

1. SAT solving

What is SAT solving

Does this formula have a solution and can you find it

Validity = tautology

Satisfiability: $\exists \vec{x}. \Phi(\vec{x})$

Equivalence of formulas: $\forall \vec{x}. \llbracket \Phi \rrbracket(\vec{x}) = \llbracket \Psi \rrbracket(\vec{x})$

Negation normal form: Negation appears only directly in front of propositional variables

$$\Phi ::= x \mid \neg x \mid \Phi \vee \Phi \mid \Phi \wedge \Phi \mid \perp \mid \top$$

Conjunctive normal form: finite conjunction of clauses (the useful one)

Disjunctive normal form: finite conjunction of co clauses

The SAT problem

Input: Prop Φ

Output: yes/no, better: satisfying assignment

The DNF SAT problem

Input: Φ in DNF

Output: yes/no, better: satisfying assignment

Algorithm: Check if clause is satisfiable.

It is if doesn't contain $x \wedge \neg x$.

Solvable in $O(n)$ (with the table)

The CNF SAT problem

Input: Φ in CNF

Output: yes/no, better: satisfying assignment

!! There is no efficient way to transform Φ to equivalent formula in CNF !!

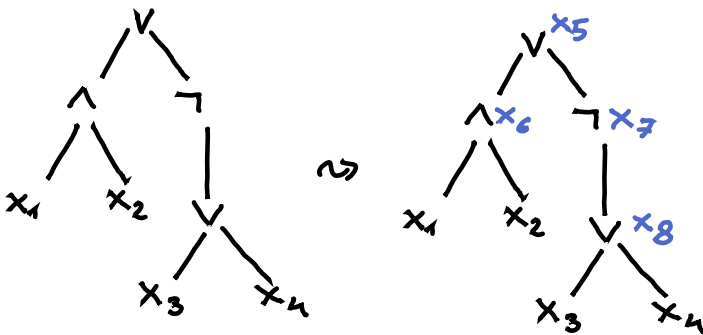
The Tseytin transformation

Maps $\Phi(x_1, \dots, x_k)$ to $\Phi^T(x_1, \dots, x_N)$ $N \geq k$ such that:

1. Φ^T is 3-CNF (each clause contains 3 literals)
2. $(x_1, \dots, x_k) \mapsto (x_1, \dots, x_N)$ is bijection between satisfying assignments
3. Φ^T is computable in $O(|\Phi|)$ time

Example:

$$\Phi \equiv (x_1 \wedge x_2) \vee \neg(x_3 \vee x_4)$$



- ① $x_6 \Leftrightarrow x_1 \wedge x_2$
- ② $x_5 \Leftrightarrow x_6 \vee x_7$
- ③ $x_7 \Leftrightarrow \neg x_8$
- ④ $x_8 \Leftrightarrow x_3 \vee x_4$

$$(A \Leftrightarrow B) \Leftrightarrow (\neg A \vee B) \wedge (\neg B \vee A)$$

$$\begin{aligned} \text{①} &= (\neg x_6 \vee (x_1 \wedge x_2)) \wedge (\neg(x_1 \wedge x_2) \vee x_6) = \\ &= (\neg x_6 \vee x_1) \wedge (\neg x_6 \vee x_2) \wedge (\neg x_1 \vee \neg x_2 \vee x_6) \\ &\vdots \end{aligned}$$

$$\Phi \equiv \text{①} \wedge \text{②} \wedge \text{③} \wedge \text{④}$$

Tseytin transformation does not give equivalent formula, but an equisatisfiable one. So bijection is not an identity.

3-colorability of a graph

$E \subseteq V \times V$ symmetric, irreflexive

$\exists \text{ col}: V \rightarrow \{R, G, B\}, \forall (u, v) \in E, \text{col}(u) \neq \text{col}(v)$

SAT problem:

variables $X_{1,1} \dots X_{n,1}$
 $X_{1,2} \dots X_{n,2}$
 $X_{1,3} \dots X_{n,3}$

① $(X_{1,1} \vee X_{1,2} \vee X_{1,3}) \wedge \dots \wedge (X_{n,1} \vee X_{n,2} \vee X_{n,3})$
every vertex is colored.

② $(\neg X_{1,1} \vee \neg X_{1,2}) \wedge (\neg X_{1,2} \vee \neg X_{1,3}) \wedge (\neg X_{1,3} \vee \neg X_{1,1})$
every vertex has at most one color

③ for $(x_i, x_j) \in E$ we include

$$(\neg X_{i,1} \vee \neg X_{j,1}) \wedge \dots \wedge (\neg X_{i,3} \vee \neg X_{j,3})$$

The DPLL algorithm

def DPLL(Φ, φ) $\varphi \dots$ assignment of truth variables
not in Φ

- Φ is empty conjunction $\mapsto$ return φ
 - Φ contains empty clause $\mapsto$ return NO
 - Φ contains unit clause $L \mapsto$ DPLL(Simplify(Φ, L), $L :: \varphi$)
- if none apply:
- choose literal L in Φ
 - DPLL(Simplify(Φ, L), $L :: \varphi$)

2. Predicate logic

$$\Phi ::= \overset{\text{relation or predicate}}{p}(t, \dots, t) \mid t = t \mid \neg \Phi \mid \Phi \wedge \Phi \mid \Phi \vee \Phi \mid \forall x. \Phi \mid \exists x. \Phi$$
$$t ::= x \mid f(x, \dots, x)$$

Sentence: formula with no free variables

Structure $\mathcal{M} = (M, (p^{\mathcal{M}})_{p \in \mathcal{P}}, (f^{\mathcal{M}})_{f \in \mathcal{F}})$

- M set
- For every relation $p \in \mathcal{P}$ we have $p^{\mathcal{M}}: M^n \rightarrow \{t, f\}$, that evaluates relation between n elements.
- For every function symbol f we have $f^{\mathcal{M}}: M^n \rightarrow M$, so f defines function $f^{\mathcal{M}}$ on a set M

Coincidence lemma: $\mathcal{M}$ and $\mathcal{M}'$ coincide on free variables of Φ , then $\mathcal{M} \models \Phi \Leftrightarrow \mathcal{M}' \models \Phi$

Satisfiability: Φ is satisfiable if there exists $\mathcal{M}$ such that $\mathcal{M} \models \Phi$

Gödel completeness theorem: If Φ is valid it has a natural deduction proof

Soundness theorem: If Φ has a proof in natural deduction, then Φ is valid.

Church/Turing's theorem

The validity problem is undecidable (there is no algorithm).

It follows: Satisfiability is undecidable

semidecidability: A problem is semidecidable, if there exists an algorithm, that returns yes if ϕ is satisfiable and runs infinitely otherwise.
A problem is co-semidecidable if no and yes are inverted.

Finite satisfiability: Can we satisfy ϕ with finite structure $\mathcal{M}$, such that $\mathcal{M} \models \phi$
Finitary satisfiability is semidecidable.

Bounded satisfiability: Does there exist a structure $\mathcal{M}$ such that $|\mathcal{M}| \leq N$ and $\mathcal{M} \models \phi$

Reducing bounded satisfiability to SAT problem

We have ϕ . We are looking for $\mathcal{M}$ with we get from $\mathcal{M}$

We add $c_1, \dots, c_N$ constants, that represent elements of $\mathcal{M}$. And for each predicate P of arity k

We add $P_{i_1, \dots, i_k}$; $i_j \in \{1, \dots, N\}$. Each one will assert $P(c_{i_1}, \dots, c_{i_k}) = \text{true}$

We define ϕ^* recursively:

$$(\exists x. \psi)^* := (\psi[c_1/x])^* \vee \dots \vee (\psi[c_N/x])^*$$

$$(\forall x. \psi)^* := (\psi[c_1/x])^* \wedge \dots \wedge (\psi[c_N/x])^*$$

$$(\psi_1 \vee \psi_2)^* := \psi_1^* \vee \psi_2^*$$

$$(P(c_{i_1}, \dots, c_{i_k}))^* := P_{i_1, \dots, i_k}$$

If we can satisfy ϕ^* , then we get $P_{i_1, \dots, i_k}$.

From there we can create a structure

$$\mathcal{M} = [N] \quad \mathcal{P} = \{ (i_1, \dots, i_k) ; P_{i_1, \dots, i_k} = \text{true} \}$$

3. SMT solving (modulo a theory T)

The SMT problem:

SAT	Propositional	Not expressive, practical
FOL	- -	not practical, expressive
SMT	- -	sweet spot

Basic setting $T =$
Function symbols + relation symbols $\swarrow$ predicate
without quantifiers

Is formula ϕ satisfiable modulo a theory T ?
= Does there exist $\mathcal{M} \in \text{Mod } T$ and environment ρ ,
such that $\mathcal{M} \models_{\rho} \phi$

Solving EUF for closure by congruence
closure $\nearrow$ equality between uninterpreted functions

Congruence axiom $\forall x, y. x = y \Rightarrow f(x) = f(y)$

1. We flatten EUF formula

$$x = f(y, g(z)) \leadsto t = g(z) \wedge x = f(y, t)$$

2. We create classes, of variables.

When $x = y$ we merge class x and y

3. The formula is unsatisfiable if for $u \neq v$ u and v
are in the same class

Reducing Array to EUF

read: $A \rightarrow I \rightarrow V$

write: $A \rightarrow I \rightarrow V \rightarrow A$

$$i=j \Rightarrow \text{read}(\text{write}(a, i, v), j) = v$$

$$i \neq j \Rightarrow \text{read}(\text{write}(a, i, v), j) = \text{read}(a, j)$$

that way we can flatten formulas. For example

$$i \neq j \wedge u = \text{read}(\text{write}(a, i, v), j) \wedge v = \text{read}(a, j) \\ \wedge u \neq v$$

We reduce this using the above rules

$$i \neq j \wedge u = f(\delta) \wedge v = \text{read}(a, j) \wedge u \neq v$$

$$i \neq j \wedge u = \text{if } i \neq j \text{ then } \text{read}(a, j) \text{ else } v \wedge v = \text{read}(a, j) \\ \wedge u \neq v$$

$$i \neq j \wedge u = \text{read}(a, j) \wedge v = \text{read}(a, j) \wedge u \neq v$$

$$|i|j|u|v| \rightsquigarrow |i|j|u|v| \neq$$

Eager vs Lazy SMT

Eager: Translate to propositional SAT

Lazy: Modify modern SAT solvers.

Use the clause decision procedure

For EUF lazy: (Ackerman's reduction)

1. Flatten terms

2. replace function instances $f(u) \rightsquigarrow f_u$

3. add constraints $u=v \rightsquigarrow f_u = f_v$

4. Solve

Example

Flattend $x = f(y) \wedge y = f(x) \wedge x \neq y$

2., 3. $x = f_y \wedge y = f_x \wedge (x = y \Rightarrow f_x = f_y) \wedge x \neq y$

$|x| |y| |f_x| |f_y| \rightsquigarrow |x f_y| |y f_x|$

Bit blasting: Replace every variable x with

k propositional variables $x_1, \dots, x_k$. ($2^k \geq \text{variables}$)

Replace every equality $x=y$ with

$(x_1 \Leftrightarrow y_1) \wedge \dots \wedge (x_k \Leftrightarrow y_k)$ and $x \neq y$ with

$\neg((x_1 \Leftrightarrow y_1) \wedge \dots \wedge (x_k \Leftrightarrow y_k))$

Submit to SAT solver.

This is satisfiable $\Leftrightarrow$ original formula is satisfiable

$(x_1 \Leftrightarrow f_{y_1}) \wedge (x_2 \Leftrightarrow f_{y_2}) \wedge (y_1 \Leftrightarrow f_{x_1}) \dots$

4. Linear Time Temporal logic

What will I talk about

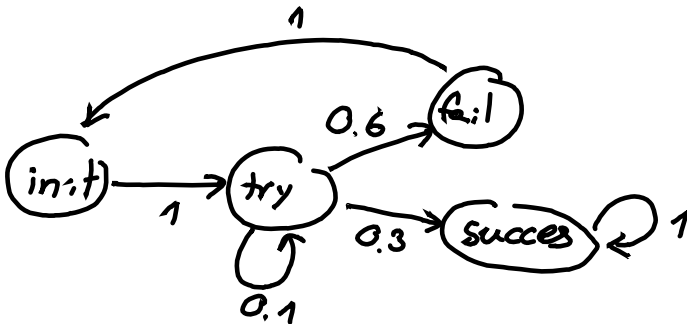
- DTMC - discrete time Markov chain

↑
two processes can
be trying at the same time

DTMCS is transition system (S, s_0, P, L)

- S ... states
- s_0 ... initial state
- $L: S \rightarrow \mathcal{P}(\text{Atoms})$... labeling function
- $P: S \times S \rightarrow [0, 1]$ transition probability matrix

$$\forall s \in S, \sum_{t \in S} P(s, t) = 1$$



A run thru DTMC $\mathcal{K}$ is a sequence

$$\pi: s_0 \xrightarrow{p_1} s_1 \xrightarrow{p_2} s_2 \xrightarrow{p_3} s_3 \xrightarrow{\dots}$$

$$\forall n \in \mathbb{N}_0. \mathcal{P}(s_n, s_{n+1}) > 0$$

What is a probability, that a randomly generated run π satisfies ϕ

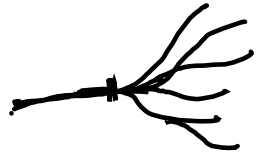
$$\phi ::= p \mid \neg \phi \mid \phi \wedge \phi \mid X\phi \mid F\phi \mid G\phi \mid \phi \cup \phi$$

$$\mathcal{P}(\pi) = \prod_{n=0}^{\infty} p_n$$

Cylinder set: $\langle w \rangle = \{ \pi \in \text{Runs} ; \pi \text{ extends } w \}$

where w is a finite run

$$\mathcal{P}(\langle w \rangle) = \prod_{i=1}^n p_i$$



We are interested in

$$\mathcal{P}[\phi] = \mathcal{P}(\{ \pi \in \text{Runs} ; \pi \models \phi \})$$

- Convert Φ to NBA
- Convert NBA to deterministic Rabin automata
- Construct a product DTMC combining Rabin automata and original DTMC
- Get strongly connected components. Find the end component that satisfies Φ and calculate the probability of getting there.

