Questionnaire “Logic and Computability”

Summer Term 2024

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.

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.

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.

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

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?

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

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

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 $\mathcal{A}_E$.

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?

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?

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.

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

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 $\mathcal{T}_E$.

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 $\mathcal{T}_E$.

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 $\mathcal{T}_E$.

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

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

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

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

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 $\mathcal{T}_E$.

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 $\mathcal{T}_E$.

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

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

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 $\mathcal{T}_E$.

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 $\mathcal{T}_E$.

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.

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

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

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}.

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 $\mathcal{T}_E$.

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 $\mathcal{T}_E$.

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 $\mathcal{T}_E$.

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

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

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}.

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).

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

9.3.3 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.

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.

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 $\mathcal{T}_{EUF}$.

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

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

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?

9.3.9 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.

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.

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.

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.

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.

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.

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.

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

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

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

$$\begin{aligned} \varphi = & ((f(a) = b) \vee (f(a) = c) \vee \neg(b = c)) \wedge ((b = c) \vee (a = b) \vee (f(a) = b)) \wedge \\ & (\neg(f(a) = b) \vee (a = b)) \wedge ((b = c) \vee \neg(a = b) \vee \neg(f(a) = b)) \wedge \\ & (\neg(f(a) = c) \vee (b = c)) \wedge (\neg(f(a) = c) \vee (b = c) \vee \neg(a = b)) \wedge \\ & ((f(a) = b) \vee (f(a) = c)) \end{aligned}$$

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

$$\begin{aligned} \varphi = & (\neg(f(a) = f(b)) \vee (f(a) = c) \vee (a = b)) \wedge (\neg(f(a) = c) \vee (a = b) \vee \neg(f(c) = a)) \wedge \\ & ((f(a) = f(b)) \vee \neg(f(a) = c)) \wedge (\neg(f(a) = f(b)) \vee (a = b)) \wedge \\ & (\neg(a = b) \vee \neg(f(c) = a)) \wedge ((f(a) = f(b)) \vee (a = b)) \wedge \\ & (\neg(a = b) \vee \neg(f(a) = c)) \end{aligned}$$

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

$$\begin{aligned}\varphi =&((a = x) \vee (a = y) \vee (x = y)) \wedge (\neg(a = x) \vee (a = y)) \wedge \\ &(\neg(a = y) \vee (x = y)) \wedge ((x = y) \vee (z = a)) \wedge \\ &(\neg(x = y) \vee (b = z)) \wedge (\neg(z = a) \vee \neg(b = z))\end{aligned}$$

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

$$\begin{aligned}\varphi =&((f(x) = x) \vee (f(x) = y) \vee (x = y)) \wedge (\neg(f(x) = x) \vee (f(x) = y)) \wedge \\ &(\neg(f(x) = y) \vee (x = y)) \wedge ((x = y) \vee (z = f(x))) \wedge \\ &(\neg(x = y) \vee (f(x) = y)) \wedge (\neg(z = f(x)) \vee \neg(f(x) = y))\end{aligned}$$

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

$$\begin{aligned}\varphi =&((a = b) \vee (a = f(a)) \vee (b = f(a))) \wedge (\neg(a = b) \vee (a = f(a))) \wedge \\ &(\neg(a = f(a)) \vee (b = f(a))) \wedge ((b = f(a)) \vee (d = a)) \wedge \\ &(\neg(b = f(a)) \vee (a = f(a))) \wedge (\neg(d = a) \vee \neg(a = f(a)))\end{aligned}$$

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

$$\begin{aligned}\varphi =&((a = b) \vee (a = f(a))) \wedge ((f(a) = b) \vee (c = b) \vee (f(a) = c)) \wedge \\ &((f(b) = c) \vee \neg(f(a) = b) \vee \neg(f(a) = c)) \wedge (\neg(a = b) \vee \neg(a = f(a))) \wedge \\ &((a = f(a)) \vee \neg(c = b))\end{aligned}$$

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

$$\begin{aligned}\varphi =&(\neg(y = f(x)) \vee \neg(y = f(y))) \wedge ((y = f(x)) \vee (y = f(y))) \wedge \\ &((f(x) = z) \vee (f(y) = x) \vee \neg(f(y) = z)) \wedge ((y = f(y)) \vee \neg(z = x)) \wedge \\ &((f(y) = x) \vee (z = x) \vee (f(y) = z)) \wedge ((y = f(x)) \vee (z = x))\end{aligned}$$