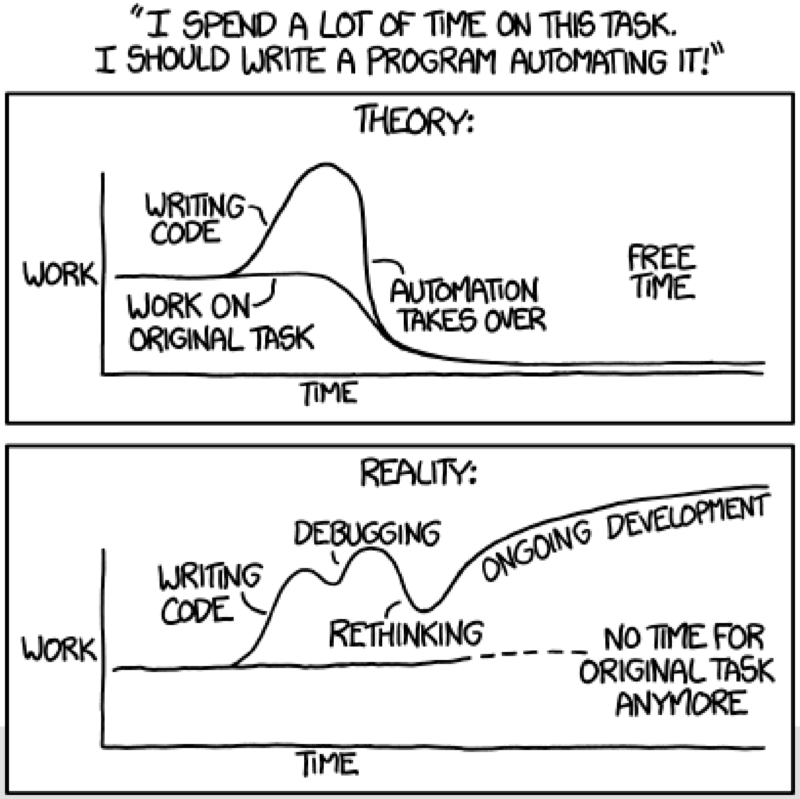
Theories in Predicate Logic

and Satisfiability Modulo Theories

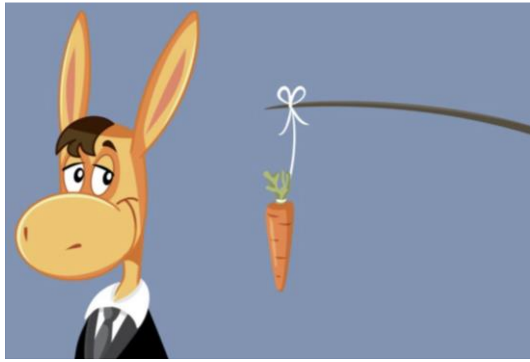
Bettina Könighofer
bettina.koenighofer@iaik.tugraz.at

Stefan Pranger
stefan.pranger@iaik.tugraz.at



https://xkcd.com/2323/

Motivation – Satisfiability Modulo Theory



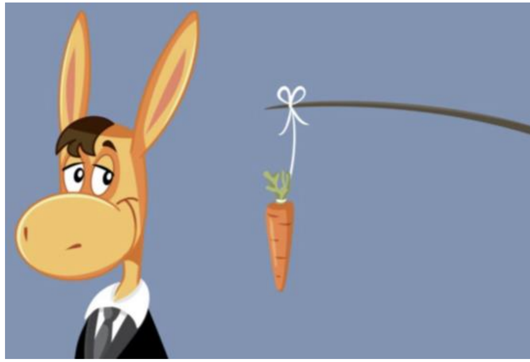
- We want solve formulas
  - over Real Numbers, Integers,…
  - that use functions and predicates like +,-,<,=,>…
  - E.g., $\varphi = x \geq 0 \wedge (x + y \leq 2 \vee x + y \geq 6) \wedge (x + y \geq 1 \vee x - y \geq 4)$

- Solving a formula = Find a **model** that makes the formula true

- The following model makes the formula true:
  - $\mathrm{A} = \{0,1,2,3,4,5,6\}$,
  - $\geq^{\mathrm{M}}, \leq^{\mathrm{M}}$: always return true (e.g., $0 \leq 1, 1 \leq 0, 0 \leq 2, 2 \leq 0, \ldots$)
  - $+^{\mathrm{M}}, -^{\mathrm{M}}$: always return 5 (e.g., $0 + 0 = 5, 0 - 0 = 5, \ldots$)

- **We are typically not interested in such arbitrary models!**

Motivation – Satisfiability Modulo Theory



- Usually we are **not** interested in **arbitrary models**
  - E.g., Models in which $5 + 3 = 10$ or $20 - 2 = 1$
- Only interested in models with **well-established interpretation** of function & predicates
- Theory
  - **Axioms** that define **interpretation/meaning** for **functions** and **predicates**
  - E.g., $1 + 1 = 2, 1 + 2 = 3, 1 + 3 = 4, \ldots$
- **Satisfiable Modulo Theory (SMT)**
  - Deciding 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

- What are Theories?
  - Definition
  - Example Theories: $\mathcal{T}_{\mathrm{E}}$ and $\mathcal{T}_{\mathrm{E}UF}$
- Notations and Concepts
  - $\mathcal{T}$ -Terms, $\mathcal{T}$ -Atoms, $\mathcal{T}$ -Literals and $\mathcal{T}$ -Formulas
  - $\mathcal{T}$-Satisfiability, $\mathcal{T}$-Validity, $\mathcal{T}$-Equivalence
- Implementation of SMT Solvers
  - Eager Encoding
    - explicit encoding of axioms
    - Ackermann & Graph-based reduction
  - Lazy Encoding
    - use combination of theory solvers and SAT solver
    - congruence Closure

Learning Outcomes

After this lecture…

1. students can explain the concept of a theory in first-order logic.
2. students can state the axioms of $\mathcal{T}_E$ and $\mathcal{T}_{UE}$.
3. students can explain the meaning of “Satisfiability Modulo Theories”.
4. students can explain the concept of **eager encoding**.
5. students can solve formulas in $\mathcal{T}_{UE}$ by applying Ackermann’s & Graph-based reduction
6. students can explain the concept of **lazy encoding**.
7. students can solve formulas in the conjunctive fragment of $\mathcal{T}_{UE}$ using Congruence Closure.

Notion of "Theory"

- Theories define axioms often used domains and applications/problems

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**
  - **→ Do not use any non-logical symbols not contained in Σ!**
    - Logical symbols are logical connectives like $\wedge, \vee \cdots$, variables like x, y ..., and quantifiers like $\forall x$

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

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

- $\Sigma_{\text{LIA}} := \mathbb{Z} \cup \{+, -\} \cup \{=, \neq <, \leq, >, \geq\}$
- $\mathcal{A}_{LIA}$: 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: $\varphi := (x = b) \wedge (y \neq x) \rightarrow (w = b)$

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. $\forall x. \forall y. \forall z. (x = y \wedge y = z \rightarrow x = z)$ (transitivity)

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

- 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
  - Example
    - $a \cdot \left(f(b) + f(c)\right) = d \land b \cdot \left(f(a) + f(c)\right) \neq \mathrm{d} \land a = b$
  - Using uninterpreted functions we get:
    - $m\left(a, p\left(f(b), f(c)\right)\right) = d \land m\left(b, p\left(f(a), f(c)\right)\right) \neq d \land a = b$
  - Can be used to show UNSAT of the formula

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

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

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

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

Outline

- What are Theories?
  - Definition
  - Example Theories: $\mathcal{T}_{\mathrm{E}}$ and $\mathcal{T}_{\mathrm{E}UF}$
- Notations and Concepts
  - $\mathcal{T}$ -Terms, $\mathcal{T}$ -Atoms, $\mathcal{T}$ -Literals and $\mathcal{T}$ -Formulas
  - $\mathcal{T}$-Satisfiability, $\mathcal{T}$-Validity, $\mathcal{T}$-Equivalence
- Implementation of SMT Solvers
  - Eager Encoding
  - Lazy Encoding

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

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

$\mathcal{T}$-formulas

- $\varphi = x \geq 0 \land \neg(x + y \leq 2 \lor x + y \geq 6) \land (x + y \geq 1 \lor x - y \geq 4)$

- **$\mathcal{T}$-formula:**
  - Predicate logic formula consisting of **$\mathcal{T}$-literals** and logical connectives, and quantifiers.

Models within a Theory

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

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

A model M within a theory $\mathcal{T}$ is therefore an assignment of all free variables to a constant in Σ.

Models within a Theory

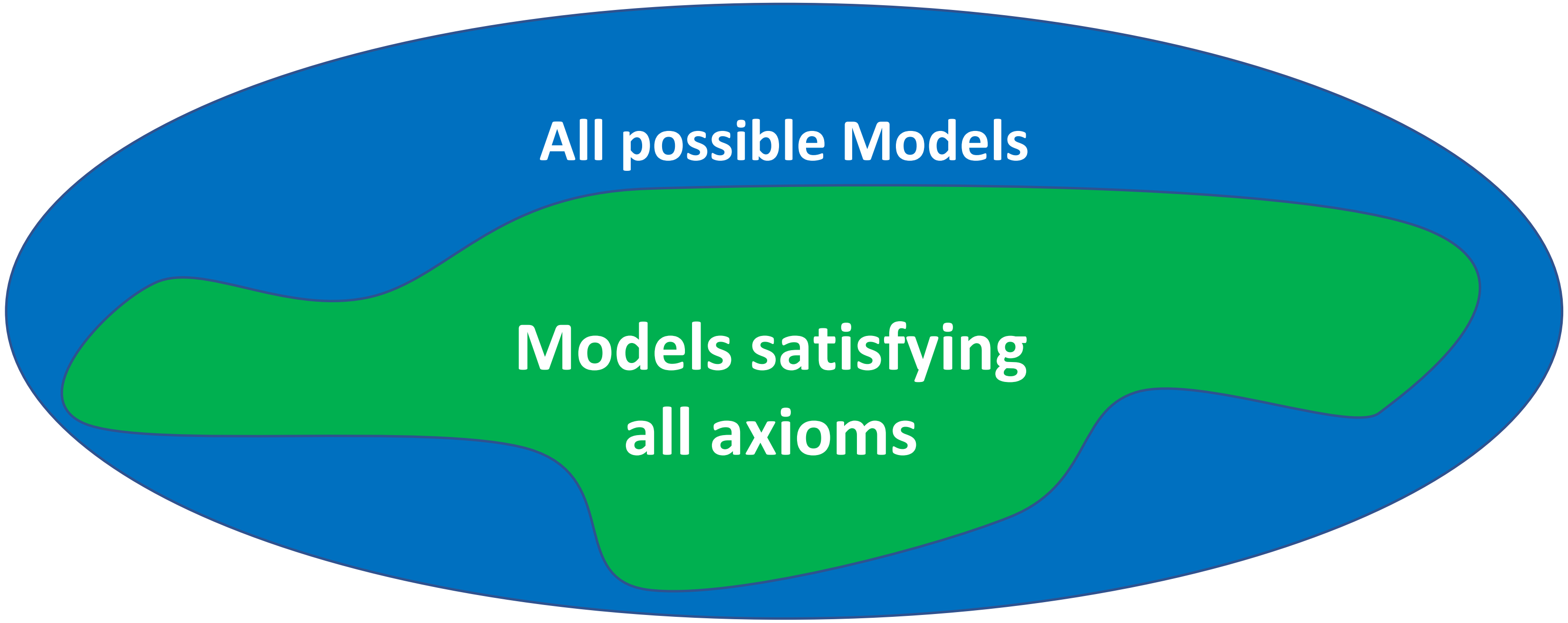
- Example: consider the formula φ in $\mathcal{T}_{\mathbf{LIA}}$
  - $\varphi \ := \ (x + y > 0) \land (x = 0)$

- Give a model for φ in $\mathcal{T}_{\mathbf{LIA}}$?

  - E.g., $M_0 = \{x \to 5, y \to 1\}$
  - $\mathrm{M}_1 = \{x \to 0, y \to 1\}$

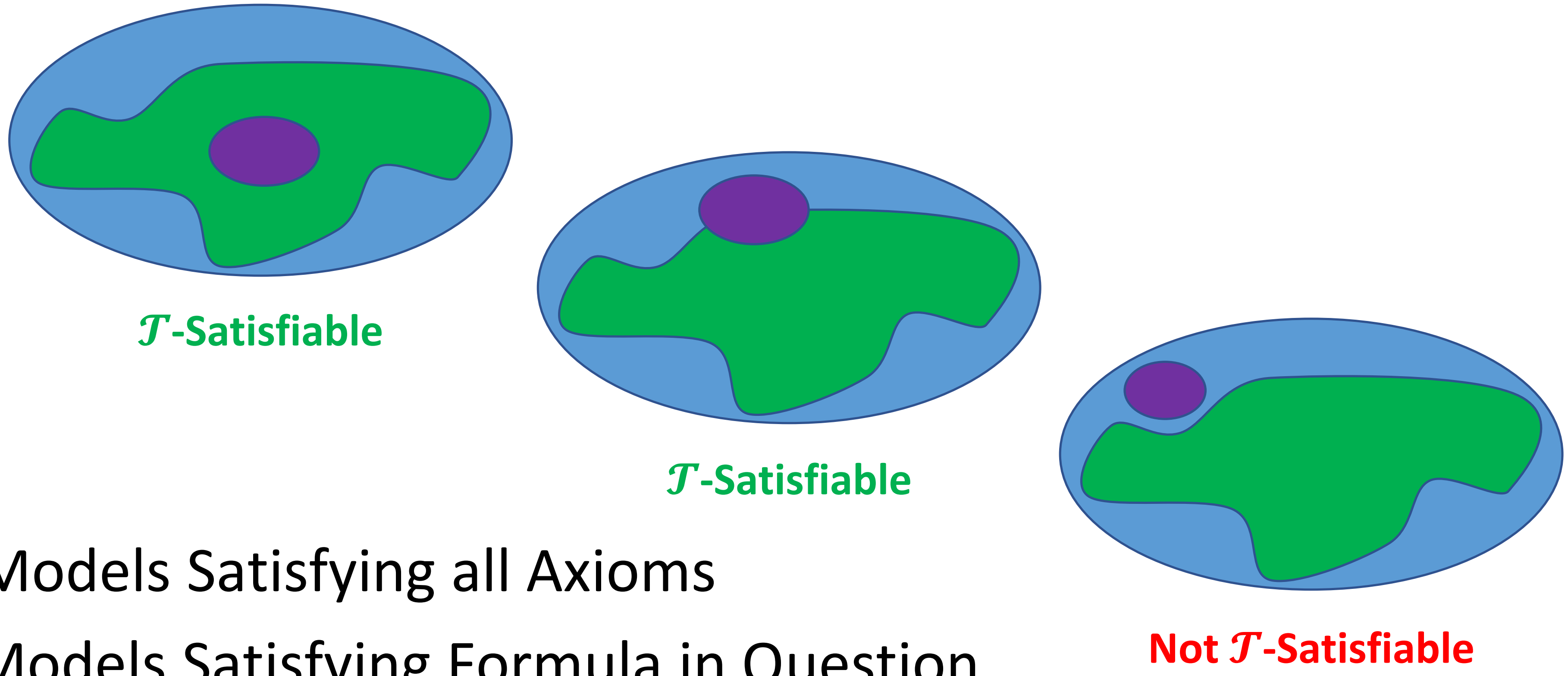
$\mathcal{T}$-Satisfiability, $\mathcal{T}$-validity, $\mathcal{T}$-Equivalence

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



$\mathcal{T}$-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.



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

Models within a Theory

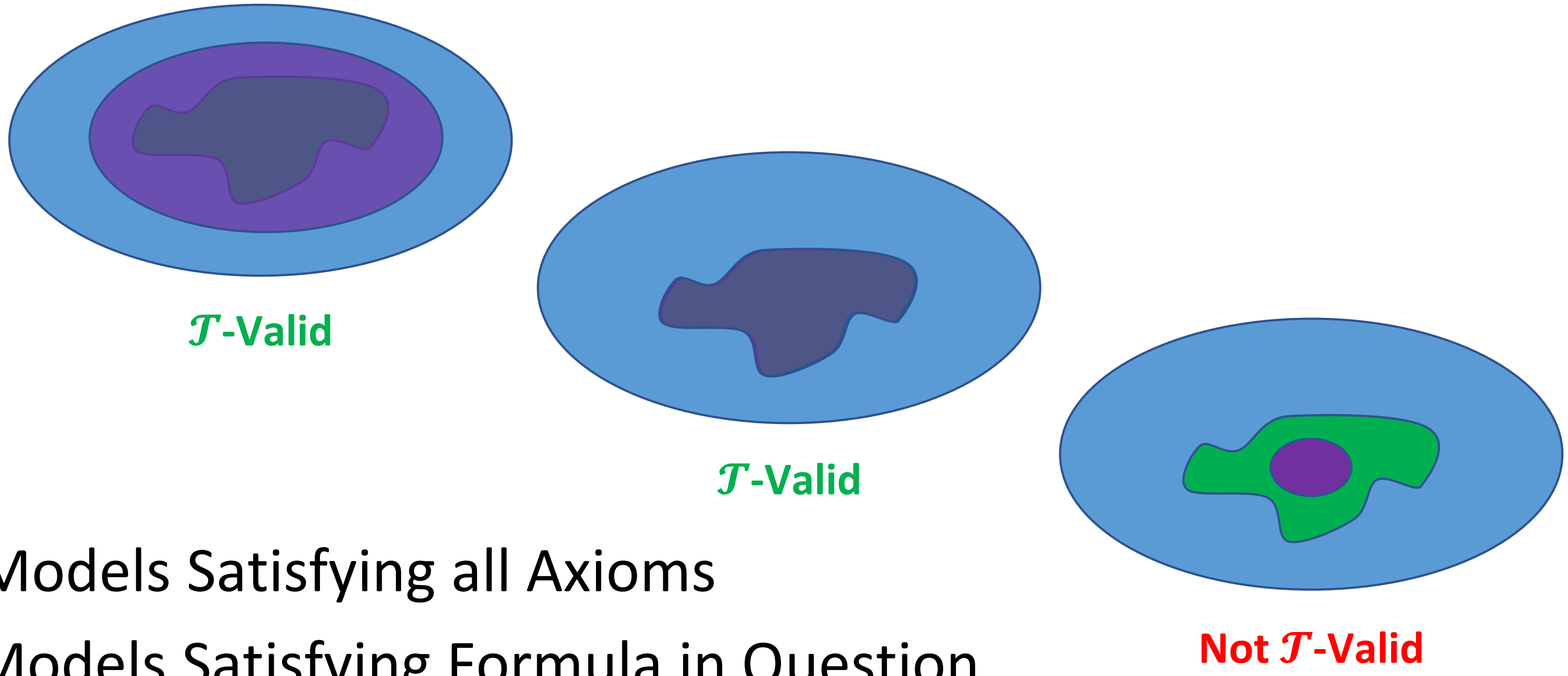
- Example: consider the formula φ in $\mathcal{T}_{\mathbf{LIA}}$
  - $\varphi \;:=\; (x + y > 0) \wedge (x = 0)$

- Give a satisfying and a falsifying model for φ in $\mathcal{T}_{\mathbf{LIA}}$?

- Falsifying model: $M_f = \{x \rightarrow 5, y \rightarrow 1\}$
- Satisfying model: $M_s = \{x \rightarrow 0, y \rightarrow 1\}$

$\mathcal{T}$-Validity

A formula φ is $\mathcal{T}$-valid, if and only if all models within $\mathcal{T}$ satisfy φ.



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

$\mathcal{T}$-Equivalence

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

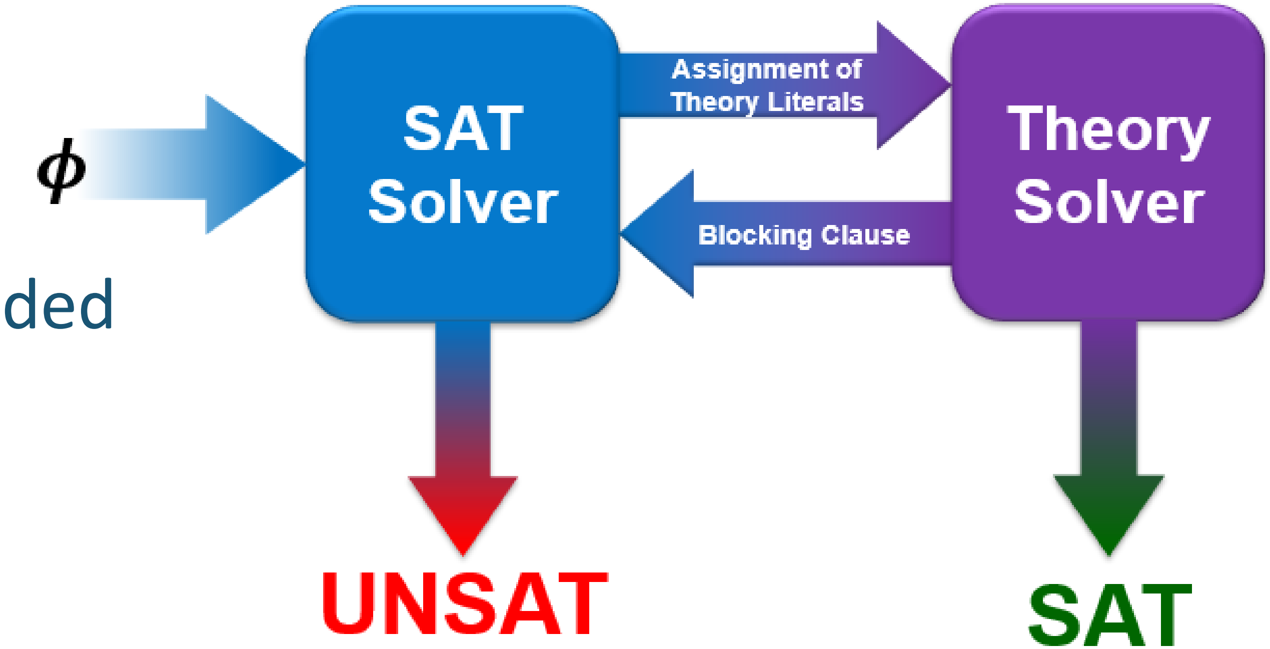
Two formulas φ and ψ are $\mathcal{T}$-equivalent, if and only if they evaluate to ture for the exact same models in $\mathcal{T}$.

Outline

- What are Theories?
  - Definition
  - Example Theories: $\mathcal{T}_{\mathrm{E}}$ and $\mathcal{T}_{\mathrm{E}UF}$
- Notations and Concepts
  - $\mathcal{T}$ -Terms, $\mathcal{T}$ -Atoms, $\mathcal{T}$ -Literals and $\mathcal{T}$ -Formulas
  - $\mathcal{T}$-Satisfiability, $\mathcal{T}$-Validity, $\mathcal{T}$-Equivalence
- Implementation of SMT Solvers
  - Eager Encoding
  - Lazy Encoding

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 $\mathcal{T}_{EUF}$

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

Eliminate Function Applications — Ackermann's Reduction

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

Eliminate Equality Applications — Graph-based Reduction

equisatisfiable propositional formula

Theory Formula ϕ^T

equisatisfiable

$\mathcal{A} \wedge \phi$

Ackermann's Reduction

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

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

Example of Ackermann's Reduction

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

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

2. $f : a, b, c$

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

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

Example of Ackermann's Reduction

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

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

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

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

Example of Ackermann's Reduction

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

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

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

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

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

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

↓ **Ackermann's Reduction** ✓

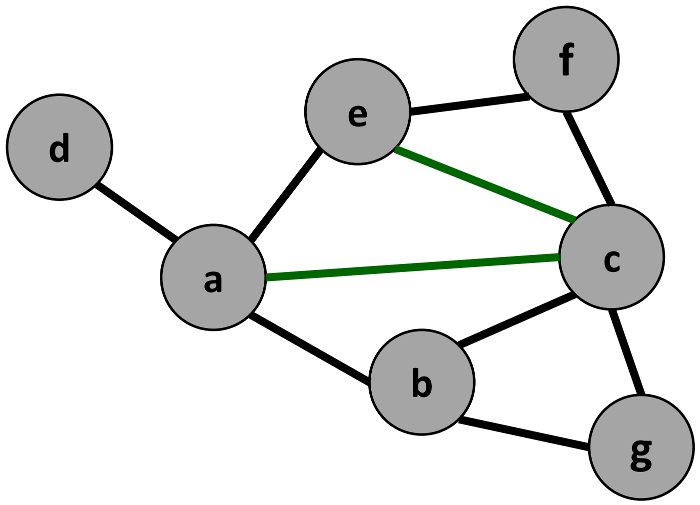
equisatisfiable quant.-free $\mathcal{T}_{E}$-formula

↓ **Graph-based Reduction**

equisatisfiable propositional formula

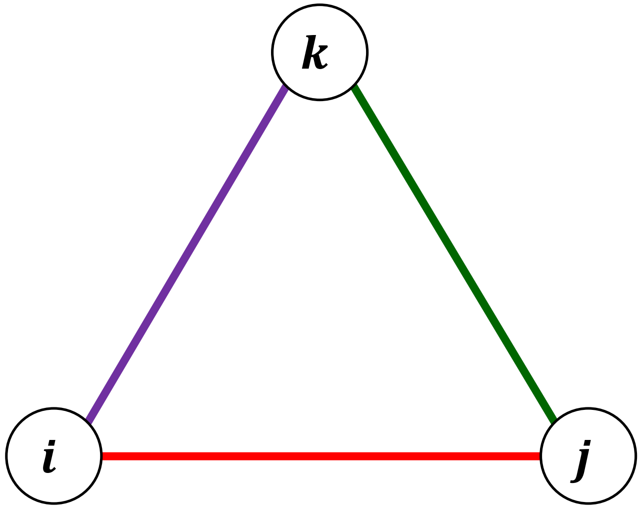
Graph-Based Reduction

- Step 1: Draw a non-polar equality graph
  - Node per variable
  - Edge per (dis)equality
- Step 2: Make graph chordal
  - No cycles size > 3



Graph-Based Reduction

- Step 3: Introduce fresh propositional variable per equation
  - $a = b \rightsquigarrow e_{a=b}$
  - Order! (To ensure symmetry)
    $b = a \rightsquigarrow e_{a=b}$
- Step 4: For each triangle (i, j, k):
  - Add transitivity constraints
    $$\left(e_{i=j} \wedge e_{j=k} \rightarrow e_{i=k}\right) \wedge$$
    $$\left(e_{i=j} \wedge e_{i=k} \rightarrow e_{j=k}\right) \wedge$$
    $$\left(e_{i=k} \wedge e_{j=k} \rightarrow e_{i=j}\right)$$

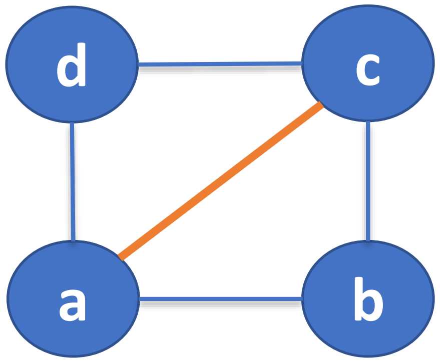


➔ SAT Solver

- Step 5: $\phi_{prop} = \phi_{TC} \wedge \hat{\phi}_E$

Example 1. Graph-Based Reduction

$$\phi_E \coloneqq a = b \wedge b = c \wedge c = d \wedge d \neq a$$



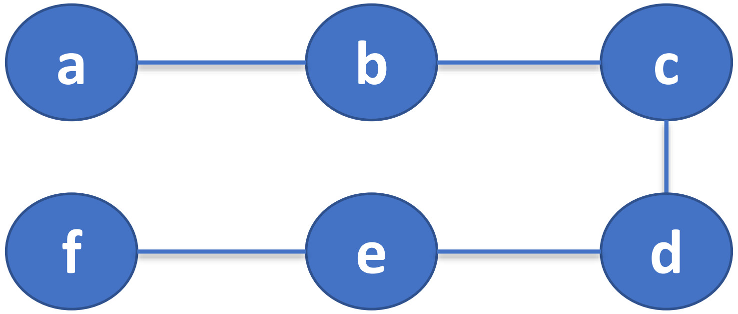
$$\begin{aligned}\phi_{TC} \coloneqq\ & (e_{ab} \wedge e_{bc} \rightarrow e_{ac}) \wedge \\ & (e_{bc} \wedge e_{ac} \rightarrow e_{ab}) \wedge \\ & (e_{ab} \wedge e_{ac} \rightarrow e_{bc}) \wedge \\ & (e_{ac} \wedge e_{cd} \rightarrow e_{ad}) \wedge \\ & (e_{cd} \wedge e_{ad} \rightarrow e_{ac}) \wedge \\ & (e_{ad} \wedge e_{ac} \rightarrow e_{cd})\end{aligned}$$

$$\hat{\phi}_E \coloneqq e_{ab} \wedge e_{bc} \wedge e_{cd} \wedge \neg e_{ad}$$

$$\phi_{prop} \coloneqq \phi_{TC} \wedge \phi_E$$

Example 3. Graph-Based Reduction

$$\phi_E \coloneqq a = b \land b \neq c \to \neg\,(c \neq d \lor d = e \land e = f)$$



$$\phi_{prop} \coloneqq e_{ab} \land \neg e_{bc} \to \neg\,(\neg e_{cd} \lor e_{de} \land e_{ef})$$

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** ✓

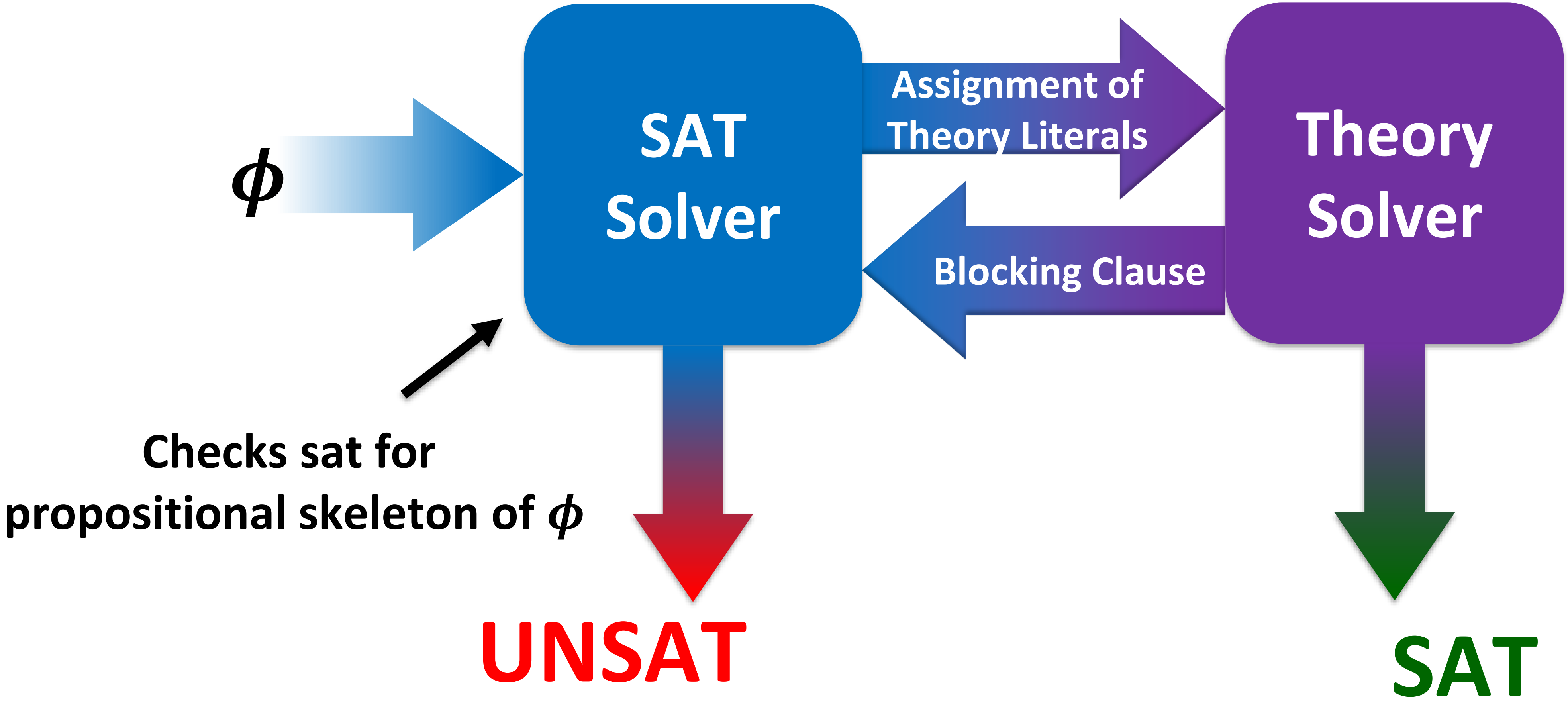
equisatisfiable propositional formula

→ **SAT Solver**

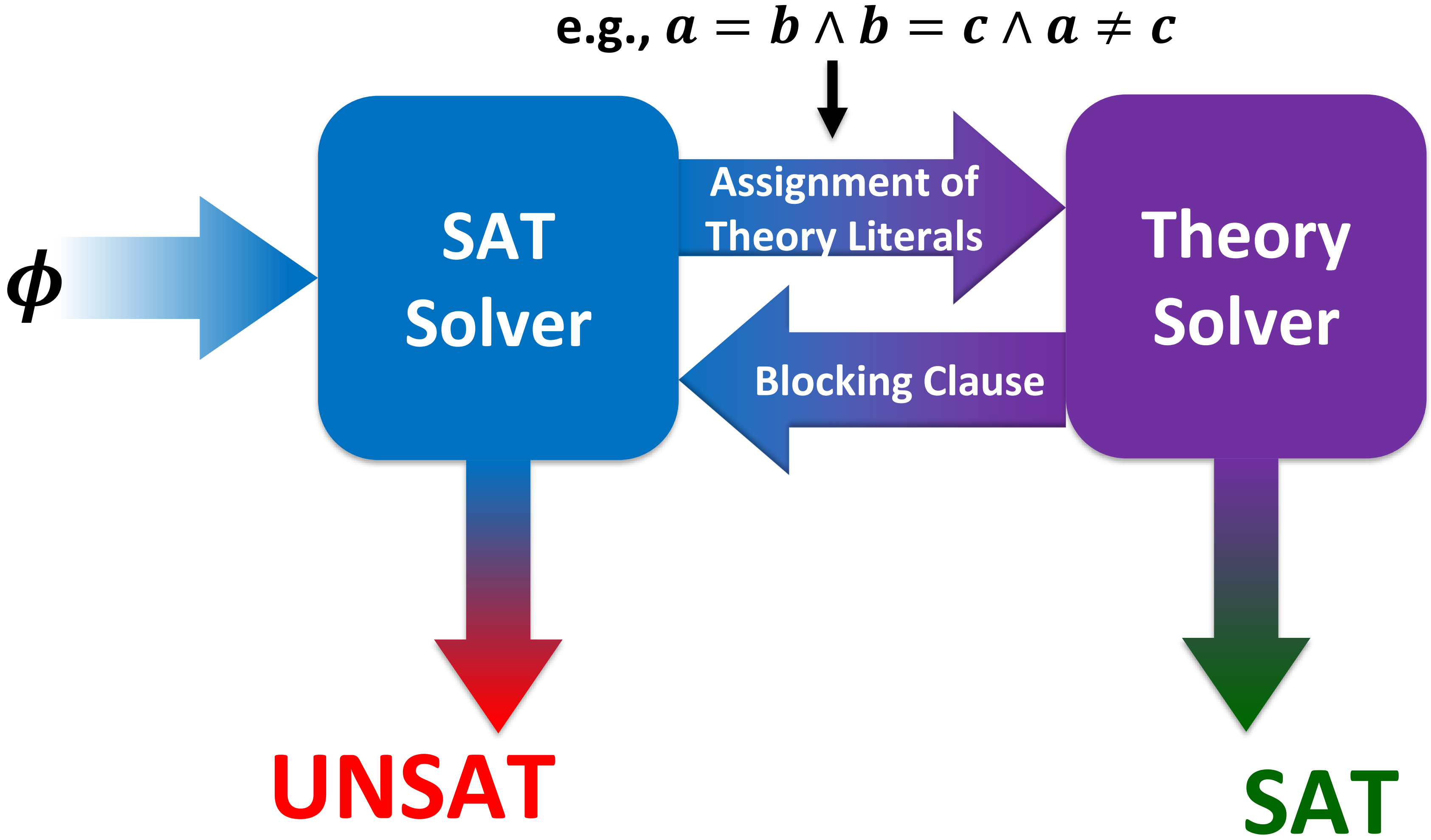
Outline

- What are Theories?
  - Definition
  - Example Theories: $\mathcal{T}_{\mathrm{E}}$ and $\mathcal{T}_{\mathrm{E}UF}$
- Notations and Concepts
  - $\mathcal{T}$ -Terms, $\mathcal{T}$ -Atoms, $\mathcal{T}$ -Literals and $\mathcal{T}$ -Formulas
  - $\mathcal{T}$-Satisfiability, $\mathcal{T}$-Validity, $\mathcal{T}$-Equivalence
- Implementation of SMT Solvers
  - Eager Encoding
  - Lazy Encoding

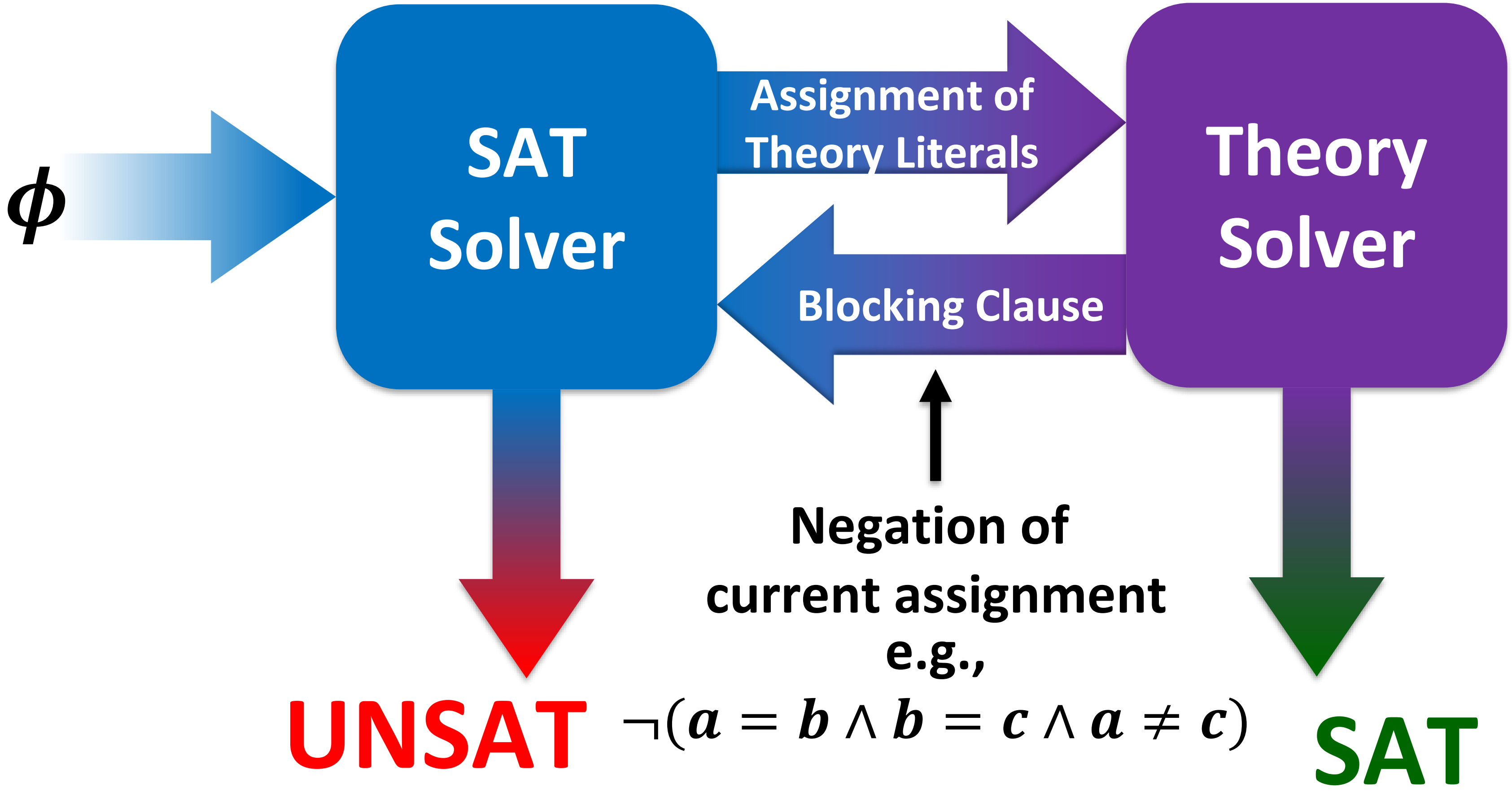
Lazy Encoding



Lazy Encoding



Lazy Encoding



Theory Solver for $\mathcal{T}_{UE}$

- 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 $\mathbf{f}(t_i), \mathbf{f}(t_j)$ (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 1. CC-Algorithm

- $\varphi \coloneqq x_1 = x_2 \wedge x_2 = x_3 \wedge x_4 = x_5 \wedge x_5 \neq x_1 \wedge f(x_1) \neq f(x_3)$

- $\{x_1, x_2\}, \ \{x_2, x_3\}, \ \{x_4, x_5\}, \ \{f(x_1)\}, \ \{f(x_3)\}$
- $\{x_1, x_2, x_3\}, \ \{x_4, x_5\}, \ \{f(x_1)\}, \ \{f(x_3)\}$
- $\{x_1, x_2, x_3\}, \ \{x_4, x_5\}, \ \{f(x_1), f(x_3)\}$

- Check: $f(x_1) \neq f(x_3)$: both are in the same class →
  φ is $\mathcal{T}_{EUF}$-UNSAT

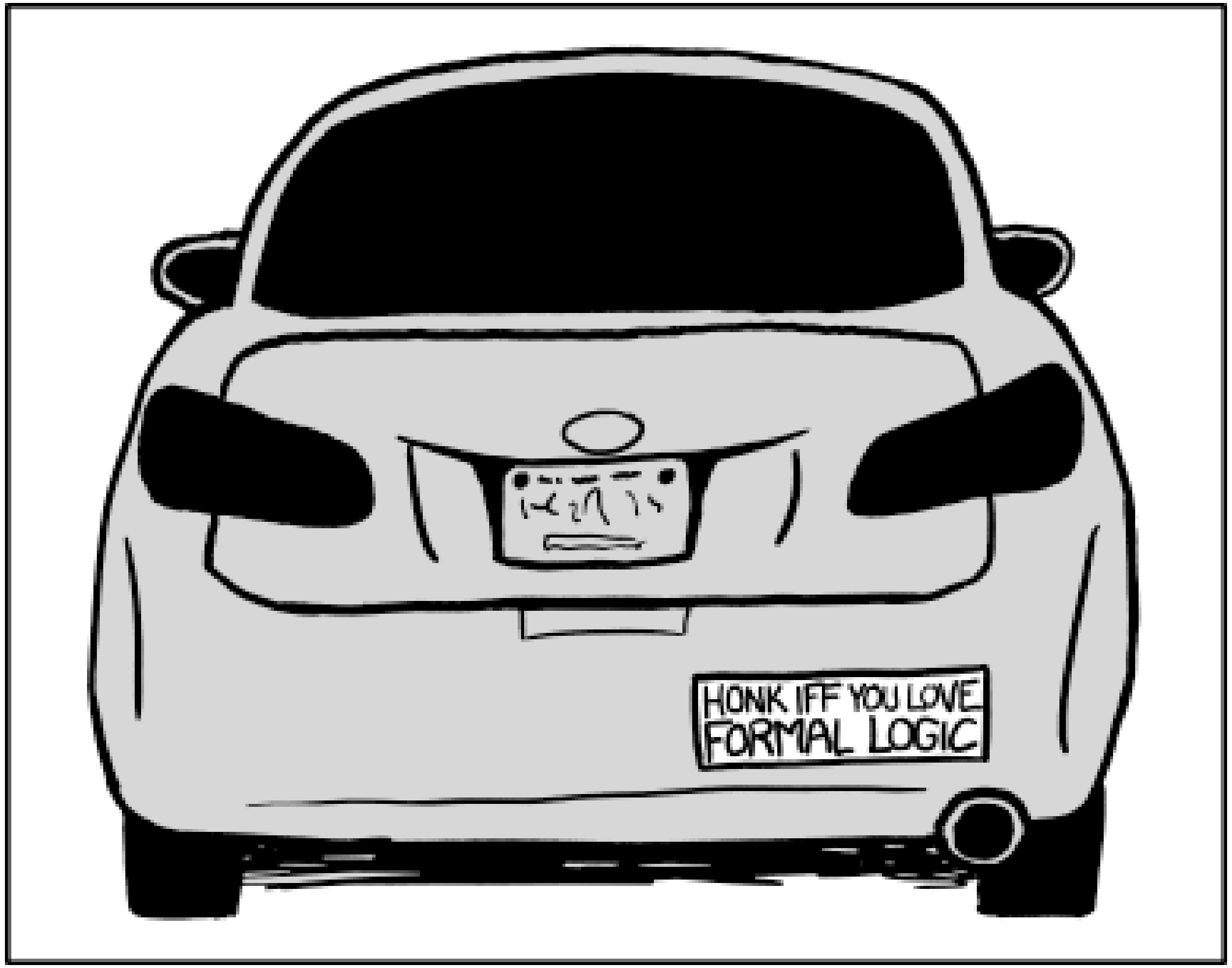
Example 2. CC-Algorithm

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

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

- Check: $f(x) \neq f(z)$ both are in the same class → φ is $\mathcal{T}_{EUF}$-UNSAT

Thank You



https://xkcd.com/1033/