Lecture 22

Pete Manolios
Northeastern

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 $\{x{=}t\} \uplus S \Longrightarrow \{x{=}t\} \cup S|t{\leftarrow}x$, if $x{\in}\mathrm{Vars}(S)$ - $\mathrm{Vars}(t)$
- If $V \Longrightarrow T$ then $\mathrm{U}(V){=}\mathrm{U}(T)$: Easy: delete, decompose, orient;
  - Let $\sigma{\in}\mathrm{U}(V)$, $Y = x{=}t$, $\theta = t{\leftarrow}x$; note $\theta{=}(Y){\downarrow}$
  - Recall Lemma: Y is in solved form and $\sigma{\in}U(Y)$, then $\sigma{=}\sigma Y{\downarrow}$
  - Apply lemma to $x{=}t$ (solved form), $\sigma{=}\sigma\theta$ (since $x\sigma = \{\sigma{\in}U(Y)\}\ t\sigma = \{t{=}x\theta\}\ x\sigma\theta$)
  - $\sigma{\in}\mathrm{U}(V) \equiv \sigma{\in}\mathrm{U}(\{x{=}t\}{\uplus}S) \equiv x\sigma{=}t\sigma \wedge \sigma{\in}\mathrm{U}(S) \equiv x\sigma{=}t\sigma \wedge \sigma\theta{\in}\mathrm{U}(S) \equiv$ $x\sigma{=}t\sigma \wedge \sigma{\in}\mathrm{U}(S\theta) \equiv \sigma{\in}\mathrm{U}(\{x{=}t\} \cup S\theta) \equiv \sigma{\in}\mathrm{U}(T)$
- Soundness: If Unify returns σ, then σ is an idempotent mgu of S
  - We showed that Unify does not change unifiers
  - By previous lemma: if S is in solved form, then $S{\downarrow}$ is an idempotent mgu

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 $\{x{=}t\} \uplus S \Longrightarrow \{x{=}t\} \cup S|t{\leftarrow}x$, if $x \in \text{Vars}(S)$ - $\text{Vars}(t)$
- 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 $x \in \text{Vars}(t)$ 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 $x \notin \text{Vars}(t)$ (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 $\{t=t\} \uplus S \quad \Longrightarrow S$
  - 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 $\{x=t\} \uplus S \Longrightarrow \{x=t\} \cup S|t{\leftarrow}x$, 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 $\exists$ 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 $\{x=t\} \uplus S \Longrightarrow \{x=t\} \cup S|t \leftarrow x$, if $x \in \mathrm{Vars}(S)$ - $\mathrm{Vars}(t)$
- 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
- Higher-order theorem proving
  - Undecidable for second-order logic
- Natural language processing
- Unification-based grammars
- Equational theories
  - Commutative, Associative, Distributive, etc
  - Term rewrite systems
- Type inference (eg ML)
- Logic programming
- Machine learning: generalization is a dual of unification