Lecture 15

Pete Manolios
Northeastern

Unification Basics

- **Unification Problem**: Given a set of pairs of terms $S = \{(s_1,t_1), \ldots, (s_n,t_n)\}$ a *unifier* of S is a substitution σ such that $s_i|\sigma = t_i|\sigma$ (we'll write $s_i\sigma = t_i\sigma$)
- ***U*(*S*)** is the set of all unifiers of *S;* notice that if σ is a unifier, so is $\tau \circ \sigma$
- σ is ***more general*** than τ, $\boldsymbol{\sigma \leq \tau}$, iff $\tau = \delta\sigma$ $(\delta \circ \sigma)$ for some substitution δ
- $\leq$ is a preorder; let δ be the identify for reflexivity
  - transitivity: if $\sigma \leq \tau$, $\tau \leq \theta$ then $\tau = \delta\sigma$, $\theta = \gamma\tau = \gamma(\delta\sigma) = (\gamma\delta)\sigma$
  - $\sigma \sim \tau$ iff $\sigma \leq \tau$, $\tau \leq \sigma$. Notice that if $\sigma = x \leftarrow y$, $\tau = y \leftarrow x$, then $\sigma \sim \tau$
  - $\sigma \sim \tau$ iff there is a *renaming* (bijection on Vars) θ s.t. $\sigma = \theta\tau$
- A ***most general unifier* (mgu)** is $\sigma \in U(S)$ s.t. for all $\tau \in U(S)$, $\sigma \leq \tau$
  - What is an mgu for $x=y$? $x \leftarrow y$? $y \leftarrow x$? $x \leftarrow z, y \leftarrow z$? $x \leftarrow y, z \leftarrow w, w \leftarrow z$?
- A substitution is ***idempotent*** if $\sigma\sigma = \sigma$ (rules out last case above)
  - σ is idempotent iff Domain(σ) is disjoint from Vars(Range(σ))
- If a unification problem has a solution, then it has an idempotent mgu
- We want an algorithm that finds an mgu, if a unifier exists

Unification Algorithm

- $S = \{(x_1,t_1), \ldots, (x_n,t_n)\}$ is in ***solved form*** if the x_i are distinct variables and don't occur in any of the t_i. Then $S\!\downarrow = \{x_1 \leftarrow t_1, \ldots, x_n \leftarrow t_n\}$
- If S is in solved form and $\sigma \in U(S)$, then $\sigma = \sigma S\!\downarrow$ (σ, $\sigma S\!\downarrow$ agree on all vars)
- If S is in solved form, then $S\!\downarrow$ is an idempotent mgu
- Algorithm: *Nondeterministic transition system* based on the following rules
  - Delete $\{t=t\} \uplus S \quad \Longrightarrow S$ useful way of thinking about algorithms: SMT/IMT
  - Decompose $\{f(t_1, \ldots, t_n) = f(s_1, \ldots, s_n)\} \uplus S \Longrightarrow \{t_1=s_1, \ldots, t_n=s_n\} \cup S$
  - Orient $\{t=x\} \uplus S \quad \Longrightarrow \{x=t\} \cup S$, if t is not a variable
  - Eliminate $\{x=t\} \uplus S \Longrightarrow \{x=t\} \cup S|x \leftarrow t$, if $x \in \mathrm{Vars}(S) - \mathrm{Vars}(t)$
- **Unify(S)** = apply rules nondeterministically; if solved return $S\!\downarrow$, else fail
- Try it with: $\{x=f(a), g(x,x)=g(x,y)\}$

Unification Algorithm

- Algorithm: Nondeterministic transition system based on the following rules
  - Delete $\{t=t\} \uplus S \implies S$
  - Decompose $\{f(t_1, \ldots, t_n) = f(s_1, \ldots, s_n)\} \uplus S \implies \{t_1=s_1, \ldots, t_n=s_n\} \cup S$
  - Orient $\{t=x\} \uplus S \implies \{x=t\} \cup S$, if t is not a variable
  - Eliminate $\{x=t\} \uplus S \implies \{x=t\} \cup S|x{\leftarrow}t$, if $x \in \text{Vars}(S) - \text{Vars}(t)$

$x=f(a), g(x,x)=g(x,y)$	$\implies$ decompose	what other rules can I use?
$x=f(a), x=x, x=y$	$\implies$ delete	can't use eliminate on $x=x$; why?
$x=f(a), x=y$	$\implies$ eliminate x	can't use orient on $x=y$; why? can eliminate using $x=f(a)$
$y=f(a), x=y$	$\implies$ eliminate y	
$y=f(a), x=f(a)$	$\implies$ return $S{\downarrow}$	

- Try it with: $\{(x, f(y)), (y, g(x))\}$
- Try it with: $\{(P(f(w), f(y)), P(x, f(g(u)))), (P(x,u), P(v,g(v))\}$
- Try it with: $\{(f(a,b,g(x,x),g(y,y),z), f(g(v,v),g(a,a),y,z,b))\}$

Unification Algorithm Termination

- Algorithm: Nondeterministic transition system based on the following rules
  - Delete $\{t{=}t\} \uplus S \quad \Longrightarrow S$
  - Decompose $\{f(t_1, \ldots, t_n) = f(s_1, \ldots, s_n)\} \uplus S \Longrightarrow \{t_1{=}s_1, \ldots, t_n{=}s_n\} \cup S$
  - Orient $\{t{=}x\} \uplus S \quad \Longrightarrow \{x{=}t\} \cup S$, if t is not a variable
  - Eliminate $\{x{=}t\} \uplus S \Longrightarrow \{x{=}t\} \cup S|x{\leftarrow}t$, if $x \in \mathrm{Vars}(S) - \mathrm{Vars}(t)$
- Termination: our measure function will be on ordinals (infinite numbers)
  - 0,1, 2, …, ω the first infinite ordinal (why stop with the naturals?)
  - Keep going: $\omega{+}1,\ \omega{+}2,\ \ldots,\ \omega{+}\omega = \omega 2,\ \omega 2{+}1,\ \ldots,\ \omega 3,\ \ldots,\ \omega\omega = \omega^2,$ $\ldots,\ \omega^3,\ \ldots,\ \omega^\omega,\ \ldots, \omega^{\omega^{\omega^{\cdots}}} = \epsilon_0$ ACL2s measures can use ordinals
  - Lexicographic ordering on tuples of natural numbers is $\approx \omega^\omega$
    - $\langle x_0, \ldots, x_{n-1}, x_n \rangle \longmapsto \omega^n x_0 + \cdots + \omega x_{n-1} + x_n$
    - There is an order-preserving bijection from n+1-tuples of Nats to ω^n
    - There is a theorem of this in the ACL2 ordinals books; you can define a relation, prove it is well-founded and use it in termination proofs

Unification Algorithm Termination

- Algorithm: Nondeterministic transition system based on the following rules
  - Delete $\{t{=}t\} \uplus S \implies S$
  - Decompose $\{f(t_1, \ldots, t_n) = f(s_1, \ldots, s_n)\} \uplus S \implies \{t_1{=}s_1, \ldots, t_n{=}s_n\} \cup S$
  - Orient $\{t{=}x\} \uplus S \implies \{x{=}t\} \cup S$, if t is not a variable
  - Eliminate $\{x{=}t\} \uplus S \implies \{x{=}t\} \cup S|x{\leftarrow}t$, if $x \in \mathrm{Vars}(S) - \mathrm{Vars}(t)$
- Termination: our measure function will be on ordinals (infinite numbers)
  - x is solved in S iff $x{=}t \in S$ and x only appears once in S
  - Measure: $\langle$vars in S not solved, size of S, # of equations $t{=}x$ in $S\rangle$
    - Delete $\leq$ why not =? $<$ Maybe $x \in t$, $x \notin S$
    - Decompose $\leq$ $<$
    - Orient $\leq$ $=$ $<$
    - Eliminate $<$

for every rule we have $(\leq \mid =)^*<$, so the lexicographic order is decreasing (and well-founded), i.e., any algorithm based on these rules terminates

Unification Algorithm Soundness

- Algorithm: Nondeterministic transition system based on the following rules
  - Delete $\{t=t\} \uplus S \quad \Longrightarrow S$
  - Decompose $\{f(t_1, \ldots, t_n) = f(s_1, \ldots, s_n)\} \uplus S \Longrightarrow \{t_1=s_1, \ldots, t_n=s_n\} \cup S$
  - Orient $\{t=x\} \uplus S \quad \Longrightarrow \{x=t\} \cup S$, if t is not a variable
  - Eliminate $\{x=t\} \uplus S \Longrightarrow \{x=t\} \cup S|x{\leftarrow}t$, if $x \in \text{Vars}(S)$ - $\text{Vars}(t)$
- If $V \Longrightarrow T$ then $U(V)=U(T)$: Easy: delete, decompose, orient; for eliminate:
  - let $\sigma \in U(V)$, $\theta = x{\leftarrow}t$. By lemma, $\sigma=\sigma\theta$ if $x\sigma=t\sigma$, since $x=t$ is in solved form
    - lemma: If X is in solved form then $\sigma=\sigma X{\downarrow}$ for all $\sigma \in U(X)$
      - Proof: σ, $\sigma X{\downarrow}$ agree on all vars by case analysis on $y \in \text{Domain}(X{\downarrow})$
  - $\sigma \in U(\{x=t\} \uplus S)$ iff $x\sigma=t\sigma \wedge \sigma \in U(S)$ iff $x\sigma=t\sigma \wedge \sigma\theta \in U(S)$ iff $x\sigma=t\sigma \wedge \sigma \in U(S\theta)$ iff $\sigma \in U(\{x=t\} \cup S\theta)$
- Soundness: If Unify returns σ, then σ is an idempotent mgu of S

Unification Algorithm Completeness

- Algorithm: Nondeterministic transition system based on the following rules
  - Delete $\{t{=}t\} \uplus S \quad \Longrightarrow S$
  - Decompose $\{f(t_1, \ldots, t_n) = f(s_1, \ldots, s_n)\} \uplus S \Longrightarrow \{t_1{=}s_1, \ldots, t_n{=}s_n\} \cup S$
  - Orient $\{t{=}x\} \uplus S \quad \Longrightarrow \{x{=}t\} \cup S$, if t is not a variable
  - Eliminate $\{x{=}t\} \uplus S \Longrightarrow \{x{=}t\} \cup S|x{\leftarrow}t$, if $x \in$ Vars(S) - Vars(t)
- Completeness: If S is solvable, then Unify(S) does not fail
  - Lemmas
    - $f(\ldots) = g(\ldots)$ has no solution if $f \neq g$
    - $x{=}t$, where $x \neq t$ and $x \in$ Vars(t) has no solution ($|x\sigma| < |t\sigma|$ for all σ)
- Proof: If S is solvable and in normal form wrt $\Longrightarrow$, then S is in solved form. S cannot contain pairs of form $f(\ldots) = f(\ldots)$ (decompose) or $f(\ldots) = g(\ldots)$ (lemma) or $x{=}x$ (delete) or $t{=}x$ where t is not a var (orient), so all equations are of form $x{=}t$ where $x \notin$ Vars(t) (lemma). Also x cannot occur twice in S (eliminate), so S is in solved form.

Unification Algorithm Improvements

- Algorithm: Nondeterministic transition system based on the following rules
  - Delete $\{t=t\} \uplus S \Longrightarrow S$
  - Decompose $\{f(t_1, \ldots, t_n) = f(s_1, \ldots, s_n)\} \uplus S \Longrightarrow \{t_1=s_1, \ldots, t_n=s_n\} \cup S$
  - Orient $\{t=x\} \uplus S \Longrightarrow \{x=t\} \cup S$, if t is not a variable
  - Eliminate $\{x=t\} \uplus S \Longrightarrow \{x=t\} \cup S|x{\leftarrow}t$, if $x \in \mathrm{Vars}(S)$ - $\mathrm{Vars}(t)$
  - Clash $\{f(t_1, \ldots, t_n) = g(s_1, \ldots, s_m)\} \uplus S \Longrightarrow \perp$ if $f \neq g$
  - *Occurs-Check* $\{x=t\} \uplus S \Longrightarrow \perp$ if $x \in \mathrm{Vars}(t) \wedge x \neq t$
- This is justified by the lemmas used for completeness
  - $f(\ldots) = g(\ldots)$ has no solution if $f \neq g$
  - $x=t$, where $x \neq t$ and $x \in \mathrm{Vars}(t)$ has no solution ($|x\sigma| < |t\sigma|$ for all σ)
- Early termination when ∃ no solution, saving (how much?) time

Complexity of Unification

- Algorithm: Nondeterministic transition system based on the following rules
  - Delete $\{t=t\} \uplus S \quad \Longrightarrow S$
  - Decompose $\{f(t_1, \ldots, t_n) = f(s_1, \ldots, s_n)\} \uplus S \Longrightarrow \{t_1=s_1, \ldots, s_n=t_n\} \cup S$
  - Orient $\{t=x\} \uplus S \quad \Longrightarrow \{x=t\} \cup S$, if t is not a variable
  - Eliminate $\{x=t\} \uplus S \Longrightarrow \{x=t\} \cup S|x{\leftarrow}t$, if $x \in \mathrm{Vars}(S) - \mathrm{Vars}(t)$
  - Clash $\{f(t_1, \ldots, t_n) = g(s_1, \ldots, s_m)\} \uplus S \Longrightarrow \perp$ if $f \neq g$
  - *Occurs-Check* $\{x=t\} \uplus S \Longrightarrow \perp$ if $x \in \mathrm{Vars}(t) \wedge x \neq t$
- Exponential blow up: $\{(x_1=f(x_0,x_0)), x_2=f(x_1,x_1), x_3=f(x_2,x_2), \ldots, x_n=f(x_{n-1},x_{n-1})\}$
- Notice that the output is exponential
- Can we do better?
  - Yes, by using a dag to represent terms and returning a dag
  - General idea: operate on a concise representation of problem
    - BDDs are concise representations of truth tables, decision trees, etc
    - Model checking searches an implicitly given graph (transition system)

History of Unification

- What we have studied is *syntactic*, *first-order* unification
  - syntactic: substitutions should make terms syntactically equal
  - equational unification: unification modulo an equational theory
    - eg for commutative f, $f(x,f(x,x)) = f(f(x,x),x)$ is E-unifiable not syntactically unifiable
  - first-order: no higher-order variables (no variables ranging over functions)
- Herbrand gave a nondeterministic algorithm in his 1930 thesis
- Robinson (1965) introduced FO theorem proving using resolution, unification
  - Required exponential time & space
- Robinson (1971) & Boyer-Moore (1972): structure sharing algorithms that were space efficient, but required exponential time
- Venturini-Zilli (1975): reduction to quadratic time using marking scheme
- Huet (1976) worked on higher-order unification led to $n\alpha(n)$ time: almost linear Robinson also discovered this algorithm
- Paterson and Wegman (1976) linear time algorithm
- Martelli and Montanari (1976) linear time algorithm based on Boyer-Moore

Unification Applications

- First-order theorem proving
  - Matching (ACL2) is a special case: given s,t find σ s.t. $s\sigma=t$
- Prolog (logic programming)
- Higher-order theorem proving
  - Undecidable for third-order logic (Huet 1973)
  - Undecidable for second-order logic (Goldfarb 1981)
- Natural language processing
- Unification-based grammars
- Equational theories
  - Commutative, Associative, Distributative, etc
  - Term rewrite systems
- Type inference (eg ML, Haskell, etc)
- Logic programming
- Machine learning: generalization is a dual of unification

Schedule

- 11/8: FOL/SMT
- ~~11/11: Temporal Logic/ Safety & Liveness/ Buchi~~ (Veteran's Day)
- 11/15: Refinement
- 11/18: Paper Presentations
- 11/22: Paper Presentations
- 11/29: Term Rewriting
- 12/2: Projects, Exam 2 (Take home)
- 12/6: Projects

Projects & Presentations

- 3 Minute Pitch, discussion