Künstliche Intelligenz

Christoph Benzmüller und Christoph Schommer

Freie Universität Berlin

Sommer/Herbst 2019

First-Order Unification

Freie Universität Berlin

First-Order Terms:

Given a (countable) set of variable symbols V, a (countable) set of constant symbols C, and a (countable) set of n-ary ($n > 0$) function symbols F.

- Each variable symbol $x \in V$ is a term.
- Each constant symbol $c \in C$ is a term.
- Given terms $t_1 \ldots t_n$ ($n > 0$) and an n-ary function symbol $f \in F$, then $f(t_1, \ldots, t_n)$ is a term.

First-Order Formulas:

Given a (countable) set of n-ary ($n \geq 0$) predicate symbols P.

- Given terms $t_1 \ldots t_n$ ($n \geq 0$) and an n-ary predicate symbol $p \in P$, then $p(t_1, \ldots, t_n)$ is a formula.
- Given formulas s and t, then $\neg s$, $s \vee t$, $s \wedge t$ and $s \rightarrow t$ are formulas.

First-Order Terms and Formulas

First-Order Terms:

Given a (countable) set of variable symbols V, a (countable) set of constant symbols C, and a (countable) set of n-ary ($n > 0$) function symbols F.

- Each variable symbol $x \in V$ is a term.
- Each constant symbol $c \in C$ is a term.
- Given terms $t_1 \ldots t_n$ ($n > 0$) and an n-ary function symbol $f \in F$, then $f(t_1, \ldots, t_n)$ is a term.

First-Order Formulas:

Given a (countable) set of n-ary ($n \geq 0$) predicate symbols P.

- Given terms $t_1 \ldots t_n$ ($n \geq 0$) and an n-ary predicate symbol $p \in P$, then $p(t_1, \ldots, t_n)$ is a formula.
- Given formulas s and t, then $\neg s$, $s \vee t$, $s \wedge t$ and $s \rightarrow t$ are formulas.

Substitution:

- replacement of a variable by a (possibly complex) term
- substitutions are functions σ that operate on variables, terms and formulas; instead of $\sigma(t)$ we will write $t\sigma$

Definition — Substitution **1**

A *substitution* is a mapping $\sigma : V \longrightarrow \mathit{Terms}$ from variables to terms.

Freie Universität Berlin

Substitution:

- replacement of a variable by a (possibly complex) term
- substitutions are functions σ that operate on variables, terms and formulas; instead of $\sigma(t)$ we will write $t\sigma$

Definition — Substitution 1

A *substitution* is a mapping $\sigma : V \longrightarrow Terms$ from variables to terms.

Definition — Substitution lifted to Terms 2

Let σ be a substitution. We define:

- If c is a constant symbol, then $c\sigma = c$
- $[f(t_1, \ldots, t_n)]\sigma = f(t_1\sigma, \ldots, t_n\sigma)$ for any $f \in \mathbf{F}$ and $t_1, \ldots, t_n \in \mathbf{T}$

Definition — Composition of Substitutions 3

Let σ and τ be substitutions. By the *composition* of σ and τ, denoted $\sigma\tau$, we mean that substitution such that for each variable x we have $x(\sigma\tau) = (x\sigma)\tau$.

Proposition — Substitution 4

For every term t we have: $t(\sigma\tau) = (t\sigma)\tau$

Proof: By structural induction on t

Proposition — Associativity of Substitution Composition 5

$(\sigma_1\sigma_2)\sigma_3 = \sigma_1(\sigma_2\sigma_3)$

Proof: Let $v \in \mathbf{V}$.

$$v(\sigma_1\sigma_2)\sigma_3 = [v(\sigma_1\sigma_2)]\sigma_3 = [(v\sigma_1)\sigma_2]\sigma_3 = (v\sigma_1)(\sigma_2\sigma_3) = v\sigma_1(\sigma_2\sigma_3)$$

Definition — Composition of Substitutions 3

Let σ and τ be substitutions. By the *composition* of σ and τ, denoted $\sigma\tau$, we mean that substitution such that for each variable x we have $x(\sigma\tau) = (x\sigma)\tau$.

Proposition — Substitution 4

For every term t we have: $t(\sigma\tau) = (t\sigma)\tau$

Proof: By structural induction on t

Proposition — Associativity of Substitution Composition 5

$(\sigma_1\sigma_2)\sigma_3 = \sigma_1(\sigma_2\sigma_3)$

Proof: Let $v \in \mathbf{V}$.

$$v(\sigma_1\sigma_2)\sigma_3 = [v(\sigma_1\sigma_2)]\sigma_3 = [(v\sigma_1)\sigma_2]\sigma_3 = (v\sigma_1)(\sigma_2\sigma_3) = v\sigma_1(\sigma_2\sigma_3)$$

Definition — Composition of Substitutions 3

Let σ and τ be substitutions. By the *composition* of σ and τ, denoted $\sigma\tau$, we mean that substitution such that for each variable x we have $x(\sigma\tau) = (x\sigma)\tau$.

Proposition — Substitution 4

For every term t we have: $t(\sigma\tau) = (t\sigma)\tau$

Proof: By structural induction on t

Proposition — Associativity of Substitution Composition 5

$(\sigma_1\sigma_2)\sigma_3 = \sigma_1(\sigma_2\sigma_3)$

Proof: Let $v \in \mathbf{V}$.

$$v(\sigma_1\sigma_2)\sigma_3 = [v(\sigma_1\sigma_2)]\sigma_3 = [(v\sigma_1)\sigma_2]\sigma_3 = (v\sigma_1)(\sigma_2\sigma_3) = v\sigma_1(\sigma_2\sigma_3)$$

Substitution

Definition — Support of Substitution **6**

The *support* of a substitution σ is the set of variables x for which $x\sigma \neq x$. A substitution has a *finite support* if its support set is finite.

Proposition **7**

The composition of two substitutions with a finite support has again a finite support.

Proof: trivial

Remark: We are typically interested in substitutions with finite support.

Notation: Let $\{x_1 \ldots, x_n\}$ be the finite support of substitution σ. Moreover, assume that $x_i\sigma = t_i$ (for $1 \leq i \leq n$). Then, our notation for σ is: $\{x_1/t_1, \ldots, x_n/t_n\}$.

Definition — Support of Substitution 6

The *support* of a substitution σ is the set of variables x for which $x\sigma \neq x$. A substitution has a *finite support* if its support set is finite.

Proposition 7

The composition of two substitutions with a finite support has again a finite support.

Proof: trivial

Remark: We are typically interested in substitutions with finite support.

Notation: Let $\{x_1 \ldots, x_n\}$ be the finite support of substitution σ. Moreover, assume that $x_i\sigma = t_i$ (for $1 \leq i \leq n$). Then, our notation for σ is: $\{x_1/t_1, \ldots, x_n/t_n\}$.

Definition — Support of Substitution **6**

The *support* of a substitution σ is the set of variables x for which $x\sigma \neq x$. A substitution has a *finite support* if its support set is finite.

Proposition **7**

The composition of two substitutions with a finite support has again a finite support.

Proof: trivial

Remark: We are typically interested in substitutions with finite support.

Notation: Let $\{x_1 \ldots, x_n\}$ be the finite support of substitution σ. Moreover, assume that $x_i\sigma = t_i$ (for $1 \leq i \leq n$). Then, our notation for σ is: $\{x_1/t_1, \ldots, x_n/t_n\}$.

Substitution

Definition — Support of Substitution 6

The *support* of a substitution σ is the set of variables x for which $x\sigma \neq x$. A substitution has a *finite support* if its support set is finite.

Proposition 7

The composition of two substitutions with a finite support has again a finite support.

Proof: trivial

Remark: We are typically interested in substitutions with finite support.

Notation: Let $\{x_1 \ldots, x_n\}$ be the finite support of substitution σ. Moreover, assume that $x_i\sigma = t_i$ (for $1 \leq i \leq n$). Then, our notation for σ is: $\{x_1/t_1, \ldots, x_n/t_n\}$.

Proposition 8

Let $\sigma_1 = \{x_1/t_1, \ldots, x_n/t_n\}$ and $\sigma_2 = \{y_1/u_1, \ldots, y_k/u_k\}$ be substitutions with finite support. The composition $\sigma_1\sigma_2$ has notation $\{x_1/(t_1\sigma_2), \ldots, x_n/(t_n\sigma_2), z_1/(z_1\sigma_2), \ldots, z_m/(z_m\sigma_2)\}$, where $z_1, \ldots, z_m$ are those variables y_i that are not amongst the x_j. (Trivial entries x/x are always deleted).

Example — Substitution 9

$\sigma_1 = \{x/f(x,y), y/h(a), z/g(c,h(x))\}$
$\sigma_2 = \{x/b, y/g(a,x), w/z\}$

Exercise:

... implement substitutions and substitution composition yourself

Proposition 8

Let $\sigma_1 = \{x_1/t_1, \ldots, x_n/t_n\}$ and $\sigma_2 = \{y_1/u_1, \ldots, y_k/u_k\}$ be substitutions with finite support. The composition $\sigma_1\sigma_2$ has notation $\{x_1/(t_1\sigma_2), \ldots, x_n/(t_n\sigma_2), z_1/(z_1\sigma_2), \ldots, z_m/(z_m\sigma_2)\}$, where $z_1, \ldots, z_m$ are those variables y_i that are not amongst the x_j. (Trivial entries x/x are always deleted).

Example — Substitution 9

$\sigma_1 = \{x/f(x,y), y/h(a), z/g(c,h(x))\}$
$\sigma_2 = \{x/b, y/g(a,x), w/z\}$

Exercise:

...implement substitutions and substitution composition yourself

Freie Universität Berlin

Proposition 8

Let $\sigma_1 = \{x_1/t_1, \ldots, x_n/t_n\}$ and $\sigma_2 = \{y_1/u_1, \ldots, y_k/u_k\}$ be substitutions with finite support. The composition $\sigma_1\sigma_2$ has notation $\{x_1/(t_1\sigma_2), \ldots, x_n/(t_n\sigma_2), z_1/(z_1\sigma_2), \ldots, z_m/(z_m\sigma_2)\}$, where $z_1, \ldots, z_m$ are those variables y_i that are not amongst the x_j. (Trivial entries x/x are always deleted).

Example — Substitution 9

$\sigma_1 = \{x/f(x,y), y/h(a), z/g(c,h(x))\}$
$\sigma_2 = \{x/b, y/g(a,x), w/z\}$

Exercise:

…implement substitutions and substitution composition yourself

Unification

- Given two terms t_1 and t_2 of language; both terms may have free variable occurrences, let's say the free variables of t_1 are $u_1, \ldots, u_n$ and the free variables of t_2 are $v_1, \ldots, v_m$.
- Can we instantiate $u_1, \ldots, u_n$ and $v_1, \ldots, v_m$ with terms in such a way that t_1 and t_2 become (syntactically) equal.
- Papers on unification:
  - Jacques Herbrand, Investigations in proof theory, 1930. (For an overview on Herbrand's work see: C.P. Wirth, J. Siekmann, C. Benzmüller, and S. Autexier, Jacques Herbrand: Life, Logic, and Automated Deduction. Handbook of the History of Logic, Volume 5, 2009.)
  - J. A. Robinson, A machine-oriented logic based on the resolution principle, Journal of the ACM 12, 1965.
  - F.Baader and J.Siekmann. Unification theory, Handbook of Logic in Artificial Intelligence and Logic Programming, 1994.
  - F.Baader and W.Snyder, Unification theory, Handbook of Automated Reasoning, 2001.

Unification

- Given two terms t_1 and t_2 of language; both terms may have free variable occurrences, let's say the free variables of t_1 are $u_1, \ldots, u_n$ and the free variables of t_2 are $v_1, \ldots, v_m$.
- Can we instantiate $u_1, \ldots, u_n$ and $v_1, \ldots, v_m$ with terms in such a way that t_1 and t_2 become (syntactically) equal.
- Papers on unification:
  - Jacques Herbrand, Investigations in proof theory, 1930. (For an overview on Herbrand's work see: C.P. Wirth, J. Siekmann, C. Benzmüller, and S. Autexier, Jacques Herbrand: Life, Logic, and Automated Deduction. Handbook of the History of Logic, Volume 5, 2009.)
  - J. A. Robinson, A machine-oriented logic based on the resolution principle, Journal of the ACM 12, 1965.
  - F.Baader and J.Siekmann. Unification theory, Handbook of Logic in Artificial Intelligence and Logic Programming, 1994.
  - F.Baader and W.Snyder, Unification theory, Handbook of Automated Reasoning, 2001.

Unification

- Given two terms t_1 and t_2 of language; both terms may have free variable occurrences, let's say the free variables of t_1 are $u_1, \ldots, u_n$ and the free variables of t_2 are $v_1, \ldots, v_m$.
- Can we instantiate $u_1, \ldots, u_n$ and $v_1, \ldots, v_m$ with terms in such a way that t_1 and t_2 become (syntactically) equal.
- Papers on unification:
  - Jacques Herbrand, Investigations in proof theory, 1930. (For an overview on Herbrand's work see: C.P. Wirth, J. Siekmann, C. Benzmüller, and S. Autexier, Jacques Herbrand: Life, Logic, and Automated Deduction. Handbook of the History of Logic, Volume 5, 2009.)
  - J. A. Robinson, A machine-oriented logic based on the resolution principle, Journal of the ACM 12, 1965.
  - F.Baader and J.Siekmann. Unification theory, Handbook of Logic in Artificial Intelligence and Logic Programming, 1994.
  - F.Baader and W.Snyder, Unification theory, Handbook of Automated Reasoning, 2001.

Unification

- Given two terms t_1 and t_2 of language; both terms may have free variable occurrences, let's say the free variables of t_1 are $u_1, \ldots, u_n$ and the free variables of t_2 are $v_1, \ldots, v_m$.
- Can we instantiate $u_1, \ldots, u_n$ and $v_1, \ldots, v_m$ with terms in such a way that t_1 and t_2 become (syntactically) equal.
- Papers on unification:
  - Jacques Herbrand, Investigations in proof theory, 1930. (For an overview on Herbrand's work see: C.P. Wirth, J. Siekmann, C. Benzmüller, and S. Autexier, Jacques Herbrand: Life, Logic, and Automated Deduction. Handbook of the History of Logic, Volume 5, 2009.)
  - J. A. Robinson, A machine-oriented logic based on the resolution principle, Journal of the ACM 12, 1965.
  - F.Baader and J.Siekmann. Unification theory, Handbook of Logic in Artificial Intelligence and Logic Programming, 1994.
  - F.Baader and W.Snyder, Unification theory, Handbook of Automated Reasoning, 2001.

Unification

- Given two terms t_1 and t_2 of language; both terms may have free variable occurrences, let's say the free variables of t_1 are $u_1, \ldots, u_n$ and the free variables of t_2 are $v_1, \ldots, v_m$.
- Can we instantiate $u_1, \ldots, u_n$ and $v_1, \ldots, v_m$ with terms in such a way that t_1 and t_2 become (syntactically) equal.
- Papers on unification:
  - Jacques Herbrand, Investigations in proof theory, 1930. (For an overview on Herbrand's work see: C.P. Wirth, J. Siekmann, C. Benzmüller, and S. Autexier, Jacques Herbrand: Life, Logic, and Automated Deduction. Handbook of the History of Logic, Volume 5, 2009.)
  - J. A. Robinson, A machine-oriented logic based on the resolution principle, Journal of the ACM 12, 1965.
  - F.Baader and J.Siekmann. Unification theory, Handbook of Logic in Artificial Intelligence and Logic Programming, 1994.
  - F.Baader and W.Snyder, Unification theory, Handbook of Automated Reasoning, 2001.

Unification

- Given two terms t_1 and t_2 of language; both terms may have free variable occurrences, let's say the free variables of t_1 are $u_1, \ldots, u_n$ and the free variables of t_2 are $v_1, \ldots, v_m$.
- Can we instantiate $u_1, \ldots, u_n$ and $v_1, \ldots, v_m$ with terms in such a way that t_1 and t_2 become (syntactically) equal.
- Papers on unification:
  - Jacques Herbrand, Investigations in proof theory, 1930. (For an overview on Herbrand's work see: C.P. Wirth, J. Siekmann, C. Benzmüller, and S. Autexier, Jacques Herbrand: Life, Logic, and Automated Deduction. Handbook of the History of Logic, Volume 5, 2009.)
  - J. A. Robinson, A machine-oriented logic based on the resolution principle, Journal of the ACM 12, 1965.
  - F.Baader and J.Siekmann. Unification theory, Handbook of Logic in Artificial Intelligence and Logic Programming, 1994.
  - F.Baader and W.Snyder, Unification theory, Handbook of Automated Reasoning, 2001.

Unification

- Given two terms t_1 and t_2 of language; both terms may have free variable occurrences, let's say the free variables of t_1 are $u_1, \ldots, u_n$ and the free variables of t_2 are $v_1, \ldots, v_m$.
- Can we instantiate $u_1, \ldots, u_n$ and $v_1, \ldots, v_m$ with terms in such a way that t_1 and t_2 become (syntactically) equal.
- Papers on unification:
  - Jacques Herbrand, Investigations in proof theory, 1930. (For an overview on Herbrand's work see: C.P. Wirth, J. Siekmann, C. Benzmüller, and S. Autexier, Jacques Herbrand: Life, Logic, and Automated Deduction. Handbook of the History of Logic, Volume 5, 2009.)
  - J. A. Robinson, A machine-oriented logic based on the resolution principle, Journal of the ACM 12, 1965.
  - F.Baader and J.Siekmann. Unification theory, Handbook of Logic in Artificial Intelligence and Logic Programming, 1994.
  - F.Baader and W.Snyder, Unification theory, Handbook of Automated Reasoning, 2001.

Freie Universität Berlin

Definition — More General Substitution **10**

Let σ_1 and σ_2 be substitutions. We say σ_2 is more general than σ_1 if, for some substitution τ, $\sigma_1 = \sigma_2\tau$.

Example **11**

1. Show that $\sigma_2 = \{x/f(g(x,y)), y/g(z,b)\}$ is more general than $\sigma_1 = \{x/f(g(a,h(z))), y/g(h(x),b), z/h(x)\}$.
2. Is σ_2 more general than σ_1?

Definition — More General Substitution **10**

Let σ_1 and σ_2 be substitutions. We say σ_2 is more general than σ_1 if, for some substitution τ, $\sigma_1 = \sigma_2\tau$.

Example **11**

1. Show that $\sigma_2 = \{x/f(g(x,y)), y/g(z,b)\}$ is more general than $\sigma_1 = \{x/f(g(a,h(z))), y/g(h(x),b), z/h(x)\}$.
2. Is σ_2 more general than σ_1?

Proposition — Transitivity of 'More general' 12

If σ_3 is more general than σ_2 and σ_2 is more general than σ_1, then σ_3 is more general than σ_1.

Proof: We know $\sigma_1 = \sigma_2\tau$ and $\sigma_2 = \sigma_3\theta$.
But then $\sigma_1 = \sigma_2\tau = (\sigma_3\theta)\tau = \sigma_3(\theta\tau)$.

Definition — Unifier/Most General Unifier (MGU) 13

Let t_1 and t_2 be terms. A substitution σ is a *unifier for t_1 and t_2* is $t_1\sigma = t_2\sigma$. t_1 and t_2 are *unifiable* if they have a unifier. A substitution is a *most general unifier MGU (of t_1 and t_2)* if it is a unifier and more general than any other unifier of t_1 and t_2. (These notions do extend to sets of terms in the obvious way).

Proposition — Transitivity of 'More general' 12

If σ_3 is more general than σ_2 and σ_2 is more general than σ_1, then σ_3 is more general than σ_1.

Proof: We know $\sigma_1 = \sigma_2\tau$ and $\sigma_2 = \sigma_3\theta$.
But then $\sigma_1 = \sigma_2\tau = (\sigma_3\theta)\tau = \sigma_3(\theta\tau)$.

Definition — Unifier/Most General Unifier (MGU) 13

Let t_1 and t_2 be terms. A substitution σ is a *unifier for t_1 and t_2* is $t_1\sigma = t_2\sigma$. t_1 and t_2 are *unifiable* if they have a unifier. A substitution is a *most general unifier MGU (of t_1 and t_2)* if it is a unifier and more general than any other unifier of t_1 and t_2. (These notions do extend to sets of terms in the obvious way).

Proposition — Transitivity of 'More general' 12

If σ_3 is more general than σ_2 and σ_2 is more general than σ_1, then σ_3 is more general than σ_1.

Proof: We know $\sigma_1 = \sigma_2\tau$ and $\sigma_2 = \sigma_3\theta$.
But then $\sigma_1 = \sigma_2\tau = (\sigma_3\theta)\tau = \sigma_3(\theta\tau)$.

Definition — Unifier/Most General Unifier (MGU) 13

Let t_1 and t_2 be terms. A substitution σ is a *unifier for t_1 and t_2* is $t_1\sigma = t_2\sigma$. t_1 and t_2 are *unifiable* if they have a unifier. A substitution is a *most general unifier MGU (of t_1 and t_2)* if it is a unifier and more general than any other unifier of t_1 and t_2. (These notions do extend to sets of terms in the obvious way).

Freie Universität Berlin

Proposition — Transitivity of 'More general' 12

If σ_3 is more general than σ_2 and σ_2 is more general than σ_1, then σ_3 is more general than σ_1.

Proof: We know $\sigma_1 = \sigma_2\tau$ and $\sigma_2 = \sigma_3\theta$.
But then $\sigma_1 = \sigma_2\tau = (\sigma_3\theta)\tau = \sigma_3(\theta\tau)$.

Definition — Unifier/Most General Unifier (MGU) 13

Let t_1 and t_2 be terms. A substitution σ is a *unifier for t_1 and t_2* is $t_1\sigma = t_2\sigma$. t_1 and t_2 are *unifiable* if they have a unifier. A substitution is a *most general unifier MGU (of t_1 and t_2)* if it is a unifier and more general than any other unifier of t_1 and t_2.
(These notions do extend to sets of terms in the obvious way).

Example **14**

$f(y, h(a))$ and $f(h(x), h(z))$ unifiable with

1. $\{y/h(x), z/a\}$.
2. $\{x/k(w), y/h(k(w)), z/a\}$.

Which one is more general?

Note: Technically, two terms t_1 and t_2 may have more than just one most general unifier (consider $g(x, x)$ and $g(y, z)$), but if so then they are the same up to a variable renaming.

Example **14**

$f(y, h(a))$ and $f(h(x), h(z))$ unifiable with

1. $\{y/h(x), z/a\}$.
2. $\{x/k(w), y/h(k(w)), z/a\}$.

Which one is more general?

Note: Technically, two terms t_1 and t_2 may have more than just one most general unifier (consider $g(x, x)$ and $g(y, z)$), but if so then they are the same up to a variable renaming.

Definition — Variable Renaming **15**

A substitution η is a *variable renaming* for a set V of variables if

1. For each $x \in V$, $x\eta$ is a variable.
2. For $x, y \in V$ with $x \neq y$, $x\eta$ and $y\eta$ are distinct.

Definition — Variable Range **16**

The *variable range* for a substitution σ is the set of variables that occur in terms of the forms $x\sigma$, where x is a variable.

Proposition — Most General Unifiers **17**

Suppose both σ_1 and σ_2 are most general unifiers of t_1 and t_2. Then there is a variable renaming η for the variable range of σ such that $\sigma_1\eta = \sigma_2$.

Proof: ... straightforward, not here ...

Definition — Variable Renaming **15**

A substitution η is a *variable renaming* for a set V of variables if

1. For each $x \in V$, $x\eta$ is a variable.
2. For $x, y \in V$ with $x \neq y$, $x\eta$ and $y\eta$ are distinct.

Definition — Variable Range **16**

The *variable range* for a substitution σ is the set of variables that occur in terms of the forms $x\sigma$, where x is a variable.

Proposition — Most General Unifiers **17**

Suppose both σ_1 and σ_2 are most general unifiers of t_1 and t_2. Then there is a variable renaming η for the variable range of σ such that $\sigma_1\eta = \sigma_2$.

Proof: ...straightforward, not here ...

Freie Universität Berlin

Definition — Variable Renaming **15**

A substitution η is a *variable renaming* for a set V of variables if

1. For each $x \in V$, $x\eta$ is a variable.
2. For $x, y \in V$ with $x \neq y$, $x\eta$ and $y\eta$ are distinct.

Definition — Variable Range **16**

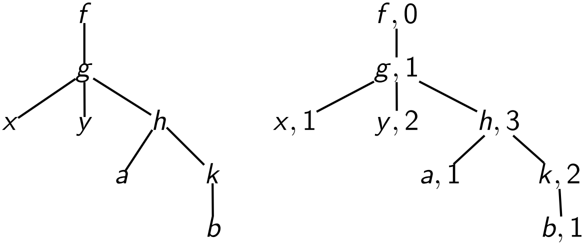
The *variable range* for a substitution σ is the set of variables that occur in terms of the forms $x\sigma$, where x is a variable.

Proposition — Most General Unifiers **17**

Suppose both σ_1 and σ_2 are most general unifiers of t_1 and t_2. Then there is a variable renaming η for the variable range of σ such that $\sigma_1\eta = \sigma_2$.

Proof: ... straightforward, not here ...

Augmented Tree Representation for: $f(g(x, y, h(a, k(b))))$.



Allows us to talk about paths trough a term, e.g.
$\langle f, 0\rangle, \langle g, 1\rangle, \langle h, 3\rangle, \langle k, 2\rangle$

Freie Universität Berlin

Definition — Disagreement Pair 18

A *disagreement pair* for terms t_1 and t_2 is a pair of terms $[d_1, d_2]$, such that

- d_1 is a subterm of t_1 and d_2 is a subterm of t_2, and
- thinking of terms as augmented trees, d_1 and d_2 have distinct labels at their roots,
- while the path from the root of t_1 down to the root of d_1 and the path from the root of t_2 down to the root of d_2 are the same.

First-Order Logic: Unification (Robinson)

Definition — Disagreement Pair 18

A *disagreement pair* for terms t_1 and t_2 is a pair of terms $[d_1, d_2]$, such that

- d_1 is a subterm of t_1 and d_2 is a subterm of t_2, and
- thinking of terms as augmented trees, d_1 and d_2 have distinct labels at their roots,
- while the path from the root of t_1 down to the root of d_1 and the path from the root of t_2 down to the root of d_2 are the same.

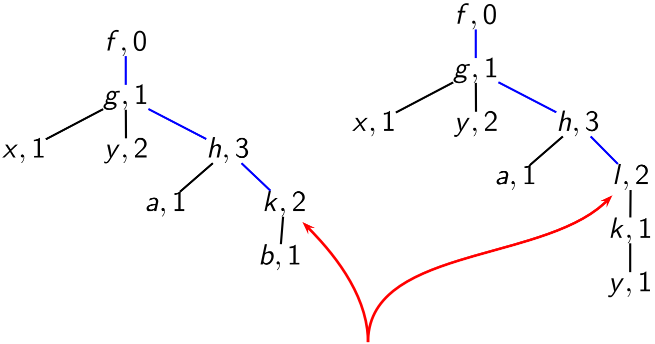
First-Order Logic: Unification (Robinson)

Definition — Disagreement Pair 18

A *disagreement pair* for terms t_1 and t_2 is a pair of terms $[d_1, d_2]$, such that

- d_1 is a subterm of t_1 and d_2 is a subterm of t_2, and
- thinking of terms as augmented trees, d_1 and d_2 have distinct labels at their roots,
- while the path from the root of t_1 down to the root of d_1 and the path from the root of t_2 down to the root of d_2 are the same.

Disagreement Pair for terms
$f(g(x, y, h(a, k(b))))$ and $f(g(x, y, h(a, l(k(y)))))$.



Disagreement pair: $[k(b), l(k(y))]$

Unification Algorithm (Robinson)

Let $\sigma := \epsilon$;
While $t_1\sigma \neq t_2\sigma$ do
  begin
  choose a disagreement pair $[d_1, d_2]$ for $t_1\sigma$ and $t_2\sigma$;
  if neither d_1 nor d_2 is a variable then FAIL;
  let x be whichever of d_1 and d_2 is a variable
    (if both are, choose one)
    and let t be the other one of d_1, d_2;
  if x occurs in t then FAIL;
  let $\sigma := \sigma\{x/t\}$;
  end.

Theorem — Unification Theorem **19**

Given two terms t_1 and t_2.

- *If t_1 and t_2 are not unifiable, then the Unification Algorithm will FAIL.*
- *If t_1 and t_2 are unifiable, then the Unification Algorithm will terminate without FAILure and the final value of σ will be a most general unifier of t_1 and t_2.*

Proof: ...not here ...

Definition — Idempotent Substitution **20**

A substitution σ is called *idempotent* if $\sigma = \sigma\sigma$

Corollary **21**

If t_1 and t_2 are unifiable, the Unification Algorithm terminates with a final value that is an idempotent most general unifier for them.

Unification (Robinson)

Theorem — Unification Theorem **19**

Given two terms t_1 and t_2.

- *If t_1 and t_2 are not unifiable, then the Unification Algorithm will FAIL.*
- *If t_1 and t_2 are unifiable, then the Unification Algorithm will terminate without FAILure and the final value of σ will be a most general unifier of t_1 and t_2.*

Proof: ...not here ...

Definition — Idempotent Substitution **20**

A substitution σ is called *idempotent* if $\sigma = \sigma\sigma$

Corollary **21**

If t_1 and t_2 are unifiable, the Unification Algorithm terminates with a final value that is an idempotent most general unifier for them.

Unification (Robinson)

Theorem — Unification Theorem **19**

Given two terms t_1 and t_2.

- *If t_1 and t_2 are not unifiable, then the Unification Algorithm will FAIL.*
- *If t_1 and t_2 are unifiable, then the Unification Algorithm will terminate without FAILure and the final value of σ will be a most general unifier of t_1 and t_2.*

Proof: . . . not here . . .

Definition — Idempotent Substitution **20**

A substitution σ is called *idempotent* if $\sigma = \sigma\sigma$

Corollary **21**

If t_1 and t_2 are unifiable, the Unification Algorithm terminates with a final value that is an idempotent most general unifier for them.

Unification (Robinson)

Theorem — Unification Theorem **19**

Given two terms t_1 and t_2.

- *If t_1 and t_2 are not unifiable, then the Unification Algorithm will FAIL.*
- *If t_1 and t_2 are unifiable, then the Unification Algorithm will terminate without FAILure and the final value of σ will be a most general unifier of t_1 and t_2.*

Proof: . . . not here . . .

Definition — Idempotent Substitution **20**

A substitution σ is called *idempotent* if $\sigma = \sigma\sigma$

Corollary **21**

If t_1 and t_2 are unifiable, the Unification Algorithm terminates with a final value that is an idempotent most general unifier for them.

Unification (Robinson)

Idempotent most general unifiers have some nice features, e.g.:

Proposition **22**

Suppose σ is an idempotent most general unifier for t_1 and t_2, and τ is any unifier. Then $\tau = \sigma\tau$.

Unification (Robinson)

Multiple Unification as a sequence of binary unifications:
Given: set of terms $\{t_0, t_1, t_2, \ldots, t_n\}$
Unifier: substitution σ such that $t_0\sigma = t_1\sigma = t_2\sigma = \ldots = t_n\sigma$
Most general unifier: one that is more general than any other unifier

Suppose $\{t_0, t_1, t_2, \ldots, t_n\}$ has a unifier. Then the computation of a most general unifier for this set of terms can be reduced to a sequence of binary unification problems as follows:

- σ_1: idempotent most general unifier of t_o and t_1
- σ_2: idempotent most general unifier of $t_o\sigma_1$ and $t_2\sigma_1$
- σ_3: idempotent most general unifier of $t_o\sigma_2$ and $t_3\sigma_2$

...

- σ_n: idempotent most general unifier of $t_o\sigma_{n-1}$ and $t_n\sigma_{n-1}$

Then, $\sigma := \sigma_1\sigma_2\sigma_3\ldots\sigma_n$ is a MGU of $\{t_0, t_1, t_2, \ldots, t_n\}$.

Unification (Robinson)

Multiple Unification as a sequence of binary unifications:
Given: set of terms $\{t_0, t_1, t_2, \ldots, t_n\}$
Unifier: substitution σ such that $t_0\sigma = t_1\sigma = t_2\sigma = \ldots = t_n\sigma$
Most general unifier: one that is more general than any other unifier

Suppose $\{t_0, t_1, t_2, \ldots, t_n\}$ has a unifier. Then the computation of a most general unifier for this set of terms can be reduced to a sequence of binary unification problems as follows:

- σ_1: idempotent most general unifier of t_o and t_1
- σ_2: idempotent most general unifier of $t_o\sigma_1$ and $t_2\sigma_1$
- σ_3: idempotent most general unifier of $t_o\sigma_2$ and $t_3\sigma_2$

...

- σ_n: idempotent most general unifier of $t_o\sigma_{n-1}$ and $t_n\sigma_{n-1}$

Then, $\sigma := \sigma_1\sigma_2\sigma_3\ldots\sigma_n$ is a MGU of $\{t_0, t_1, t_2, \ldots, t_n\}$.

Unification (Robinson)

Multiple Unification as a sequence of binary unifications:
Given: set of terms $\{t_0, t_1, t_2, \ldots, t_n\}$
Unifier: substitution σ such that $t_0\sigma = t_1\sigma = t_2\sigma = \ldots = t_n\sigma$
Most general unifier: one that is more general than any other unifier

Suppose $\{t_0, t_1, t_2, \ldots, t_n\}$ has a unifier. Then the computation of a most general unifier for this set of terms can be reduced to a sequence of binary unification problems as follows:

- σ_1: idempotent most general unifier of t_o and t_1
- σ_2: idempotent most general unifier of $t_o\sigma_1$ and $t_2\sigma_1$
- σ_3: idempotent most general unifier of $t_o\sigma_2$ and $t_3\sigma_2$

...

- σ_n: idempotent most general unifier of $t_o\sigma_{n-1}$ and $t_n\sigma_{n-1}$

Then, $\sigma := \sigma_1\sigma_2\sigma_3\ldots\sigma_n$ is a MGU of $\{t_0, t_1, t_2, \ldots, t_n\}$.

Multiple Unification as a sequence of binary unifications:
Given: set of terms $\{t_0, t_1, t_2, \ldots, t_n\}$
Unifier: substitution σ such that $t_0\sigma = t_1\sigma = t_2\sigma = \ldots = t_n\sigma$
Most general unifier: one that is more general than any other unifier

Suppose $\{t_0, t_1, t_2, \ldots, t_n\}$ has a unifier. Then the computation of a most general unifier for this set of terms can be reduced to a sequence of binary unification problems as follows:

- σ_1: idempotent most general unifier of t_o and t_1
- σ_2: idempotent most general unifier of $t_o\sigma_1$ and $t_2\sigma_1$
- σ_3: idempotent most general unifier of $t_o\sigma_2$ and $t_3\sigma_2$

...

- σ_n: idempotent most general unifier of $t_o\sigma_{n-1}$ and $t_n\sigma_{n-1}$

Then, $\sigma := \sigma_1\sigma_2\sigma_3\ldots\sigma_n$ is a MGU of $\{t_0, t_1, t_2, \ldots, t_n\}$.

Unification (Robinson)

Multiple Unification as a sequence of binary unifications:
Given: set of terms $\{t_0, t_1, t_2, \dots, t_n\}$
Unifier: substitution σ such that $t_0\sigma = t_1\sigma = t_2\sigma = \dots = t_n\sigma$
Most general unifier: one that is more general than any other unifier

Suppose $\{t_0, t_1, t_2, \dots, t_n\}$ has a unifier. Then the computation of a most general unifier for this set of terms can be reduced to a sequence of binary unification problems as follows:

- σ_1: idempotent most general unifier of t_o and t_1
- σ_2: idempotent most general unifier of $t_o\sigma_1$ and $t_2\sigma_1$
- σ_3: idempotent most general unifier of $t_o\sigma_2$ and $t_3\sigma_2$

...

- σ_n: idempotent most general unifier of $t_o\sigma_{n-1}$ and $t_n\sigma_{n-1}$

Then, $\sigma := \sigma_1\sigma_2\sigma_3 \dots \sigma_n$ is a MGU of $\{t_0, t_1, t_2, \dots, t_n\}$.

Unification (Robinson)

Multiple Unification as a sequence of binary unifications:
Given: set of terms $\{t_0, t_1, t_2, \ldots, t_n\}$
Unifier: substitution σ such that $t_0\sigma = t_1\sigma = t_2\sigma = \ldots = t_n\sigma$
Most general unifier: one that is more general than any other unifier

Suppose $\{t_0, t_1, t_2, \ldots, t_n\}$ has a unifier. Then the computation of a most general unifier for this set of terms can be reduced to a sequence of binary unification problems as follows:

- σ_1: idempotent most general unifier of t_o and t_1
- σ_2: idempotent most general unifier of $t_o\sigma_1$ and $t_2\sigma_1$
- σ_3: idempotent most general unifier of $t_o\sigma_2$ and $t_3\sigma_2$

...

- σ_n: idempotent most general unifier of $t_o\sigma_{n-1}$ and $t_n\sigma_{n-1}$

Then, $\sigma := \sigma_1\sigma_2\sigma_3\ldots\sigma_n$ is a MGU of $\{t_0, t_1, t_2, \ldots, t_n\}$.

Unification (Martelli/Montanari)

Given: multi-set $E := \{s_1 = t_1, \ldots, s_n = t_n\}$
Unifier of E: substitution σ such that $s_i\sigma = t_i\sigma$ (for all $1 \leq i \leq n$)

Unification after Martinelli/Montanari

$$t = t, E \longrightarrow_{mm} E$$

$$f(s_1, \ldots, s_n) = f(t_1, \ldots, t_n), E \longrightarrow_{mm} s_1 = t_1, \ldots, s_n = t_n, E$$

$$f(\ldots) = g(\ldots) \longrightarrow_{mm} FAIL$$

$$x = t, E \longrightarrow_{mm} x = t, E\{x/t\} \quad \text{(if } x \text{ does not occur free in } t\text{)}$$

$$x = t, E \longrightarrow_{mm} FAIL \quad \text{(if } x \text{ occurs free in } t\text{)}$$

$$t = x, E \longrightarrow_{mm} x = t, E \quad \text{(if } t \text{ is not a variable)}$$

Unification (Martelli/Montanari)

Given: multi-set $E := \{s_1 = t_1, \ldots, s_n = t_n\}$
Unifier of E: substitution σ such that $s_i\sigma = t_i\sigma$ (for all $1 \leq i \leq n$)

Unification after Martinelli/Montanari

$$t = t, E \longrightarrow_{mm} E$$

$$f(s_1, \ldots, s_n) = f(t_1, \ldots, t_n), E \longrightarrow_{mm} s_1 = t_1, \ldots, s_n = t_n, E$$

$$f(\ldots) = g(\ldots) \longrightarrow_{mm} FAIL$$

$$x = t, E \longrightarrow_{mm} x = t, E\{x/t\} \quad \text{(if } x \text{ does not occur free in } t\text{)}$$

$$x = t, E \longrightarrow_{mm} FAIL \quad \text{(if } x \text{ occurs free in } t\text{)}$$

$$t = x, E \longrightarrow_{mm} x = t, E \quad \text{(if } t \text{ is not a variable)}$$

Given: multi-set $E := \{s_1 = t_1, \ldots, s_n = t_n\}$
Unifier of E: substitution σ such that $s_i\sigma = t_i\sigma$ (for all $1 \leq i \leq n$)

Unification after Martinelli/Montanari

$$t = t, E \longrightarrow_{mm} E$$

$$f(s_1, \ldots, s_n) = f(t_1, \ldots, t_n), E \longrightarrow_{mm} s_1 = t_1, \ldots, s_n = t_n, E$$

$$f(\ldots) = g(\ldots) \longrightarrow_{mm} FAIL$$

$$x = t, E \longrightarrow_{mm} x = t, E\{x/t\} \quad \text{(if } x \text{ does not occur free in } t\text{)}$$

$$x = t, E \longrightarrow_{mm} FAIL \quad \text{(if } x \text{ occurs free in } t\text{)}$$

$$t = x, E \longrightarrow_{mm} x = t, E \quad \text{(if } t \text{ is not a variable)}$$

Unification (Martelli/Montanari)

Given: multi-set $E := \{s_1 = t_1, \ldots, s_n = t_n\}$
Unifier of E: substitution σ such that $s_i\sigma = t_i\sigma$ (for all $1 \leq i \leq n$)

Unification after Martinelli/Montanari

$$t = t, E \longrightarrow_{mm} E$$

$$f(s_1, \ldots, s_n) = f(t_1, \ldots, t_n), E \longrightarrow_{mm} s_1 = t_1, \ldots, s_n = t_n, E$$

$$f(\ldots) = g(\ldots) \longrightarrow_{mm} FAIL$$

$$x = t, E \longrightarrow_{mm} x = t, E\{x/t\} \quad \text{(if } x \text{ does not occur free in } t\text{)}$$

$$x = t, E \longrightarrow_{mm} FAIL \quad \text{(if } x \text{ occurs free in } t\text{)}$$

$$t = x, E \longrightarrow_{mm} x = t, E \quad \text{(if } t \text{ is not a variable)}$$

Unification (Martelli/Montanari)

Given: multi-set $E := \{s_1 = t_1, \ldots, s_n = t_n\}$
Unifier of E: substitution σ such that $s_i\sigma = t_i\sigma$ (for all $1 \leq i \leq n$)

Unification after Martinelli/Montanari

$$t = t, E \longrightarrow_{mm} E$$

$$f(s_1, \ldots, s_n) = f(t_1, \ldots, t_n), E \longrightarrow_{mm} s_1 = t_1, \ldots, s_n = t_n, E$$

$$f(\ldots) = g(\ldots) \longrightarrow_{mm} FAIL$$

$$x = t, E \longrightarrow_{mm} x = t, E\{x/t\} \quad \text{(if } x \text{ does not occur free in } t\text{)}$$

$$x = t, E \longrightarrow_{mm} FAIL \quad \text{(if } x \text{ occurs free in } t\text{)}$$

$$t = x, E \longrightarrow_{mm} x = t, E \quad \text{(if } t \text{ is not a variable)}$$

Unification (Martelli/Montanari)

Given: multi-set $E := \{s_1 = t_1, \ldots, s_n = t_n\}$
Unifier of E: substitution σ such that $s_i\sigma = t_i\sigma$ (for all $1 \leq i \leq n$)

Unification after Martinelli/Montanari

$$t = t, E \longrightarrow_{mm} E$$

$$f(s_1, \ldots, s_n) = f(t_1, \ldots, t_n), E \longrightarrow_{mm} s_1 = t_1, \ldots, s_n = t_n, E$$

$$f(\ldots) = g(\ldots) \longrightarrow_{mm} FAIL$$

$$x = t, E \longrightarrow_{mm} x = t, E\{x/t\} \quad \text{(if } x \text{ does not occur free in } t\text{)}$$

$$x = t, E \longrightarrow_{mm} FAIL \quad \text{(if } x \text{ occurs free in } t\text{)}$$

$$t = x, E \longrightarrow_{mm} x = t, E \quad \text{(if } t \text{ is not a variable)}$$

Unification (Martelli/Montanari)

Given: multi-set $E := \{s_1 = t_1, \ldots, s_n = t_n\}$
Unifier of E: substitution σ such that $s_i\sigma = t_i\sigma$ (for all $1 \leq i \leq n$)

Unification after Martinelli/Montanari

$$t = t, E \longrightarrow_{mm} E$$

$$f(s_1, \ldots, s_n) = f(t_1, \ldots, t_n), E \longrightarrow_{mm} s_1 = t_1, \ldots, s_n = t_n, E$$

$$f(\ldots) = g(\ldots) \longrightarrow_{mm} FAIL$$

$$x = t, E \longrightarrow_{mm} x = t, E\{x/t\} \quad \text{(if } x \text{ does not occur free in } t\text{)}$$

$$x = t, E \longrightarrow_{mm} FAIL \quad \text{(if } x \text{ occurs free in } t\text{)}$$

$$t = x, E \longrightarrow_{mm} x = t, E \quad \text{(if } t \text{ is not a variable)}$$

Definition — Solved Form **23**

If $E := \{x_1 = t_1, \ldots, x_n = t_n\}$, with x_i being pairwise distinct variables and where x_i does not occur in the free variables of t_i, then E is called in *solved form* representing a solution $\sigma_E = \{x_1/t_1, \ldots, x_n/t_n\}$.

Theorem **24**

If E is in solved form then σ_E is a most general unifier of E.

Theorem **25**

1. *If $E \longrightarrow_{mm} E'$ then σ is a unifier of E iff σ is a unifier of E'*
2. *If $E \longrightarrow^*_{mm}$ FAIL then E is not unifiable.*
3. *If $E \longrightarrow^*_{mm} E'$ with E' in solved form, then σ_E is a MGU of E*

Unification (Martelli/Montanari)

Definition — Solved Form **23**

If $E := \{x_1 = t_1, \ldots, x_n = t_n\}$, with x_i being pairwise distinct variables and where x_i does not occur in the free variables of t_i, then E is called in *solved form* representing a solution $\sigma_E = \{x_1/t_1, \ldots, x_n/t_n\}$.

Theorem **24**

If E is in solved form then σ_E is a most general unifier of E.

Theorem **25**

1. *If $E \longrightarrow_{mm} E'$ then σ is a unifier of E iff σ is a unifier of E'*
2. *If $E \longrightarrow^*_{mm}$ FAIL then E is not unifiable.*
3. *If $E \longrightarrow^*_{mm} E'$ with E' in solved form, then σ_E is a MGU of E*

Unification (Martelli/Montanari)

Definition — Solved Form **23**

If $E := \{x_1 = t_1, \ldots, x_n = t_n\}$, with x_i being pairwise distinct variables and where x_i does not occur in the free variables of t_i, then E is called in *solved form* representing a solution $\sigma_E = \{x_1/t_1, \ldots, x_n/t_n\}$.

Theorem **24**

If E is in solved form then σ_E is a most general unifier of E.

Theorem **25**

1. *If $E \longrightarrow_{mm} E'$ then σ is a unifier of E iff σ is a unifier of E'*
2. *If $E \longrightarrow^*_{mm}$ FAIL then E is not unifiable.*
3. *If $E \longrightarrow^*_{mm} E'$ with E' in solved form, then σ_E is a MGU of E*

Complexity of Unification

Some Literature

- **Paterson, Wegman: Linear Unification, JCSS 17, 1978**
  Unifiability is decidable in linear time. A most general unifier can be computed in linear time.
- **Dwork, Kanellakis, Mitchell: On the sequential nature of unification, J.Log.Progr. 1, 1984**
  Unifiability is log-space complete for P, that is, every problem in P can be reduced in log-space to a unifiability problem. Thus, most likely, unifiability cannot be efficiently paralllized.
- **Baader, Nipkow: Term rewriting and all that. 1998.**
  A very good introduction and overview.