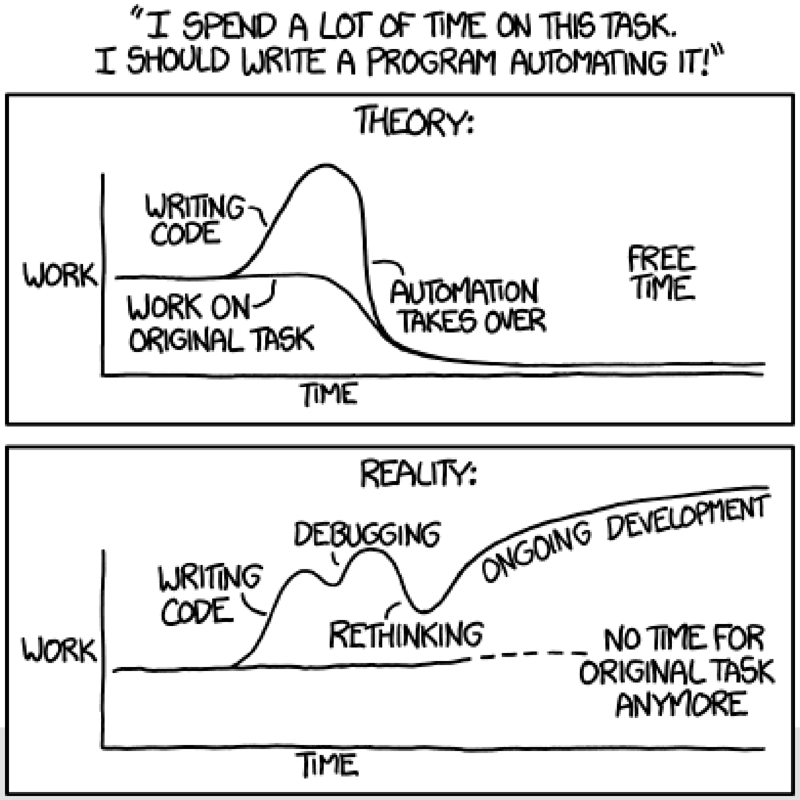
SCIENCE
PASSION
TECHNOLOGY

Theories in Predicate Logic

and Satisfiability Modulo Theories

Bettina Könighofer
bettina.koenighofer@iaik.tugraz.at

Stefan Pranger
stefan.pranger@iaik.tugraz.at



https://xkcd.com/2323/

Motivation

- We want write formulas like
  - $\varphi = x \geq 0 \land (x + y \leq 2 \lor x + y \geq 6) \land (x + y \geq 1 \lor x - y \geq 4)$
- using
  - Real Numbers, Integers, Function and Predicates like +,-,<,=,>…

- Theory
  - **Axioms** that define interpretation/meaning for functions and predicates

- Satisfiability Modulo Theory
  - Solving first-order formulas within a theory
  - → Checking whether a formula logic is **satisfiable modulo theory** means that we only consider models that interpret functions and predicates as defined by the **axioms in the theory**.

Outline

- Definition and Notations
  - What is a theory?
  - …
- Implementation of SMT Solvers
  - Eager Encoding
    explicit encoding of axioms

vs

  - Lazy Encoding
    use specialized theory solvers
    in combination with SAT solvers

Notion of "Theory"

Application Domain	Structures & Objects	Predicates & Functions
Arithmetic	Numbers (Integers, Rationals, Reals)	$= \quad < \quad > \quad \leq \quad \geq$ $+ \quad \cdot$
Computer Programs	Arrays, Bitvectors, Lists,…	Array-Read, Array-Write, …

Definition of a Theory

Definition of a First-Order Theory $\mathcal{T}$:

- Signature Σ
  - is a set of **constants, predicate and function symbols**
  - besides the logical symbols *(logical connectives like* $\wedge, \vee \cdots$*, variables like* x, y *...,* *and quantifiers like* $\forall x$), a formula only has symbols from Σ
  - → Do not use any non-logical symbols (constants, predicates or functions) not contained in Σ

- Set of Axioms $\mathcal{A}$
  - Sentences (=Formulas without free variables) with symbols from Σ only
  - Gives **meaning** to the predicate and function symbols

Theory of Linear Integer Arithmetic $\mathcal{T}_{\mathbf{LIA}}$

Example: $\varphi := x \geq 0 \wedge (x + y \leq 2 \vee x + y \geq 6)$

Definition of $\mathcal{T}_{\mathbf{LIA}}$:

- $\Sigma_{\text{LIA}} := \{\dots, -3, -2, -1, 0, 1, 2, 3 \dots, =, +, -, \neq, <, >, \leq, \geq\}$
- $\mathcal{A}_{LIA}$: defines the usual meaning to all symbols
  - Maps constants to their corresponding value in $\mathbb{Z}$
  - E.g., The function **+** is interpreted as the addition function, e.g.
    - …
    - 0+0 → 0
    - 0+1 → 1….

Theory of Equality $\mathcal{T}_{\mathrm{E}}$

Example: $\varphi := (x = b) \land (y \neq x) \rightarrow (w = b)$

Definition of $\mathcal{T}_{\mathrm{E}}$:

- $\Sigma_{\mathrm{E}} := \{a_0, b_0, c_0, \ldots, =\}$
  - Binary equality predicate $=$
  - Arbitrary constant symbols
- $\mathcal{A}_E$:
  1. $\forall x. x = x$ (reflexivity)
  2. $\forall x. \forall y. (x = y \rightarrow y = x)$ (symmetry)
  3. $\forall x. \forall y. \forall z. (x = y \land y = z \rightarrow x = z)$ (transitivity)

Uninterpreted Functions

- An uninterpreted function has no other property than its name, its arity and the function congruence property
  - Given the same inputs, it gives the same outputs
- Used for abstractions
  - $a \cdot \left(f(b) + f(c)\right) = d \wedge b \cdot \left(f(a) + f(c)\right) \neq \mathrm{d} \wedge a = b$
  - Using uninterpreted functions we get:
  - $m\left(a, p\left(f(b), f(c)\right)\right) = d \wedge m\left(b, p\left(f(a), f(c)\right)\right) \neq d \wedge a = b$
  - Can be used to show UNSAT of the formula

Theory of Equality & Uninterpreted Functions $\mathcal{T}_{\mathbf{EUF}}$

Example: $\varphi \coloneqq ((f(x) = g(b)) \land (f(y) \neq f(x))) \rightarrow P(x)$

Definition of $\mathcal{T}_{EUF}$:

- $\Sigma_{\mathrm{EUF}} = \{a_0, b_0, c_0, \dots, =\}$
  - Binary equality predicate $=$
  - Arbitrary constant, function and predicate symbols

- $\mathcal{A}_{EUF}$

1-3 same as in $\mathcal{A}_E$ (reflexivity), (symmetry), (transitivity)

4 $\forall \overline{x}. \forall \overline{y}. ((\bigwedge_i x_i = y_i) \rightarrow f(\overline{x}) = f(\overline{y}))$ (function congruence)

5 $\forall \overline{x}. \forall \overline{y}. ((\bigwedge_i x_i = y_i) \rightarrow P(\overline{x}) = P(\overline{y}))$ (predicate equivalence)

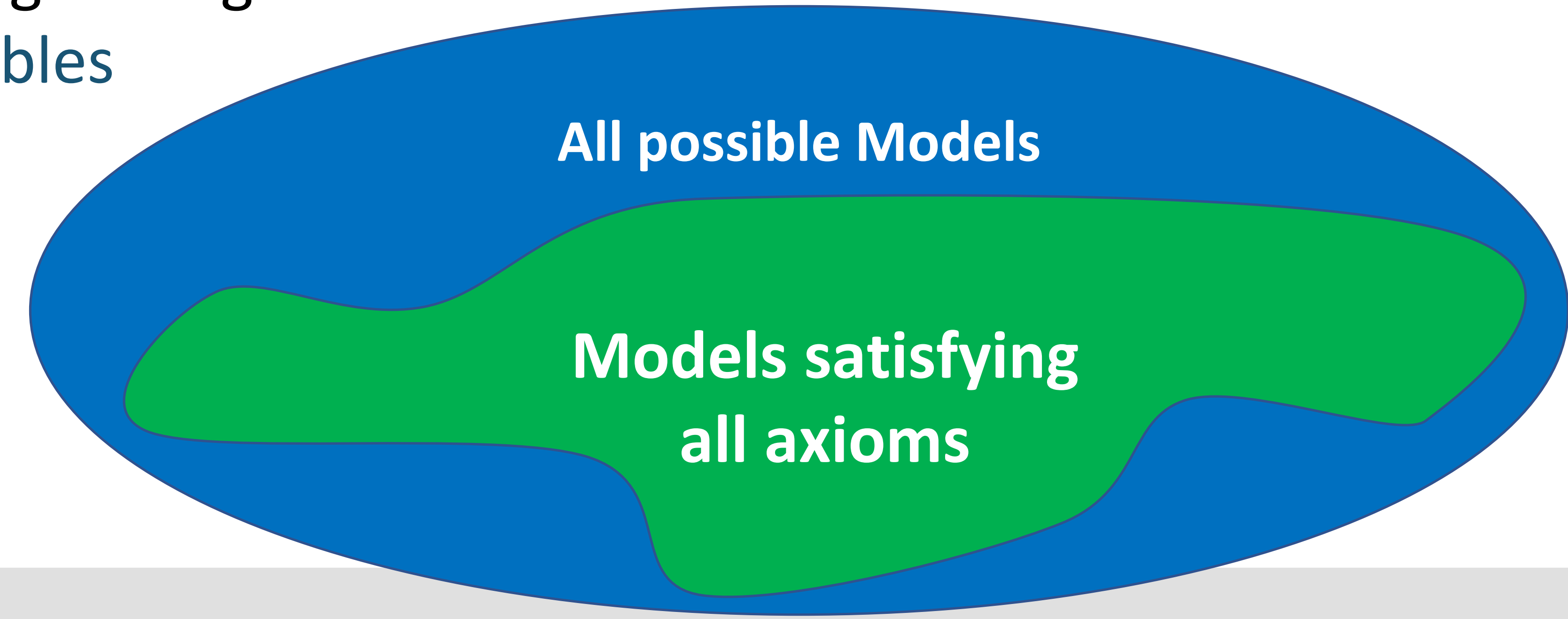
$\mathcal{T}$-terms, $\mathcal{T}$-atoms and $\mathcal{T}$-literals

- $\varphi = x \geq 0 \land \neg(x + y \leq 2 \lor x + y \geq 6) \land (x + y \geq 1 \lor x - y \geq 4)$
- **$\mathcal{T}$-term:**
  - Constants in Σ, variables, function instances with function symbols and inputs in Σ
  - $0, \mathrm{x}, x + y, x - y$
- **$\mathcal{T}$-atom:**
  - Predicate instances with predicate symbol and inputs in Σ
  - $x \geq 0, x + y \leq 2, \ldots.$
- **$\mathcal{T}$-literal:**
  - $\mathcal{T}$-atom or its negation
  - $x + y \leq 2, \neg(x + y \leq 2), \ldots$

Models within a Theory

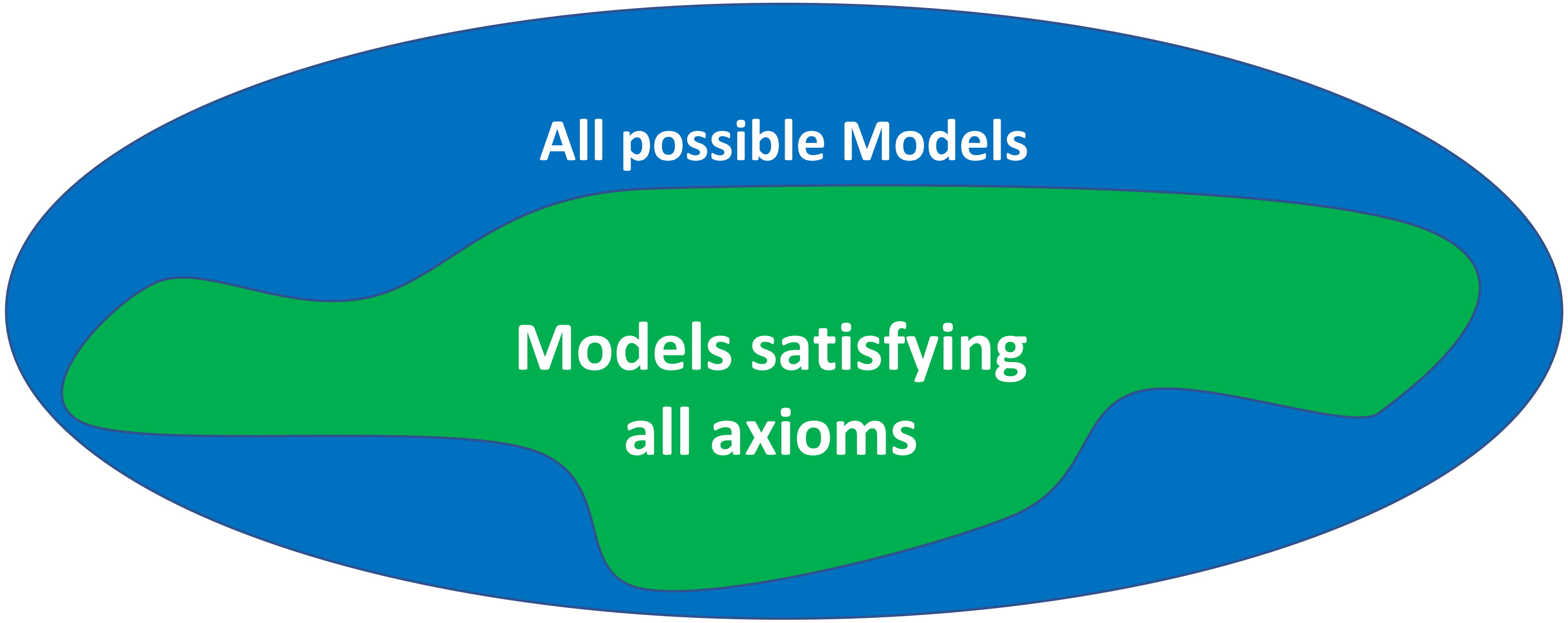
- Model in Predicate Logic
  - Defines domain
  - Value of free variables
  - Concrete implementation of functions and predicates

- Model in Predicate Logic using Theories
  - Value of free variables



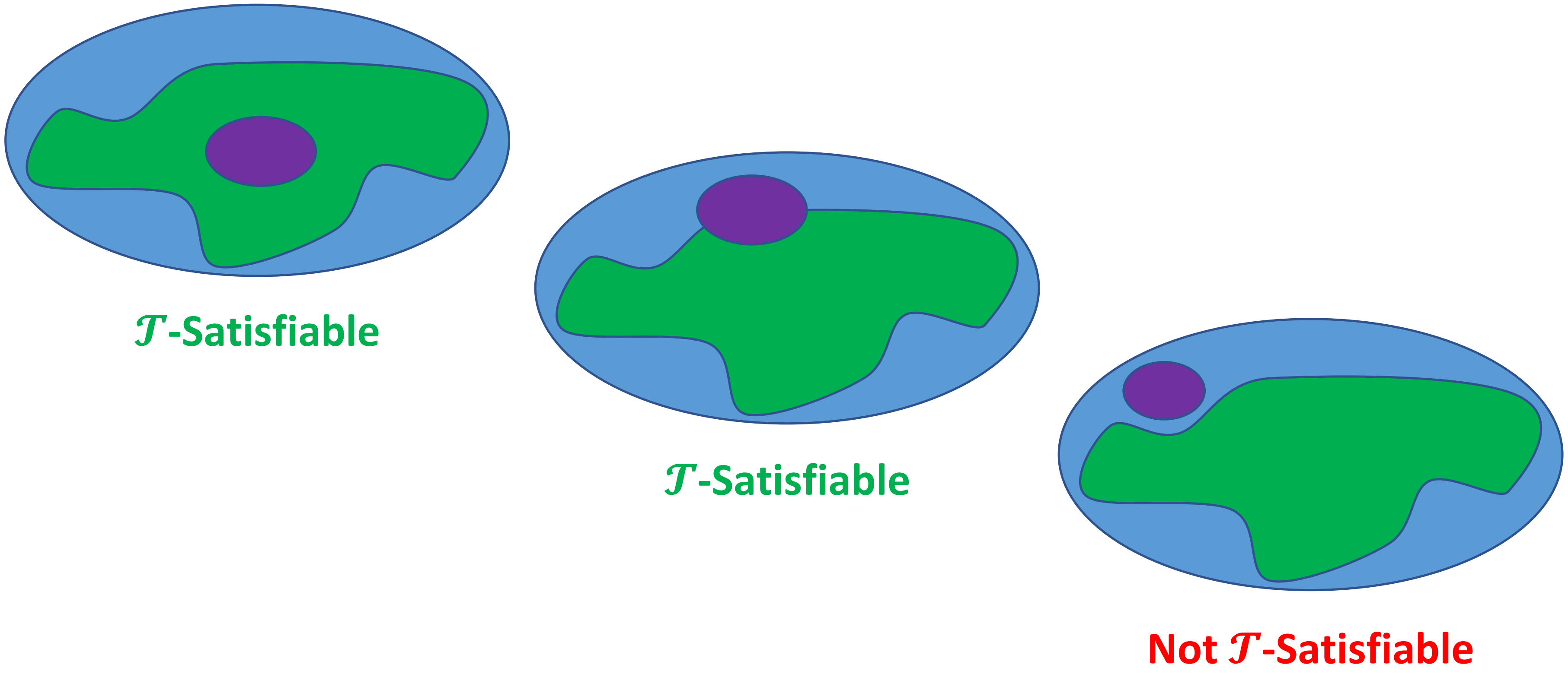
$\mathcal{T}$-Satisfiability, $\mathcal{T}$-validity, $\mathcal{T}$-Equivalence, $\mathcal{T}$-Entailment

- Only models satisfying axioms are relevant
- ➔ “Satisfiability ***modulo*** (=‘with respect to’) theories”



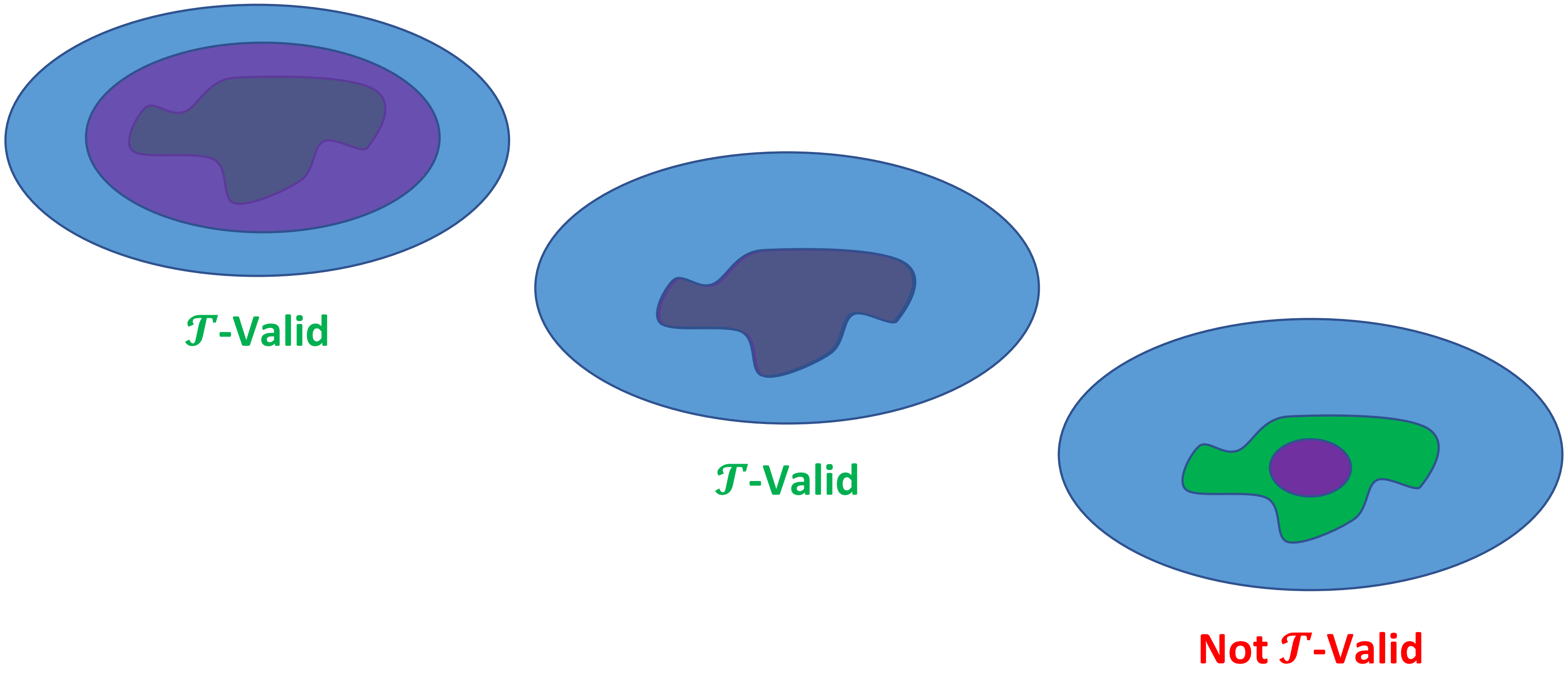
$\mathcal{T}$-Satisfiability

- **Green:** Models Satisfying all Axioms
- **Violet:** Models Satisfying Formula in Question



$\mathcal{T}$-Validity

- **Green:** Models Satisfying all Axioms
- **Violet:** Models Satisfying Formula in Question



$\mathcal{T}$-Entailment and $\mathcal{T}$-Equivalence

- Similar to Satisfiability & Validity
- Only consider models that satisfy all axioms
  - Models not satisfying (at least) one axiom: Irrelevant Model!

Outline

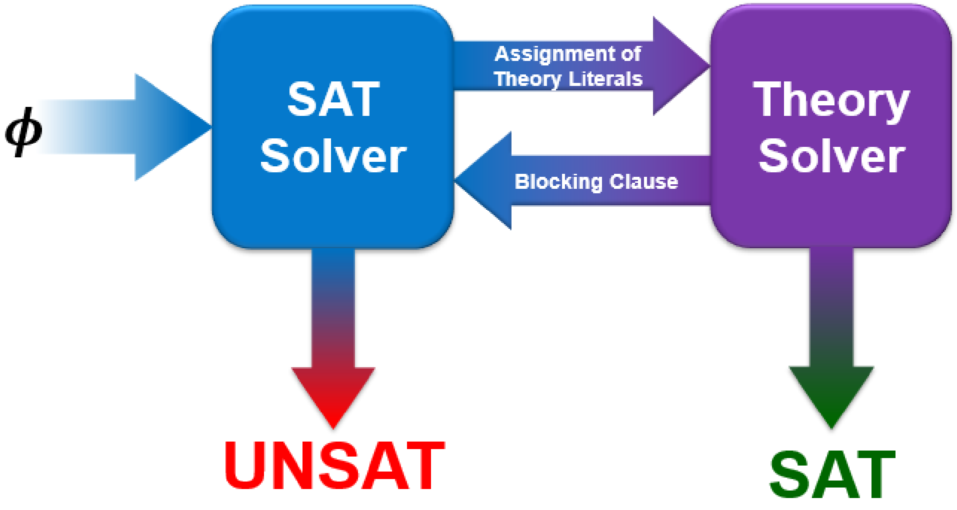
- Definition and Notations
  - What is a theory?
  - …
- Implementation of SMT Solvers
  - Eager Encoding
    explicit encoding of axioms

vs

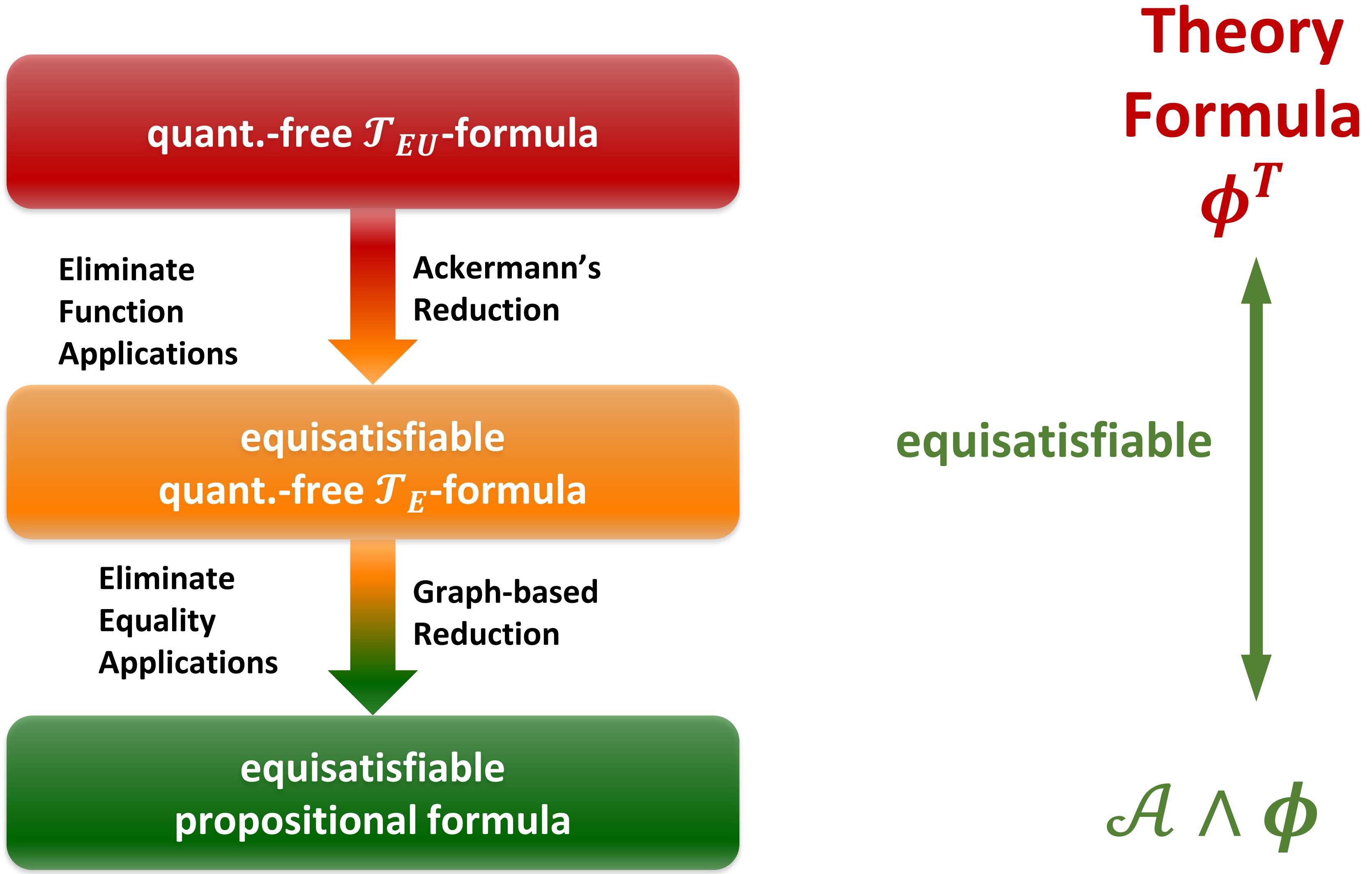
  - Lazy Encoding
    use specialized theory solvers
    in combination with SAT solvers

Implementations of SMT Solvers

- Eager Encoding
  - Equisatisfiable propositional formula
    - Adds all constraints that could be needed at once
  - SAT Solver
- Lazy Encoding
  - SAT Solver and Theory Solver
  - Add constrains only when needed



Eager Encoding for Formulas in $\mathcal{T}_{EUF}$



Ackermann's Reduction

Input: Formula $\boldsymbol{\phi}_{\mathbf{EUF}}$ in $\mathcal{T}_{EUF}$ Output: Formula $\boldsymbol{\phi}_{\boldsymbol{E}}$ in $\mathcal{T}_{E}$

- Replace each function instance via a fresh variable
  - $f(x) \rightsquigarrow f_x$
  - Form formula $\widehat{\boldsymbol{\phi}}_{\mathbf{EUF}}$
- Add functional-consistency constraints
  - $(x = y) \rightarrow (f_x = f_y)$
  - Form formula ϕ_{FC}
- $\phi_E = \phi_{FC} \wedge \hat{\phi}_{EUF}$

Example of Ackermann's Reduction

- $\boldsymbol{\phi_{EUF} \coloneqq \big(f(a) = f(b)\big) \land \neg\big(f(b) = f(c)\big)}$

1. $\hat{\phi}_{EUF} \coloneqq (f_a = f_b) \land \neg(f_b = f_c)$

2. $f: a, b, c$

   $\phi_{FC} \coloneqq \big((a = b) \rightarrow f_a = f_b\big) \land \big((b = c) \rightarrow f_b = f_c\big) \land$
   $\big((a = c) \rightarrow f_a = f_c\big)$

3. $\phi_E = \phi_{FC} \land \hat{\phi}_{EUF}$

Example of Ackermann's Reduction

[Lecture] Given the formula

$$\varphi_{EUF} \quad := \quad f(g(x)) = f(y) \;\vee\; (z = g(y) \wedge z \neq f(z))$$

Apply the *Ackermann* reduction algorithm to compute an equisatisfiable formula in $\mathcal{T}_E$.

$$\begin{aligned}\varphi_{FC} \quad := \quad & (x = y \to g_x = g_y)\wedge \\ & (g_x = y \to f_{gx} = f_y)\wedge \\ & (g_x = z \to f_{gx} = f_z)\wedge \\ & (y = z \to f_y = f_z)\end{aligned}$$

$$\hat{\varphi}_{EUF} \quad := \quad f_{gx} = f_y \;\vee\; (z = g_y \wedge z \neq f_z)$$

$$\varphi_E \quad := \hat{\varphi}_{EUF} \wedge \varphi_{FC}$$

Example of Ackermann's Reduction

[Lecture] Perform the graph-based reduction on the following formula to compute an equsatisfiable formula in propositional logic.

Given the formula

$$\varphi_{EUF} \quad := \quad f(x,y) = f(y,z) \ \vee \ (z = f(y,z) \wedge f(x,x) \neq f(x,y))$$

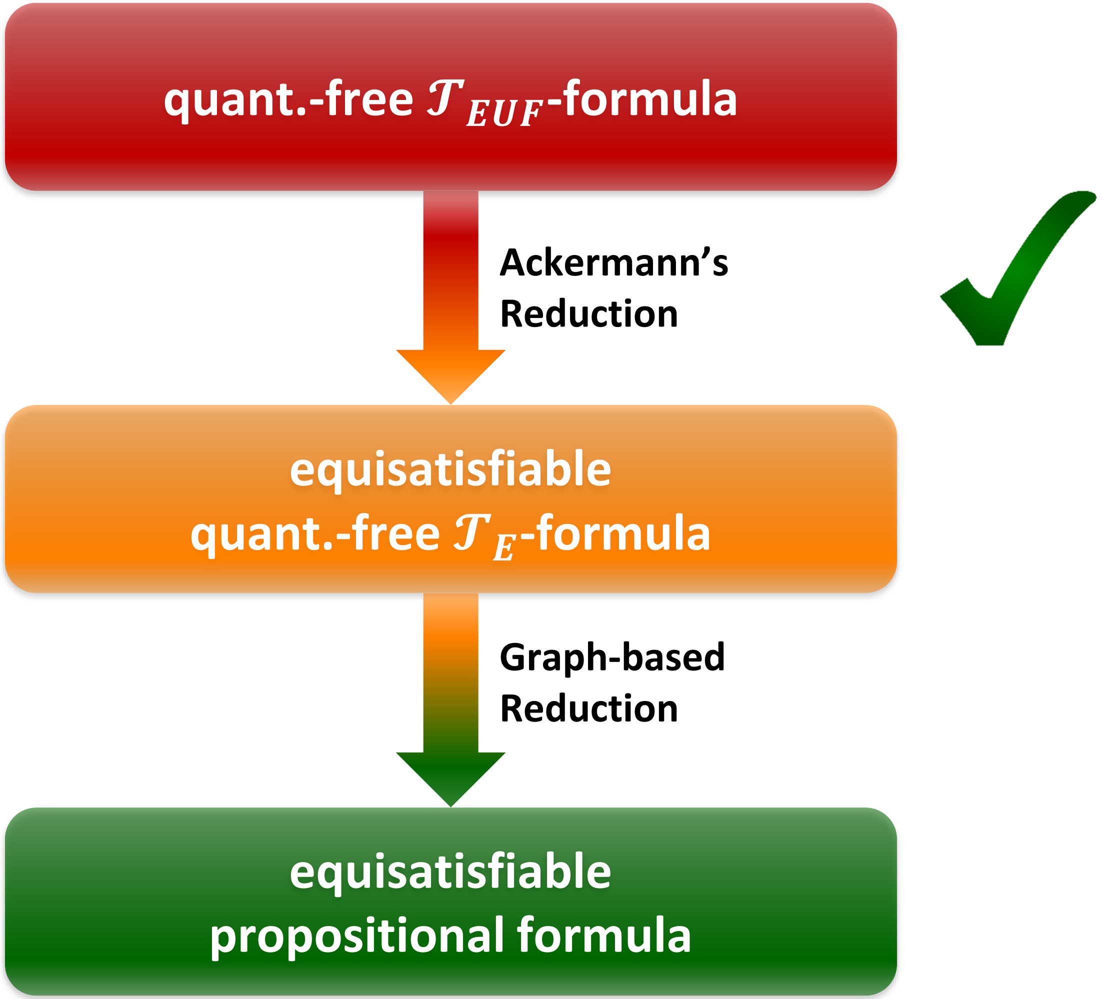
Apply the *Ackermann* reduction algorithm to compute an equisatisfiable formula in $\mathcal{T}_E$.

$$\begin{aligned} \varphi_{FC} \quad := \quad & (x = y \wedge y = z \rightarrow f_{xy} = f_{yz})\wedge \\ & (x = x \wedge y = x \rightarrow f_{xy} = f_{xx})\wedge \\ & (y = x \wedge z = x \rightarrow f_{yz} = f_{xx}) \end{aligned}$$

$$\hat{\varphi}_{EUF} \quad := \quad f_{xy} = f_{yz} \ \vee \ (z = f_{yz} \wedge f_{xx} \neq f_{xy})$$

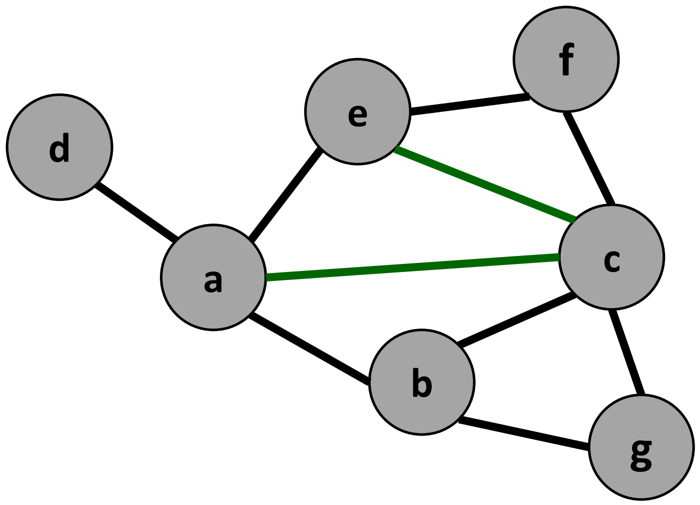
$$\varphi_E \quad := \hat{\varphi}_{EUF} \wedge \varphi_{FC}$$

Eager Encoding for Formulas in $\mathcal{T}_{EUF}$



Graph-Based Reduction

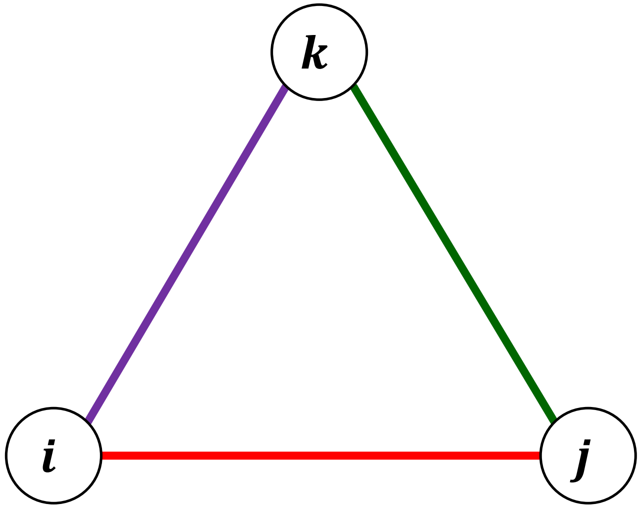
- Non-Polar Equality Graph
  - Node per variable
  - Edge per (dis)equality
- Make it chordal
  - No cycles size > 3



Graph-Based Reduction

- Fresh Propositional Variables
  - $a = b \rightsquigarrow e_{a=b}$
  - Order! (To ensure symmetry)
    $b = a \rightsquigarrow e_{a=b}$

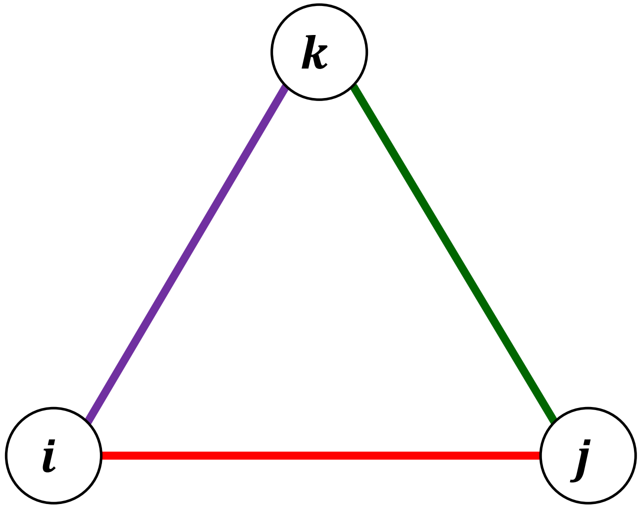
- Triangle (i, j, k):
  - Transitivity Constraints
    $(e_{i=j} \land e_{j=k} \rightarrow e_{i=k}) \land$
    $(e_{i=j} \land e_{i=k} \rightarrow e_{j=k}) \land$
    $(e_{i=k} \land e_{j=k} \rightarrow e_{i=j})$



Graph-Based Reduction

- Fresh Propositional Variables
  - $a = b \rightsquigarrow e_{a=b}$
  - Order! (To ensure symmetry)
    $b = a \rightsquigarrow e_{a=b}$
- Triangle (i, j, k):
  - Transitivity Constraints
    $(e_{i=j} \land e_{j=k} \to e_{i=k}) \land$
    $(e_{i=j} \land e_{i=k} \to e_{j=k}) \land$
    $(e_{i=k} \land e_{j=k} \to e_{i=j})$

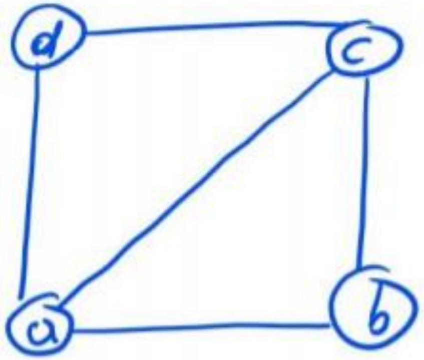
- $\phi_{prop} = \phi_{TC} \land \hat{\phi}_E$



→ SAT Solver

Example Graph-Based Reduction

- $\phi_E \coloneqq a = b \land b = c \land c = d \land d \neq a$



Triangles:
- a, b, c
- a, b, d

fresh vars: $e_{ab}, e_{bc}, e_{cd}, e_{ad}, e_{ac}$

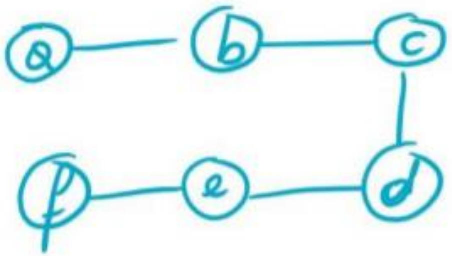
$$\phi_{TC} := (e_{ab} \land e_{bc} \rightarrow e_{ac}) \land (e_{ab} \land e_{ac} \rightarrow e_{bc}) \land (e_{bc} \land e_{ac} \rightarrow e_{ab}) \land (e_{ac} \land e_{cd} \rightarrow e_{ad}) \land (e_{cd} \land e_{ad} \rightarrow e_{ac}) \land (e_{ad} \land e_{ac} \rightarrow e_{cd})$$

$$\hat{\phi}_E := e_{ab} \land e_{bc} \land e_{cd} \land \neg e_{ad}$$

$$\phi_{prop} := \phi_{TC} \land \hat{\phi}_E$$

Example Graph-Based Reduction

- $\phi_E := a = b \wedge b \neq c \rightarrow \neg(c \neq d \vee d = e \wedge e = f)$



$\phi_{prop} := e_{ab} \vee e_{bc} \rightarrow (\neg e_{cd} \vee e_{de} \wedge e_{ef})$

Example Graph-Based Reduction

[Lecture] Perform graph-based reduction to translate a formula in $\mathcal{T}_E$ into an equisatisfiable formula in propositional logic.

$$\varphi_E \;:=\; (a = b \lor a = d) \rightarrow (b = c \land c \neq e \land e \neq d)$$

$$\begin{aligned}\varphi_{TC} := & (e_{a=b} \land e_{b=c} \rightarrow e_{a=c}) \land \\ & (e_{a=b} \land e_{a=c} \rightarrow e_{b=c}) \land \\ & (e_{b=c} \land e_{a=c} \rightarrow e_{a=b}) \land \\ \\ & (e_{a=c} \land e_{c=d} \rightarrow e_{a=d}) \land \\ & (e_{a=c} \land e_{a=d} \rightarrow e_{c=d}) \land \\ & (e_{c=d} \land e_{a=d} \rightarrow e_{a=c})\end{aligned}$$

- Triangle 1: a-b-c
- Triangle 2: a-c-d

$$\hat{\varphi}_E := (e_{a=b} \lor e_{a=d} \rightarrow (e_{b=c} \land \neg e_{c=d})$$

$$\varphi_{prop} := \varphi_{TC} \land \hat{\varphi}_E$$

Eager Encoding for Formulas in $\mathcal{T}_{EUF}$

quant.-free $\mathcal{T}_{UE}$-formula

↓ Ackermann's Reduction ✓

equisatisfiable quant.-free $\mathcal{T}_{E}$-formula

↓ Graph-based Reduction ✓

equisatisfiable propositional formula

→ SAT Solver

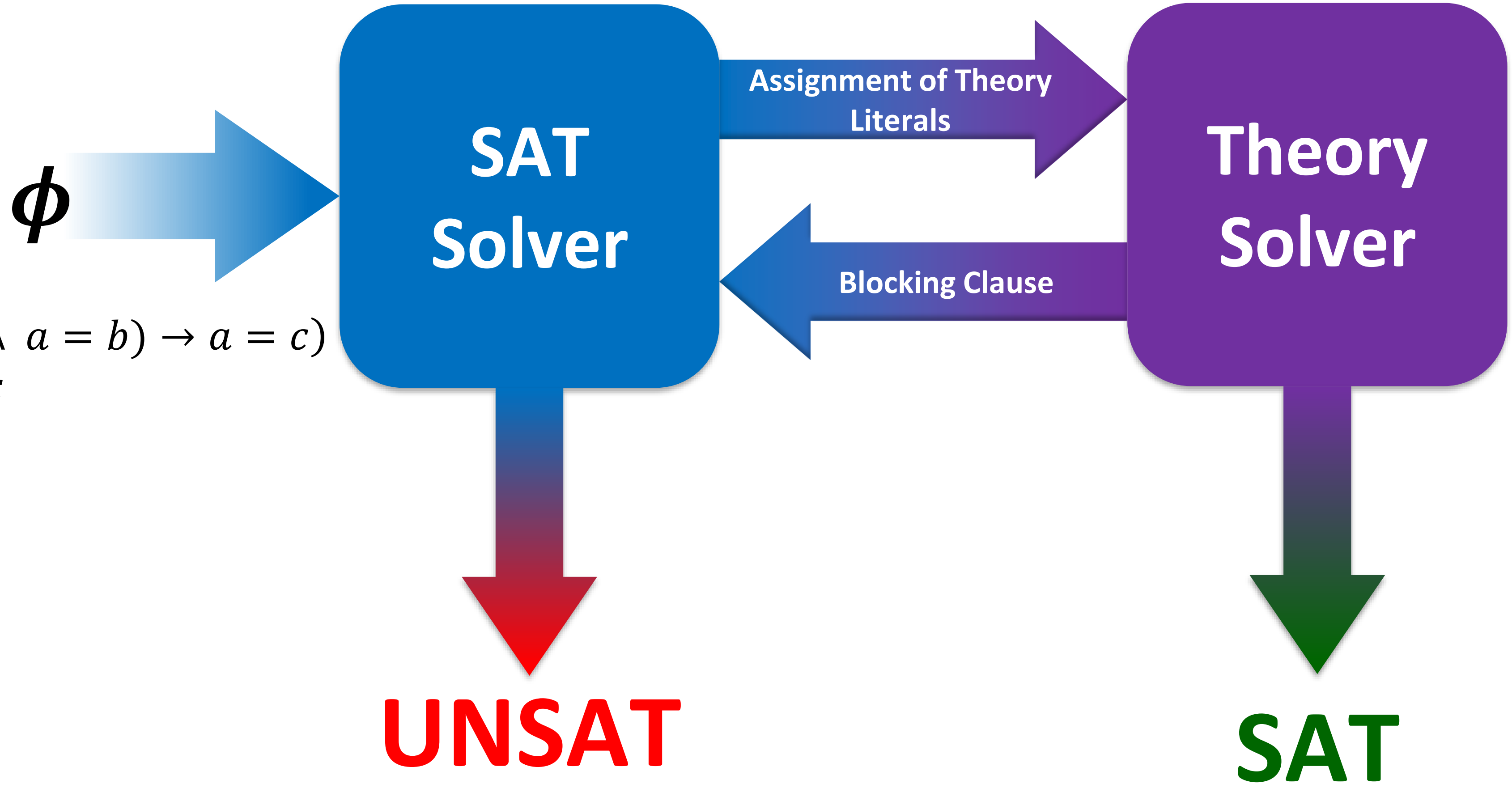
Outline

- Definition and Notations ✓
  - What is a theory?
  - …
- Implementation of SMT Solvers
  - Eager Encoding

    vs ✓

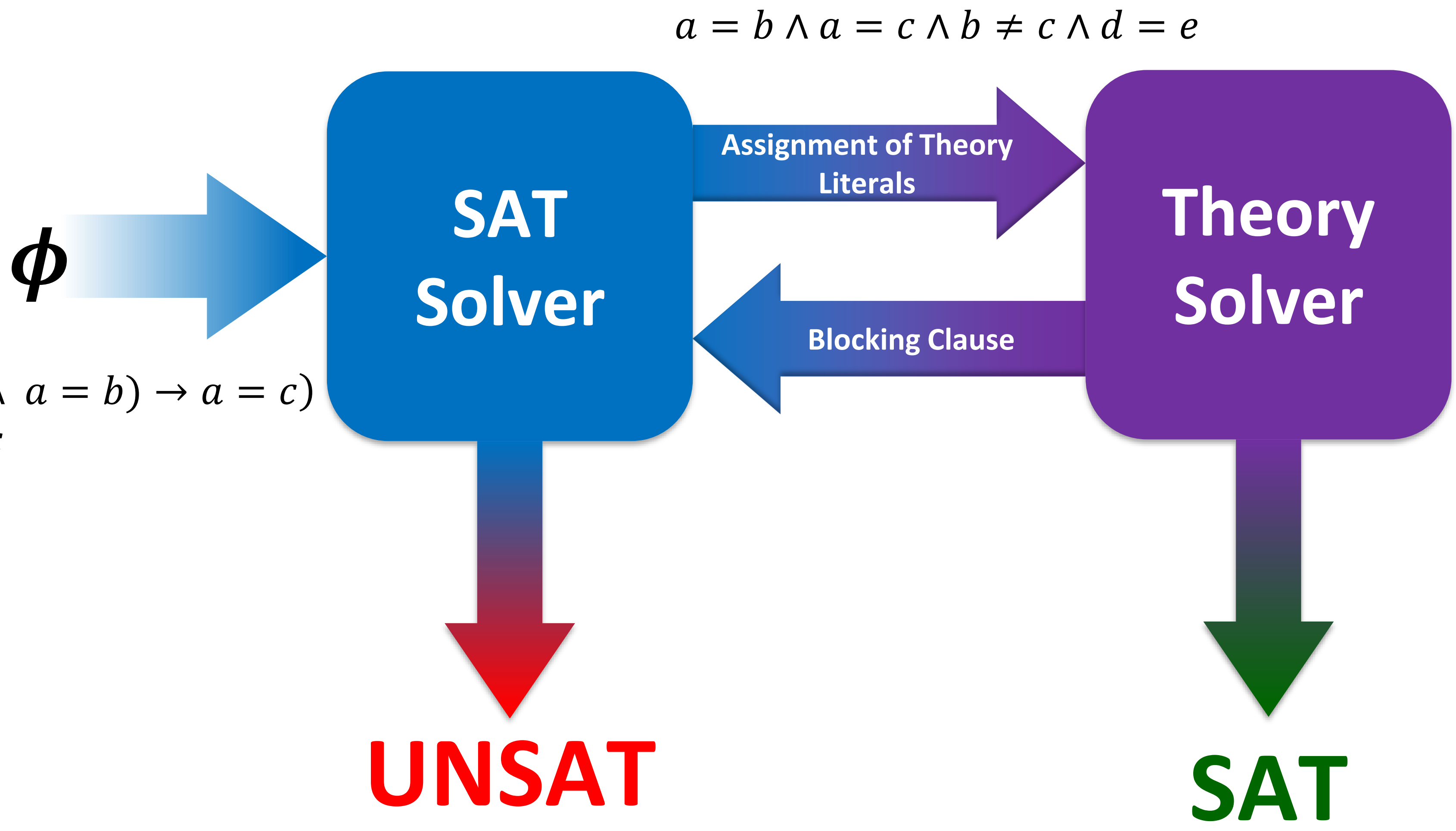
  - Lazy Encoding

(Very) Lazy Encoding



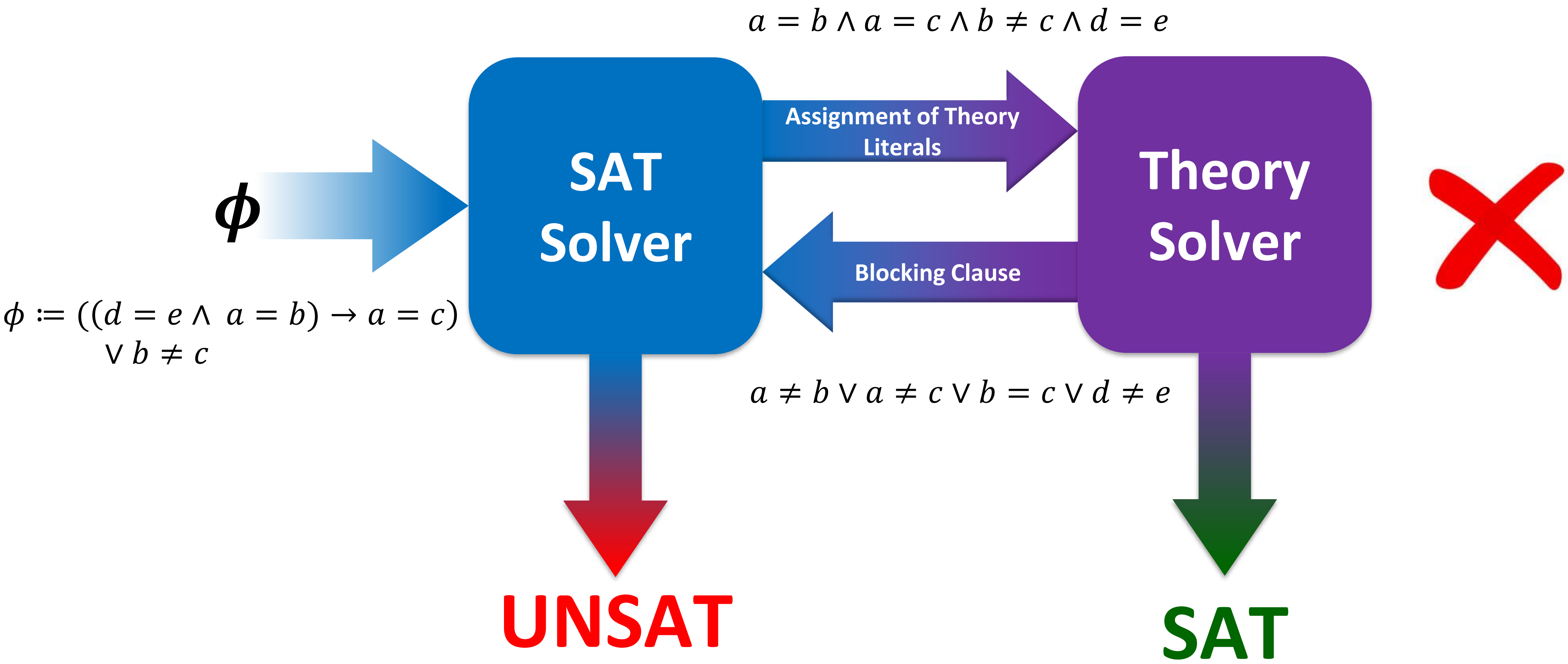
$\phi \coloneqq ((d = e \land a = b) \rightarrow a = c) \lor b \neq c$

(Very) Lazy Encoding

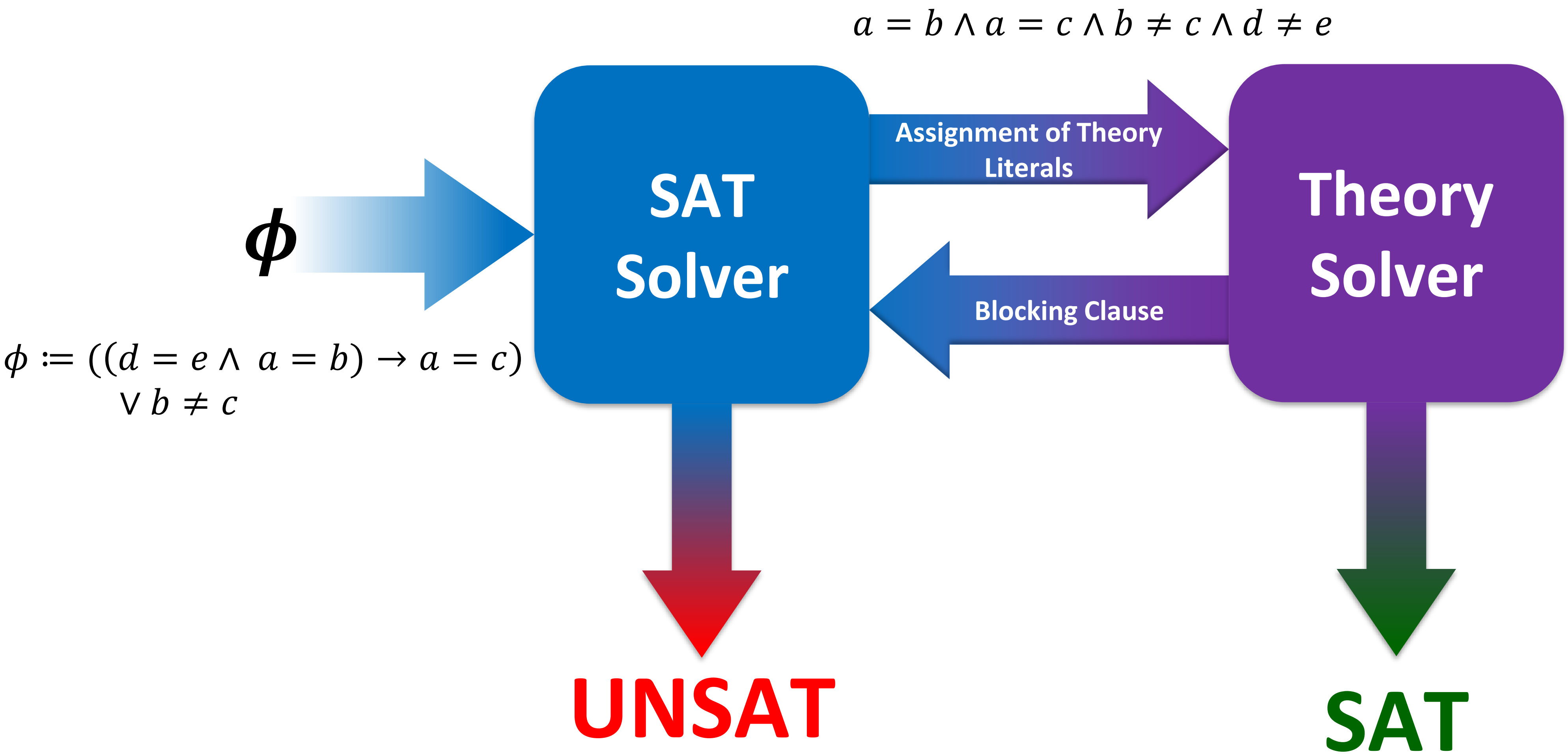


$\phi := ((d = e \wedge\ a = b) \rightarrow a = c) \vee b \neq c$

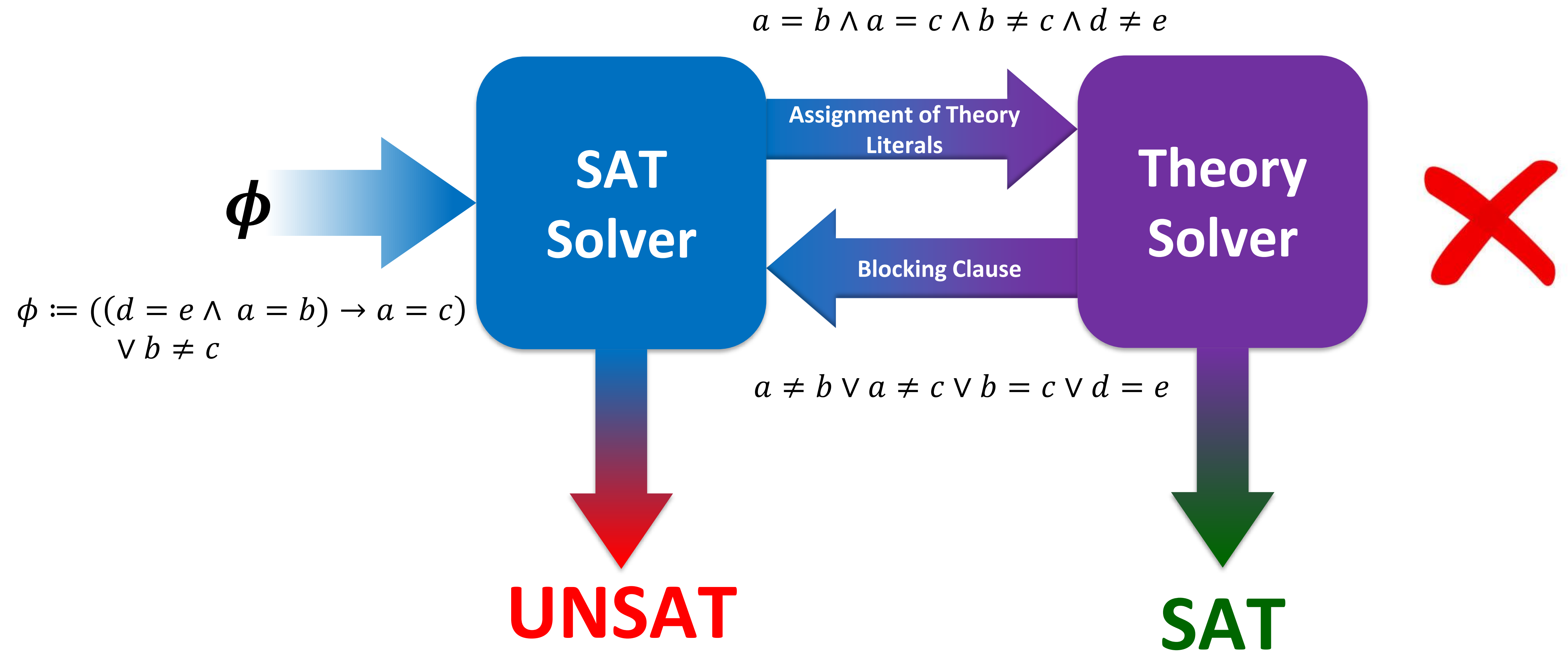
(Very) Lazy Encoding



(Very) Lazy Encoding

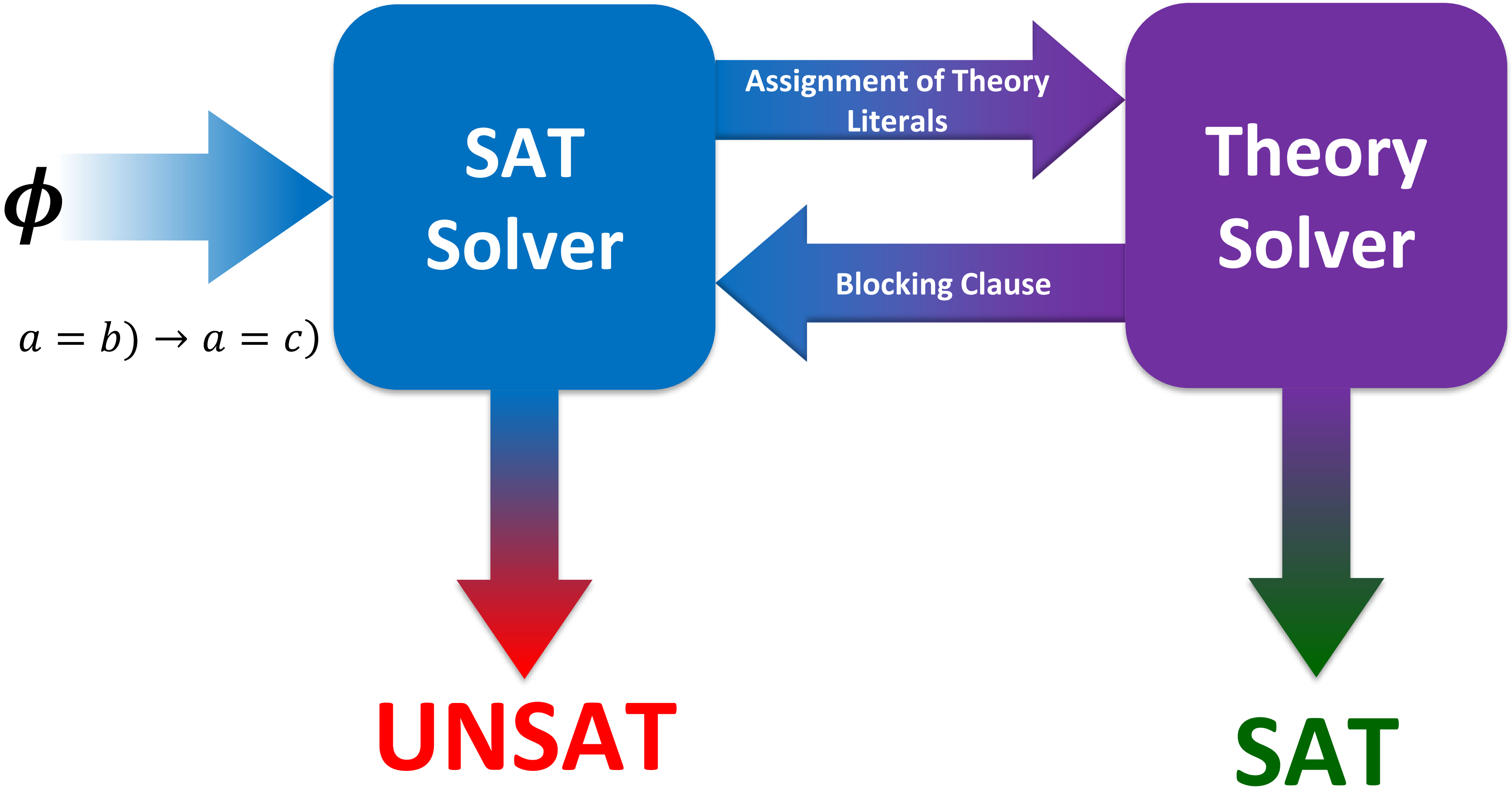


(Very) Lazy Encoding



(Very) Lazy Encoding

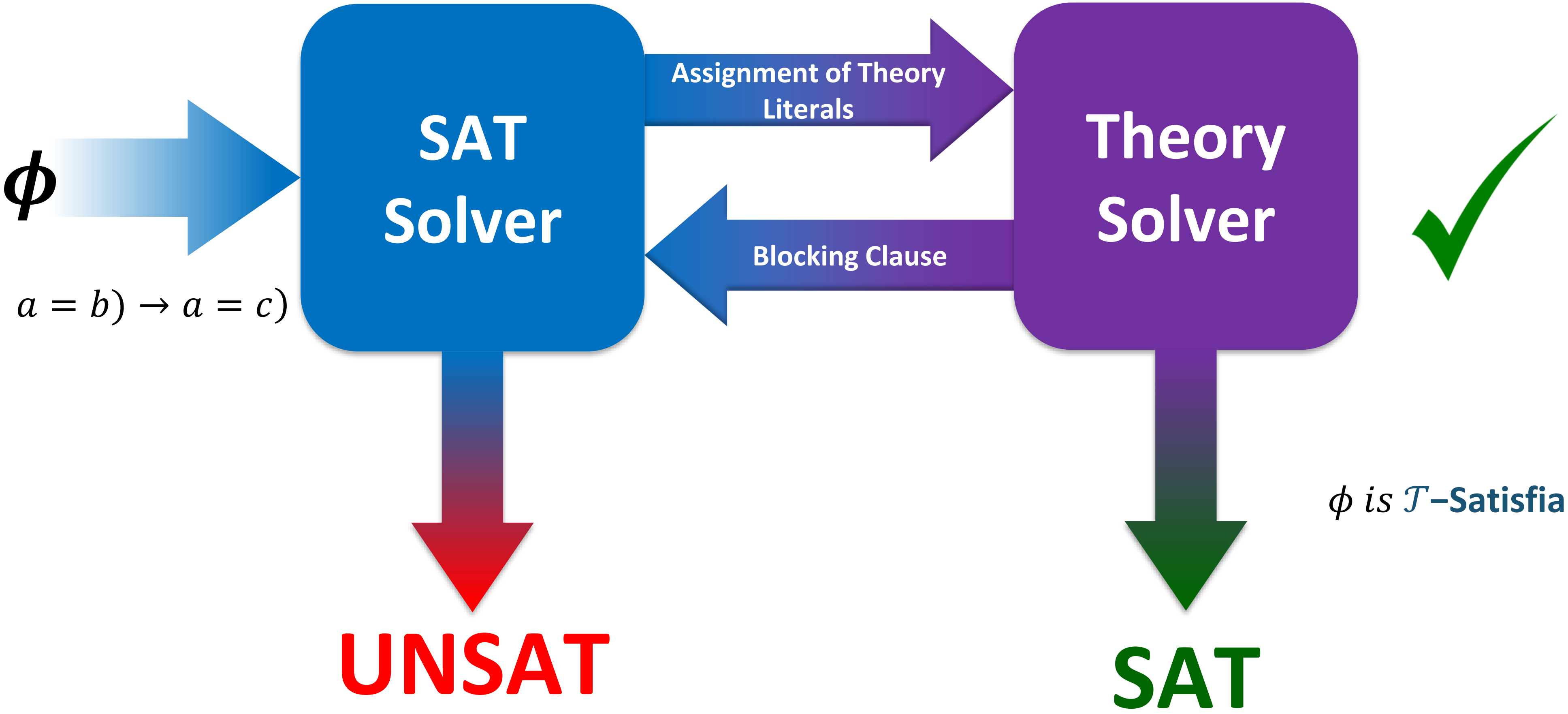
$$a = b \land a = c \land b = c \land d \neq e$$



$$\phi \coloneqq ((d = e \land a = b) \rightarrow a = c) \lor b \neq c$$

(Very) Lazy Encoding

$a = b \wedge a = c \wedge b = c \wedge d \neq e$



$\phi \coloneqq ((d = e \wedge \ a = b) \rightarrow a = c) \vee b \neq c$

ϕ *is* $\mathcal{T}$**–Satisfiability**

Conjunctive Fragment of $\mathcal{T}_{UE}$

- Theory solver takes conjunctions of theory literals as input
  - Equalities ($t_1 = t_2$)
  - Disequalities ($t_1 \neq t_2$)

- Terms t_i
  - Constants
    - $a, b, c, d, \ldots$
  - Uninterpreted Function instance
    - $f(a), g(b), h(c, d), \ldots$

Congruence-Closure Algorithm

1. For every equality, create a congruence class
   - E.g. $t_1 = t_2$: create class for t_1, t_2

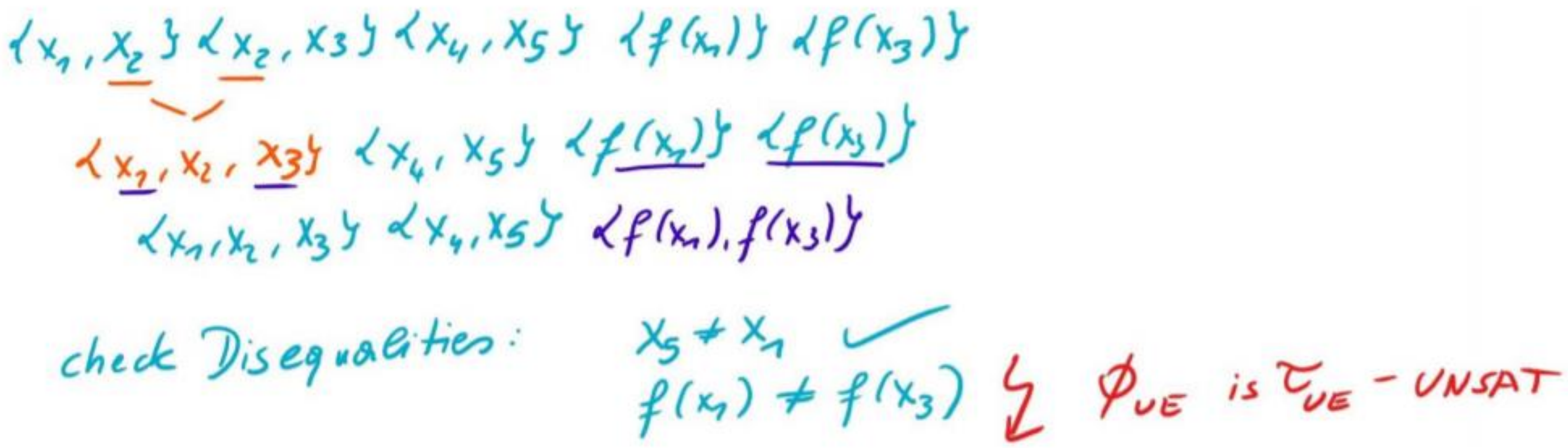
2. Create a singleton class for every term that only appears in disequalites

3. Merge clases:
   - Shared term between classes: Merge classes! (repeat)
   - t_i, t_j from same class: Merge classes of $f(t_i), f(t_j)$ (repeat)
   - No merging possible anymore, go to step 4

4. Check Disequalities $t_k \neq t_l$
   - t_k, t_l in same class: **UNSAT!**
   - Otherwise: **SAT!**

Example for CC-Algorithm

- $x_1 = x_2 \land x_2 = x_3 \land x_4 = x_5 \land x_5 \neq x_1 \land f(x_1) \neq f(x_3)$



Example for CC-Algorithm

- $x = f(y) \land y = f(u) \land u = v \land v = z \land v = f(y) \land f(x) \neq f(z)$

$\{x, f(y)\}\ \{y, f(u)\}\ \{u, v\}\ \{v, z\}\ \{v, f(y)\}\ \{f(x)\}\ \{f(z)\}$

$\{x, f(y), v\}\ \{y, f(u)\}\ \{u, v, z\}\ \{f(x)\}\ \{f(z)\}$

$\{x, v, z, f(y)\}\ \{y, f(u)\}\ \{f(x)\}\ \{f(z)\}$

$\{x, v, z, f(y)\}\ \{y, f(u)\}\ \{f(x), f(z)\}$

Check: $f(x) \neq f(z)$ ↯ T_{UE} - UNSAT

Example for CC-Algorithm

[Lecture] Consider the following formula in the conjunctive fragment of $\mathcal{T}_{EUF}$.

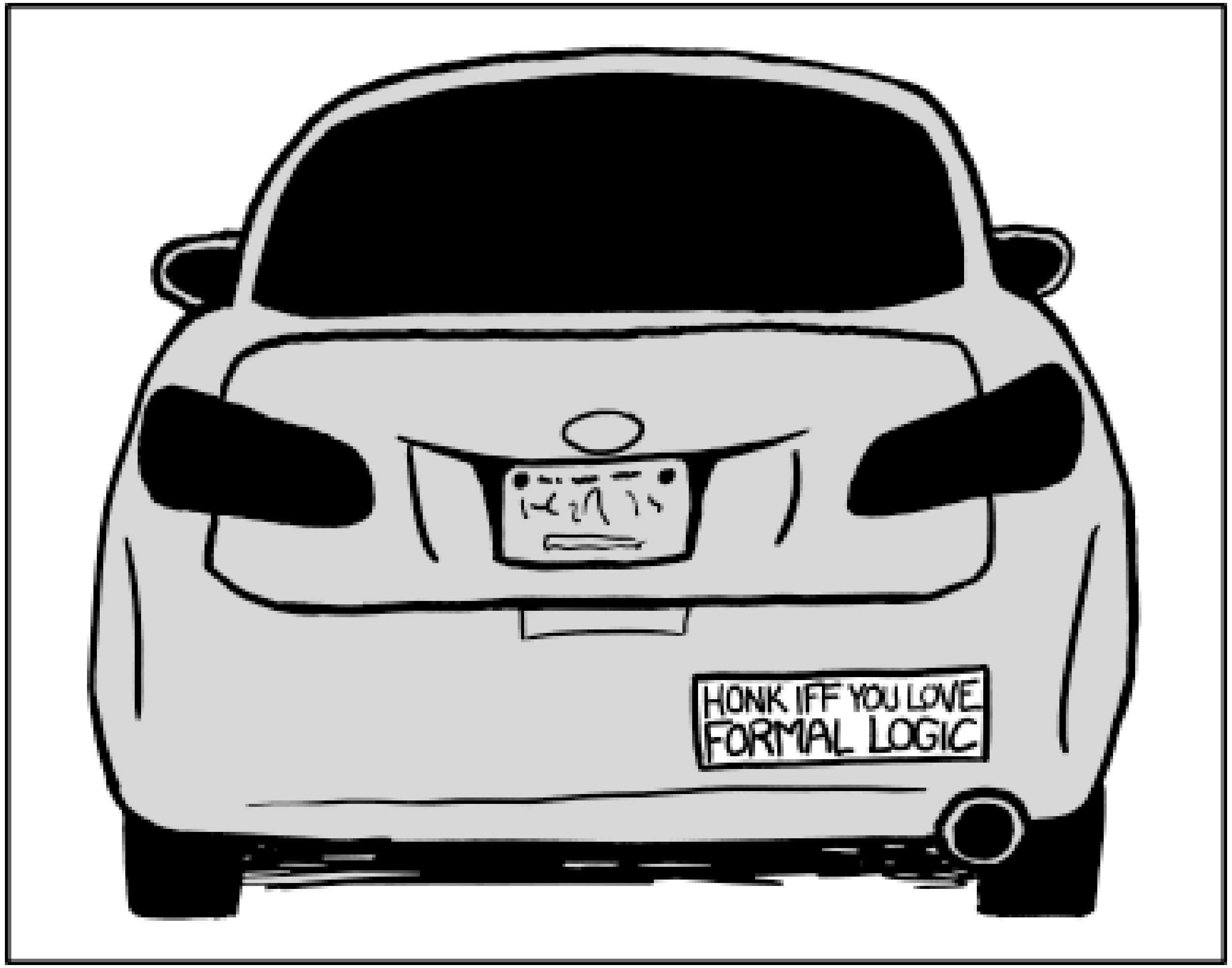
$$x = f(y) \wedge x \neq y \wedge y \neq u \wedge y = f(u) \wedge z \neq f(u) \wedge$$
$$u = v \wedge v = z \wedge v = f(y) \wedge v \neq f(z) \wedge f(x) \neq f(z)$$

Use the *Congruence Closure* algorithm to determine whether this formula is satisfiable.

$$\{x, f(y)\}, \{y, f(u)\}, \{u, \underline{v}\}, \{\underline{v}, z\}, \{\underline{v}, f(y)\}, \{f(x)\}, \{f(z)\}$$
$$\{x, \underline{f(y)}\}, \{y, f(u)\}, \{u, v, z, v, \underline{f(y)}, \{f(x)\}, \{f(z)\}\}$$
$$\{\underline{x}, f(y), u, v, \underline{z}, v\}, \{y, f(u)\}, \{\underline{f(x)}\}, \{\underline{f(z)}\}$$
$$\{x, f(y), \underline{u}, v, \underline{z}, v\}, \{y, \underline{f(u)}\}, \{f(x), \underline{f(z)}\}$$
$$\{x, f(y), u, v, z, v\}, \{y, f(u)\}, \{f(x), f(z)\}$$

Checking the disequality $f(x) \neq f(z)$ leads to the result that the assignment is UNSAT since $f(x)$ and $f(z)$ are in the same congruence class.

Thank You



https://xkcd.com/1033/