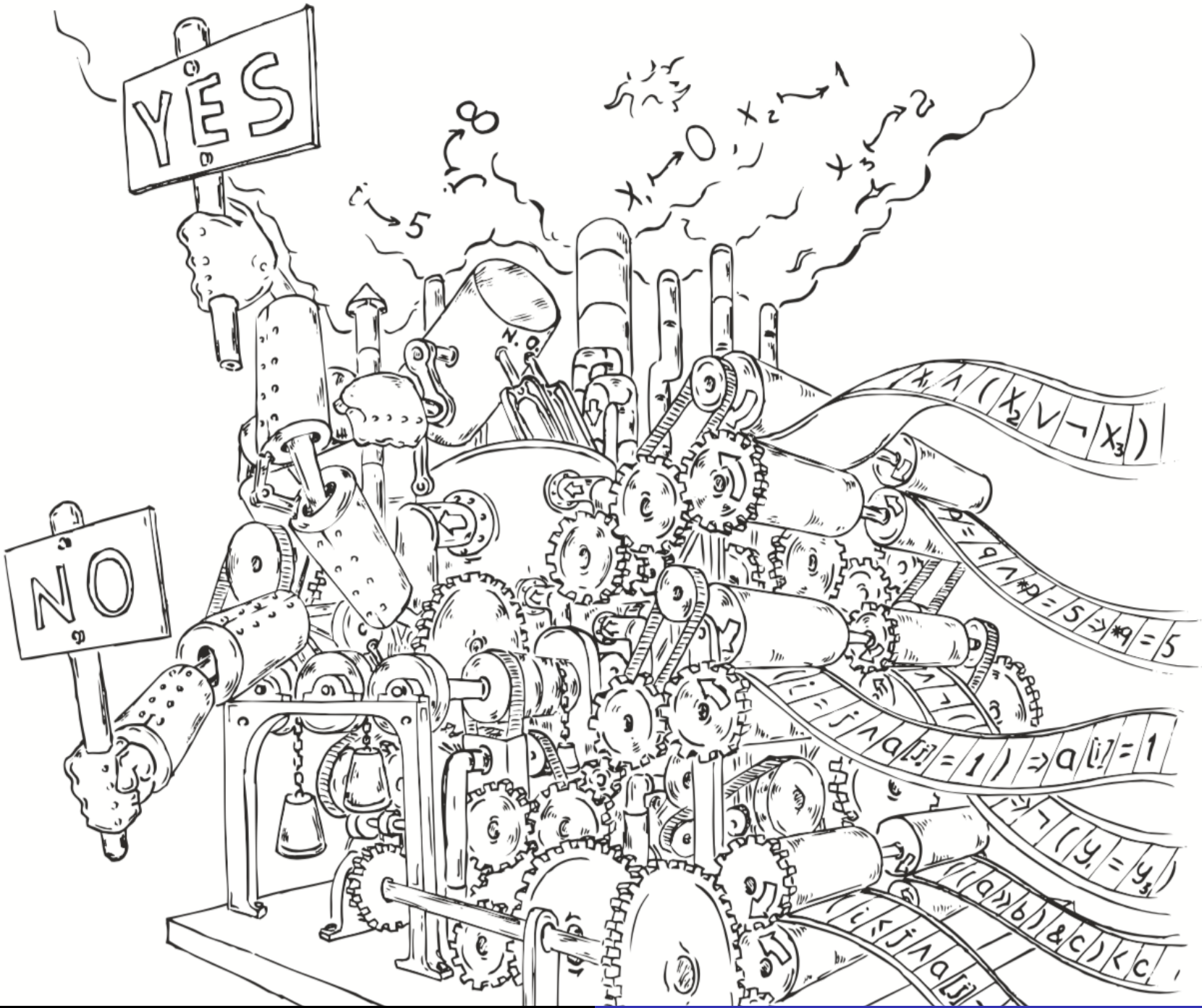

YES
NO

SAT/SMT solvers
1. Introduction

Roman Kholin

Lomonosov Moscow State University

Moscow, 2023

Outline

- What are SAT and SMT
- How they are made
- Where to use they

Edmund Clarke:

> A key technology of the 21st century

Donald Knuth

> Evidently a killer app, because it is key to the solution of so mane other problems

Examples

You are the chief of protocol and you make a diner for an ambassadors. The prince wants to invite either an ambassador from Peru or not to invite ambassador from Qatar. The queen wants to see the ambassadors of Qatar or Romania. The king does not want to see ambassadors from Romania or Peru. Whom to invite for dinner?

Examples

- The set of natural numbers was divided into subsets. Does any contain Pythagorean triple?
- Let's the number of subsets is three and original set is $\{1, \ldots, N\}$. The minimal N such that it can't be diveded is 7825
- There is no answer for number of subsets 3. It's could be shown that $\{1, \ldots, 10^7\}$ could be diveded

Examples

```
if(!a && !b) h();
else
    if(!a) g();
    else f();
```

```
if(a) f();
else
    if(b) g();
    else h();
```

Examples

```
if(!a && !b) h();
else
    if(!a) g();
    else f();
```

```
if(a) f();
else
    if(b) g();
    else h();
```

$$(\texttt{if } x \texttt{ then } y \texttt{ else } z) \equiv (x \wedge y) \vee (\neg x \wedge z)$$

Examples

```
if(!a && !b) h();
else
    if(!a) g();
    else f();
```

```
if(a) f();
else
    if(b) g();
    else h();
```

(if x then y else z) $\equiv$ $(x \wedge y) \vee (\neg x \wedge z)$

```
if ¬a ∧ ¬b then h
else
   if ¬a then g
   else f
```

```
if a then f
else
   if b then g
   else h
```

Examples

```
if(!a && !b) h();
else
    if(!a) g();
    else f();
```

```
if(a) f();
else
    if(b) g();
    else h();
```

$$(\texttt{if } x \texttt{ then } y \texttt{ else } z) \quad \equiv \quad (x \wedge y) \vee (\neg x \wedge z)$$

```
if ¬a ∧ ¬b then h
else
   if ¬a then g
   else f
```

```
if a then f
else
   if b then g
   else h
```

$$\begin{aligned} & (\neg a \wedge \neg b) \wedge h \;\vee\; \neg(\neg a \wedge \neg b) \wedge (\neg a \wedge g \;\vee\; a \wedge f) \\ \iff \quad & a \wedge f \;\vee\; \neg a \wedge (b \wedge g \;\vee\; \neg b \wedge h)\,. \end{aligned}$$

Definitions

Propositional formula

Given a set V, brackets, $\vee$, $\wedge$, $\rightarrow$, $\neg$, let define an formulas as:

- all variables V are formula.
- if A is formula, then $\neg(A)$ - fromula
- if A, B are formulas, then $(A) \vee (B)$, $(A) \wedge (B)$, $(A) \rightarrow (B)$ is formula.

Assignment

Given a formula φ, an $assignment$ of φ from a domain D is a function mapping φ 's variables to elements of D. An assignment to φ is full if all of φ 's variables are assigned, and partial otherwise.

Definitions

Satisfiability, Validity, and Contradiction

- A formula is $satisfiable$ if there exists an assignment of its variables under which the formula evaluates to true.
- A formula is a $contradiction$ if it is not $satisfiable$.
- A formula is $valid$ (also called a $tautology$) if it evaluates to true under all assignments.

The decision problem for formulas

The decision problem for a given formula φ is to determine whether φ is valid.

Soundness of a procedure

A procedure for a decision problem is sound if, when it returns $Valid$, the input formula is $valid$.

Definitions

Completeness of a procedure

A procedure for a decision problem is $complete$ if

- it always terminates, and
- it returns $Valid$ when the input formula is valid

Decision procedure

A procedure is called a decision procedure for T if it is sound and complete with respect to every formula of T

Decidability of a theory

A theory is decidable if and only if there is a decision procedure for it

Definitions

Negation Normal Form (NNF)

A formula is in negation normal form (NNF) if negation is allowed only over variables, and $\vee$, $\wedge$, $\neg$ are the only allowed Boolean connectives

Literal

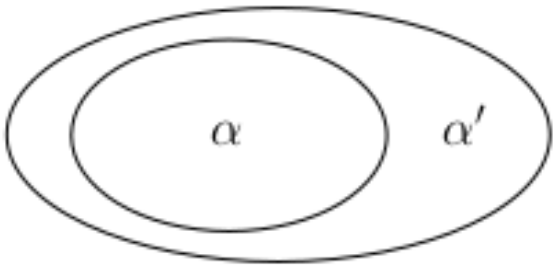
A *literal* is either an variable or its negation. We say that a literal is negative if it is a negated variable, and positive otherwise

State of a literal under an assignment

A positive literal is satisfied if its variable is assigned TRUE. Similarly, a negative literal is satisfied if its variable is assigned FALSE

Monotonicity of NNF

Let φ be a formula in NNF and let α be an assignment of its variables. Let the positive set of α with respect to φ, denoted $pos(\alpha, \varphi)$, be the literals that are satisfied by α. For every assignment α' to φ's variables such that $pos(\alpha, \varphi) \subseteq pos(\alpha', \varphi), \alpha \models \varphi \rightarrow \alpha' \models \varphi$



$$\alpha \models \varphi \implies \alpha' \models \varphi$$

Definitions

Disjunctive Normal Form (DNF)

A formula is in disjunctive normal form if it is a disjunction of conjunctions of literals, i.e., a formula of the form $\bigvee_i \bigwedge_j l_{ij}$, where l_{ij} is the j-th literal in the i-th term (a term is a conjunction of literals)

Conjunctive Normal Form (CNF)

A formula is in conjunctive normal form if it is a conjunction of disjunctions of literals, i.e., it has the form $\bigwedge_i \bigvee jl_{ij}$, where l_{ij} is the j-th literal in the i-th clause (a clause is a disjunction of literals)

Tseitin Transformation

$$\phi := ((p \vee q) \wedge r) \rightarrow (\neg s)$$

Equisatisfiability

Two formulas are equisatisfiable if they are both satisfiable or they are both unsatisfiable

SAT

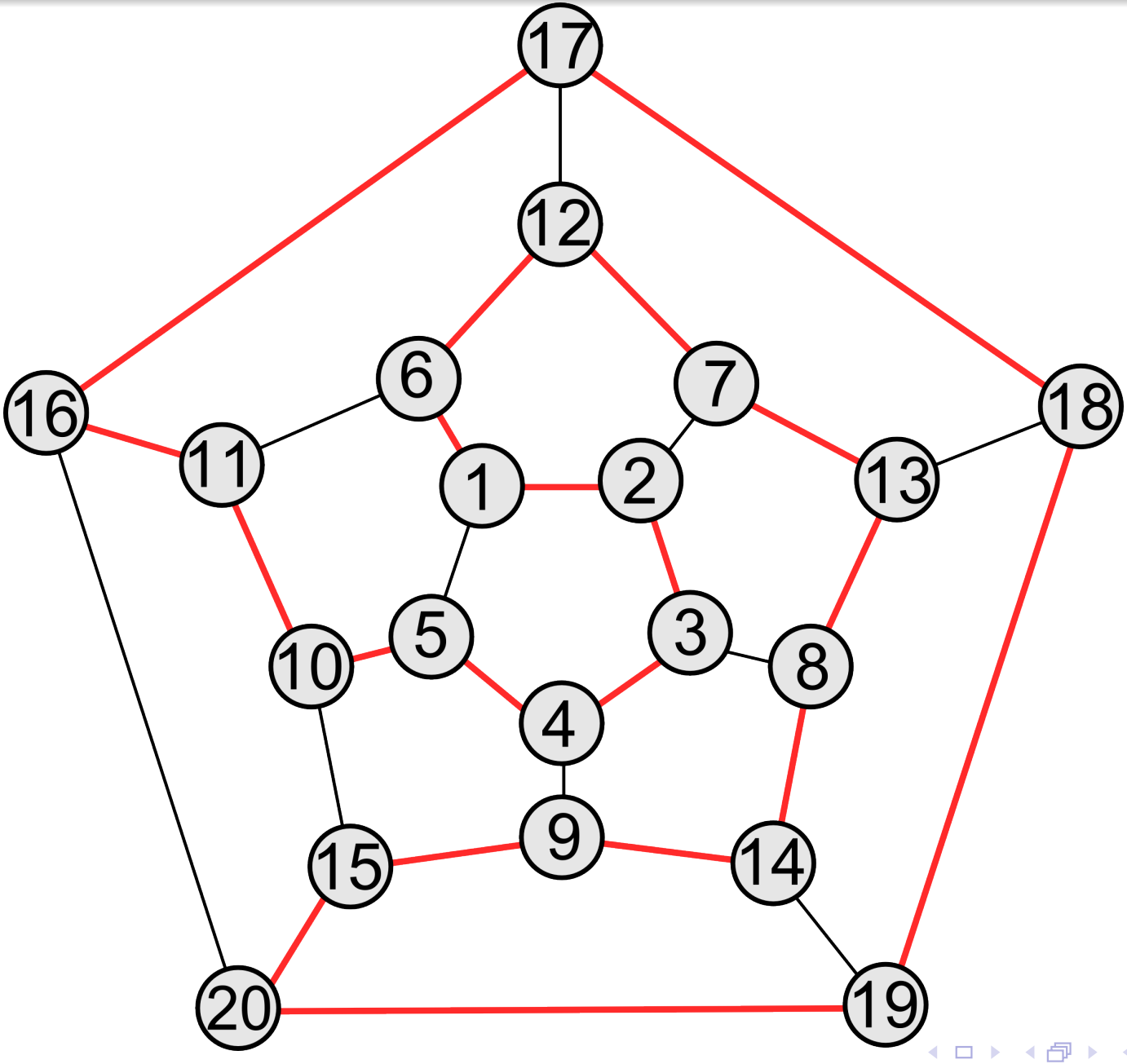
Given a propositional formula f over n propositional variables $V = x_1, \ldots, x_n$, is there are an assignment $\alpha : V \to \{0, 1\}$ with $\alpha(f) = 1$?

SAT

Given a propositional formula f over n propositional variables $V = x_1, \ldots, x_n$, is there are an assignment $\alpha : V \to \{0, 1\}$ with $\alpha(f) = 1$?

- SAT belongs to NP (for n > 2)
- SAT is complete for NP

Examples



Examples

- $AtLeastOne(x_1, \ldots, x_n)$
- $AtMostOne(x_1, \ldots, x_n)$
- Lazy approach

- $(x = 4) \wedge ((y = 7) \vee (x = y))$
- $(x + y = 3) \wedge (y - z = 7) \wedge (z * 2 = 4)$
- $(lenght(s) = 3) \wedge (s[0] =' a') \wedge (s[1] =' b') \wedge (s[2] =' c')$

Definitions

First-order logic

- Variables: a set of variables
- Logical symbols: the standard Boolean connectives (e.g., $\wedge, \vee, \neg$), quantifiers ($\exists$ and $\forall$), and parentheses
- Nonlogical symbols: function, predicate, and constant symbols
- Syntax: rules for constructing formulas. Formulas adhering to these rules are said to be well formed

- A domain
- An interpretation of the nonlogical symbols, in the form of a mapping from each function and predicate symbol to a function and a predicate, respectively, and an assignment of a domain element to each of the constant symbols
- An assignment of a domain element to each of the free (unquantified) variables

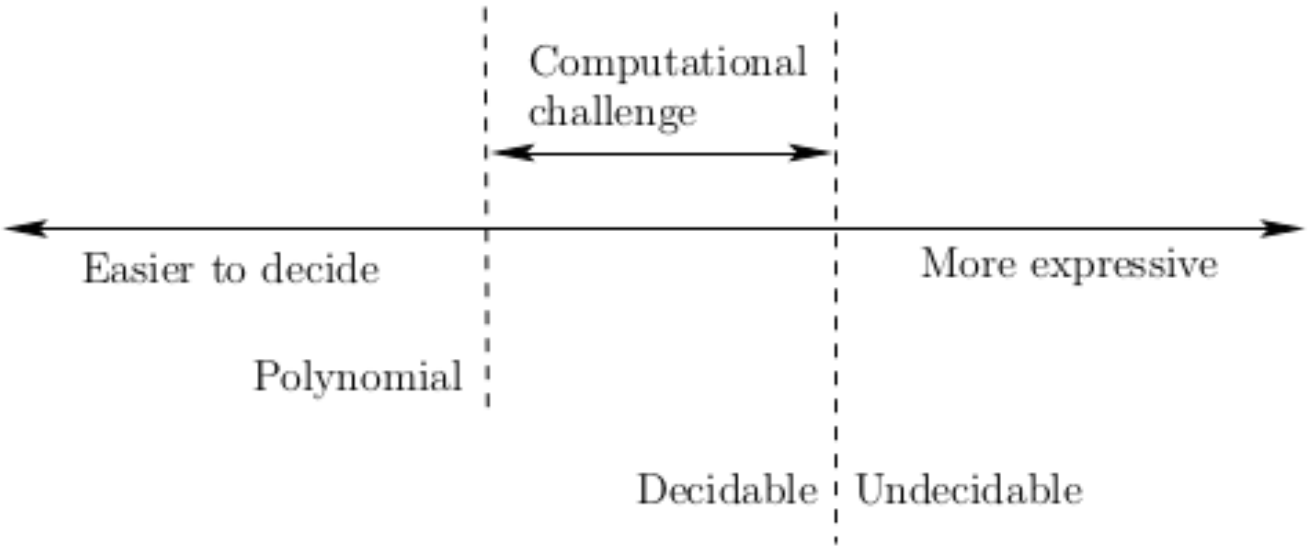
Definitions

Satisfiability

The formula φ is satisfiable if and only if there exists a structure under which the formula is true.

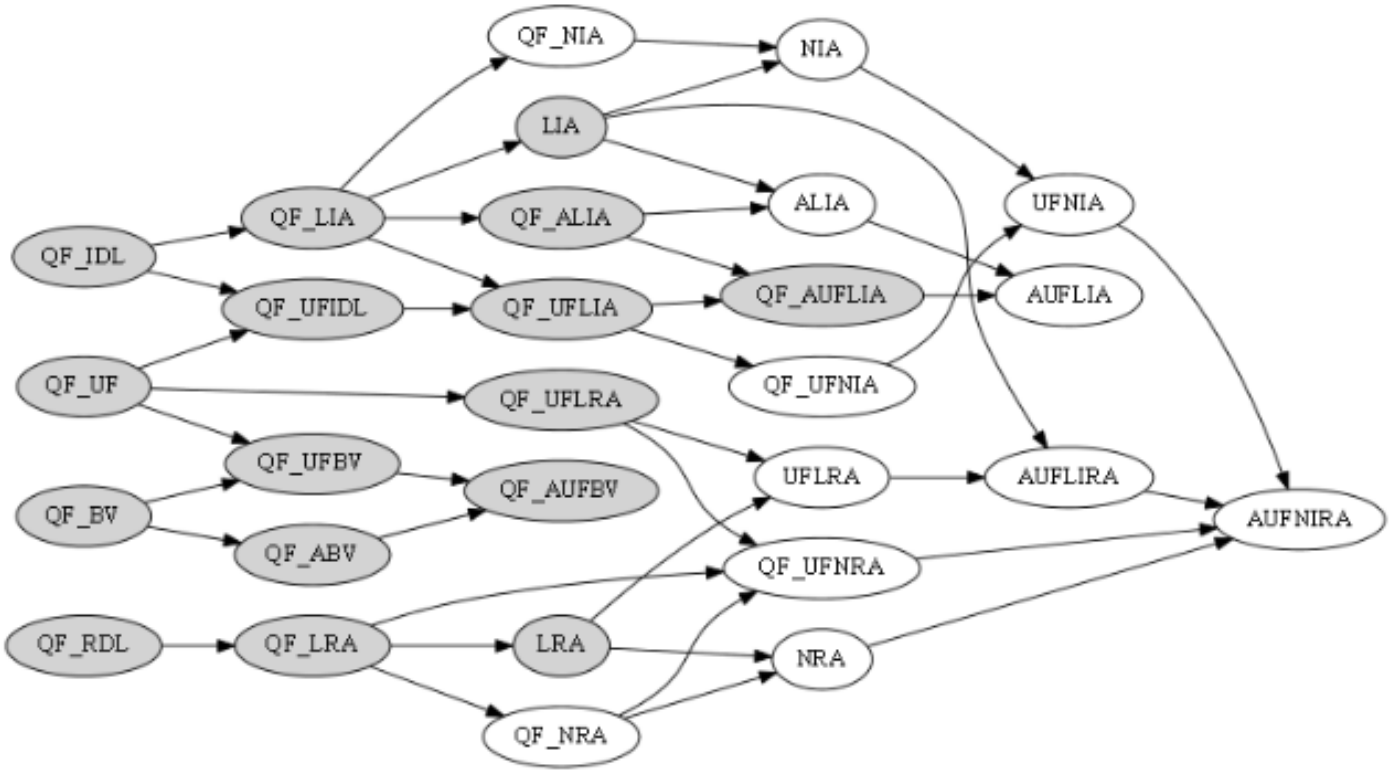
Expressiveness

Theory A is more expressive than theory B if every language that can be defined by a B-formula can also be defined by an A-formula, and there exists at least one language definable by an A-formula that cannot be defined by a B-formula. We denote the fact that theory B is less expressive than theory A by $B \prec A$.

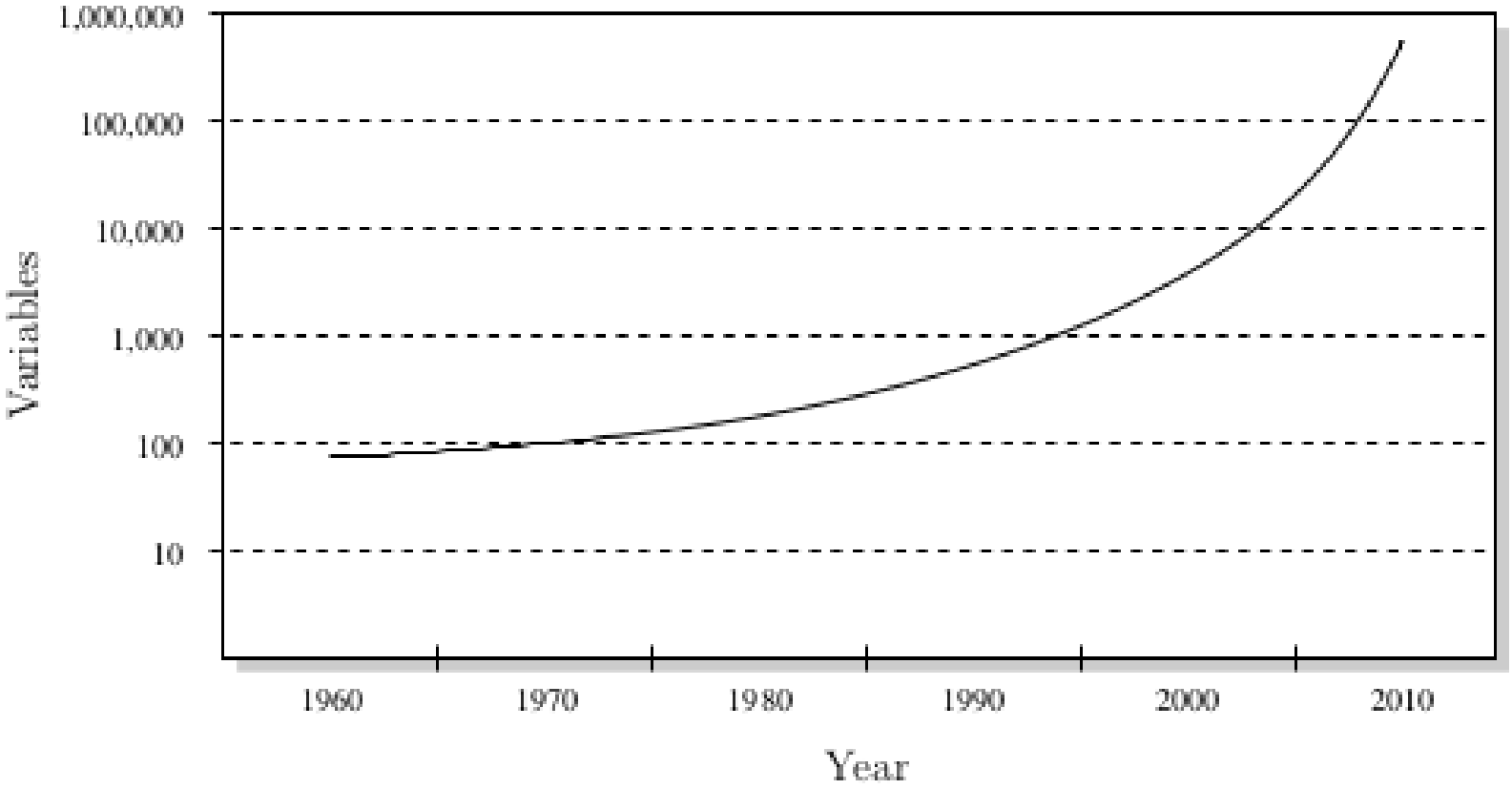


logics

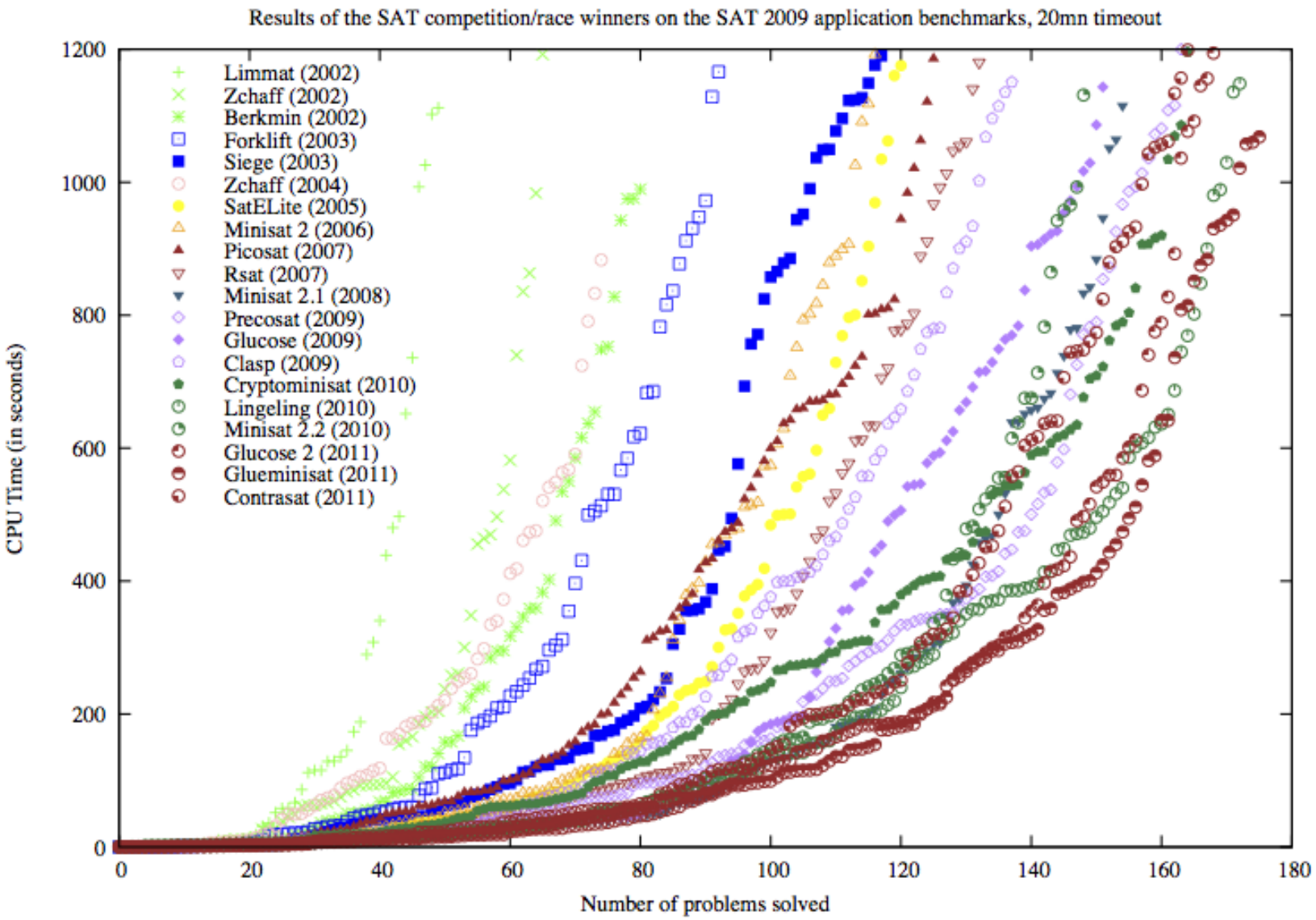
- Propositional Logic
- Equalities and Uninterpreted Functions
- Linear Arithmetic
- Bit Vectors
- Arrays
- Pointer Logic



Solvers


1,000,000
100,000
10,000
1,000
100
10
Variables
1960
1970
1980
1990
2000
2010
Year

Solvers


Results of the SAT competition/race winners on the SAT 2009 application benchmarks, 20mn timeout
CPU Time (in seconds)
Number of problems solved
0
200
400
600
800
1000
1200
0
20
40
60
80
100
120
140
160
180
Limmat (2002)
Zchaff (2002)
Berkmin (2002)
Forklift (2003)
Siege (2003)
Zchaff (2004)
SatELite (2005)
Minisat 2 (2006)
Picosat (2007)
Rsat (2007)
Minisat 2.1 (2008)
Precosat (2009)
Glucose (2009)
Clasp (2009)
Cryptominisat (2010)
Lingeling (2010)
Minisat 2.2 (2010)
Glucose 2 (2011)
Glueminisat (2011)
Contrasat (2011)

Listing 3. Integer Example in SMT-LIB

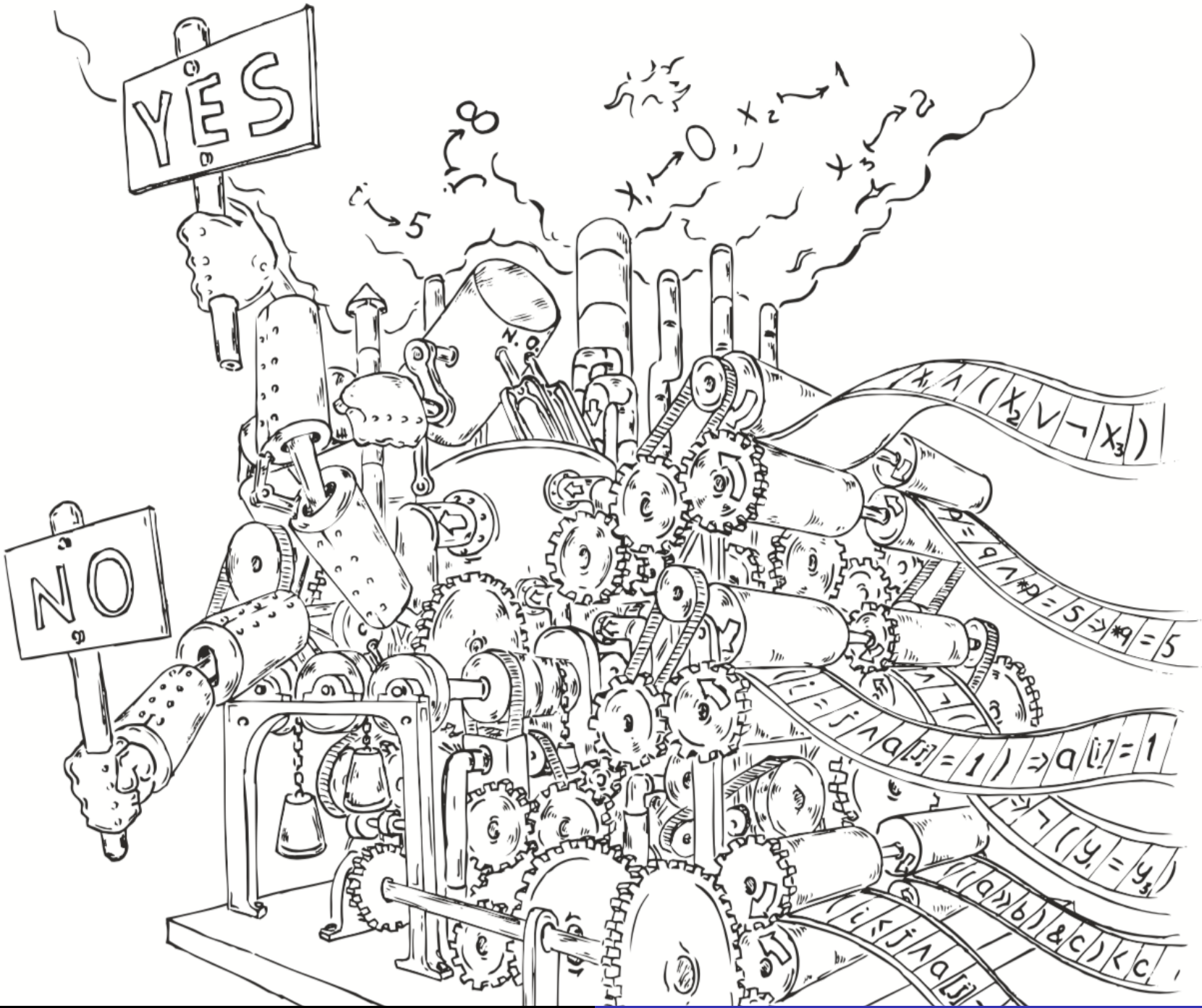
```
(set-logic QF_LIA)
(declare-fun x () Int)
(declare-fun y () Int)
(declare-fun z () Int)
(assert (= (+ (* 6 x) (* 2 y) (* 12 z)) 30))
(assert (= (+ (* 3 x) (* 6 y) (* 3 z)) 12))
(check-sat)
```

DIMACS format

- $(a \vee b \vee \neg c) \wedge (\neg a \vee \neg b \vee c) \wedge (b \vee c \vee \neg d)$
- c example
  p cnf 4 3
  1 2 -3 0
  -1 -2 3 0
  2 3 -4 0

Additional materials

- Decision Procedures An Algorithmic Point of View, Daniel Kroening, Ofer Strichman
- Handbook of satisfiability, Edmund Clarke
- The art of Computer Programming, volume 4, part 6, Donald Knuth


YES
NO