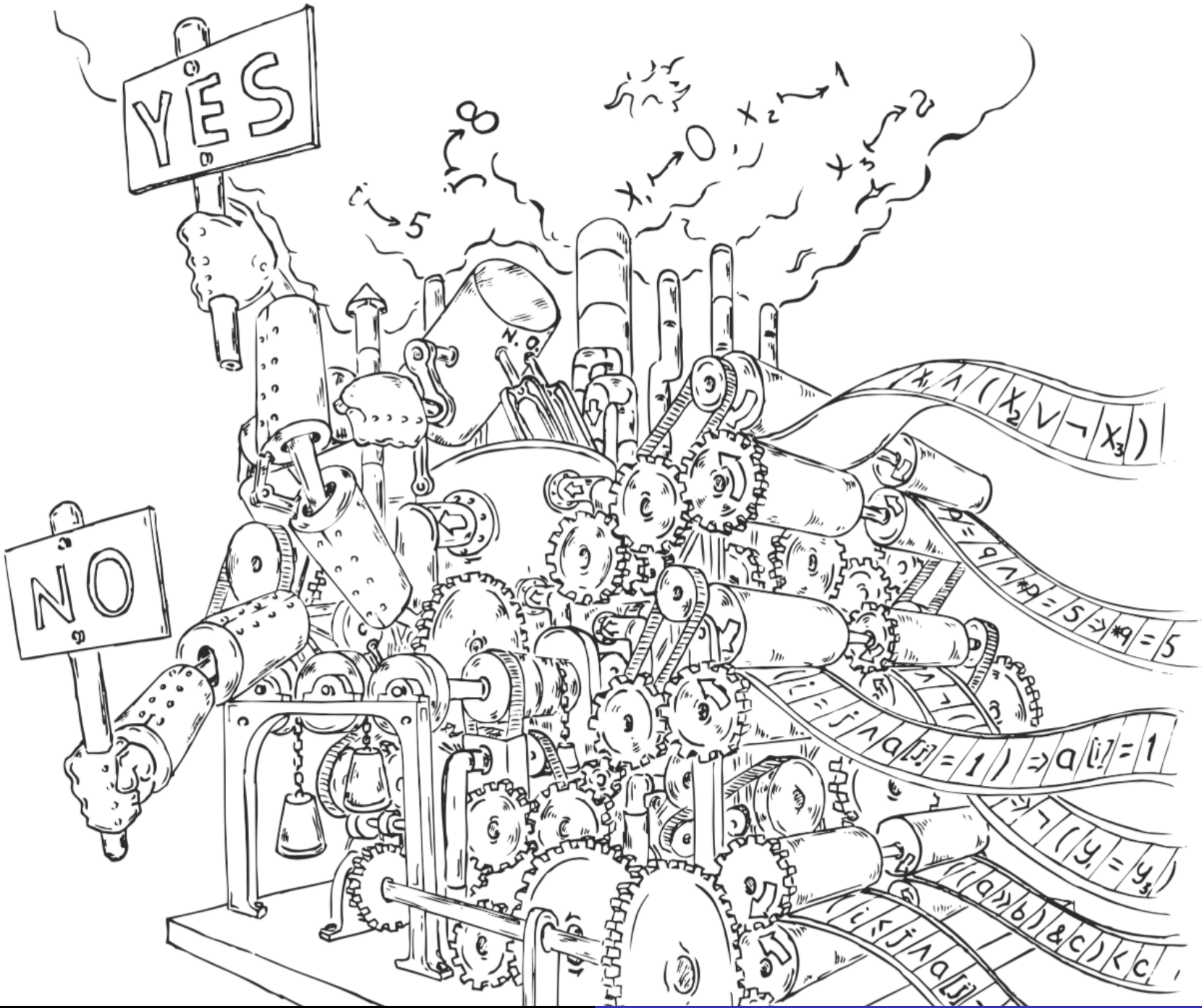

YES
NO

SAT/SMT solvers
3. Davis–Putnam–Logemann–Loveland(Theory)

Roman Kholin

Lalambda

Tbilisi, 2023

Definitions

- $at(\varphi)$ - the set of Σ-atoms in a given NNF formula φ
- $at_i(\varphi)$ - the i-th distinct atom in φ
- $e(\varphi)$ - Boolean variable called encoder of this atom
- $e(\varphi)$ - Boolean formula resulting from substituting each Σ-atom in φ with its Boolean encoder

$e(\varphi)$ is also called propositional skeleton of φ

- $\varphi := x = y \vee x = z$
- $at_1(\varphi) := x = y, at_2(\varphi) := x = z$
- $e(x = y) := b_1$
- $e(x = z) := b_2$
- $e(\varphi) := b_1 \vee b_2$

Definitions

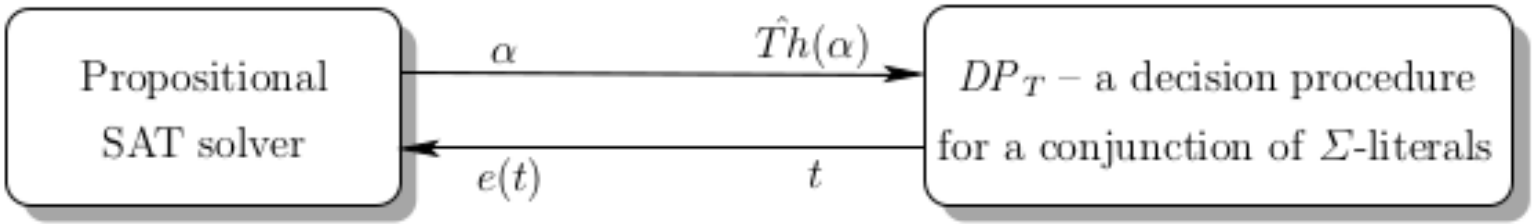
- α - assignment of $e(\phi)$
- $Th(at_i, \alpha) = at_i$, if $\alpha(at_i) = TRUE$, $\neg at_i$ otherwise
- $Th(\alpha) = \{Th(at_i, \alpha) | e(at_i), \alpha\}$
- $\overline{Th(\alpha)}$ - conjunction of $Th(\alpha)$

$$\varphi := x = y \wedge ((y = z \wedge \neg(x = z)) \vee x = z)$$


CDCL
LAZY-SMT
CDCL LAZY-SMT
DPLL(T)
imgflip.com

Lazy SMT

1. **function** LAZY-BASIC(φ)
2. $\quad \mathcal{B} := e(\varphi)$;
3. $\quad$ **while** (TRUE) **do**
4. $\quad\quad \langle \alpha, res \rangle :=$ SAT-SOLVER($\mathcal{B}$);
5. $\quad\quad$ **if** $res =$"Unsatisfiable" **then return** "Unsatisfiable";
6. $\quad\quad$ **else**
7. $\quad\quad\quad \langle t, res \rangle :=$ DEDUCTION($\hat{Th}(\alpha)$);
8. $\quad\quad\quad$ **if** $res =$"Satisfiable" **then return** "Satisfiable";
9. $\quad\quad\quad \mathcal{B} := \mathcal{B} \wedge e(t)$;



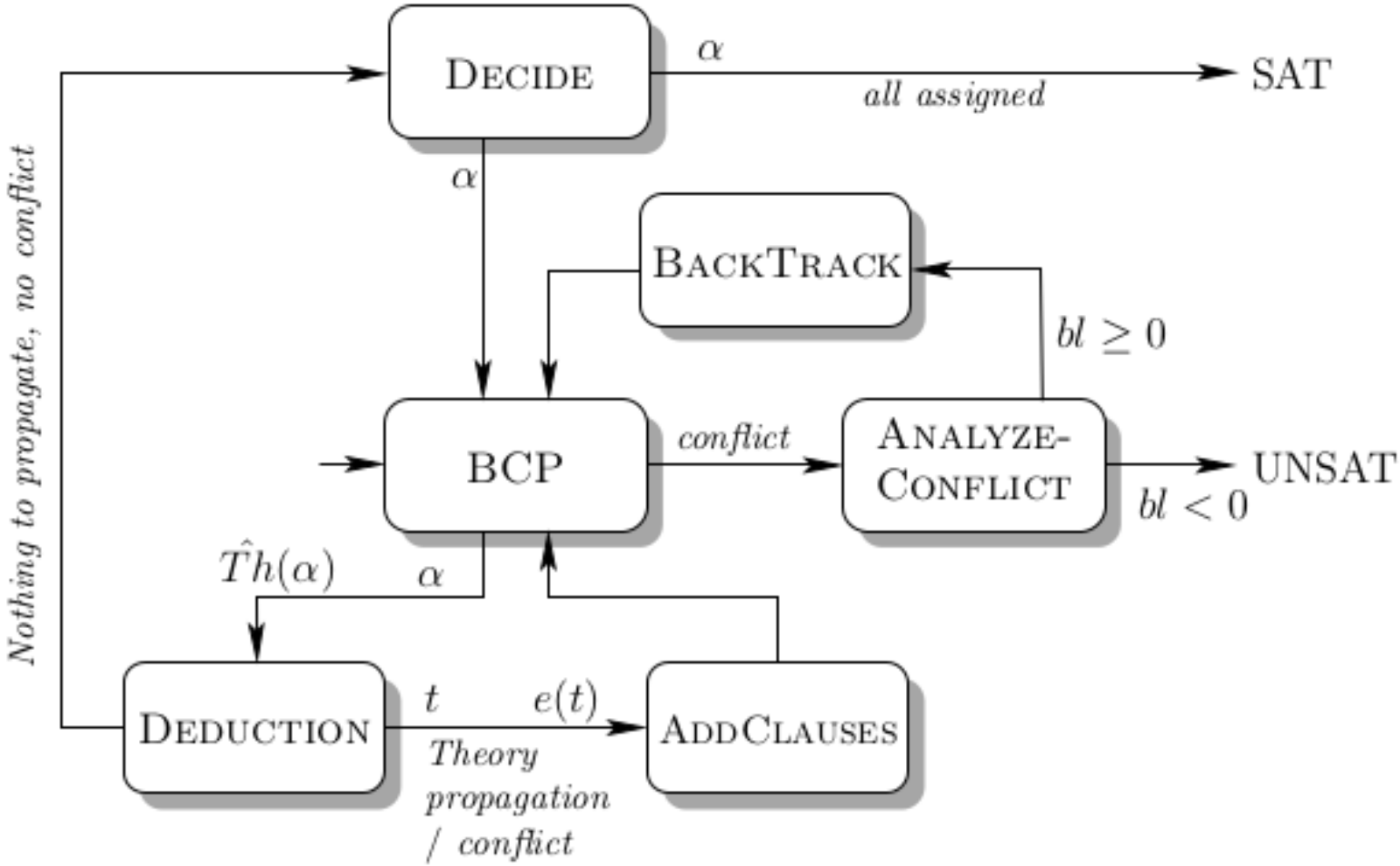
Lazy CDCL

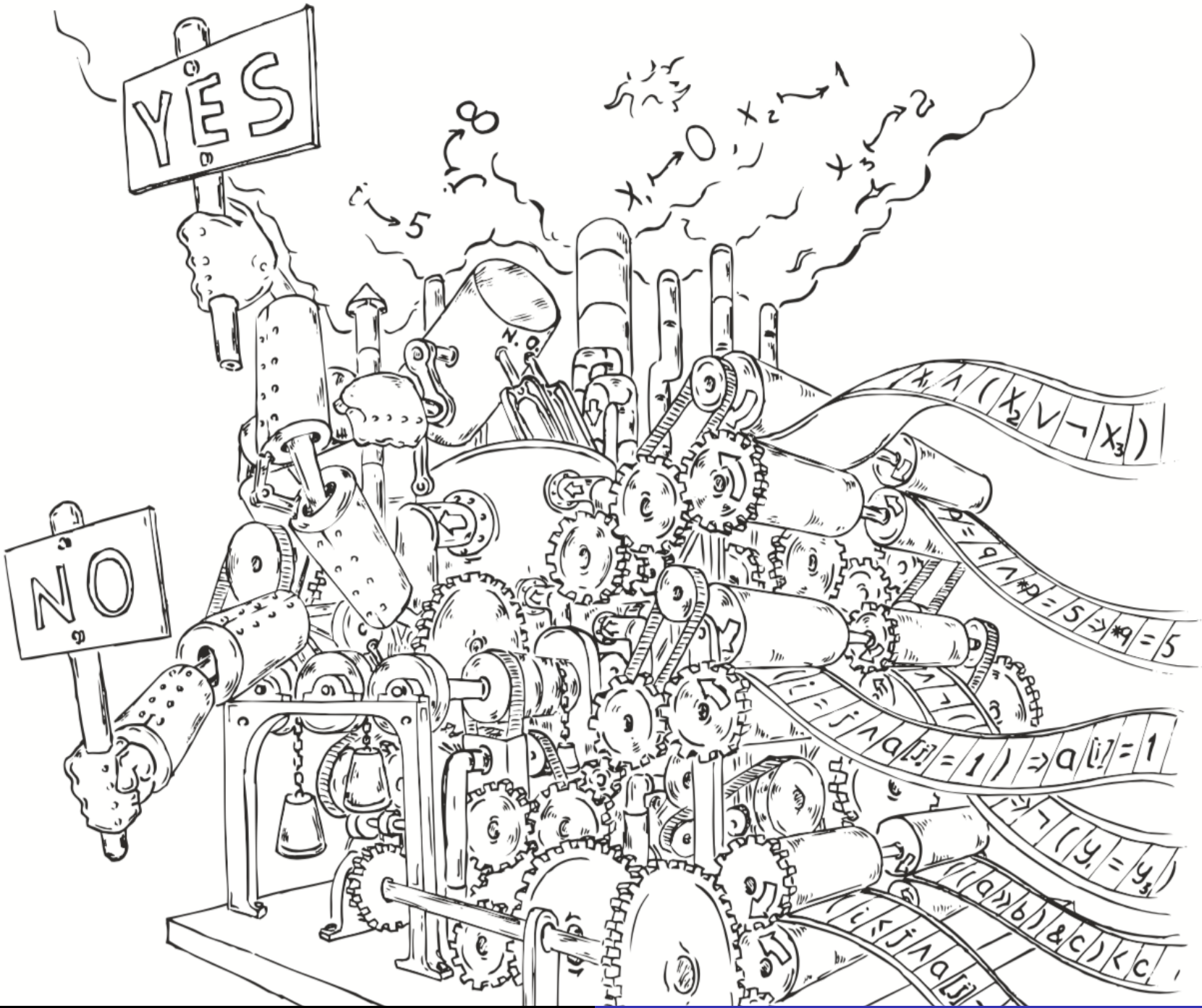
```
1.  function LAZY-CDCL
2.      ADDCLAUSES(cnf(e(φ)));
3.      while (TRUE) do
4.          while (BCP() = "conflict") do
5.              backtrack-level := ANALYZE-CONFLICT();
6.              if backtrack-level < 0 then return "Unsatisfiable";
7.              else BackTrack(backtrack-level);
8.          if ¬DECIDE() then                          ▷ Full assignment
9.              ⟨t, res⟩:=DEDUCTION(T̂h(α));            ▷ α is the assignment
10.             if res="Satisfiable" then return "Satisfiable";
11.             ADDCLAUSES(e(t));
```

DPLL(T)

```
1.  function DPLL(T)
2.      AddClauses(cnf(e(φ)));
3.      while (TRUE) do
4.          repeat
5.              while (BCP() = "conflict") do
6.                  backtrack-level := Analyze-Conflict();
7.                  if backtrack-level < 0 then return "Unsatisfiable";
8.                  else BackTrack(backtrack-level);
9.              ⟨t, res⟩:=Deduction(T̂h(α));
10.             AddClauses(e(t));
11.         until t ≡ TRUE;
12.         if α is a full assignment then return "Satisfiable";
13.         Decide();
```

DPLL(T)


Decide
α
SAT
all assigned
α
BackTrack
$bl \geq 0$
BCP
conflict
Analyze-Conflict
UNSAT
$bl < 0$
$\hat{Th}(\alpha)$
α
Deduction
t
$e(t)$
AddClauses
Theory propagation / conflict
Nothing to propagate, no conflict


YES
NO