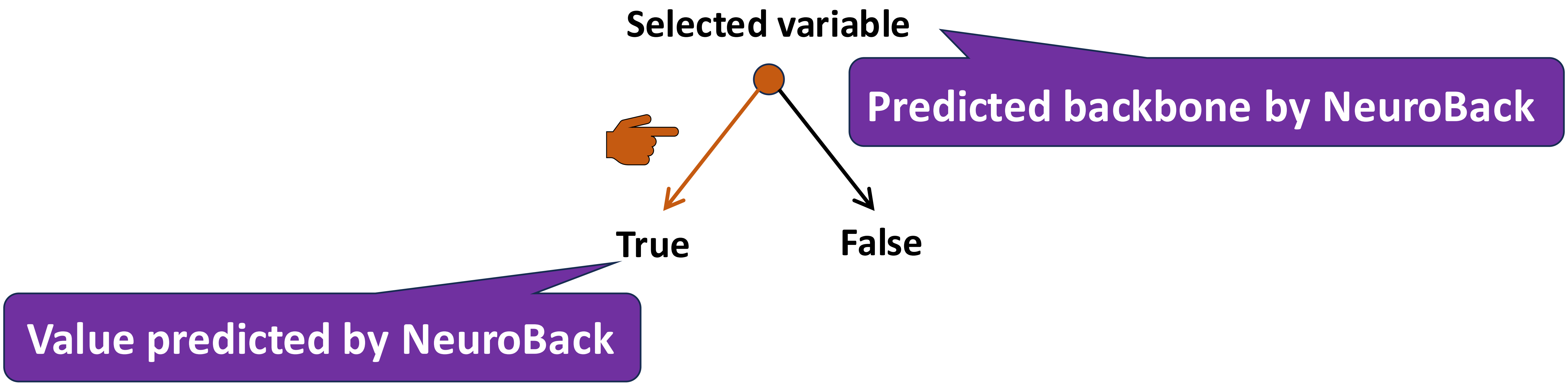
Apply backbone predictions **cleverly** to facilitate SAT

Enhance variable value selection heuristic in SAT

Our Method: NeuroBack

Apply backbone predictions **cleverly** to facilitate SAT

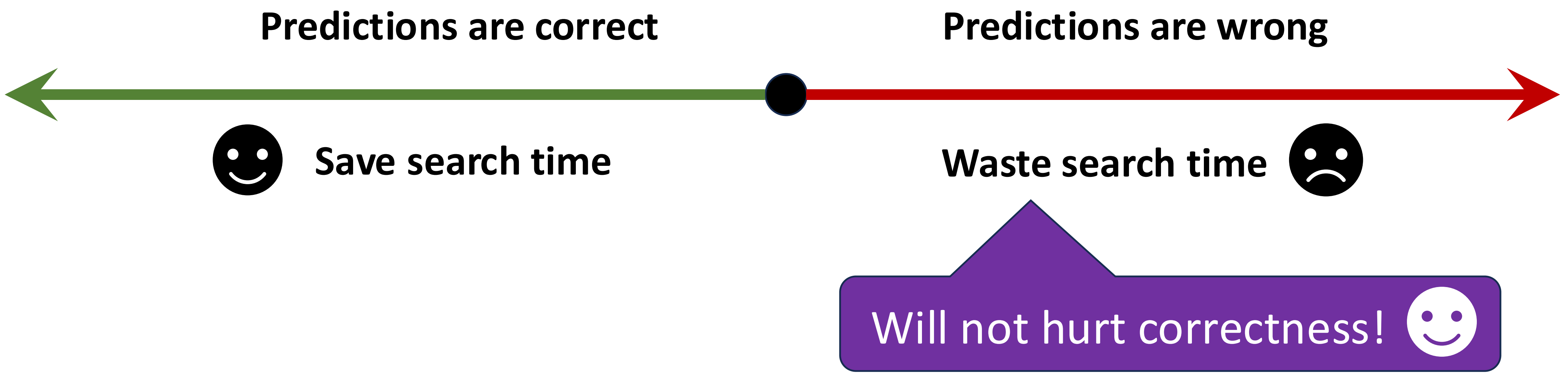
Enhance variable value selection heuristic in SAT



Our Method: NeuroBack

Apply backbone predictions **cleverly** to facilitate SAT

Can benefit from neural predictions even if they contain errors



Our Method: NeuroBack

Apply backbone predictions **cleverly** to facilitate SAT

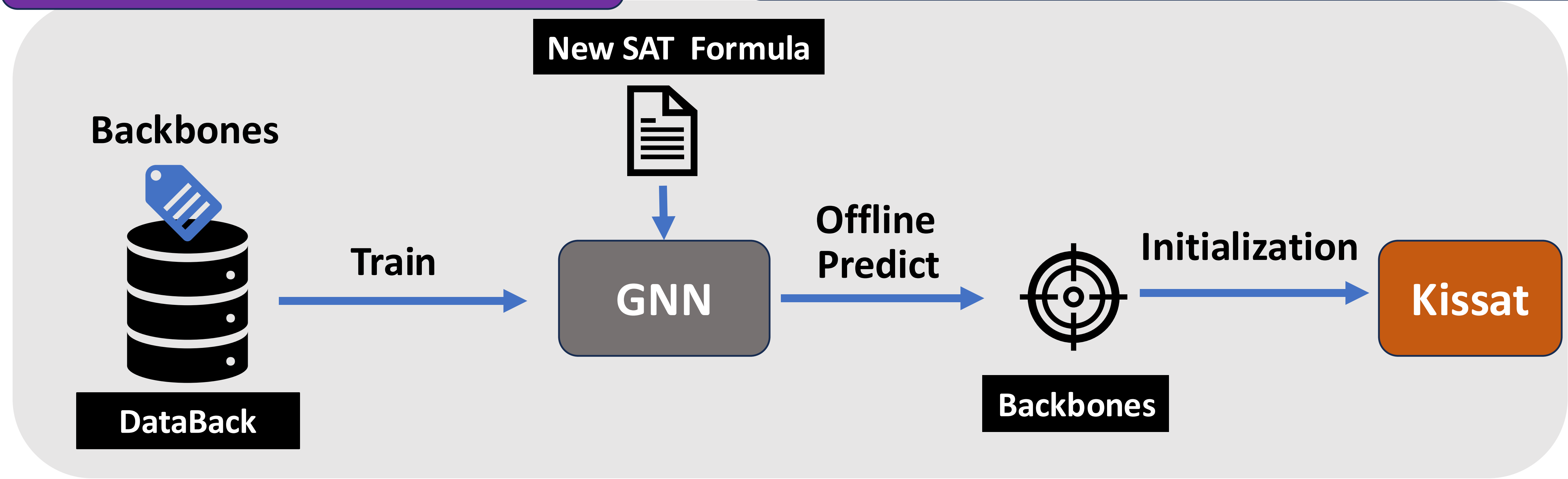
Goal: make the gain much more than the loss

Our Method: NeuroBack

First to enhance Kissat [Biere et al.] using GNN!

State-of-the-art SAT solver!

Well-engineered, very hard to optimize!



Results

The **first** success in enhancing Kissat using GNN in recent SAT competitions

Standard Time Limit per problem: 5,000 seconds

	SATCOMP-2022	SATCOMP-2023
More Problems Solved:	5.2%	7.4%
Time Saved (per problem):	117 seconds (5.0%)	246 seconds (10.4%)