Questionnaire “Logic and Computability”
Summer Term 2023

Contents

9 Satisfiability Modulo Theories

9.1 Definitions and Notations

9.1.1 Give the definition of a theory of formulas in first-order logic.

Solution

A theory is as a pair $(\Sigma; \mathcal{A})$ where Σ is a signature which defines a set of constant, function, and predicate symbols. The set of axioms $\mathcal{A}$ is a set of closed predicate logic formulas in which only constant, function, and predicate symbols of Σ appear.

9.1.2 Explain the concept of a theory in first-order logic using the *theory of Linear Integer Arithmetic* $\mathcal{T}_{LIA}$ as example.

Solution

Variables in $\mathcal{T}_{LIA}$ are of integer sort ($\mathbb{Z}$). The functions of $\mathcal{T}_{LIA}$ are $+$ and $-$ and the predicates are $=, \neq, <, >, \leq$, and $\geq$. The axioms withing $\mathcal{T}_{LIA}$ define the meaning for these functions and predicates.
Therefore, for the theory of Linear Integer Arithmetic $\mathcal{T}_{LIA}$ we have:

- $\Sigma = \mathbb{Z} \cup \{+, -\} \cup \{=, \neq, <, \leq, >, \geq\}$
- $\mathcal{A}$ defines the usual meaning to all symbols:
  - Constant symbols are mapped to the corresponding value in $\mathbb{Z}$.
  - $+$ is interpreted as the function $0+0 \to 0, 0+1 \to 1, \ldots$. $-$ follows it analogous interpretation.
  - The predicate symbols are interpreted as their respective comparison operator.

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

Solution

The satisfiability modulo theories (SMT) problem refers to the problem of determining whether a formula in predicate logic is satisfiable with respect to some theory. A theory fixes the interpretation/meaning of certain predicate and function symbols. Checking whether a formula in predicate logic is satisfiable with respect to a theory means that we are not interested in arbitrary models but in models that interpret the functions and predicates contained in the theory as defined by the axioms in the theory.

9.1.4 Give the definitions of $\mathcal{T}$*-terms*, $\mathcal{T}$*-atoms* and $\mathcal{T}$*-literals* for SMT formulas.

Solution

- $\mathcal{T}$-terms: A $\mathcal{T}$*-term* is either a constant or variables $x, y, \ldots$. An application of a function symbol in Σ where all inputs are $\mathcal{T}$-terms is a $\mathcal{T}$-term.

  Examples for $\mathcal{T}$-terms in $\mathcal{T}_{LIA}$ are: $x + 2$, 5, $x - y$.
- $\mathcal{T}$-atom: A $\mathcal{T}$*-atom* is the application of a predicate symbol in Σ where all inputs are $\mathcal{T}$-terms.

  Examples for $\mathcal{T}$-atoms in $\mathcal{T}_{LIA}$ are: $x + 2 > 0$, $5 \leq 2$, $x - y > 10$.
- $\mathcal{T}$-literal: A $\mathcal{T}$*-literal* is a $\mathcal{T}$-atoms or its negation.

9.1.5 What is the difference between a model of an SMT formula and a model of a predicate logic formula without a theory?

Solution

A model in predicate logic needs to define the domain of the variables and needs to define a concrete meaning to all predicate and function symbols and free variables involved. In SMT, the domain and the interpretation of the predicate and function symbols is fixed. A model for an SMT formula only defines an assignment to all free variables within the formula.

9.1.6 Given the signature $\Sigma_{EUF} \coloneqq \{a, b, c, \dots\} \cup \{f, g, h, \dots\} \cup \{=, P, Q, R, \dots\}$, of the *Theory of Equality and Uninterpreted Functions* $\mathcal{T}_{EUF}$. State the axioms $\mathcal{A}_{EUF}$ of $\mathcal{T}_{EUF}$.

Solution

The axioms $\mathcal{A}_{EUF}$ are the following:

(a) $\forall x.\ \ x = x$ (reflexivity)

(b) $\forall x, y.\ \ x = y \rightarrow y = x$ (symmetry)

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

(d) $\forall \overline{x}, \overline{y}.\ \ (\bigwedge_{i=1}^{n} x_i = y_i) \rightarrow f(\overline{x}) = f(\overline{y})$ (congruence)

(e) $\forall \overline{x}, \overline{y}.\ \ (\bigwedge_{i=1}^{n} x_i = y_i) \rightarrow (P(\overline{x}) \leftrightarrow P(\overline{y}))$ (equivalence)

9.1.7 Explain the concepts of eager encoding and lazy encoding in the context of solving formulas in SMT.

Solution

- In eager encoding, all axioms of the theory are explicitly incorporated into the input formula. The resulting equisatisfiable propositional formula is then given to a SAT solver.
- SMT solvers that use lazy encoding use specialized theory solvers in combination with SAT solvers to decide the satisfiability of formulas within a given theory. In contrast to eager encoding, where a sufficient set of constraints is computed at the beginning, lazy encoding starts with no constraints at all, and lazily adds constraints only when required.

9.1.8 In the following list tick all formulas that are axioms of the theory of equalities and uninterpreted functions $\mathcal{T}_{EUF}$.

- ☐ $\forall x\,(x = x)$
- ☐ $\forall x\,\forall y\,(x = y \vee y = x)$
- ☐ $\forall x\,\forall y\,\forall z\,(x = y \wedge y = z \rightarrow x = z)$
- ☐ $\forall x\,\forall y\,(f(x) = f(y) \rightarrow x = y)$

9.1.9 A first-order theory $\mathcal{T}$ is defined by a signature Σ and a set of axioms $\mathcal{A}$. Consider the *Theory of Equality* $\mathcal{T}_E$. Give its signature Σ_E and its axioms $\mathcal{A}_E$.

Solution

There is no solution available for this question yet.

9.1.10 What is an uninterpreted function? What is the difference between an uninterpreted and an interpreted function? What are the properties of an uninterpreted function?

Solution

There is no solution available for this question yet.

9.1.11 Considering formulas φ and ψ regarding a theory $\mathcal{T}$.

- When is a formula φ $\mathcal{T}$-valid?
- When is a formula φ $\mathcal{T}$-satisfiable?
- When does φ $\mathcal{T}$-entail ψ?

Solution

There is no solution available for this question yet.

9.2 Eager Encoding

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

Solution

The main idea of eager encoding is that the input formula is translated into a propositional formula with all relevant theory-specific information encoded into the formula.

(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 $\varphi_{prop} \coloneqq \hat{\varphi} \wedge \varphi_{cons}$ that corresponds to an equisatisfiable propositional formula to φ.

9.2.2 Explain the specific translations used in *eager encoding* to decide formulas in the theory of equality and uninterpreted functions.

Solution

The translations used in the eager approach for $\mathcal{T}_{EUF}$ are:

(a) Ackermann Reduction: to remove all function instances, resulting in an equisatisfiable formula in $\mathcal{T}_E$.

(b) Graph-Based Reduction: to remove all equality instances, resulting in an equisatisfiable formula in propositional logic.

9.2.3 Given the formula

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

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

Solution

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

$$\hat{\varphi}_{EUF} \quad := \quad f_x = f_y \ \vee \ (z = y \wedge z \neq f_z)$$

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

9.2.4 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 to compute an equisatisfiable formula in $\mathcal{T}_E$.

Solution

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

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

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

9.2.5 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 to compute an equisatisfiable formula in $\mathcal{T}_E$.

Solution

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

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

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

9.2.6 Perform the graph-based reduction to translate the following formula in $\mathcal{T}_E$ into an equisatisfiable formula in propositional logic.

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

Solution

We choose:

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

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

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

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

9.2.7 Perform the graph-based reduction to translate the following formula in $\mathcal{T}_E$ into an equisatisfiable formula in propositional logic.

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

Solution

We choose:

- Triangle 1: a-b-c
- Triangle 2: a-c-d
- Triangle 3: c-d-e

$$
\begin{aligned}
\varphi_{TC} \coloneqq &(e_{a=b} \wedge e_{b=c} \rightarrow e_{a=c})\wedge \\
&(e_{a=b} \wedge e_{a=c} \rightarrow e_{b=c})\wedge \\
&(e_{b=c} \wedge e_{a=c} \rightarrow e_{a=b})\wedge \\
\\
&(e_{a=c} \wedge e_{c=d} \rightarrow e_{a=d})\wedge \\
&(e_{a=c} \wedge e_{a=d} \rightarrow e_{c=d})\wedge \\
&(e_{c=d} \wedge e_{a=d} \rightarrow e_{a=c})\wedge \\
\\
&(e_{c=e} \wedge e_{c=d} \rightarrow e_{d=e})\wedge \\
&(e_{c=e} \wedge e_{d=e} \rightarrow e_{c=d})\wedge \\
&(e_{c=d} \wedge e_{d=e} \rightarrow e_{c=e})
\end{aligned}
$$

$$\hat{\varphi}_E \coloneqq (e_{a=b} \ \vee \ e_{a=d} \rightarrow (e_{b=c} \ \wedge \ \neg e_{c=e} \ \wedge \ \neg e_{e=d})$$

$$\varphi_{prop} \coloneqq \varphi_{TC} \wedge \hat{\varphi}_E$$

9.2.8 Given the formula

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

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

Solution

$$\hat{\varphi}_{EUF} \quad := \quad f_x = y \wedge x = g_x \vee x \neq f_x \wedge g_x = f_{g_x} \vee y \neq g_x \wedge x = f_y \wedge g_y = f_{g_x}$$

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

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

9.2.9 Given the formula

$$\varphi_{EUF} := f(a,b) = x \wedge f(x,y) \neq g(a) \vee f(m,n) = b \vee f(g(a),y) \neq a$$

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

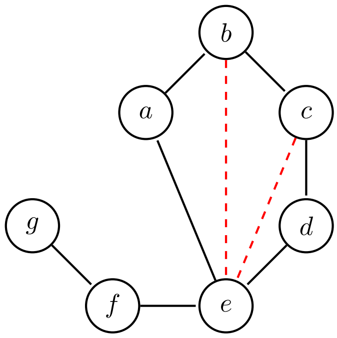
Solution

There is no solution available for this question yet.

9.2.10 Perform the graph-based reduction to translate the following formula in $\mathcal{T}_E$ into an equisatisfiable formula in propositional logic.

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

Solution



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

$$\hat{\varphi}_E \coloneqq \neg e_{a=b} \wedge e_{b=c} \vee e_{c=d} \rightarrow \neg(\neg e_{d=e} \vee e_{e=f}) \wedge \neg(e_{f=g} \wedge \neg e_{a=e})$$

$$\varphi_{prop} \coloneqq \varphi_{TC} \wedge \hat{\varphi}_E$$

9.2.11 In the following list tick all statements that conform to the eager encoding approach for the implementation of SMT solver.

- ☐ Eager encoding is based on the interaction between a SAT solver and a so-called theory solver.
- ☐ Eager encoding involves translating the original formula to an equisatisfiable boolean formula in a single step.

☐ Eager encoding is based on the direct encoding of axioms.

☐ Eager encoding starts with no constraints at all and adds constraints only when needed.

9.2.12 Given the formula

$$\varphi_{EUF} \quad := \quad f(x,y) = g(x) \rightarrow [f(g(y),z) = x \vee \neg(g(z) = y)]\,.$$

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

Solution

There is no solution available for this question yet.

9.2.13 Given the formula

$$\varphi_{EUF} \quad := \quad f(g(x),h(y)) = a \ \vee \ b = f(u,v) \ \rightarrow \ k(a,b) = u \wedge v = k(x,y)$$

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

Solution

There is no solution available for this question yet.

9.2.14 When applying eager encoding to decide the satisfiability of a formula in $\mathcal{T}_{EUF}$, explain how reflexivity, symmetry and transitivity are handled within the graph-based reduction.

Solution

There is no solution available for this question yet.

9.2.15 Perform the graph-based reduction to translate the following formula in $\mathcal{T}_E$ into an equisatisfiable formula in propositional logic.

$$\varphi_{EUF} \quad := \quad x \neq y \wedge y = g_x \vee g_x = g_y \rightarrow \neg(g_y \neq z \vee z = f_x) \wedge \neg(f_x = f_y \wedge x \neq z)$$

Solution

There is no solution available for this question yet.

9.2.16 Consider the following formula in $\mathcal{T}_{EUF}$.

$$\begin{aligned}\varphi_{EUF} \quad := \quad & f(x) = f(y) \wedge f(y) = y \vee f(g(x)) = f(f(y)) \wedge g(x) = x \\ & \vee f(x) \neq f(y) \wedge y \neq g(f(y)) \wedge x \neq g(x)\end{aligned}$$

- Use Ackermann's reduction to compute an equisatisfiable formula in $\mathcal{T}_E$.
- Then perform the graph-based reduction on the outcome of Ackermann's reduction to construct an equisatisfiable propositional formula φ_{prop}.

Solution

There is no solution available for this question yet.

9.2.17 Given the formula

$$f(x) = g(x) \vee z = f(y) \rightarrow f(z) \neq g(y) \wedge x = z$$

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

Solution

There is no solution available for this question yet.

9.2.18 Given the formula

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

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

Solution

There is no solution available for this question yet.

9.2.19 Given the formula

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

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

Solution

There is no solution available for this question yet.

9.2.20 Perform the graph-based reduction to translate the following formula in $\mathcal{T}_E$ into an equisatisfiable formula in propositional logic.

$$\varphi_E := x \neq y \wedge y = c \vee c = d \rightarrow \neg(d \neq z \vee z = a) \wedge \neg(a = b \wedge x \neq z).$$

Solution

There is no solution available for this question yet.

9.2.21 Consider the following formula in $\mathcal{T}_{EUF}$.

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

- Use Ackermann's reduction to compute an equisatisfiable formula in $\mathcal{T}_E$.
- Then perform the graph-based reduction on the outcome of Ackermann's reduction to construct an equisatisfiable propositional formula φ_{prop}.

Solution

There is no solution available for this question yet.

9.3 Lazy Encoding

9.3.1 Give the definition of the propsitional skeleton of a formula φ in a given theory $\mathcal{T}$. Give an example for a formula φ in $\mathcal{T}_{LIA}$ and its corresponding propositional skeleton skel(φ).

Solution

The propositional skeleton skel(φ) of a formula φ is obtained by replacing each occurance of a $\mathcal{T}$-literal with a propositional variable.
An example for a formula φ in $\mathcal{T}_{LIA}$:

$$\varphi := (x > y) \lor (x > z),$$

and the corresponding skeleton skel(φ):

$$e_1 \lor e_2,$$

where $e_1 \equiv x > y$ and $e_2 \equiv x > z$.

9.3.2 Explain the concept of lazy encoding to decide satisfiability of formulas in a first-order theory.

Solution

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 $\mathcal{T}$-satisfiable. Otherwise, a blocking clause is generated and the SAT solver searches for a new assignment. This is repeated until either a $\mathcal{T}$-consistent assignment is found, or the SAT solver cannot find any more assignments.
See figure in lecture notes on page 11.

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

$$\begin{aligned} \varphi_{EUF} \quad := \quad & x = f(y) \land x \neq y \land y \neq u \land y = f(u) \land z \neq f(u) \land \\ & u = v \land v = z \land v = f(y) \land v \neq f(z) \land f(x) \neq f(z) \end{aligned}$$

Use the congruence closure algorithm to determine whether this formula is satisfiable.

Solution

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

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.

9.3.4 In the following list tick all statements that conform to the lazy encoding approach for the implementation of SMT solver.

☐ Lazy encoding is based on the interaction between a SAT solver and a so-called theory solver.

☐ Lazy encoding involves translating the original formula to an equisatisfiable Boolean formula in a single step.

☐ Lazy encoding is based on the direct encoding of axioms.

☐ Lazy encoding starts with no constraints at all and adds constraints only when needed.

9.3.5 To decide SMT formulas, the lazy approach uses a theory solver in combination with a SAT solver. Explain what a theory solver is. Explain what the inputs and outputs of a theory solver are and how it is used within the lazy encoding approach.

Solution

There is no solution available for this question yet.

9.3.6 In the following list, mark all items that are true for an *eager encoding* procedure for $\mathcal{T}_{UE}$ with **E**, mark all items that are true for a *lazy encoding* procedure with **L**, and mark all items which neither belong to an eager nor a lazy encoding procedure with **N**.

☐ Only one call to a propositional SAT solver is required.

☐ A propositional formula that is equisatisfiable to the original theory formula is constructed before calling any solver.

☐ A propositional SAT solver and a theory solver for the conjunctive fragment of the theory interact with each other.

☐ For a theory-inconsistent assignment of literals, a blocking clause is created.

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

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

Use the congruence closure algorithm to determine whether this formula is satisfiable.

Solution

There is no solution available for this question yet.

9.3.8 What does the congruence closure algorithm compute? State the inputs and output of the algorithm.

In the context of deciding satisfiability of formulas in $\mathcal{T}_{EUF}$, what is the congruence closure algorithm used for?

Solution

There is no solution available for this question yet.

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

$$\varphi_{EUF} \quad := \quad f(a) = c \land f(c) \neq f(d) \land b = f(c) \land a \neq f(c) \land c = d \land b \neq d \land a = c$$

Use the congruence closure algorithm to determine whether this formula is satisfiable.

Solution

There is no solution available for this question yet.

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

$$\varphi_{EUF} \quad := \quad a = b \wedge c \neq d \wedge f(a) = c \wedge f(b) \neq f(c) \wedge f(a) = f(d) \wedge f(b) = c \wedge f(d) = f(c)$$

Use the congruence closure algorithm to determine whether this formula is satisfiable.

Solution

There is no solution available for this question yet.

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

$$\begin{aligned} f(b) = a \wedge c \neq d \wedge f(e) = b \wedge d \neq f(b) \wedge f(a) = f(e) \wedge \\ b \neq f(b) \wedge a \neq e \wedge f(a) = e \wedge a = c \wedge f(b) \neq e \wedge d = f(c) \end{aligned}$$

Use the congruence closure algorithm to determine whether this formula is satisfiable.

Solution

There is no solution available for this question yet.

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

$$\varphi_{EUF} \quad := \quad f(b) = a \wedge e = b \wedge c = f(c) \wedge d \neq f(e) \wedge f(a) = f(d) \wedge a \neq f(c) \wedge d = f(a)$$

Use the congruence closure algorithm to determine whether this formula is satisfiable.

Solution

There is no solution available for this question yet.

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

$$\begin{aligned} \varphi_{EUF} \quad := \quad & f(o) = k \wedge l \neq f(m) \wedge n \neq l \wedge f(k) = m \wedge f(o) = f(k) \wedge o \neq k \wedge \\ & l \neq f(n) \wedge f(m) \neq k \wedge m \neq f(m) \wedge o = n \wedge f(m) = o \end{aligned}$$

Use the congruence closure algorithm to determine whether this formula is satisfiable.

Solution

There is no solution available for this question yet.

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

$$\varphi_{EUF} \quad := \quad f(b) = a \wedge e = b \wedge c = f(c) \wedge d \neq f(e) \wedge f(a) = f(d) \wedge a \neq f(c) \wedge d = f(a)$$

Use the congruence closure algorithm to determine whether this formula is satisfiable.

Solution

$$\begin{aligned} &\{f(b), a\}, \{e, b\}, \{c, f(c)\}, \{f(e)\}, \{\underline{f(a)}, f(d)\}, \{d, \underline{f(a)}\} \\ &\{\underline{f(b)}, a\}, \{\underline{e, b}\}, \{c, f(c)\}, \{\underline{f(e)}\}, \{f(a), f(d), d\} \\ &\{f(b), a, f(e)\}, \{e, b\}, \{c, f(c)\}, \{f(a), f(d), d\} \end{aligned}$$

Checking the disequalities $d \neq f(e)$ and $a \neq f(c)$ leads to the result that the assignment is SAT, since neither d and $f(e)$ nor a and $f(c)$ are in the same congruence class.

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

$$\begin{aligned}\varphi_{EUF} \quad := \quad & f(o) = k \wedge l \neq f(m) \wedge n \neq l \wedge f(k) = m \wedge f(o) = f(k) \wedge o \neq k \wedge \\ & l \neq f(n) \wedge f(m) \neq k \wedge m \neq f(m) \wedge o = n \wedge f(m) = o\end{aligned}$$

Use the congruence closure algorithm to determine whether this formula is satisfiable.

Solution

$$\begin{aligned}
&\{k, \underline{f(o)}\}, \{l\}, \{m, f(k)\}, \{f(k), \underline{f(o)}\}, \{f(n)\}, \{n, o\}, \{o, f(m)\} \\
&\{k, f(k), f(o)\}, \{l\}, \{m, f(k)\}, \{f(n)\}, \{n, \underline{o}\}, \{\underline{o}, f(m)\} \\
&\{k, \underline{f(k)}, f(o)\}, \{l\}, \{m, \underline{f(k)}\}, \{f(n)\}, \{n, o, f(m)\} \\
&\{k, m, f(k), \underline{f(o)}\}, \{l\}, \{\underline{f(n)}\}, \{\underline{n, o}, f(m)\} \\
&\{\underline{k, m}, \underline{f(k)}, f(n), f(o)\}, \{l\}, \{n, o, \underline{f(m)}\} \\
&\{k, m, n, o, f(k), f(m), f(n), f(o)\}, \{l\}
\end{aligned}$$

Checking the disequalities $o \neq k$, $f(m) \neq k$, $m \neq f(m)$ leads to the result that the assignment is UNSAT, since o and k, $f(m)$ and k, m and $f(m)$ are in the same congruence class.

9.3.16 Use the lazy encoding approach to check whether the formula φ in $\mathcal{T}_{EUF}$ is satisfiable.

$$\varphi := (x = y) \wedge (y = f(y)) \wedge (y \neq f(x)) \wedge (z = f(z)) \wedge (f(z) = f(x))$$

Solution

We start by computing skel(φ):

- $e_0 \Leftrightarrow (x = y)$
- $e_1 \Leftrightarrow (y = f(y))$
- $e_2 \Leftrightarrow (y = f(x))$
- $e_3 \Leftrightarrow (z = f(z))$
- $e_4 \Leftrightarrow (f(z) = f(x))$

$$\text{skel}(\varphi) = e_0 \wedge e_1 \wedge \neg e_2 \wedge e_3 \wedge e_4$$

Step	1	2	3	4	5	6
Decision Level	0	0	0	0	0	0
Assignment	-	e_0	e_0, e_1	$e_0, e_1, \neg e_2$	$e_0, e_1, \neg e_2,$ e_3	$e_0, e_1, \neg e_2,$ e_3, e_4
Cl. 1: e_0	e_0	✓	✓	✓	✓	✓
Cl. 2: e_1	e_1	e_1	✓	✓	✓	✓
Cl. 3: $\neg e_2$	$\neg e_2$	$\neg e_2$	$\neg e_2$	✓	✓	✓
Cl. 4: e_3	e_3	e_3	e_3	e_3	✓	✓
Cl. 5: e_4	e_4	e_4	e_4	e_4	e_4	✓
BCP	e_0	e_1	$\neg e_2$	e_3	e_4	-
PL	-	-	-	-	-	-
Decision	-	-	-	-	-	SAT

The SAT solver has computed that skel(φ) is satisfiable, we are therefore going to check for consistency with the theory:

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

The $\mathcal{T}_{EUF}$-Solver returned SAT, therefore φ is satisfiable.

9.3.17 Use the lazy encoding approach to check whether the formula φ in $\mathcal{T}_{EUF}$ is satisfiable.

$$\varphi := (g(a) = c) \wedge \big((f(g(a)) \neq f(c)) \vee (g(a) = d))\big) \wedge (c \neq d)$$

Solution

We start by computing skel(φ):

- $e_0 \Leftrightarrow (g(a) = c)$
- $e_1 \Leftrightarrow (f(g(a)) = f(c))$
- $e_2 \Leftrightarrow (g(a) = d)$
- $e_3 \Leftrightarrow (c = d)$

$$\text{skel}(\varphi) = e_0 \wedge \neg e_1 \wedge e_2 \wedge \neg e_3$$

Step	1	2	3	4
Decision Level	0	0	0	0
Assignment	-	e_0	$e_0, \neg e_3$	$e_0, \neg e_3, \neg e_1$
Cl. 1: e_0	e_0	✓	✓	✓
Cl. 2: $\neg e_1, e_2$	$\neg e_1, e_2$	$\neg e_1, e_2$	$\neg e_1, e_2$	✓
Cl. 3: $\neg e_3$	$\neg e_3$	$\neg e_3$	✓	✓
BCP	e_0	$\neg e_3$	-	-
PL	-	-	$\neg e_1$	-
Decision	-	-	-	-

A satisfying assignment $e_0 \wedge \neg e_1 \wedge \neg e_3$ has been found and we have to check consistency of $(g(a) = c) \wedge (f(g(a)) \neq f(c)) \wedge (c \neq d)$:

$$\{g(a), c\}, \{f(g(a))\}, \{f(c)\}, \{d\}$$
$$\{g(a), c\}, \{d\}, \{f(c), f(g(a))\}$$

Congruence Closure returns UNSAT because of: $(f(g(a)) \neq f(c))$. We therefore add $\neg e_0 \vee e_1 \vee e_3$ as a blocking clause and continue.

Step	5	6	7	8	9
Decision Level	0	0	0	0	0
Assignment	-	e_0	$e_0, \neg e_3$	$e_0, \neg e_3, e_1$	$e_0, \neg e_3, e_1, e_2$
Cl. 1: e_0	e_0	✓	✓	✓	✓
Cl. 2: $\neg e_1, e_2$	$\neg e_1, e_2$	$\neg e_1, e_2$	$\neg e_1, e_2$	e_2	✓
Cl. 3: $\neg e_3$	$\neg e_3$	$\neg e_3$	✓	✓	✓
Cl. 4: $\neg e_0, e_3, e_1$	$\neg e_0, e_3, e_1$	e_3, e_1	e_1	✓	✓
BCP	e_0	$\neg e_3$	e_1	e_2	-
PL	-	-	-	-	-
Decision	-	-	-	-	-

We have to check consistency for $(g(a) = c) \wedge (f(g(a)) = f(c)) \wedge (g(a) = d) \wedge (c \neq d)$:

$$\{g(a), c\}, \{f(g(a)), f(c)\}, \{g(a), d\}$$
$$\{f(g(a)), f(c)\}, \{c, d, g(a)\}$$

Congruence Closure returns UNSAT because of: $(c \neq d)$

We therefore add $\neg e_0 \vee e_3 \vee \neg e_1 \vee \neg e_2$ as a blocking clause and continue.

Step	10	11	12	13	14
Decision Level	0	0	0	0	0
Assignment	-	e_0	$e_0, \neg e_3$	$e_0, \neg e_3, e_1$	$e_0, \neg e_3, e_1, \neg e_2$
Cl. 1: e_0	e_0	✓	✓	✓	✓
Cl. 2: $\neg e_1, e_2$	$\neg e_1, e_2$	$\neg e_1, e_2$	$\neg e_1, e_2$	e_2	{} ✗
Cl. 3: $\neg e_3$	$\neg e_3$	$\neg e_3$	✓	✓	✓
Cl. 4: $\neg e_0, e_3, e_1$	$\neg e_0, e_3, e_1$	e_3, e_1	e_1	✓	✓
Cl. 5: $\neg e_0, e_3, \neg e_1, \neg e_2$	$\neg e_0, e_3, \neg e_1, \neg e_2$	$e_3, \neg e_1, \neg e_2$	$\neg e_1, \neg e_2$	$\neg e_2$	✓
BCP	e_0	$\neg e_3$	e_1	$\neg e_2$	-
PL	-	-	-	-	-
Decision	-	-	-	-	UNSAT

Conflict in step 14