Unification Modulo Builtins

Ștefan Ciobâcă, Andrei Arusoaie and Dorel Lucanu

Faculty of Computer Science
Alexandru Ioan Cuza University, Iași, Romania
stefan.ciobaca@info.uaic.ro

July 24th, 2018
Bogotà, WoLLIC 2018

This work was supported by a grant of the Romanian National Authority for Scientific Research and Innovation, CNCS/CCCDI - UEFISCDI, project number PN-III-P2-2.1-BG-2016-0394, within PNCDI III.

Outline

Unification

Unification is solving an equation in a term algebra.

Unification

Unification is solving an equation in a term algebra.

Example: $\Sigma = \{f, a, b, c\}$ and $X = \{x, y, z, \ldots\}$.
To unify $f(x, f(a, y))$ and $f(z, z)$ is to solve the equation

$$f(x, f(a, y)) \doteq f(z, z).$$

Unification

Unification is solving an equation in a term algebra.

Example: $\Sigma = \{f, a, b, c\}$ and $X = \{x, y, z, \ldots\}$.
To unify $f(x, f(a, y))$ and $f(z, z)$ is to solve the equation

$$f(x, f(a, y)) \doteq f(z, z).$$

A solution to the equation is a *unifier* of $f(x, f(a, y))$ and $f(z, z)$.

Unification

Unification is solving an equation in a term algebra.

Example: $\Sigma = \{f, a, b, c\}$ and $X = \{x, y, z, \ldots\}$.
To unify $f(x, f(a, y))$ and $f(z, z)$ is to solve the equation

$$f(x, f(a, y)) \doteq f(z, z).$$

A solution to the equation is a *unifier* of $f(x, f(a, y))$ and $f(z, z)$.

Example unifier: $\{x \mapsto f(a, a), y \mapsto a, z \mapsto f(a, a)\}$.

Unification

Unification is solving an equation in a term algebra.

Example: $\Sigma = \{f, a, b, c\}$ and $X = \{x, y, z, \ldots\}$.
To unify $f(x, f(a, y))$ and $f(z, z)$ is to solve the equation

$$f(x, f(a, y)) \doteq f(z, z).$$

A solution to the equation is a *unifier* of $f(x, f(a, y))$ and $f(z, z)$.

Example unifier: $\{x \mapsto f(a, a), y \mapsto a, z \mapsto f(a, a)\}$.

Well known: if two terms are unifiable, there is a most general unifier ($\{x \mapsto f(a, y), z \mapsto f(a, y)\}$).

E-Unification

E-Unification is solving en equation in the quotient algebra of terms (equivalence classes given by an equational theory E).

E-Unification

E-Unification is solving en equation in the quotient algebra of terms (equivalence classes given by an equational theory E).

Example: For $E = \{f(x,y) = f(y,x), f(x,f(y,z)) = f(f(x,y),z)\}$ (the AC axioms for f), we have that $f(a,f(b,c)) =_E f(b,f(a,c))$.

E-Unification

E-Unification is solving en equation in the quotient algebra of terms (equivalence classes given by an equational theory E).

Example: For $E = \{f(x,y) = f(y,x), f(x,f(y,z)) = f(f(x,y),z)\}$ (the AC axioms for f), we have that $f(a,f(b,c)) =_E f(b,f(a,c))$.

Example: $f(a,f(x,c))$ and $f(f(a,y),c)$ are not unifiable, but they are E-unifiable.

E-Unification

E-Unification is solving en equation in the quotient algebra of terms (equivalence classes given by an equational theory E).

Example: For $E = \{f(x,y) = f(y,x), f(x,f(y,z)) = f(f(x,y),z)\}$ (the AC axioms for f), we have that $f(a,f(b,c)) =_E f(b,f(a,c))$.

Example: $f(a,f(x,c))$ and $f(f(a,y),c)$ are not unifiable, but they are E-unifiable.

Well-known: any two AC-unifiable terms have a complete set of most general unifiers of size at most double exponential.

E-Unification

E-Unification is solving en equation in the quotient algebra of terms (equivalence classes given by an equational theory E).

Example: For $E = \{f(x,y) = f(y,x), f(x,f(y,z)) = f(f(x,y),z)\}$ (the AC axioms for f), we have that $f(a,f(b,c)) =_E f(b,f(a,c))$.

Example: $f(a,f(x,c))$ and $f(f(a,y),c)$ are not unifiable, but they are E-unifiable.

Well-known: any two AC-unifiable terms have a complete set of most general unifiers of size at most double exponential.

Other results for various equational theories E of interest.

Applications of (E-)Unification

1. first-order theorem proving

$$\frac{P(\tilde{s}) \vee C \qquad \neg P(\tilde{t}) \vee D}{(C \vee D)\sigma}\sigma = mgu(\tilde{s} \doteq \tilde{t})$$

Applications of (E-)Unification

1. first-order theorem proving

$$\frac{P(\tilde{s}) \vee C \qquad \neg P(\tilde{t}) \vee D}{(C \vee D)\sigma}\sigma = mgu(\tilde{s} \doteq \tilde{t})$$

2. symbolic reachability analysis in term rewriting systems

$$f(x, a) \Rightarrow g(x)$$
$$f(b, b) \Rightarrow a$$
$$f(b, c) \Rightarrow b$$

Applications of (E-)Unification

1. first-order theorem proving

$$\frac{P(\tilde{s}) \vee C \qquad \neg P(\tilde{t}) \vee D}{(C \vee D)\sigma}\sigma = mgu(\tilde{s} \doteq \tilde{t})$$

2. symbolic reachability analysis in term rewriting systems

$$f(x, a) \Rightarrow g(x)$$
$$f(b, b) \Rightarrow a$$
$$f(b, c) \Rightarrow b$$

What steps can (instances of) the term $f(y, y)$ take?

Applications of (E-)Unification

1. first-order theorem proving

$$\frac{P(\tilde{s}) \vee C \qquad \neg P(\tilde{t}) \vee D}{(C \vee D)\sigma}\sigma = mgu(\tilde{s} \doteq \tilde{t})$$

2. symbolic reachability analysis in term rewriting systems

$$f(x, a) \Rightarrow g(x)$$
$$f(b, b) \Rightarrow a$$
$$f(b, c) \Rightarrow b$$

What steps can (instances of) the term $f(y, y)$ take?

Solve $f(y, y) \doteq f(x, a)$; $f(y, y) \doteq f(b, b)$; $f(y, y) \doteq f(b, c)$.

E-Unification Modulo Builtins

E-unification modulo builtins is a generalization of E-unification.

E-Unification Modulo Builtins

E-unification modulo builtins is a generalization of E-unification.

Allows mixed terms: terms with free function symbols, interpreted symbols, and an equational theory E.

E-Unification Modulo Builtins

E-unification modulo builtins is a generalization of E-unification.

Allows mixed terms: terms with free function symbols, interpreted symbols, and an equational theory E.

Example: $f(x \times y, a) \doteq f(6, z)$.

E-Unification Modulo Builtins

E-unification modulo builtins is a generalization of E-unification.

Allows mixed terms: terms with free function symbols, interpreted symbols, and an equational theory E.

Example: $f(x \times y, a) \doteq f(6, z)$.

Solution: $\{x \mapsto 2, y \mapsto 3, z \mapsto a\}$.

SMT Solvers

Decide whether a (quantifier-free) first-order formula over a particular theory is satisfiable.

SMT Solvers

Decide whether a (quantifier-free) first-order formula over a particular theory is satisfiable.

LIA examples: $\exists x.(x + 2 * y >= 10)$; $x + x + 1 = y * 2$.

SMT Solvers

Decide whether a (quantifier-free) first-order formula over a particular theory is satisfiable.

LIA examples: $\exists x.(x + 2 * y >= 10)$; $x + x + 1 = y * 2$.

Many theories supported: AUFLIA, AUFLIRA, AUFNIRA, LIA, LRA, QF_ABV, QF_AUFBV, QF_AUFLIA, QF_AX, QF_BV, QF_IDL, QF_LIA, QF_LRA, QF_NIA, QF_NRA, QF_RDL, QF_UF, QF_UFBV, QF_UFIDL, QF_UFLIA, QF_UFLRA, QF_UFNRA, UFLRA, UFNIA.

E-Unification Modulo Builtins

Example: $f(x \times y, a) \doteq f(6, z)$.

E-Unification Modulo Builtins

Example: $f(x \times y, a) \doteq f(6, z)$.

Main ideas:

1. rely on an SMT solver to handle the interpreted symbols (e.g., $\times$);
2. handle the free symbols in the usual way.

Outline

Builtins

A *builtin signature* $\Sigma^{\mathsf{b}} \triangleq (S^{\mathsf{b}}, F^{\mathsf{b}})$ is any many-sorted signature that includes the following distinguished objects:

Builtins

A *builtin signature* $\Sigma^{\mathsf{b}} \triangleq (S^{\mathsf{b}}, F^{\mathsf{b}})$ is any many-sorted signature that includes the following distinguished objects:

1. a sort *Bool*, together with two constants $\top$ and $\bot$ of sort *Bool*;

Builtins

A *builtin signature* $\Sigma^{\mathsf{b}} \triangleq (S^{\mathsf{b}}, F^{\mathsf{b}})$ is any many-sorted signature that includes the following distinguished objects:

1. a sort *Bool*, together with two constants $\top$ and $\bot$ of sort *Bool*;
2. the propositional operation symbols $\neg : \mathit{Bool} \to \mathit{Bool}$, $\wedge, \vee, \to \ : \mathit{Bool} \times \mathit{Bool} \to \mathit{Bool}$ and

Builtins

A *builtin signature* $\Sigma^{\mathsf{b}} \triangleq (S^{\mathsf{b}}, F^{\mathsf{b}})$ is any many-sorted signature that includes the following distinguished objects:

1. a sort *Bool*, together with two constants $\top$ and $\bot$ of sort *Bool*;
2. the propositional operation symbols $\neg : \mathit{Bool} \to \mathit{Bool}$, $\wedge, \vee, \to \ : \mathit{Bool} \times \mathit{Bool} \to \mathit{Bool}$ and
3. an equality predicate symbol $= : s \times s \to \mathit{Bool}$ for each sort $s \in S^{\mathsf{b}}$.

Builtins

A *builtin signature* $\Sigma^{\mathsf{b}} \triangleq (S^{\mathsf{b}}, F^{\mathsf{b}})$ is any many-sorted signature that includes the following distinguished objects:

1. a sort *Bool*, together with two constants $\top$ and $\bot$ of sort *Bool*;
2. the propositional operation symbols $\neg : \mathit{Bool} \to \mathit{Bool}$, $\wedge, \vee, \to \ : \mathit{Bool} \times \mathit{Bool} \to \mathit{Bool}$ and
3. an equality predicate symbol $= : s \times s \to \mathit{Bool}$ for each sort $s \in S^{\mathsf{b}}$.

A *builtin model* M^{b} is a model of a builtin signature Σ^{b}, where the interpretation of the distinguished objects of the builtin signature is fixed as follows: $M^{\mathsf{b}}_{\mathit{Bool}} = \{\top, \bot\}$, $M^{\mathsf{b}}_{\top} = \top$, $M^{\mathsf{b}}_{\bot} = \bot$, $M^{\mathsf{b}}_{=}(a, b) = \top$ iff $a = b$, $M^{\mathsf{b}}_{\neg}(\top) = \bot$, $M^{\mathsf{b}}_{\neg}(\bot) = \top$, $M^{\mathsf{b}}_{\wedge}(\top, b) = M^{\mathsf{b}}_{\wedge}(b, \top) = b$, $M^{\mathsf{b}}_{\wedge}(\bot, b) = M^{\mathsf{b}}_{\wedge}(b, \bot) = \bot$, and so on.

Signature Modulo a Builtin Model

A *signature modulo a builtin model* is a tuple $\Sigma \triangleq (S, \leq, F, M^{\mathsf{b}})$ consisting of

Signature Modulo a Builtin Model

A *signature modulo a builtin model* is a tuple $\Sigma \triangleq (S, \leq, F, M^{\mathsf{b}})$ consisting of

1. an order-sorted signature $(S, \leq, F)$, and

Signature Modulo a Builtin Model

A *signature modulo a builtin model* is a tuple $\Sigma \triangleq (S, \leq, F, M^{\mathsf{b}})$ consisting of

1. an order-sorted signature $(S, \leq, F)$, and
2. a builtin Σ^{b}-model M^{b}, where $\Sigma^{\mathsf{b}} \triangleq (S^{\mathsf{b}}, F^{\mathsf{b}})$ is a builtin subsignature of $(S, \leq, F)$, and

Signature Modulo a Builtin Model

A *signature modulo a builtin model* is a tuple $\Sigma \triangleq (S, \leq, F, M^{\mathsf{b}})$ consisting of

1. an order-sorted signature $(S, \leq, F)$, and
2. a builtin Σ^{b}-model M^{b}, where $\Sigma^{\mathsf{b}} \triangleq (S^{\mathsf{b}}, F^{\mathsf{b}})$ is a builtin subsignature of $(S, \leq, F)$, and
3. the set $F \setminus F^{\mathsf{b}}$ is partitioned into two subsignatures: *constructors* F^{c} and *defined operations* F^{der} such that $F^{\mathsf{c}}_{w,s} = \emptyset$ for each $s \in S^{\mathsf{b}}$.

Signature Modulo a Builtin Model

A *signature modulo a builtin model* is a tuple $\Sigma \triangleq (S, \leq, F, M^{\mathsf{b}})$ consisting of

1. an order-sorted signature $(S, \leq, F)$, and
2. a builtin Σ^{b}-model M^{b}, where $\Sigma^{\mathsf{b}} \triangleq (S^{\mathsf{b}}, F^{\mathsf{b}})$ is a builtin subsignature of $(S, \leq, F)$, and
3. the set $F \setminus F^{\mathsf{b}}$ is partitioned into two subsignatures: *constructors* F^{c} and *defined operations* F^{der} such that $F^{\mathsf{c}}_{w,s} = \emptyset$ for each $s \in S^{\mathsf{b}}$.

We further assume that the only builtin constant symbols in Σ are the elements of the builtin model, i.e., $F^{\mathsf{b}}_{\varepsilon,s} = M^{\mathsf{b}}_s$. Σ^{b} is called the *builtin subsignature* of Σ and $\Sigma^{\mathsf{c}} = (S, \leq, F^{\mathsf{c}} \cup \bigcup_{s \in S^{\mathsf{b}}} F^{\mathsf{b}}_{\varepsilon,s})$ the *constructor subsignature* of Σ.

Model M^{Σ} Generated by a Signature Modulo a Builtin Model

Let $\Sigma \triangleq (S, \leq, F, M^{\mathsf{b}})$ be a signature modulo a builtin model. The model M^{b} is extended to a $(S, \leq, F)$-model M^{Σ}, defined as follows:

Model M^{Σ} Generated by a Signature Modulo a Builtin Model

Let $\Sigma \triangleq (S, \leq, F, M^{\mathsf{b}})$ be a signature modulo a builtin model. The model M^{b} is extended to a $(S, \leq, F)$-model M^{Σ}, defined as follows:

1. $M^{\Sigma}_s = T_{\Sigma^{\mathsf{c}},s}$ for each $s \in S \setminus S^{\mathsf{b}}$, i.e. M^{Σ}_s includes the constructor terms;

Model M^{Σ} Generated by a Signature Modulo a Builtin Model

Let $\Sigma \triangleq (S, \leq, F, M^{\mathsf{b}})$ be a signature modulo a builtin model. The model M^{b} is extended to a $(S, \leq, F)$-model M^{Σ}, defined as follows:

1. $M^{\Sigma}_s = T_{\Sigma^{\mathsf{c}},s}$ for each $s \in S \setminus S^{\mathsf{b}}$, i.e. M^{Σ}_s includes the constructor terms;
2. $M^{\Sigma}_f = M^{\mathsf{b}}_f$ for each $f \in F^{\mathsf{b}}$;

Model M^{Σ} Generated by a Signature Modulo a Builtin Model

Let $\Sigma \triangleq (S, \leq, F, M^{\mathsf{b}})$ be a signature modulo a builtin model. The model M^{b} is extended to a $(S, \leq, F)$-model M^{Σ}, defined as follows:

1. $M^{\Sigma}_s = T_{\Sigma^{\mathsf{c}},s}$ for each $s \in S \setminus S^{\mathsf{b}}$, i.e. M^{Σ}_s includes the constructor terms;
2. $M^{\Sigma}_f = M^{\mathsf{b}}_f$ for each $f \in F^{\mathsf{b}}$;
3. M^{Σ}_f is the term constructor $M^{\Sigma}_f(t_1, \ldots, t_n) = f(t_1, \ldots, t_n)$ for each $f \in F^{\mathsf{c}}$;

Model M^{Σ} Generated by a Signature Modulo a Builtin Model

Let $\Sigma \triangleq (S, \leq, F, M^{\mathsf{b}})$ be a signature modulo a builtin model. The model M^{b} is extended to a $(S, \leq, F)$-model M^{Σ}, defined as follows:

1. $M^{\Sigma}_s = T_{\Sigma^{\mathsf{c}},s}$ for each $s \in S \setminus S^{\mathsf{b}}$, i.e. M^{Σ}_s includes the constructor terms;
2. $M^{\Sigma}_f = M^{\mathsf{b}}_f$ for each $f \in F^{\mathsf{b}}$;
3. M^{Σ}_f is the term constructor $M^{\Sigma}_f(t_1, \ldots, t_n) = f(t_1, \ldots, t_n)$ for each $f \in F^{\mathsf{c}}$;
4. M^{Σ}_f is a function $M^{\Sigma}_f : M^{\Sigma}_{s_1} \times \cdots \times M^{\Sigma}_{s_n} \to M^{\Sigma}_s$ for each $f \in F^{\mathrm{der}}_{s_1 \ldots s_n, s}$.

The Quotient Model

We consider a set E of identities such as associativity, commutativity or idempotence over constructor symbols in Σ.

The Quotient Model

We consider a set E of identities such as associativity, commutativity or idempotence over constructor symbols in Σ.

We let $\cong$ be the equivalence relation induced by E on M^{Σ}. We make the assumption that $\cong$ is a congruence on M^{Σ}, i.e., that it is compatible with the functions.

The Quotient Model

We consider a set E of identities such as associativity, commutativity or idempotence over constructor symbols in Σ.

We let $\cong$ be the equivalence relation induced by E on M^{Σ}. We make the assumption that $\cong$ is a congruence on M^{Σ}, i.e., that it is compatible with the functions.

We define $M^{\Sigma}_{\cong} \triangleq M^{\Sigma}/\cong$ to be the quotient algebra induced by the congruence $\cong$ on M^{Σ}.

The Quotient Model

We consider a set E of identities such as associativity, commutativity or idempotence over constructor symbols in Σ.

We let $\cong$ be the equivalence relation induced by E on M^{Σ}. We make the assumption that $\cong$ is a congruence on M^{Σ}, i.e., that it is compatible with the functions.

We define $M^{\Sigma}_{\cong} \triangleq M^{\Sigma}/\cong$ to be the quotient algebra induced by the congruence $\cong$ on M^{Σ}.

E-Unification Modulo Builtins is solving equations in $M^{\Sigma}_{\cong}$.

Outline

A Working Example

Consider the E-unification modulo builtins problem

$$\mathsf{n} \rightsquigarrow 2 \times N + 1, \mathsf{cnt} \rightsquigarrow C \doteq \mathsf{cnt} \rightsquigarrow C' + N', \mathsf{n} \rightsquigarrow N' + 3.$$

A Working Example

Consider the E-unification modulo builtins problem

$$\mathsf{n} \rightsquigarrow 2 \times N + 1, \mathsf{cnt} \rightsquigarrow C \doteq \mathsf{cnt} \rightsquigarrow C' + N', \mathsf{n} \rightsquigarrow N' + 3.$$

1. n, cnt are constant symbols of builtin sort Id;

A Working Example

Consider the E-unification modulo builtins problem

$$\mathsf{n} \rightsquigarrow 2 \times N + 1, \mathsf{cnt} \rightsquigarrow C \doteq^{?} \mathsf{cnt} \rightsquigarrow C' + N', \mathsf{n} \rightsquigarrow N' + 3.$$

1. n, cnt are constant symbols of builtin sort Id;
2. $\rightsquigarrow$ is a free binary function symbol;

A Working Example

Consider the E-unification modulo builtins problem

$$\mathsf{n} \rightsquigarrow 2 \times N + 1, \mathsf{cnt} \rightsquigarrow C \doteq^{?} \mathsf{cnt} \rightsquigarrow C' + N', \mathsf{n} \rightsquigarrow N' + 3.$$

1. n, cnt are constant symbols of builtin sort Id;
2. $\rightsquigarrow$ is a free binary function symbol;
3. , is an AC symbol (E contains the AC axioms for ,);

A Working Example

Consider the E-unification modulo builtins problem

$$\mathsf{n} \rightsquigarrow 2 \times N + 1, \mathsf{cnt} \rightsquigarrow C \doteq^{?} \mathsf{cnt} \rightsquigarrow C' + N', \mathsf{n} \rightsquigarrow N' + 3.$$

1. n, cnt are constant symbols of builtin sort Id;
2. $\rightsquigarrow$ is a free binary function symbol;
3. , is an AC symbol (E contains the AC axioms for ,);
4. $\times, +$ are builtin operations on integers and $1, 2, 3$ are builtin constant symbols;

A Working Example

Consider the E-unification modulo builtins problem

$$\mathsf{n} \rightsquigarrow 2 \times N + 1, \mathsf{cnt} \rightsquigarrow C \doteq \mathsf{cnt} \rightsquigarrow C' + N', \mathsf{n} \rightsquigarrow N' + 3.$$

1. n, cnt are constant symbols of builtin sort Id;
2. $\rightsquigarrow$ is a free binary function symbol;
3. , is an AC symbol (E contains the AC axioms for ,);
4. $\times, +$ are builtin operations on integers and $1, 2, 3$ are builtin constant symbols;
5. N, C, N', C' are variables of builtin sort Int.

Abstractions of Terms

Consider the E-unification modulo builtins problem

$$\mathsf{n} \rightsquigarrow \quad 2 \times N + 1,\ \ \mathsf{cnt} \rightsquigarrow C \quad \doteq \quad \mathsf{cnt} \rightsquigarrow \quad C' + N',\ \ \mathsf{n} \rightsquigarrow \quad N' + 3$$

Abstractions of Terms

$$\mathsf{n} \rightsquigarrow \quad 2 \times N + 1,\ \ \mathsf{cnt} \rightsquigarrow C \quad \doteq \quad \mathsf{cnt} \rightsquigarrow \quad C' + N',\ \ \mathsf{n} \rightsquigarrow \quad N' + 3$$

Consider the E-unification modulo builtins problem

$$\mathsf{n} \rightsquigarrow X, \qquad \mathsf{cnt} \rightsquigarrow C \quad \doteq \quad \mathsf{cnt} \rightsquigarrow Y, \qquad \mathsf{n} \rightsquigarrow Z$$

Abstractions of Terms

$$\mathsf{n} \rightsquigarrow \quad 2 \times N + 1,\ \ \mathsf{cnt} \rightsquigarrow C \quad \doteq \quad \mathsf{cnt} \rightsquigarrow \quad C' + N',\ \ \mathsf{n} \rightsquigarrow \quad N' + 3$$

Consider the E-unification modulo builtins problem

$$I \rightsquigarrow X, \qquad J \rightsquigarrow C \doteq K \rightsquigarrow Y, \qquad L \rightsquigarrow Z$$

Abstractions of Terms

$$\mathsf{n} \rightsquigarrow \quad 2 \times N + 1,\ \ \mathsf{cnt} \rightsquigarrow C \quad \doteq \quad \mathsf{cnt} \rightsquigarrow \quad C' + N',\ \ \mathsf{n} \rightsquigarrow \quad N' + 3$$

Consider the E-unification modulo builtins problem

$$\begin{array}{lll} I \rightsquigarrow X, & J \rightsquigarrow C & \doteq \quad K \rightsquigarrow Y, \quad L \rightsquigarrow Z \\ X = N + 1 \wedge I = \mathsf{n} \wedge J = \mathsf{cnt} & & Y = C' + N' \wedge Z = N' + 3 \wedge \\ & & K = \mathsf{cnt} \wedge L = \mathsf{n} \end{array}$$

Abstractions of Terms

Solve E-unification problem

$$I \rightsquigarrow X, J \rightsquigarrow C \doteq K \rightsquigarrow Y, L \rightsquigarrow Z.$$

Abstractions of Terms

Solve E-unification problem

$$I \rightsquigarrow X, J \rightsquigarrow C \doteq K \rightsquigarrow Y, L \rightsquigarrow Z.$$

1. Obtain a complete set of E-unifiers $\{\tau_1, \ldots, \tau_n\}$.

Abstractions of Terms

Solve E-unification problem

$$I \rightsquigarrow X, J \rightsquigarrow C \doteq K \rightsquigarrow Y, L \rightsquigarrow Z.$$

1. Obtain a complete set of E-unifiers $\{\tau_1, \ldots, \tau_n\}$.
   In our example: $n = 2$ and

$$\tau_1 = \{I \mapsto K, J \mapsto L, X \mapsto Y, Z \mapsto C\},$$

$$\tau_2 = \{I \mapsto L, J \mapsto K, X \mapsto Z, Y \mapsto C\}.$$

Abstractions of Terms

Solve E-unification problem

$$I \rightsquigarrow X, J \rightsquigarrow C \doteq K \rightsquigarrow Y, L \rightsquigarrow Z.$$

1. Obtain a complete set of E-unifiers $\{\tau_1, \ldots, \tau_n\}$.
   In our example: $n = 2$ and

$$\tau_1 = \{I \mapsto K, J \mapsto L, X \mapsto Y, Z \mapsto C\},$$

$$\tau_2 = \{I \mapsto L, J \mapsto K, X \mapsto Z, Y \mapsto C\}.$$

2. Turn builtin parts of the substitution into constraints:

Abstractions of Terms

Solve E-unification problem

$$I \rightsquigarrow X, J \rightsquigarrow C \doteq K \rightsquigarrow Y, L \rightsquigarrow Z.$$

1. Obtain a complete set of E-unifiers $\{\tau_1, \ldots, \tau_n\}$.
   In our example: $n = 2$ and

   $$\tau_1 = \{I \mapsto K, J \mapsto L, X \mapsto Y, Z \mapsto C\},$$

   $$\tau_2 = \{I \mapsto L, J \mapsto K, X \mapsto Z, Y \mapsto C\}.$$

2. Turn builtin parts of the substitution into constraints:
   We already have: $X = N + 1 \wedge I = \mathsf{n} \wedge J = \mathsf{cnt} \wedge Y = C' + N' \wedge Z = N' + 3 \wedge K = \mathsf{cnt} \wedge L = \mathsf{n}$
   We add: $I = K \wedge J = L \wedge X = Y \wedge Z = C$

Abstractions of Terms

Solve E-unification problem

$$I \rightsquigarrow X, J \rightsquigarrow C \doteq K \rightsquigarrow Y, L \rightsquigarrow Z.$$

1. Obtain a complete set of E-unifiers $\{\tau_1, \ldots, \tau_n\}$.
   In our example: $n = 2$ and

   $$\tau_1 = \{I \mapsto K, J \mapsto L, X \mapsto Y, Z \mapsto C\},$$

   $$\tau_2 = \{I \mapsto L, J \mapsto K, X \mapsto Z, Y \mapsto C\}.$$

2. Turn builtin parts of the substitution into constraints:
   We already have: $X = N + 1 \wedge I = \mathsf{n} \wedge J = \mathsf{cnt} \wedge Y = C' + N' \wedge Z = N' + 3 \wedge K = \mathsf{cnt} \wedge L = \mathsf{n}$
   We add: $I = K \wedge J = L \wedge X = Y \wedge Z = C$ - unsatisfiable

Abstractions of Terms

Solve E-unification problem

$$I \rightsquigarrow X, J \rightsquigarrow C \doteq K \rightsquigarrow Y, L \rightsquigarrow Z.$$

1. Obtain a complete set of E-unifiers $\{\tau_1, \ldots, \tau_n\}$.
   In our example: $n = 2$ and

   $$\tau_1 = \{I \mapsto K, J \mapsto L, X \mapsto Y, Z \mapsto C\},$$

   $$\tau_2 = \{I \mapsto L, J \mapsto K, X \mapsto Z, Y \mapsto C\}.$$

2. Turn builtin parts of the substitution into constraints:
   We already have: $X = N + 1 \wedge I = \mathsf{n} \wedge J = \mathsf{cnt} \wedge Y = C' + N' \wedge Z = N' + 3 \wedge K = \mathsf{cnt} \wedge L = \mathsf{n}$
   We add: $I = L \wedge J = K \wedge X = Z \wedge Y = C$

Abstractions of Terms

Solve E-unification problem

$$I \rightsquigarrow X, J \rightsquigarrow C \doteq K \rightsquigarrow Y, L \rightsquigarrow Z.$$

1. Obtain a complete set of E-unifiers $\{\tau_1, \ldots, \tau_n\}$.
   In our example: $n = 2$ and

   $$\tau_1 = \{I \mapsto K, J \mapsto L, X \mapsto Y, Z \mapsto C\},$$

   $$\tau_2 = \{I \mapsto L, J \mapsto K, X \mapsto Z, Y \mapsto C\}.$$

2. Turn builtin parts of the substitution into constraints:
   We already have: $X = N + 1 \wedge I = \mathsf{n} \wedge J = \mathsf{cnt} \wedge Y = C' + N' \wedge Z = N' + 3 \wedge K = \mathsf{cnt} \wedge L = \mathsf{n}$
   We add: $I = L \wedge J = K \wedge X = Z \wedge Y = C$ - satisfiable

Abstractions of Terms

Solve E-unification problem

$$I \rightsquigarrow X, J \rightsquigarrow C \doteq K \rightsquigarrow Y, L \rightsquigarrow Z.$$

1. Obtain a complete set of E-unifiers $\{\tau_1, \ldots, \tau_n\}$.
   In our example: $n = 2$ and

   $$\tau_1 = \{I \mapsto K, J \mapsto L, X \mapsto Y, Z \mapsto C\},$$

   $$\tau_2 = \{I \mapsto L, J \mapsto K, X \mapsto Z, Y \mapsto C\}.$$

2. Turn builtin parts of the substitution into constraints:
   Conclusion: the equation holds only if
   $X = N + 1 \wedge I = \mathsf{n} \wedge J = \mathsf{cnt} \wedge Y = C' + N' \wedge Z = N' + 3 \wedge K = \mathsf{cnt} \wedge L = \mathsf{n} \wedge I = L \wedge J = K \wedge X = Z \wedge Y = C$
   is true.

E-Unifiers Modulo Builtins

An *E-unifier modulo builtins* (E-umb) of two terms t_1, t_2 is a pair $u = (\sigma, \phi)$, where σ is a substitution and ϕ is a builtin constraint, such that

$$M_{\cong}^{\Sigma} \models \phi \rightarrow \sigma(t_1) = \sigma(t_2).$$

E-Unifiers Modulo Builtins

An *E-unifier modulo builtins* (E-umb) of two terms t_1, t_2 is a pair $u = (\sigma, \phi)$, where σ is a substitution and ϕ is a builtin constraint, such that

$$M^{\Sigma}_{\cong} \models \phi \rightarrow \sigma(t_1) = \sigma(t_2).$$

A set C of pairs of substitutions and builtin logical constraints is called a *complete set of E-unifiers* of t_1 and t_2 if:

1. each pair $(\sigma, \phi) \in C$ is an E-umb of t_1 and t_2: $M^{\Sigma}_{\cong} \models \phi \rightarrow \sigma(t_1) = \sigma(t_2)$;
2. for any partial valuation $\alpha : var(t_1, t_2) \rightarrow M^{\Sigma}_{\cong}$ such that $\alpha(t_1) = \alpha(t_2)$, there is an E-unifier $(\sigma, \phi) \in C$ and a valuation α^r such that $M^{\Sigma}_{\cong}, \alpha^r \models \phi$ and $\alpha = (\alpha^r \circ \sigma)|_{var(t_1,t_2)}$.

Algorithm for E-Unification Modulo Builtins

1: **function** UNIFICATION(t_1, t_2)
2: ▷ returns: a complete set of E-umbs of t_1 and t_2
3: compute $\langle s_1 \mid \phi^{\sigma_1} \rangle$, an abstraction of t_1
4: compute $\langle s_2 \mid \phi^{\sigma_2} \rangle$, an abstraction of t_2
5: compute $\{\tau_1, \ldots, \tau_n\}$, a complete set of E-unifiers of s_1 and s_2
6: **for** $i \in \{1, \ldots, n\}$ **do**
7: $\tau_i' \leftarrow \tau_i|_{\mathcal{X} \setminus \mathcal{X}^b}$
8: $\phi_i' \leftarrow \phi^{\sigma_1} \wedge \phi^{\sigma_2} \wedge \bigwedge_{x \in dom(\tau_i) \cap \mathcal{X}^b} \tau_i(x) = x$
9: **return** $\{(\tau_1', \phi_1'), \ldots, (\tau_n', \phi_n')\}$

Algorithm for E-Unification Modulo Builtins

1: **function** UNIFICATION(t_1, t_2)
2: ▷ returns: a complete set of E-umbs of t_1 and t_2
3: compute $\langle s_1 \mid \phi^{\sigma_1} \rangle$, an abstraction of t_1
4: compute $\langle s_2 \mid \phi^{\sigma_2} \rangle$, an abstraction of t_2
5: compute $\{\tau_1, \ldots, \tau_n\}$, a complete set of E-unifiers of s_1 and s_2
6: **for** $i \in \{1, \ldots, n\}$ **do**
7: $\tau_i' \leftarrow \tau_i|_{\mathcal{X} \setminus \mathcal{X}^b}$
8: $\phi_i' \leftarrow \phi^{\sigma_1} \wedge \phi^{\sigma_2} \wedge \bigwedge_{x \in dom(\tau_i) \cap \mathcal{X}^b} \tau_i(x) = x$
9: **return** $\{(\tau_1', \phi_1'), \ldots, (\tau_n', \phi_n')\}$

Theorem

The set $\{(\tau_1', \phi_1'), \ldots, (\tau_n', \phi_n')\}$ computed by the algorithm above is a complete set of E-unifiers modulo builtins of t_1 and t_2.

Outline

Conclusion and Future Work

1. introduced E-unification modulo builtins;
2. generalized standard notions in E-unification to E-unification modulo builtins;
3. prototype implementation in Maude: `https://github.com/andreiarusoaie/unification-modulo-builtins`.

Conclusion and Future Work

1. introduced E-unification modulo builtins;
2. generalized standard notions in E-unification to E-unification modulo builtins;
3. prototype implementation in Maude: `https://github.com/andreiarusoaie/unification-modulo-builtins`.

4. Future work: full implementation, extend to defined symbols, applications to program verification.