Questionnaire "Logic and Computability"

Summer Term 2022

Contents

9 Satisfiability Modulo Theories

9.1 Lecture

9.1.1 Definitions and Notations

1. [Lecture] 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.

2. [Lecture] 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:
   - $\Sigma = \{....., -3, -2, -1, 0, 1, 2, 3..., =, +, -, \neq, <, >, \leq, \geq\}$
   - $\mathcal{A}$ defines the usual meaning to all symbols. (Constant number symbols are mapped to the corresponding value in $\mathbb{Z}$, $+$ is interpreted as the function $0+0 \rightarrow 0$, $0+1 \rightarrow 1$, etc.).

3. [Lecture] 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.

4. [Lecture] Explain the terms $\mathcal{T}$*-terms*, $\mathcal{T}$*-atoms* and $\mathcal{T}$*-literals* for SMT formulas.
   Solution:

   - *Constants* in Σ, *variables*, and applications of function symbols in Σ where all inputs are $\mathcal{T}$-terms are $\mathcal{T}$-terms.
   - A $\mathcal{T}$-atom is the application of a predicate symbol in Σ where all inputs are $\mathcal{T}$-terms.
   - A $\mathcal{T}$-literal is a $\mathcal{T}$-atoms or its negation.

5. [Lecture] 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.

6. [Lecture] Given the signature $\Sigma_{EUF} := \{=, a, b, c, d, ..., f, g, h, ..., P, Q, R, ...\}$ 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)

7. [Lecture] 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.

9.1.2 Eager Encoding

8. [Lecture] 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.
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.
Given a formula φ, algorithms based on eager encoding operate in three steps:

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

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

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

9. [Lecture] 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.

10. [Lecture] Given the formula

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

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

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

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

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

11. [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 $\mathcal{T}_E$.
Solution:

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

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

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

12. [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 $\mathcal{T}_E$.

Solution:

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

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

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

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

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

Solution:

We choose:

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

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

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

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

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

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

Solution:

We choose:

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

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

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

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

9.1.3 Lazy Encoding

15. [Lecture] 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.

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

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

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

Solution:

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

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.

9.2 Practicals

17. [Practicals] **[3 Points]** 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 algorithm to compute an equisatisfiable formula in $\mathcal{T}_E$.
Solution:

There is no solution available for this question yet.

18. [Practicals] **[3 Points]** Given the formula:

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

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

There is no solution available for this question yet.

19. [Practicals] **[3 Points]** Perform 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.

20. [Practicals] **[5 Points]** Consider the following formula in $\mathcal{T}_{EUF}$:

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

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.
Solution:

There is no solution available for this question yet.

21. [Practicals] **[3 Points]** Use the Congruence-Closure algorithm to check if the following assignment for the equalities is satisfiable.

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

Solution:

There is no solution available for this question yet.

22. [Practicals] **[3 Points]** Use the Congruence-Closure algorithm to check if the following assignment for the equalities is satisfiable.

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

Solution:

There is no solution available for this question yet.

9.3 Self-Assessment

9.3.1 Definitions and Notations

23. [Self-Assessment] Explain the concept of a theory in first-order logic using the *theory of Linear Real Arithmetic* $\mathcal{T}_{LRA}$ as example.

Solution:

There is no solution available for this question yet.

24. [Self-Assessment] In the following list tick all formulas that are axioms of the theory of equalities and uninterpreted functions $\mathcal{T}_{EUF}$.

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

25. [Self-Assessment] 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.

26. [Self-Assessment] 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.

27. [Self-Assessment] Provide the answers to the following questions:

- When is a formula $\mathcal{T}$-valid?
- When is a formula $\mathcal{T}$-satisfiable?
- When do we have $\mathcal{T}$-entailment of two formulas?

Solution:

There is no solution available for this question yet.

9.3.2 Eager Encoding

28. [Self-Assessment] 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.

29. [Self-Assessment]

    - Explain the concept of *Eager Encoding* to decide satisfiability of formulas in a first-order theory.
    - Explain how eager encoding works on the *Theory of Equality* $\mathcal{T}_E$.

    Solution:

    There is no solution available for this question yet.

30. [Self-Assessment] In the following text fill the blanks with the missing word(s).

    The Ackermann's reduction is used to reduce a formula φ_{in} in _____ _____ to a formula in _____ _____ that is equisatisfiable. Two formulas are equisatisfiable if _____. The algorithm adds explicit constraints to the formula φ_{in} to enforce _____. These constraints say, that $\forall\bar{x}\forall\bar{y}\ (\bigwedge_i x_i = y_i) \rightarrow$ _____). The resulting equisatisfiable formula consists of two parts and is of the form: $\varphi_{out} := \varphi_C \wedge \hat{\varphi_{in}}$. The right part of the formula $\hat{\varphi_{in}}$ describes the flattening original formula in which we replace _____ with _____.

    Solution:

    There is no solution available for this question yet.

31. [Self-Assessment] For the following $\mathcal{T}_{EUF}$-formula, compute an equivalid formula φ_E in the *Theory of Equality* $\mathcal{T}_E$, by applying *Ackermann's Reduction*.

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

    Solution:

There is no solution available for this question yet.

32. [Self-Assessment] Consider the following formula in $\mathcal{T}_{EUF}$.

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

Use Ackermann's reduction to compute an equisatisfiable formula in $\mathcal{T}_E$.

Solution:

There is no solution available for this question yet.

33. [Self-Assessment] In the context of *Eager Encoding* within the *Satisfiability Modulo Theories (SMT)*, explain how instances of *reflexivity, symmetry* and *transitivity* are handled within the *Graph-based Reduction*.

Solution:

There is no solution available for this question yet.

34. [Self-Assessment] In the following text fill the blanks with the missing word(s).

The Graph Based Reduction is used to reduce a formula φ_{in} in _____ _____ to a formula in _____ _____ that is equisatisfiable. Two formulas are equisatisfiable if _____ _____. In the first step of the algorithm, we create a Non-Polar Equality Graph and in the next step we make it chordal. The graph is chordal, if _____ _____. We introduce fresh propositional variables for each equation to ensure _____. In order to ensure transitivity, the algorithm adds constraints of the form _____ for all _____ in the graph. The resulting equisatisfiable formula consists of two parts and is of the form: $\varphi_{out} := \varphi_{TC}$ __ $\widehat{\varphi_{in}}$. The right part of the formula $\widehat{\varphi_{in}}$ describes the flattening original formula in which we replace _____ with _____.

Solution:

There is no solution available for this question yet.

35. [Self-Assessment] Consider the following formula from $\mathcal{T}_E$.

$$\varphi_{EUF} := \big[f_y = g_x \wedge f_y = y\big] \vee \big[f_y = f_x \wedge y \neq f_{gy}\big] \vee \big[f_x = f_y \wedge f_y = y\big] \vee \big[f_x = f_{gy} \wedge f_y \neq y\big]$$

Use the graph-based algorithm to construct an equisatisfiable propositional formula φ_{prop}.

What would you have to change if you would want to check φ_E for *validity* instead of satisfiability?

Solution:

There is no solution available for this question yet.

36. [Self-Assessment] Consider the following formula from $\mathcal{T}_E$.

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

Use the graph-based algorithm to construct an equivalid propositional formula φ_{prop}.

Solution:

There is no solution available for this question yet.

37. [Self-Assessment] Consider the following formula in $\mathcal{T}_{EUF}$.

$$\varphi_{EUF} := f(x) = f(y) \wedge f(y) = y \vee f(g(x)) = f(f(y)) \wedge g(x) = x \quad (1)$$
$$\vee f(x) \neq f(y) \wedge y \neq g(f(y)) \wedge x \neq g(x) \quad (2)$$

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.

9.3.3 Lazy Encoding

38. [Self-Assessment] 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.

39. [Self-Assessment] *Satisfiability Modulo Theories (SMT)* solvers can be implemented via lazy encoding or via eager encoding. Give a short explanation about both approaches and point out their main differences.

Solution:

There is no solution available for this question yet.

40. [Self-Assessment] 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.

41. [Self-Assessment] In the following text fill the blanks with the missing word(s).

One way to solve *Satisfiability* _____ *Theories* problems works as follows. First, the propositional skeleton of the formula in question is given to a _____ solver. If this solver returns _____, we terminate with answer _____. In the other case, the solver returns a _____, which is a _____

of theory literals. This can be given to a _____ solver that can decide the _____ fragment of the theory in question. If this solver returns _____, we terminate with answer _____. Otherwise, we add a _____ _____ to the propositional skeleton, to prevent the same _____ from occurring again, and run the _____ solver on the augmented propositional skeleton. This loop is repeated until either the _____ solver returns _____ (in which case the answer is _____), or the _____ solver returns _____ (in which case the answer is _____). This entire procedure is called _____ encoding.

Solution:

There is no solution available for this question yet.

42. [Self-Assessment] 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.

43. [Self-Assessment] Use the Congruence-Closure algorithm to check if the following assignment for the equalities is satisfiable.

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

Solution:

There is no solution available for this question yet.

44. [Self-Assessment] Explain for what the *Congruence Closure* algorithm is used. What are the inputs and outputs of the algorithm? What does the algorithm compute? Explain the individual steps of the algorithm.

Solution:

There is no solution available for this question yet.

45. [Self-Assessment] Consider the following formula in the conjunctive fragment of $\mathcal{T}_{EUF}$.

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

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

Solution:

There is no solution available for this question yet.

46. [Self-Assessment] 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.

47. [Self-Assessment] Consider the following formula in the conjunctive fragment of $\mathcal{T}_{EUF}$. Let $a \neq b$ be a shorthand notation for $\neg(a = b)$.

$$f(b) = a \wedge c \neq d \wedge f(e) = b \wedge d \neq f(b) \wedge f(a) = f(e) \wedge$$
$$b \neq f(b) \wedge a \neq e \wedge f(a) = e \wedge a = c \wedge f(b) \neq e \wedge d = f(c)$$

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

There is no solution available for this question yet.