Computer-Aided Modelling and Reasoning

Karel Kubíček

April 25, 2018

Overview

Goal: decide, whether a security protocol leaks sensitive information.

- Modeled protocol messages as first-order algebra formulas.
- Models intruder capabilities by rules for derivation content from messages.
- Describe a unification algorithm for equation over first-order algebra.
- Run the protocol symbolically and let the unification deduce target messages.

First-order Algebra Terms

A free term algebra $\mathcal{T}_\Sigma(\mathcal{V})$ is a set of terms over variables $\mathcal{V}$ and symbols of Σ.

A substitution is a function $\sigma : \mathcal{V} \to \mathcal{T}_\Sigma(\mathcal{V})$ on variables.

A unification problem U is set of equations over $\mathcal{T}_\Sigma(\mathcal{V})$: $U = \{(t_1, u_1), \ldots, (t_n, u_n)\}$.

A unifier for U is a substitution σ such that $\sigma \cdot t = \sigma \cdot u$ for all $(t, u) \in U$.

A unifier σ is the most general unifier (MGU), iff any other $\tau = \upsilon \circ \sigma$.

Example

Let $u = f(x, h(x))$ and $t = f(y, z)$.

Let $U = \{(u, t)\}$. $\sigma = [x := y, z := h(y)]$ is a MGU for U.

A unifier $\tau = [x := a, y := a, z := h(a)]$ is less general, as $\tau = [y := a] \circ \sigma$.

Robinson-style unification algorithm

Algorithm: UNIFY(U)

Data: A unification problem U
Result: MGU substitution σ for U

1 **if** $U = \emptyset$ **then**
2   return id
3 **let** $(u, t) \in U$ **and** $U' = U \setminus \{(u, t)\}$
4 **if** u is a variable x **then**
5   **if** $x \notin fv(t)$ **then**
6   return UNIFY($[x := t] \cdot U'$) $\circ^{\perp}$ $[x := t]$)
7   **else if** $x = t$ **then**
8   return UNIFY(U')
9   **else**
10   return $\perp$
11 **else if** t is a variable x **then**
12   return UNIFY($U' \cup \{(t, u)\}$)
13 **else if** $u = f(u_1, \ldots, u_n) \wedge t = f(t_1, \ldots, t_n)$ **then**
14   return UNIFY($U' \cup \{(u_i, t_i) \mid 1 \leq i \leq n\}$)
15 **else**
16   return $\perp$

Example

Let $u = f(x, h(x))$ and $t = f(y, z)$.

1. Line 13: $u = f(\ldots), t = f(\ldots)$.
   $UNIFY(\{(x, y), (h(x), z)\})$.
2. Line 5: $u = x, t = y$, $x \notin fv(y)$.
   $UNIFY(\{(h(y), z)\}) \circ [x := y]$.
3. Line 11: $u = h(y), t = z$, $t \in \mathcal{V}$.
   $UNIFY(\{(z, h(y))\})$.
4. Line 5: $u = z, t = h(y)$, $z \notin fv(y)$.
   $UNIFY(\emptyset) \circ [z := h(y)]$.
5. Line 1: $U = \emptyset$, return id.

Solution $\sigma = [x := y] \circ [z := h(y)]$

Security protocol model

Protocol messages are terms from $\mathcal{T}_\Sigma(\mathcal{V})$.

The constants are for agent names and nonces.

The functions denote security primitives ($\mathsf{h}(t)$, $\{|t|\}_k$, $\{t\}_k$, $[t]_k$), and tuples $\langle t_1, t_2 \rangle$.

Role-based protocol specification consists of send and receive events.

Example (Needham–Schroeder Public-Key Protocol)

$$NSPK(A) = \mathrm{send}(\{A, na\}_B) \cdot \mathrm{recv}(\{na, NB\}_A) \cdot \mathrm{send}(\{NB\}_B)$$

$$NSPK(B) = \mathrm{recv}(\{A, NA\}_B) \cdot \mathrm{send}(\{NA, nb\}_A) \cdot \mathrm{recv}(\{nb\}_B)$$

The protocol is executed in threads.

Protocol state (IK, th) consists of intruder knowledge IK and thread pool th.

A $\mathrm{send}(t)$ event adds term t to IK and pop the event from the thread.

Intruder capabilities

$T \vdash u$ denotes the capability of deriving a message u from observed messages T.

Axiom rule

$$\frac{t \in T}{T \vdash t}\ \text{Ax}$$

Composition rule

$$\frac{T \vdash t_1, \ldots, T \vdash t_n}{T \vdash f(t_1, \ldots, t_n)}\ \text{Comp}\ (f \in \Sigma_C)$$

Analysis rules

$$\frac{T \vdash \langle t_1, t_2 \rangle}{T \vdash t_i}\ \text{Proj}_i \qquad \frac{T \vdash \{|t|\}_k,\ T \vdash k}{T \vdash t}\ \text{Sdec} \qquad \frac{T \vdash \{t\}_k,\ T \vdash k}{T \vdash t}\ \text{Adec}$$

Constraint-based protocol analysis

A decision procedure removes branching by symbolic execution.

The trace tr is solved later by the unification of the symbolic variables.

The trace $\sigma \cdot tr$ corresponds to the ground execution of the protocol.

Example (Needham–Schroeder Public-Key Protocol)

$$NSPK(0) = \mathrm{send}_0(\{A_0, na_0\}_{B_0}) \cdot \mathrm{recv}_0(\{na_0, NB_0\}_A) \cdot \mathrm{send}_0(\{NB_0\}_B)$$

$$NSPK(1) = \mathrm{recv}_1(\{A_1, NA_1\}_{B_1}) \cdot \mathrm{send}_1(\{NA_1, nb_1\}_{A_1}) \cdot \mathrm{recv}_1(\{nb_1\}_{B_1})$$

Define the protocol secrecy by adding extra message: $\mathrm{recv}_1(\langle NA_1, nb_1 \rangle)$.

Intruder knowledge initially contain $IK_0 \vdash \langle A_0, B_0, A_1, B_1 \rangle$.

$\mathrm{send}(t)$ extends it by t: $IK_1 = IK_0 \cup \{A_0, na_0\}_{B_0}$.

$\mathrm{recv}(t)$ adds constraint t: $IK_1 \vdash \{A_1, NA_1\}_{B_1}$.

Lowe's attack: $\sigma = [A_0 := a, B_0 := \iota, A_1 := a, B_1 := b, NB_0 := nb_1, NA_1 := na_0]$.

Constraint system

An intruder deduction constraint $c = M \mid A \rhd t$.

A constraint system *cs* is a finite set of constraints.

A constraint $M \mid A \rhd t$ is intruder derivable by σ iff $\sigma \cdot (M \cup A) \vdash \sigma \cdot t$.

A solution σ of *cs* makes all $c \in cs$ intruder derivable.

Set of all solutions is called solution set:

$$sol(cs) = \{\sigma \mid \forall (M \mid A \rhd t) \in cs.\ \sigma \cdot (M \cup A) \vdash \sigma \cdot t\}.$$

Constraint solving

Unification rule

$\mathrm{Unif}^{\ell} \quad M \mid A \rhd t \rightsquigarrow^1_\sigma \emptyset$ if $t \notin \mathcal{V}, u \in M \cup A,$ and $\sigma = UNIFY(t, u)$

Composition rule ($f \in \Sigma_{\mathcal{C}}$)

$\mathrm{Comp}^{\ell} \quad M \mid A \rhd f(t_1, \ldots, t_n) \rightsquigarrow^1_{id} \{M \mid A \rhd t_1, \ldots, M \mid A \rhd t_n\}$

Analysis rules

$\mathrm{Proj}^{\ell} \quad M \cup \{\langle u, v\rangle\} \mid A \rhd t \rightsquigarrow^1_{id} \{M \cup \{u, v\} \mid A \cup \langle u, v\rangle \rhd t\}$

$\mathrm{Sdec}^{\ell} \quad M \cup \{\{|u|\}_k\} \mid A \rhd t \rightsquigarrow^1_{id} \{M \cup \{u\} \mid A \cup \{\{|u|\}_k\} \rhd t, M \cup \{u\} \mid A \cup \{\{|u|\}_k\} \rhd k\}$

$\mathrm{Adec}^{\ell} \quad M \cup \{\{u\}_\iota\} \mid A \rhd t \rightsquigarrow^1_{id} \{M \cup \{u\} \mid A \cup \{\{u\}_\iota\} \rhd t\}$

$\mathrm{Ksub}^{\ell} \quad M \cup \{\{u\}_x\} \mid A \rhd t \rightsquigarrow^1_{[x:=\iota]} \{[x := \iota] \cdot (M \cup \{u\} \mid A \cup \{u\}_x \rhd t)\}$

Lifting c to cs

$$\frac{c \rightsquigarrow^1_\sigma cs}{c \cup cs' \rightsquigarrow_\sigma cs \cup \sigma \cdot cs'} \text{ Context}$$

Finding solutions

The relation $\rightsquigarrow_\sigma$ is single reduction step. Let $\rightsquigarrow_\sigma^*$ be reflexive and transitive closure.

The reduction stops at *simple* constraint system, where all $\rhd\, t$ are variables.

The set of reducts of a *cs* is defined as

$$red(cs) = \{\tau \circ \sigma \mid \exists cs'.\ cs \rightsquigarrow_\sigma^* cs' \wedge cs' \text{ is } simple \wedge \tau \in sol(cs')\}.$$

We want to show that $red(cs) = sol(cs)$.

Then $\sigma \in red(cs)$ is the ground substitution and $\sigma \cdot tr$ is a ground trace of the protocol.

We will use "cut rule" for intruder deduction: *If* $T \cup \{t\} \vdash u$ *and* $T \vdash t$ *then* $T \vdash u$*.*

Constraint solving soundness ($red(cs) \subseteq sol(cs)$)

Lemma (One-step reduction soundness)

If $\{c\} \rightsquigarrow_{\sigma}^{*} cs$ *and* $\tau \in sol(cs)$ *then* $\tau \circ \sigma \in sol(\{c\})$.

Proof: One-step reduction soundness.

By case on the reduction rule R.

High-level proof scheme:

- $R = M \mid A \rhd t \rightsquigarrow_{\sigma}^{1} \{c_1, \dots, c_n\}$.
- The rule gives us: $c = M \mid A \rhd t$, $cs = \{c_1, \dots, c_n\}$, and σ from $\rightsquigarrow_{\sigma}$.
- The τ makes all $c_i \in cs$ intruder derivable: $\tau \cdot (M_i \cup A_i) \vdash \tau \cdot t_i$.
- We have to show that the relation keeps the intruder derivability (for resulting c): $\tau \circ \sigma \cdot (M \cup A) \vdash \tau \circ \sigma \cdot t$.

Constraint solving soundness ($red(cs) \subseteq sol(cs)$)

Proof: One-step reduction soundness.

$R = \mathrm{Unif}^{\ell} \quad M \mid A \rhd t \rightsquigarrow^{1}_{\sigma} \emptyset \quad$ if $t \notin \mathcal{V}, u \in M \cup A,$ and $\sigma = UNIFY(t, u)$.

The rule gives us: $c = M \mid A \rhd t$, $cs = \emptyset$, $u \in M \cup A$, and $\sigma = UNIFY(t, u)$.
$UNIFY(t, u)$ ensures that $\sigma \cdot t = \sigma \cdot u$.

We have to show $\tau \circ \sigma \cdot (M \cup A) \vdash \tau \circ \sigma \cdot t$.

$$\frac{\sigma \cdot t = \sigma \cdot u \in \sigma \cdot (M \cup A)}{\sigma \cdot (M \cup A) \vdash \sigma \cdot t}\ \mathrm{Ax}$$

The axiom rule conclusion is free of τ, so $\forall \tau . \tau \circ \sigma \in sol(\{c\})$.

$\rightarrow$

Constraint solving soundness ($red(cs) \subseteq sol(cs)$)

Proof: One-step reduction soundness.

$R = \mathrm{Sdec}^{\ell} \quad M \cup \{\{|u|\}_k\} \mid A \rhd t \rightsquigarrow^1_{id} \{M \cup \{u\} \mid A \cup \{\{|u|\}_k\} \rhd t, M \cup \{u\} \mid A \cup \{\{|u|\}_k\} \rhd k\}.$

The rule gives us: $c = M \cup \{\{|u|\}_k\} \mid A \rhd t$, $\sigma = id$, and $cs = \{c_u, c_k\}$.

By $\tau \in sol(cs)$, apply τ on both constraints: $\tau \cdot (M \cup A \cup \{u, \{|u|\}_k\}) \vdash \tau \cdot t$ (c_u) and $\tau \cdot (M \cup A \cup \{\{|u|\}_k\}) \vdash \tau \cdot k$ (c_k).

$$\dfrac{\dfrac{}{\tau \cdot (M \cup A \cup \{\{|u|\}_k\}) \vdash \tau \cdot \{|u|\}_k}\,\text{Ax} \qquad \dfrac{}{\tau \cdot (M \cup A \cup \{\{|u|\}_k\}) \vdash \tau \cdot k}\,(\text{by } c_k)}{\tau \cdot (M \cup A \cup \{\{|u|\}_k\}) \vdash \tau \cdot u}\,\text{Sdec}$$

From (c_u) and the result of this derivation, we derive $\tau \cdot (M \cup A \cup \{\{|u|\}_k\}) \vdash \tau \cdot t$ (using the cut rule).

$\longrightarrow$

Constraint solving soundness ($red(cs) \subseteq sol(cs)$)

Proof: One-step reduction soundness.

$R = \mathrm{Comp}^{\ell} \quad M \mid A \rhd f(t_1, \ldots, t_n) \leadsto^1_{id} \{M \mid A \rhd t_1, \ldots, M \mid A \rhd t_n\}$.

$c = M \mid A \rhd f(t_1, \ldots, t_n)$, $cs = \{M \mid A \rhd t_1, \ldots, M \mid A \rhd t_n\}$, and $\sigma = id$.

We use $\tau \in sol(cs)$ and we have to show $\tau \circ id = \tau \in sol(\{c\})$.

$$\frac{\tau \cdot (M \cup A) \vdash \tau \cdot t_1, \ldots, \tau \cdot (M \cup A) \vdash \tau \cdot t_n}{\tau \cdot (M \cup A) \vdash \tau \cdot f(t_1, \ldots, t_n)} \ \text{Comp}\ (f \in \Sigma_C)$$

In the conclusion of this derivation, τ satisfies definition of $sol(\{c\})$.

$\rightarrow$

Constraint solving soundness ($red(cs) \subseteq sol(cs)$)

Proof: One-step reduction soundness.

$R = \mathrm{Proj}^{\ell} \quad M \cup \{\langle u, v\rangle\} \mid A \rhd t \rightsquigarrow^{1}_{id} \{M \cup \{u, v\} \mid A \cup \langle u, v\rangle \rhd t\}$.

$c = M \cup \{\langle u, v\rangle\} \mid A \rhd t$, $cs = \{M \cup \{u, v\} \mid A \cup \{\langle u, v\rangle\} \rhd t\}$, and $\sigma = id$.

We have to show $\tau \circ id = \tau \in sol(\{c\})$. We begin with the axiom rule:

$$\dfrac{\dfrac{}{\tau \cdot (M \cup \{u, v\} \cup A \cup \{\langle u, v\rangle\}) \vdash \tau \cdot \langle u, v\rangle}\,\text{Ax}}{\tau \cdot (M \cup \{u, v\} \cup A \cup \{\langle u, v\rangle\}) \vdash \tau \cdot u}\,\text{Proj}_1 \qquad \dfrac{\dfrac{}{\tau \cdot (M \cup \{u, v\} \cup A \cup \{\langle u, v\rangle\}) \vdash \tau \cdot \langle u, v\rangle}\,\text{Ax}}{\tau \cdot (M \cup \{u, v\} \cup A \cup \{\langle u, v\rangle\}) \vdash \tau \cdot v}\,\text{Proj}_2$$

As $\tau \in sol(cs)$, $\tau \cdot (M \cup \{u, v\} \cup A \cup \{\langle u, v\rangle\}) \vdash \tau \cdot t$ and the cut rule using the conclusion of the derivations above, τ satisfies definition of $sol(\{c\})$, as $\tau \cdot (M \cup \{\langle u, v\rangle\} \cup A) \vdash \tau \cdot t$.

$\rightarrow$

Constraint solving soundness ($red(cs) \subseteq sol(cs)$)

Proof: One-step reduction soundness.

$R = \mathrm{Adec}^{\ell} \quad M \cup \{\{u\}_\iota\} \mid A \rhd t \rightsquigarrow^1_{id} \{M \cup \{u\} \mid A \cup \{\{u\}_\iota\} \rhd t\}$.

$c = M \cup \{\{u\}_\iota\} \mid A \rhd t$, $cs = \{M \cup \{u\} \mid A \cup \{\{u\}_\iota\} \rhd t\}$, and $\sigma = id$.

We use $\tau \in sol(cs)$ and we have to show $\tau \circ id = \tau \in sol(\{c\})$.

$$\frac{\tau \cdot (M \cup \{u\} \cup A \cup \{\{u\}_\iota\}) \vdash \tau \cdot \{u\}_\iota}{\tau \cdot (M \cup A \cup \{\{u\}_\iota\}) \vdash \tau \cdot u}\ \text{Adec}$$

As $\tau \in sol(cs)$, $\tau \cdot (M \cup \{u\} \cup A \cup \{\{u\}_\iota\}) \vdash \tau \cdot t$ and the cut rule using the conclusion of the derivation above, τ satisfies definition of $sol(\{c\})$ using the cut rule, as $\tau \cdot (M \cup \{\{u\}_\iota\} \cup A) \vdash \tau \cdot t$.

$\longrightarrow$

Constraint solving soundness ($red(cs) \subseteq sol(cs)$)

Proof: One-step reduction soundness.

$R = \mathrm{Ksub}^{\ell} \quad M \cup \{\{u\}_x\} \mid A \rhd t \rightsquigarrow^{1}_{[x:=\iota]} \{[x := \iota] \cdot (M \cup \{u\} \mid A \cup \{u\}_x \rhd t)\}$.

$c = M \cup \{\{u\}_x\} \mid A \rhd t$, $cs = \{[x := \iota] \cdot c\}$, and $\sigma = [x := \iota]$.

We use $\tau \in sol(cs)$ and we have to show $\tau \circ [x := \iota] \in sol(\{c\})$.

The $\tau \in sol(cs)$ can be rewrote as $\tau \in sol([x := \iota] \cdot \{c\})$, then directly from Lemma 6 (*If* $\tau \in sol(\sigma \cdot cs)$ *then* $\tau \circ \sigma \in sol(cs)$.) we obtain $\tau \circ [x := \iota] \in sol(\{c\})$.

□

Constraint solving soundness ($red(cs) \subseteq sol(cs)$)

Lemma (Reduction over context soundness)

If $cs \rightsquigarrow_\sigma cs'$ and $\tau \in sol(cs')$ then $\tau \circ \sigma \in sol(cs)$.

Lemma (Transitive and reflexive closure of reduction soundness)

If $cs \rightsquigarrow_\sigma^* cs'$, cs' is simple, and and $\tau \in sol(cs')$ then $\tau \circ \sigma \in sol(cs)$.

Theorem (Soundness)

Constraint solving is sound, i.e. $red(cs) \subseteq sol(cs)$.

Constraint solving termination

Theorem (Termination)

The constraint reduction relation $\rightsquigarrow$ is well-founded.

The number of free variables in constraints never increases.

The number of constraints can increase, but it decreases complexity of constraints.

Proof by case analysis.

Conclusions

We defined formal notation messages of security protocol.

The protocol is then a queue of send and receive messages for each party.

The intruder capabilities are modeled as deduction rules.

Decision procedure based on constraint solving used to verify the protocol.

The protocol execution is symbolic. An information leak is searched by the unification.

Conclusions

We defined formal notation messages of security protocol.

The protocol is then a queue of send and receive messages for each party.

The intruder capabilities are modeled as deduction rules.

Decision procedure based on constraint solving used to verify the protocol.

The protocol execution is symbolic. An information leak is searched by the unification.

Do you have any question?

My questions

Now, the constraint-based analysis' goal is to determine all nonces. Should not be better to look for any information leakage?

The decision procedure can be programmed in Isabelle?

Why do you have ℓ in constraint reduction rules? Why not '? Does the ℓ has some meaning in the context?

Found mistakes

UNIFY algorithm, line 17: $1 \leq i \leq n$

Page 12, (rule Pair_i)

UNIFY, not unify on page 19, in Lemma 7.

Robinson-style unification algorithm

Algorithm: UNIFY(U)

Data: A unification problem U

Result: MGU substitution σ for U

if $U = \emptyset$ **then**
    return id
let $(u, t) \in U$ **and** $U' = U \setminus \{(u, t)\}$
if u is a variable x **then**
    if $x \notin fv(t)$ **then**
        return UNIFY($[x := t] \cdot U') \circ^{\perp} [x := t]$)
    else if $x = t$ **then**
        return UNIFY(U')
    else
        return $\perp$
else if t is a variable x **then**
    return UNIFY($U' \cup \{(t, u)\}$)
else if $u = f(u_1, \ldots, u_n) \wedge t = f(t_1, \ldots, t_n)$ **then**
    return UNIFY($U' \cup \{(u_i, t_i) \mid 1 \leq i \leq n\}$)
else
    return $\perp$

Robinson-style unification algorithm

Algorithm: UNIFY(U)

Data: A unification problem U
Result: MGU substitution σ for U

if $U = \emptyset$ **then**
  return *id*
let $(u, t) \in U$ **and** $U' = U \setminus \{(u, t)\}$
if u is a variable x **then**
  if $x \notin fv(t)$ **then**
    return UNIFY($[x := t] \cdot U') \circ^{\perp} [x := t]$)
  else if $x = t$ **then**
    return UNIFY(U')
  else
    return $\perp$
else if t is a variable x **then**
  return UNIFY($U' \cup \{(t, u)\}$)
else if $u = f(u_1, \ldots, u_n) \wedge t = f(t_1, \ldots, t_n)$ **then**
  return UNIFY($U' \cup \{(u_i, t_i) \mid 1 \leq i \leq n\}$)
else
  return $\perp$

Theorem (Termination)

The algorithm UNIFY terminates on $\forall U$.

	$\lvert fv(U)\rvert$	$\Sigma_{(t,u)\in U}\lvert t\rvert$	$\lvert U\rvert$
Unify	$<$	$\geq$	$<$
Simp	$\leq$	$\leq$	$<$
Occurs	$\perp$	$\perp$	$\perp$
Swap	$=$	$<$	$=$
Fun	$=$	$<$	$\geq$
Fail	$\perp$	$\perp$	$\perp$

Robinson-style unification algorithm

Algorithm: UNIFY(U)

Data: A unification problem U
Result: MGU substitution σ for U

if $U = \emptyset$ **then**
    return *id*
let $(u, t) \in U$ **and** $U' = U \setminus \{(u, t)\}$
if u is a variable x **then**
    if $x \notin fv(t)$ **then**
        return UNIFY($[x := t] \cdot U'$) $\circ^{\perp}$ $[x := t]$)
    else if $x = t$ **then**
        return UNIFY(U')
    else
        return $\perp$
else if t is a variable x **then**
    return UNIFY($U' \cup \{(t, u)\}$)
else if $u = f(u_1, \ldots, u_n) \wedge t = f(t_1, \ldots, t_n)$ **then**
    return UNIFY($U' \cup \{(u_i, t_i) \mid 1 \leq i \leq n\}$)
else
    return $\perp$

Theorem (Soundness)

When the algorithm UNIFY terminates with a σ on the U, then the σ is MGU.

Robinson-style unification algorithm

Algorithm: UNIFY(U)

Data: A unification problem U
Result: MGU substitution σ for U

if $U = \emptyset$ **then**
    return id
let $(u, t) \in U$ **and** $U' = U \setminus \{(u, t)\}$
if u is a variable x **then**
    if $x \notin fv(t)$ **then**
        return UNIFY($[x := t] \cdot U') \circ^{\perp} [x := t]$)
    else if $x = t$ **then**
        return UNIFY(U')
    else
        return $\perp$
else if t is a variable x **then**
    return UNIFY($U' \cup \{(t, u)\}$)
else if $u = f(u_1, \ldots, u_n) \wedge t = f(t_1, \ldots, t_n)$ **then**
    return UNIFY($U' \cup \{(u_i, t_i) \mid 1 \leq i \leq n\}$)
else
    return $\perp$

Theorem (Soundness)

When the algorithm UNIFY terminates with a σ on the U, then the σ is MGU.

Proof by induction:

Base: For $U = \emptyset$ is $\sigma = id$.

Assume: $U' = U \setminus \{(x, t)\}$ and $\sigma = UNIFY([x := t] \cdot U')$.

Step: Show $\sigma' = \sigma \circ^{\perp} [x := t]$ is MGU for U. Let have another unif. of U: $\tau = \rho \circ \sigma$. τ unifies (x, t), so $\tau \circ [x := t] = \rho \circ \sigma \circ [x := t]$ is the same as $\tau = \rho \circ \sigma'$.

Robinson-style unification algorithm

Algorithm: UNIFY(U)

Data: A unification problem U
Result: MGU substitution σ for U

if $U = \emptyset$ **then**
    return id
let $(u, t) \in U$ **and** $U' = U \setminus \{(u, t)\}$
if u is a variable x **then**
    if $x \notin fv(t)$ **then**
        return UNIFY($[x := t] \cdot U') \circ^{\perp} [x := t]$)
    else if $x = t$ **then**
        return UNIFY(U')
    else
        return $\perp$
else if t is a variable x **then**
    return UNIFY($U' \cup \{(t, u)\}$)
else if $u = f(u_1, \ldots, u_n) \wedge t = f(t_1, \ldots, t_n)$ **then**
    return UNIFY($U' \cup \{(u_i, t_i) \mid 1 \leq i \leq n\}$)
else
    return $\perp$

Theorem (Completeness)

If there is a unifier for U then UNIFY(U) returns u unifier for U.

Robinson-style unification algorithm

Algorithm: UNIFY(U)

Data: A unification problem U

Result: MGU substitution σ for U

if $U = \emptyset$ **then**
  return id
let $(u, t) \in U$ **and** $U' = U \setminus \{(u, t)\}$
if u is a variable x **then**
  if $x \notin fv(t)$ **then**
    return UNIFY($[x := t] \cdot U'$) $\circ^{\perp}$ $[x := t]$)
  else if $x = t$ **then**
    return UNIFY(U')
  else
    return $\perp$
else if t is a variable x **then**
  return UNIFY($U' \cup \{(t, u)\}$)
else if $u = f(u_1, \ldots, u_n) \wedge t = f(t_1, \ldots, t_n)$ **then**
  return UNIFY($U' \cup \{(u_i, t_i) \mid 1 \leq i \leq n\}$)
else
  return $\perp$

Theorem (Completeness)

If there is a unifier for U then UNIFY(U) returns u unifier for U.

Proof from soundness and by induction:

Base: For $U = \emptyset$ is $\sigma = id \neq \perp$.

Inspect cases if $\perp$ is possible.