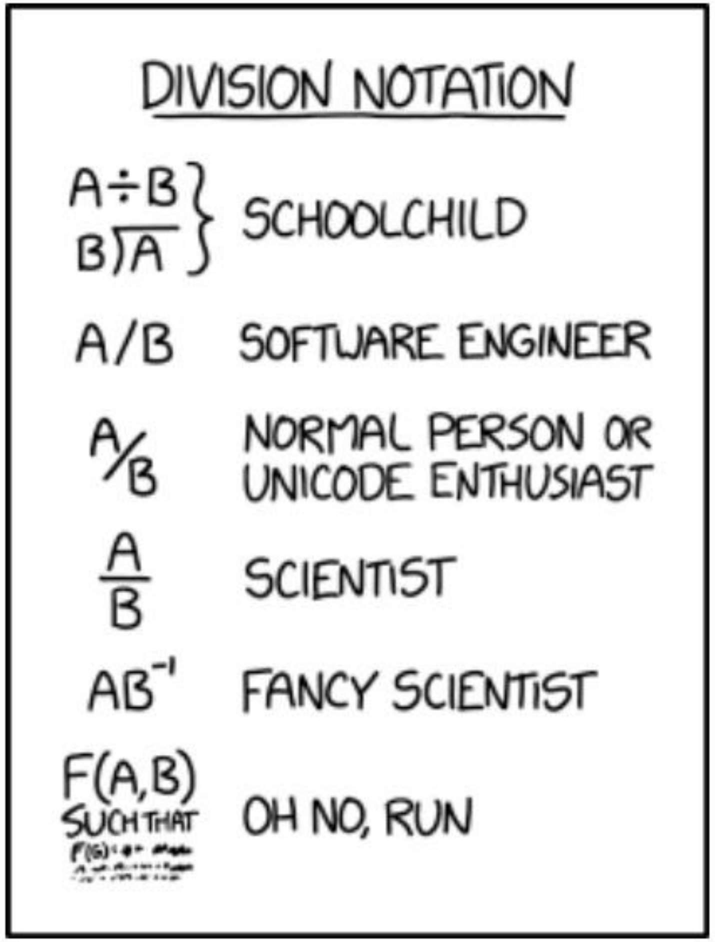
Logic and Computability

Training for the Exam

Bettina Könighofer
bettina.koenighofer@iaik.tugraz.at

Stefan Pranger
stefan.pranger@iaik.tugraz.at

Division Notation



https://xkcd.com/2687/

Example 1

Model the following sentences with predicate logic, as detailed as possible. Clearly indicate the intended meaning of all function, predicate, and constant symbols that you use.

(a) Every integer that is greater or equal than one is also greater or equal than two.
(b) For any two integers, their sum is smaller than their product

Example 1

Model the following sentences with predicate logic, as detailed as possible. Clearly indicate the intended meaning of all function, predicate, and constant symbols that you use.

(a) Every integer that is greater or equal than one is also greater or equal than two.
(b) For any two integers, their sum is smaller than their product

$x < y$ ... *x is smaller than y*
$x \geq y$ ... *x is greater or equal than y*
$x + y$... *returns the sum of x and y*
$x * y$... *returns the product of x and y*

$$A = \mathbb{Z}$$

(a) $\forall x\, (x \geq 1 \rightarrow x \geq 2)$
(b) $\forall x \forall y\, (x + y < x * y)$

Example 2

The syntax of predicate logic is defined via 2 types of sorts: *terms* and *formulas*.

- What are terms and what are formulas?
- Give the definitions and examples for both.

Example 2

Syntax of Predicate Logic

Two types of *sorts:*

- **Terms**
  - Refer to **objects** of the **domain**:
    - *constants* represent individual objects, e.g., Alice, Bob, 5, 3, 3.45…
    - *variables* like x, y represent objects
    - *functions symbols* refer to objects like $x \cdot y, f(x)$ …
- **Formulas**
  - Have a truth value
  - *prop. variables* like x, y
    predicates like $P(x, y), x = y$

$$\overbrace{\underbrace{P}_{\textbf{Predicate}}\,(\,\underbrace{x}_{\textbf{Term}}\,,\ \underbrace{f(y)}_{\textbf{Term}}\,)}^{\textbf{Formula}}$$

Example 3

Explain the algorithm of how to decide the equivalence of combinational circuits via the reduction to satisfiability.

Example 3

Explain the algorithm of how to decide the equivalence of combinational circuits via the reduction to satisfiability.

Algorithm - Circuit Equivalence based on SAT

1. Encode C_1 and C_2 into two formulas φ_1 and φ_2
2. Compute the Conjunctive Normal Form (CNF) of $\varphi_1 \oplus \varphi_2$
   - Use Tseitin Encoding
3. Give CNF($\varphi_1 \oplus \varphi_2$) to a **SAT solver**
4. C_1 and C_2 are **equivalent** if and only if $\varphi_1 \oplus \varphi_2$ is **UNSAT**

Example 4

Apply Tseitin's encoding to the following formula:

$$\varphi = (p \vee \neg q) \vee (\neg p \wedge \neg r).$$

Use the following *Tseitin-rewriting rules*:

$$\begin{aligned}
\chi \leftrightarrow (\varphi \vee \psi) \quad &\Leftrightarrow (\neg\varphi \vee \chi) \wedge (\neg\psi \vee \chi) \wedge (\neg\chi \vee \varphi \vee \psi) \\
\chi \leftrightarrow (\varphi \wedge \psi) \quad &\Leftrightarrow (\neg\chi \vee \varphi) \wedge (\neg\chi \vee \psi) \wedge (\neg\varphi \vee \neg\psi \vee \chi) \\
\chi \leftrightarrow \neg\varphi \quad &\Leftrightarrow (\neg\chi \vee \neg\varphi) \wedge (\chi \vee \varphi)
\end{aligned}$$

For each variable you introduce, clearly indicate which subformula of φ it represents.

Example 4

$$\varphi = \underbrace{\underbrace{(p \vee \underbrace{\neg q}_{x_1})}_{x_4} \vee \underbrace{(\underbrace{\neg p}_{x_2} \wedge \underbrace{\neg r}_{x_3})}_{x_5}}_{x_\varphi}.$$

$$\begin{gathered}
CNF(\varphi) = x_\varphi \wedge \\
(\neg x_4 \vee x_\varphi) \wedge (\neg x_5 \vee x_\varphi) \wedge (x_\varphi \vee x_4 \vee x_5) \wedge \\
(\neg p \vee x_4) \wedge (\neg x_1 \vee x_4) \wedge (\neg x_4 \vee p \vee x_1) \wedge \\
(\neg x_5 \vee x_2) \wedge (\neg x_5 \vee x_3) \wedge (\neg x_2 \vee \neg x_3 \vee x_5) \wedge \\
(\neg x_1 \vee \neg q) \wedge (x_1 \vee q) \wedge (\neg x_2 \vee \neg p) \wedge (x_2 \vee p) \wedge \\
(\neg x_3 \vee \neg r) \wedge (x_3 \vee r)
\end{gathered}$$

Example 5

For each of the following sequents, either provide a natural deduction proof, or a counterexample
that proves the sequent invalid.

- For proofs, clearly indicate which rule, and what assumptions/premises/ intermediate results you are using in each step. Also clearly indicate the scope of any boxes you use.
- For counterexamples, give a complete model. Show that the model satisfies the premise(s) of the sequent in question, but does not satisfy the respective conclusion.

$$\vdash (p \land q) \rightarrow \neg(\neg p \lor \neg q)$$

Example 5

$$\vdash (p \wedge q) \rightarrow \neg(\neg p \vee \neg q)$$

1.	$p \wedge q$	ass.
2.	p	$\wedge$e1 1
3.	q	$\wedge$e2 1
4.	$\neg p \vee \neg q$	ass.
5.	$\neg p$	ass.
6.	$\bot$	$\neg$e 2,5
7.	$\neg q$	ass.
8.	$\bot$	$\neg$e 3,7
9.	$\bot$	$\vee$e 4,5–6,7–8
10.	$\neg(\neg p \vee \neg q)$	$\neg$i 4–9
11.	$(p \wedge q) \rightarrow \neg(\neg p \vee \neg q)$	$\rightarrow$i 1–10

Example 6

For each of the following sequents, either provide a natural deduction proof, or a counterexample
that proves the sequent invalid.

- For proofs, clearly indicate which rule, and what assumptions/premises/ intermediate results you are using in each step. Also clearly indicate the scope of any boxes you use.
- For counterexamples, give a complete model. Show that the model satisfies the premise(s) of the sequent in question, but does not satisfy the respective conclusion.

$$\neg\exists x P(x) \lor \neg\exists y Q(y) \vdash \forall z \neg(Q(z) \land P(z))$$

Example 6

$$\neg\exists x P(x) \lor \neg\exists y Q(y) \vdash \forall z \neg(Q(z) \land P(z))$$

1.		$\neg\exists x\ P(x) \lor \neg\exists y\ Q(y)$	prem
2.	z_0	$Q(z_0) \land P(z_0)$	assum
3.		$\neg\exists x\ P(x)$	assum
4.		$P(z_0)$	$\land$e2
5.		$\exists x\ P(x)$	$\exists$i4
6.		$\perp$	$\neg$e3, 5
7.		$\neg\exists y\ Q(y)$	assum
8.		$Q(z_0)$	$\land$e2
9.		$\exists y\ Q(y)$	$\exists$i8
10.		$\perp$	$\neg$e7, 9
11.		$\perp$	$\lor$e1, 3 – 6, 7 – 10
12.		$\neg(Q(z_0) \land P(z_0))$	$\neg$i3 – 11
13.		$\forall z\ \neg(Q(z) \land P(z))$	$\forall$i3 – 12

Example 7

Consider the following natural deduction proof for the sequent

$$\forall x\ (P(x) \to Q(x)),\quad \exists x\ P(x) \quad \vdash \quad \forall x Q(x).$$

Is the proof correct? If not, explain the error in the proof and either show how to correctly prove the sequent, or give a counterexample that proves the sequent invalid.

1.	$\forall x\ (P(x) \to Q(x))$	prem.
2.	$\exists x\ P(x)$	prem.
3.	x_0	
4.	$P(x_0)$	ass.
5.	$P(x_0) \to Q(x_0)$	$\forall$e 1
6.	$Q(x_0)$	$\to$ e, 4,5
7.	$\forall x\ Q(x)$	$\forall$i 4-6
8.	$\forall x\ Q(x)$	$\exists$e 2,3-7

Example 7

Consider the following natural deduction proof for the sequent

$$\forall x\,(P(x) \to Q(x)), \quad \exists x\, P(x) \quad \vdash \quad \forall x Q(x).$$

Is the proof correct? If not, explain the error in the proof and either show how to correctly prove the sequent, or give a counterexample that proves the sequent invalid.

1.	$\forall x\,(P(x) \to Q(x))$	prem.
2.	$\exists x\, P(x)$	prem.
3.	x_0	
4.	$P(x_0)$	ass.
5.	$P(x_0) \to Q(x_0)$	$\forall$e 1
6.	$Q(x_0)$	$\to$ e, 4,5
7.	$\forall x\, Q(x)$	$\forall$i 4-6
8.	$\forall x\, Q(x)$	$\exists$e 2,3-7

Assumption should be here

No fresh variable for forall introduction

Example 7

Consider the following natural deduction proof for the sequent

$$\forall x\ (P(x) \to Q(x)), \quad \exists x\ P(x) \quad \vdash \quad \forall x Q(x).$$

The sequent is not provable! The following Model M is a counterexample:

Model $\mathcal{M}$:

$$\mathcal{A} = \{a, b\}$$
$$P^{\mathcal{M}} = \{a\}$$
$$Q^{\mathcal{M}} = \{a\}$$

$$\mathcal{M} \models \forall x\ (P(x) \to Q(x)), \quad \exists x\ P(x)$$
$$\mathcal{M} \not\models \forall x Q(x)$$

Example 8

Construct a Reduced Ordered Binary Decision Diagram (ROBDD) for the formula

$$f = (\neg p \vee r) \wedge (q \vee \neg p) \wedge (\neg q \vee p)$$

using *variable order* $r < q < p$. Use complemented edges and a node for `true` as the only constant node. To simplify drawing, you may assume that *dangling edges* point to the constant node. Write down all cofactors that you compute to obtain the final result and mark them in the graph.

Example 8

Construct a Reduced Ordered Binary Decision Diagram (ROBDD) for the formula

$$f = (\neg p \vee r) \wedge (q \vee \neg p) \wedge (\neg q \vee p)$$

$f_r = (q \vee \neg p) \wedge (\neg q \vee p)$

$f_{rq} = p$

$f_{rqp} = \top$

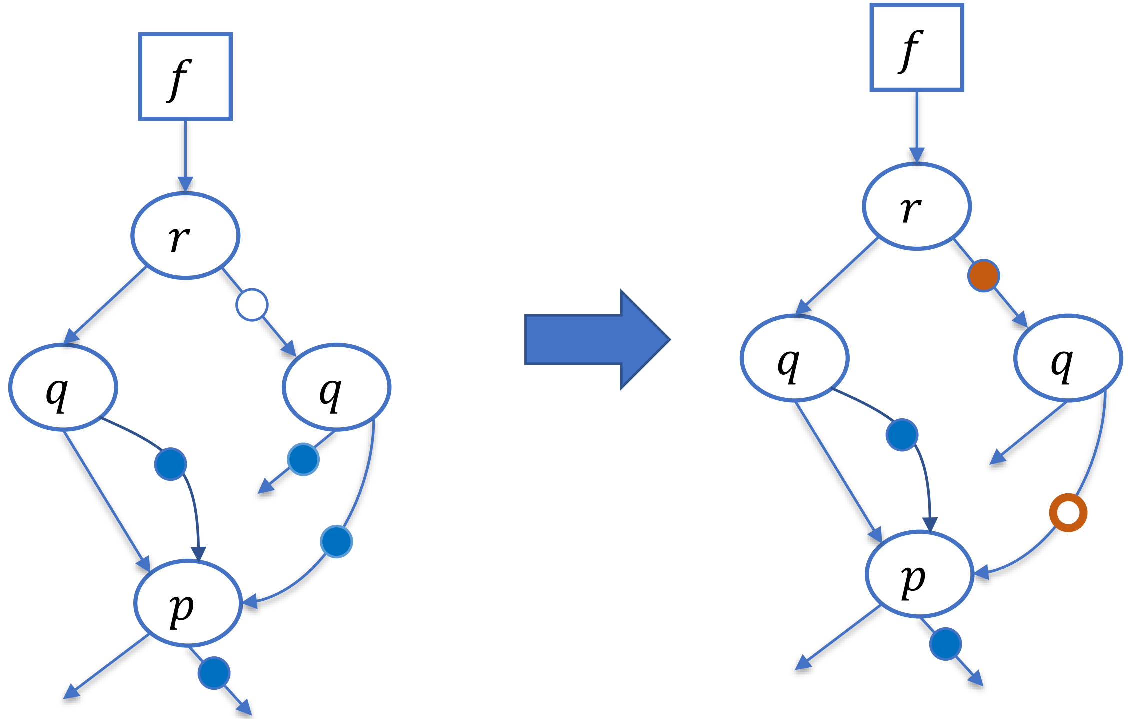
$f_{rq\neg p} = \bot$

$f_{r\neg q} = \neg p = \neg f_{rq}$

$f_{\neg r} = \neg p \wedge \neg q$

$f_{\neg rq} = \bot$

$f_{\neg r \neg q} = \neg p = \neg f_{rq}$



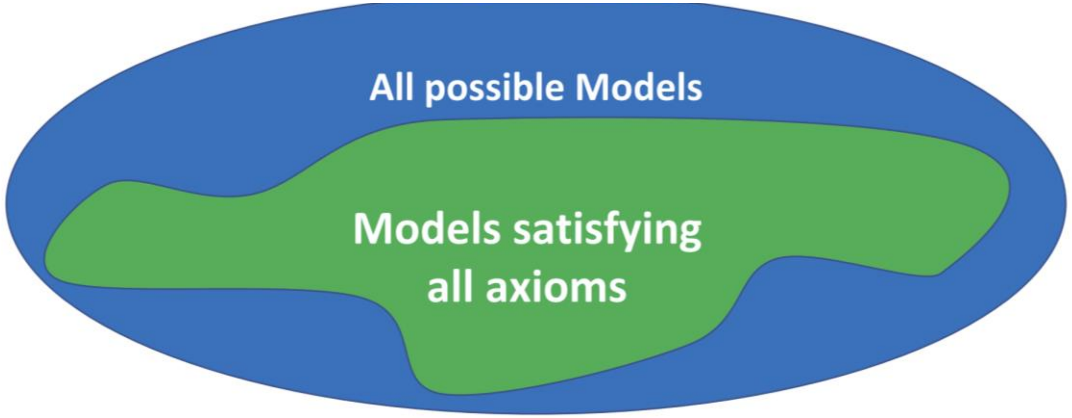
Example 9

Explain the problem of *satisfiability modulo theories*. As part of your explanation, explain what a *theory* is and explain the meaning of *theory-satisfiability*.

Example 9

Explain the problem of *satisfiability modulo theories*. As part of your explanation, explain what a *theory* is and explain the meaning of *theory-satisfiability*.

A formula φ is $\mathcal{T}$-satisfiable, if and only if there exists is a model M within $\mathcal{T}$ (satisfying all its axioms) that satisfies φ.



A theory fixes the interpretation/meaning of the predicate and function symbols that can be used in the formula. Thus, for checking *theory-satisfiability*, only models that interpret the functions and predicates as defined by the axioms in the theory are relevant.

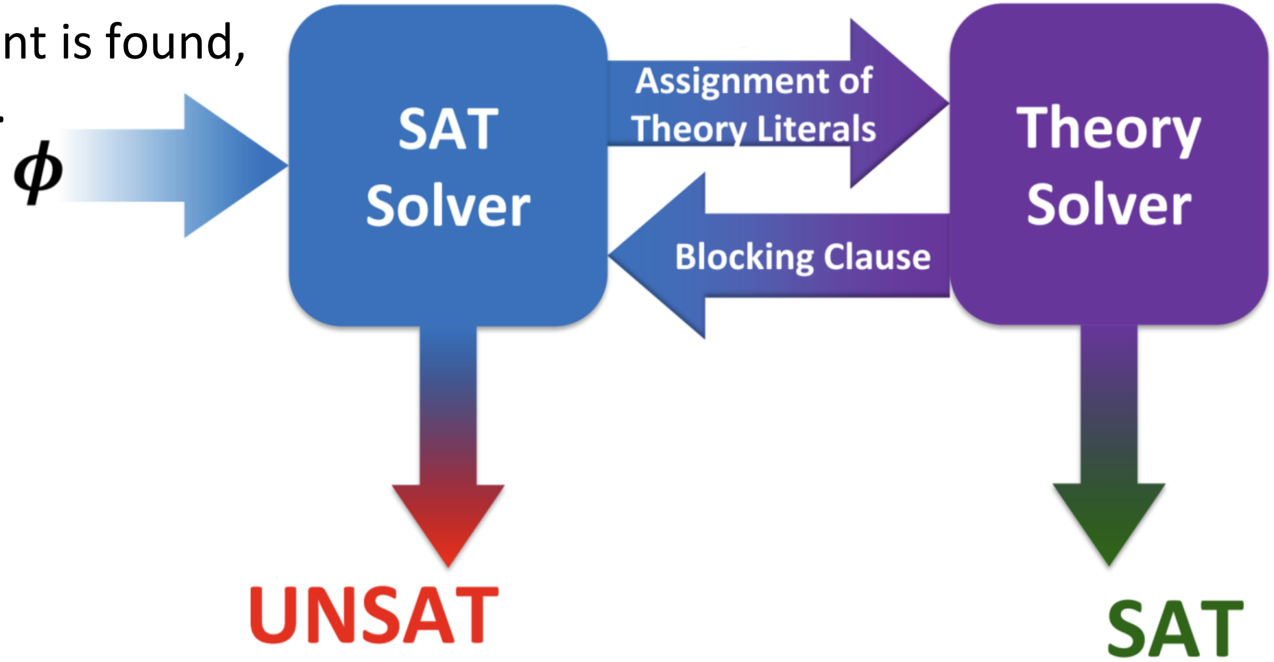
Example 10

Explain the concept of *Lazy Encoding* to decide satisfiability of formulas in a first-order theory.

Example 10

Explain the concept of *Lazy Encoding* to decide satisfiability of formulas in a first-order theory.

- The **propositional skeleton** of φ is given to a **SAT** solver.
- If a satisfying assignment is found, it is checked by a **theory solver**.
  - If the **assignment is consistent** with the theory, φ is T-satisfiable.
  - Otherwise, a **blocking clause** is generated.
- The SAT solver searches for a new assignment.
- This is repeated until either a T-consistent assignment is found, or the SAT solver cannot find any more assignments.



Example 11

Use the DPLL algorithm with conflict-driven clause learning to determine whether or not the set of clauses given is satisfiable. Decide variables in alphabetical order starting with the *negative* phase. For conflicts, draw conflict graphs after the end of the table, and add the learned clause to the table.
If the set of clauses resulted in `SAT`, give a satisfying model. If the set of clauses resulted in `UNSAT`, give a resolution proof that shows that the conjunction of the clauses from the table is unsatisfiable.

Clause 1: $\{a, b, c\}$

Clause 2: $\{\neg b, \neg c, e\}$

Clause 3: $\{b, e\}$

Clause 4: $\{b, \neg d\}$

Clause 5: $\{\neg c, d\}$

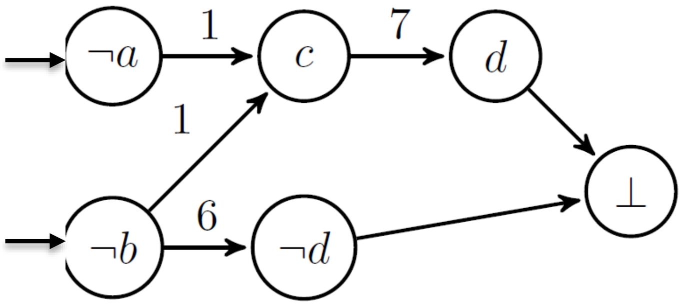
Clause 6: $\{\neg c, e\}$

Clause 7: $\{\neg a, \neg b, \neg c\}$

Clause 8: $\{a, c, \neg e\}$

Example 11

Step	1	2	3	4	5
Decision Level	0	1	2	2	2
Assignment	-	$\neg a$	$\neg a, \neg b$	$\neg a, \neg b, c$	$\neg a, \neg b, c, \neg d$
Cl. 1: a, b, c	a, b, c	b, c	c	✓	✓
Cl. 2: $\neg a, \neg b, \neg c$	$\neg a, \neg b, \neg c$	✓	✓	✓	✓
Cl. 3: $a, c, \neg e$	$a, c, \neg e$	$c, \neg e$	$c, \neg e$	✓	✓
Cl. 4: $\neg b, \neg c, e$	$\neg b, \neg c, e$	$\neg b, \neg c, e$	✓	✓	✓
Cl. 5: b, e	b, e	b, e	e	e	e
Cl. 6: $b, \neg d$	$b, \neg d$	$b, \neg d$	$\neg d$	$\neg d$	✓
Cl. 7: $\neg c, d$	$\neg c, d$	$\neg c, d$	$\neg c, d$	d	{} ✗
Cl. 8: $\neg c, e$	$\neg c, e$	$\neg c, e$	$\neg c, e$	e	e
BCP	-	-	c	$\neg d$	-
PL	-	-	-	-	-
Decision	$\neg a$	$\neg b$	-	-	-



$$\dfrac{\dfrac{7.\ \neg c \lor d \qquad 6.\ b \lor \neg d}{b \lor \neg c} \qquad 1.\ a \lor b \lor c}{a \lor b}$$

Example 11

Step	(1)	6	7	8	9
Decision Level	1	1	1	2	2
Assignment	$\neg a$	$\neg a, b$	$\neg a, b, d$	$\neg a, b, d, \neg c$	$\neg a, b, d, \neg c, \neg e$
Cl. 1: a, b, c	b, c	✓	✓	✓	✓
Cl. 2: $\neg c, d$	$\neg c, d$	$\neg c, d$	✓	✓	✓
Cl. 3: $\neg c, e$	$\neg c, e$	$\neg c, e$	$\neg c, e$	✓	✓
Cl. 4: $\neg a, \neg b, \neg c$	✓	✓	✓	✓	✓
Cl. 5: $a, c, \neg e$	$c, \neg e$	$c, \neg e$	$c, \neg e$	$\neg e$	✓
Cl. 6: $\neg b, \neg c, e$	$\neg b, \neg c, e$	$\neg c, e$	$\neg c, e$	✓	✓
Cl. 7: b, e	b, e	✓	✓	✓	✓
Cl. 8: $b, \neg d$	$b, \neg d$	✓	✓	✓	✓
Cl. 9: a, b	b	✓	✓	✓	✓
BCP	b	-	-	$\neg e$	-
PL	-	d	-	-	-
Decision	-	-	$\neg c$	-	SAT

Example 12

- Explain the concept of *eager encoding* to solve formulas in in SMT.
- Give the 3 main steps that are performed in algorithms based on eager encoding.

Example 12

- Explain the concept of *eager encoding* to solve formulas in in SMT.
- Give the 3 main steps that are performed in algorithms based on eager encoding.

Translates original formula to **equisatisfiable propositional formula** by eagerly adding **any instances of axioms** that could be needed.

Theory Formula
$\boldsymbol{\phi^T}$

$\Updownarrow$ **equisatisfiable**

$\mathcal{A} \wedge \boldsymbol{\phi}$

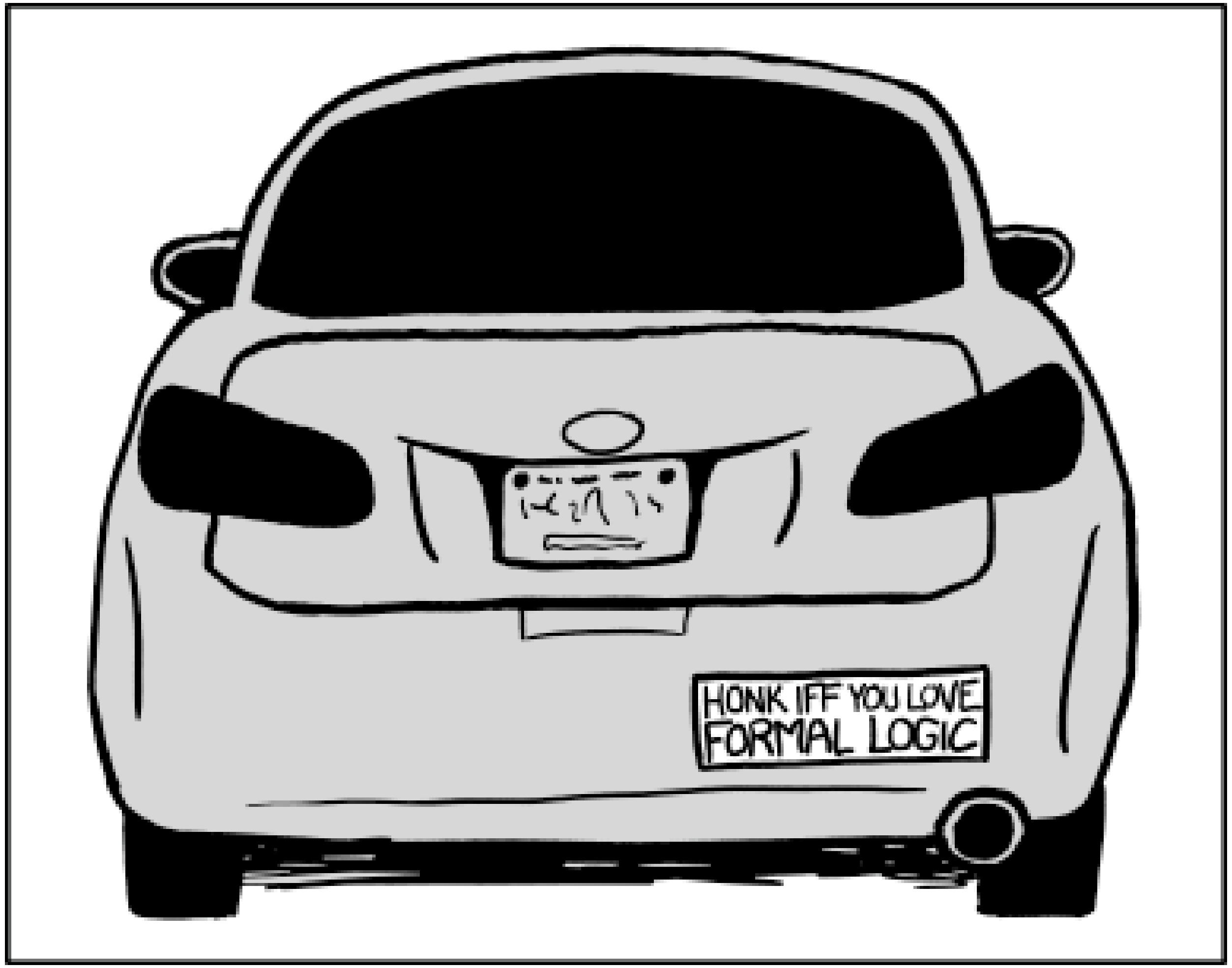
Steps:

(I) Replace any unique $\mathcal{T}$-atom in the original formula φ with a fresh propositional variable to get a propositional formula $\hat{\varphi}$.

(II) Generate a propositional formula φ_{cons} that constrains the values of the introduced propositional variables to preserve the information of the theory.

(III) Invoke a SAT solver on the propositional formula $\varphi_{prop} := \hat{\varphi} \wedge \varphi_{cons}$ that corresponds to an equisatisfiable propositional formula to φ.

Thank You



https://xkcd.com/1033/