Lecture 14 - Unification & Resolution

Vaishnavi Sundararajan

COL703 - Logic for Computer Science

Recap: Substitutions & normal forms

- A **substitution** θ is a partial map from $\mathscr{V}$ to $T(\Sigma)$, with a finite domain
- Read $\theta = \{t/x\}$ as "x is replaced by t under θ"
- Substitution lemma
- $Q_1 x_1 \dots Q_n x_n. [\varphi]$ is in **Prenex Normal Form (PNF)** if φ is **quantifier-free (qf)**.
- For any FO expression φ, there exists a logically equivalent ψ in PNF.
- PNF expression $Q_1 x_1 \dots Q_n x_n. [\varphi]$ is in **Skolem Normal Form (SNF)** if $Q_i = \forall$ for every $1 \leqslant i \leqslant n$.
- For any FO sentence φ, there exists an equisatisfiable ψ in SNF.

Recap: Herbrand models & unification

- Universe is $T^g(\Sigma)$, the set of all ground terms over the signature Σ
- Map each symbol in the syntax to itself; variables map to ground terms
- A sentence $\varphi \in FO_\Sigma$ is satisfiable iff its SNF form φ_{snf} is satisfiable iff Γ^g, the set of all ground instances of the qf subexpression in φ_{snf}, is satisfied by a Herbrand model.
- A sentence is unsatisfiable iff some finite set of ground instances of its qf subexpressions is unsatisfiable.
- Look to resolution for proving unsatisfiability
- **Unification** is the problem of finding a substitution θ so as to make some terms identical.
- One solves an equation of the form $t_1\theta = t_2\theta$ to find an appropriate θ.

Recap: Unifiability

- A finite set of terms $T = \{t_i \mid 1 \leqslant i \leqslant n\}$ is said to be **unifiable** if there exists a θ (a **unifier** for T) such that $t_i\theta = t_j\theta$ for all $1 \leqslant i, j \leqslant n$.
- A substitution that is "less constrained" than another is said to be "more general". Look for the most general unifier (mgu).
- If a set of terms is unifiable, then it has an mgu.
- Only two possible obstacles to unification:
  - Function clash (trying to unify $f(\ldots)$ with $g(\ldots)$ where $f \neq g$)
  - Occurs check (trying to unify x and t where $x \in \mathrm{vars}(t)$)
- If neither of these occurs, a set is unifiable!

Recap: Algorithm

- Start with a system of equations $l_1 = r_1, l_2 = r_2, \ldots, l_n = r_n$
- Perform the following transformations till you cannot anymore.

1. $l_i = t \notin \mathcal{V}$ and $r_i = x$: Replace $l_i = r_i$ by $x = t$
2. $l_i = x$ and $r_i = x$: Remove the equation
3. $l_i = f(\ldots)$ and $r_i = g(\ldots)$: The following cases arise.
    - $f \neq g$: Clash; no unification possible. Terminate.
    - $f = g$: Then $l_i = f(t_1, \ldots, t_k)$ and $r_i = f(u_1, \ldots, u_k)$. Replace $l_i = r_i$ by k new equations, each of the form $t_j = u_j$, for $1 \leqslant j \leqslant k$.
4. $l_i = x$ and $r_i = t$: The following cases arise.
    - $x \in \text{vars}(t)$: Occurs check; no unification possible. Terminate.
    - $x \notin \text{vars}(t)$: Replace every occurrence of x in $\{l_j \cup r_j \mid 1 \leqslant j \leqslant n, j \neq i\}$ by t.

Example

(1)

$$g(Y) = X$$

$$f(X, h(X), Y) = f(g(Z), W, Z)$$

Example

(1)

$$g(Y) = X$$

$$f(X, h(X), Y) = f(g(Z), W, Z)$$

(2)

$$X = g(Y)$$

$$f(X, h(X), Y) = f(g(Z), W, Z)$$

Example

①

$$g(Y) = X$$

$$f(X, h(X), Y) = f(g(Z), W, Z)$$

②

$$X = g(Y)$$

$$f(X, h(X), Y) = f(g(Z), W, Z)$$

③

$$X = g(Y)$$

$$X = g(Z)$$

$$h(X) = W$$

$$Y = Z$$

Example

1

$$g(Y) = X$$
$$f(X, h(X), Y) = f(g(Z), W, Z)$$

2

$$X = g(Y)$$
$$f(X, h(X), Y) = f(g(Z), W, Z)$$

3

$$X = g(Y)$$
$$X = g(Z)$$
$$h(X) = W$$
$$Y = Z$$

4

$$g(Z) = g(Y)$$
$$X = g(Z)$$
$$h(g(Z)) = W$$
$$Y = Z$$

Example

1

$$g(Y) = X$$
$$f(X, h(X), Y) = f(g(Z), W, Z)$$

2

$$X = g(Y)$$
$$f(X, h(X), Y) = f(g(Z), W, Z)$$

3

$$X = g(Y)$$
$$X = g(Z)$$
$$h(X) = W$$
$$Y = Z$$

4

$$g(Z) = g(Y)$$
$$X = g(Z)$$
$$h(g(Z)) = W$$
$$Y = Z$$

5

$$Z = Y$$
$$X = g(Z)$$
$$h(g(Z)) = W$$
$$Y = Z$$

Example

①
$$g(Y) = X$$
$$f(X, h(X), Y) = f(g(Z), W, Z)$$

②
$$X = g(Y)$$
$$f(X, h(X), Y) = f(g(Z), W, Z)$$

③
$$X = g(Y)$$
$$X = g(Z)$$
$$h(X) = W$$
$$Y = Z$$

④
$$g(Z) = g(Y)$$
$$X = g(Z)$$
$$h(g(Z)) = W$$
$$Y = Z$$

⑤
$$Z = Y$$
$$X = g(Z)$$
$$h(g(Z)) = W$$
$$Y = Z$$

⑥
$$Z = Z$$
$$X = g(Z)$$
$$h(g(Z)) = W$$
$$Y = Z$$

Example

1

$$g(Y) = X$$
$$f(X, h(X), Y) = f(g(Z), W, Z)$$

2

$$X = g(Y)$$
$$f(X, h(X), Y) = f(g(Z), W, Z)$$

3

$$X = g(Y)$$
$$X = g(Z)$$
$$h(X) = W$$
$$Y = Z$$

4

$$g(Z) = g(Y)$$
$$X = g(Z)$$
$$h(g(Z)) = W$$
$$Y = Z$$

5

$$Z = Y$$
$$X = g(Z)$$
$$h(g(Z)) = W$$
$$Y = Z$$

6

$$Z = Z$$
$$X = g(Z)$$
$$h(g(Z)) = W$$
$$Y = Z$$

7

$$X = g(Z)$$
$$h(g(Z)) = W$$
$$Y = Z$$

Example

(1)

$$g(Y) = X$$
$$f(X, h(X), Y) = f(g(Z), W, Z)$$

(2)

$$X = g(Y)$$
$$f(X, h(X), Y) = f(g(Z), W, Z)$$

(3)

$$X = g(Y)$$
$$X = g(Z)$$
$$h(X) = W$$
$$Y = Z$$

(4)

$$g(Z) = g(Y)$$
$$X = g(Z)$$
$$h(g(Z)) = W$$
$$Y = Z$$

(5)

$$Z = Y$$
$$X = g(Z)$$
$$h(g(Z)) = W$$
$$Y = Z$$

(6)

$$Z = Z$$
$$X = g(Z)$$
$$h(g(Z)) = W$$
$$Y = Z$$

(7)

$$X = g(Z)$$
$$h(g(Z)) = W$$
$$Y = Z$$

(8)

$$X = g(Z)$$
$$W = h(g(Z))$$
$$Y = Z$$

Algorithm: Termination

- Once we swap an equation of the form $t = x$, we do not swap back
- How many equations of the form $x = x$ can we get for a given input?
- How many new equations does each $f(\ldots) = g(\ldots)$ get replaced by?

Algorithm: Termination

- Once we swap an equation of the form $t = x$, we do not swap back
- How many equations of the form $x = x$ can we get for a given input?
- How many new equations does each $f(\ldots) = g(\ldots)$ get replaced by?
- So transformations (1)–(3) can only be applied finitely many times.
- (4) can be applied at most once per variable.
- So the algorithm terminates in a finite number of steps.
- When the algorithm terminates, all equations are of the form $x_i = t_i$ (each x_i only occurs once)
- This is called **a set of equations in solved form**.
- For a set of equations in solved form as above, the substitution $\{t_1/x_1, t_2/x_2, \ldots, t_n/x_n\}$ is a unifier.

Algorithm: Correctness

- **Soundness**: If the algorithm produces a θ, then θ is a unifier for S.
- **Completeness**: If S is unifiable, then the algorithm produces a unifier θ.
- Suppose I run the algorithm on a set S of equations, and get S' after one iteration (applying one instance of one transformation rule).
- **Claim**: A substitution θ is a unifier for S iff it is a unifier for S'.
- We now analyze each rule to see if this holds.
- For now, ignore the rules which cause the algorithm to terminate without returning any unifier.
- We denote by r the rule that was applied.

Algorithm: Correctness (contd.)

- $\mathsf{r} = (1)$: There exists a system of equations T s.t. $S = T \cup \{t = x\}$ and $S' = T \cup \{x = t\}$ for some $x \in \mathcal{V}$ and some $t \notin \mathcal{V}$. $t\theta = x\theta$ iff $x\theta = t\theta$, so θ is a unifier for S iff it is a unifier for S'.

Algorithm: Correctness (contd.)

- $\mathsf{r} = (1)$: There exists a system of equations T s.t. $S = T \cup \{t = x\}$ and $S' = T \cup \{x = t\}$ for some $x \in \mathscr{V}$ and some $t \notin \mathscr{V}$. $t\theta = x\theta$ iff $x\theta = t\theta$, so θ is a unifier for S iff it is a unifier for S'.
- $\mathsf{r} = (2)$: Then, $S = S' \cup \{x = x\}$. Any θ satisfies $x\theta = x\theta$, so the claim holds for this case also.

Algorithm: Correctness (contd.)

- r = (1): There exists a system of equations T s.t. $S = T \cup \{t = x\}$ and $S' = T \cup \{x = t\}$ for some $x \in \mathscr{V}$ and some $t \notin \mathscr{V}$. $t\theta = x\theta$ iff $x\theta = t\theta$, so θ is a unifier for S iff it is a unifier for S'.
- r = (2): Then, $S = S' \cup \{x = x\}$. Any θ satisfies $x\theta = x\theta$, so the claim holds for this case also.
- r = (3): There exists a T s.t. $S = T \cup \{f(t_1, \ldots, t_k) = f(u_1, \ldots, u_k)\}$ and $S' = T \cup \{t_1 = u_1, \ldots, t_k = u_k\}$. One can verify that $f(t_1, \ldots, t_k)\theta = f(u_1, \ldots, u_k)\theta$ iff $t_1\theta = u_1\theta, \ldots, t_k\theta = u_k\theta$. Thus θ is a unifier for S iff it is a unifier for S'.

Algorithm: Correctness (contd.)

- $\mathsf{r} = (4)$: There is some T s.t. $S = T \cup \{x = t\}$ and $S' = T\{t/x\} \cup \{x = t\}$.
  - Suppose we show that for any $l = r$ in T and any substitution θ s.t. $x\theta = t\theta$, we have $l\theta = r\theta$ iff $(l\{t/x\})\theta = (r\{t/x\})\theta$.
  - Then, if S' has a unifier θ, $(l\{t/x\})\theta$ is identical to $(r\{t/x\})\theta$ for every $l = r$ in T. By the above statement, $l\theta = r\theta$, so θ is also a unifier for S.
  - Similarly, if S has a unifier θ, $l\theta$ is identical to $r\theta$, and $(l\{t/x\})\theta = (r\{t/x\})\theta$, so θ is also a unifier for S'.
  - How do we show that $l\theta = r\theta$ iff $(l\{t/x\})\theta = (r\{t/x\})\theta$? Note that $x \notin \mathsf{vars}(t)$, so $x\theta = t\theta = u$ for some u.
  - If $x \notin \mathsf{vars}(l)$, then $l\{t/x\}\theta = l\theta$.
  - Now suppose $x \in \mathsf{vars}(l)$. Let $x\theta = t\theta = u$. Let $\theta = \{u/x\} \cup \theta'$.

Algorithm: Correctness (contd.)

- Suppose $x \in \text{vars}(l)$. Let $x\theta = t\theta = u$. Let $\theta = \{u/x\} \cup \theta'$. Then,
  - $t\theta = t\theta' = u$ (since $x \notin \text{vars}(t)$)
  - $l\{t/x\}\theta = l\{t/x\}(\{u/x\} \cup \theta') = l\{t/x\}\theta'$ (since $x \notin \text{vars}(t)$)
  - $l\{t/x\}\theta' = l(\{t\theta'/x\} \cup \theta')$ (replacing x by t and then applying θ' is the same as replacing x by the result of applying θ' to t "first", while replacing all other variables by their results under θ')
  - $l(\{t\theta'/x\} \cup \theta') = l(\{u/x\} \cup \theta') = l\theta$
- One can perform a similar analysis for r.
- **Claim**: If the algorithm terminates without a unifier, the original set S of equations itself has no unifier.
- **Proof sketch**: If S has a unifier, then each new set of equations S' must have a unifier too. Since S' has no unifier ("bad" termination), chase back to the fact that S has no unifier either.

Algorithm: Correctness (contd.)

- Suppose the algorithm terminates with a set of equations $S^* = \{x_1 = t_1, \dots, x_n = t_n\}$. Let $\theta = \{t_1/x_1, \dots, t_n/x_n\}$. Is θ an mgu for S^*?
- Consider any unifier τ for S^*. $x_i\tau = t_i\tau$ for each $1 \leqslant i \leqslant n$.
- Consider the function $\rho = \tau \upharpoonright \left(\mathscr{V} \setminus \{x_1, \dots, x_n\}\right)$.
- We know that $\mathsf{vars}(t_j) \cap \{x_1, \dots, x_n\} = \emptyset$. So $t_i\tau = t_i\rho = t_i$.
- Then, $x_i(\theta \circ \rho) = (x_i\theta)\rho = t_i\rho = t_i = x_i\tau$.
- Therefore, $\tau = \theta \circ \rho$ for **any** τ that unifies S^*, and so θ is an mgu for S^*.
- θ and τ are unifiers of S as well, so θ is an mgu for S also.

Resolution: Roadmap

- $\Gamma \vDash \varphi$ iff $\Gamma \cup \{\neg\varphi\}$ unsatisfiable
- Every sentence in FO has an equisatisfiable sentence in SCNF
- A sentence is unsatisfiable iff some finite set of ground instances of its qf subexpressions is unsatisfiable.
- Perform resolution to determine unsatisfiability
- What is our notion of clauses now? Literals?
- Want to apply resolution to the "clause form" of $\Gamma \cup \{\neg\varphi\}$ and obtain the empty clause to show unsatisfiability.

SCNF, clauses, and literals

- Consider an SCNF sentence $\varphi = \forall x_1 x_2 \dots x_n .\ [\psi]$ where ψ qf.
- Suppose $\psi = \bigwedge_{1 \leqslant i \leqslant m} \delta_i$ where each $\delta_i = \bigvee_{1 \leqslant j \leqslant k_i} \ell_j$
- "Ignore" the universal quantifiers, focus on ψ
- Then, we represent φ also by the set of **clauses** $\{\delta_i \mid 1 \leqslant i \leqslant m\}$.
- Each clause δ_i is represented by the set of **literals** $\{\ell_i \mid 1 \leqslant i \leqslant k_i\}$.
- Each literal is of the form $P(\dots)$ or $\neg P(\dots)$ for $P \in \mathscr{P}$.
- Perform unification on variables to eliminate contradictory literals **across clauses**.
- **Achtung**: A "bad" termination of the unification algorithm will not allow resolution to proceed. Avoid accidental bad terminations!

Models of clauses

- For a substitution θ, the result of applying it to a clause is given by $\delta_i\theta = \{\ell_i\theta \mid 1 \leqslant i \leqslant k_i\}$. The set of ground instances of a clause δ is $\Gamma^g(\delta) = \{\delta\theta \mid \theta$ is a ground substitution for $\delta\}$.
- An empty clause has no models
- An interpretation is a model of a set of clauses if it is a model for every clause in that set.
- A set S of clauses is unsatisfiable iff there is a finite subset $S' \subseteq_{\text{fin}} S$ such that $\Gamma^g(S')$ is unsatisfiable.

Clauses and literals: FO edition

- **Exercise**: Show that $\forall x_1 \ldots x_n. \left[\bigwedge_{1 \leqslant i \leqslant m} \delta_i\right] \Leftrightarrow \bigwedge_{1 \leqslant i \leqslant m} \left(\forall x_1 \ldots x_n. \ [\delta_i]\right)$
- Consider the sentence $\forall x. \ [P(x)] \wedge \forall x. \ [\neg P(f(x))]$. Is it satisfiable?

Clauses and literals: FO edition

- **Exercise**: Show that $\forall x_1 \ldots x_n.\ \left[\bigwedge_{1\leqslant i\leqslant m} \delta_i\right] \Leftrightarrow \bigwedge_{1\leqslant i\leqslant m} \left(\forall x_1 \ldots x_n.\ [\delta_i]\right)$
- Consider the sentence $\forall x.\ [P(x)] \wedge \forall x.\ [\neg P(f(x))]$. Is it satisfiable? No.
- Can I turn this into the set of clauses $\{\{P(x)\}, \{\neg P(f(x))\}\}$?

Clauses and literals: FO edition

- **Exercise**: Show that $\forall x_1 \ldots x_n.\ \left[\bigwedge_{1 \leqslant i \leqslant m} \delta_i\right] \Leftrightarrow \bigwedge_{1 \leqslant i \leqslant m} \left(\forall x_1 \ldots x_n.\ [\delta_i]\right)$
- Consider the sentence $\forall x.\ [P(x)] \wedge \forall x.\ [\neg P(f(x))]$. Is it satisfiable? No.
- Can I turn this into the set of clauses $\{\{P(x)\}, \{\neg P(f(x))\}\}$?
- What will the unification algorithm do on these clauses?
- **Occurs check!**
- So even though original expression was unsat, no way to derive the empty clause.
- Rename bound variables to keep variables across clauses distinct.
- Only consider clauses with distinct variable names from now on.

Clauses and literals: FO edition

- For resolution over PL, we resolved one literal at a time.
- Suppose I have two clauses of the form $\delta_1 = \{P(x), P(y)\}$ and $\delta_2 = \{\neg P(m), \neg P(n)\}$. Is $\{\delta_1, \delta_2\}$ satisfiable?

Clauses and literals: FO edition

- For resolution over PL, we resolved one literal at a time.
- Suppose I have two clauses of the form $\delta_1 = \{P(x), P(y)\}$ and $\delta_2 = \{\neg P(m), \neg P(n)\}$. Is $\{\delta_1, \delta_2\}$ satisfiable?
- Clearly not. But suppose we only replace y by m in our first attempt. We are then left with a single clause of the form $\{P(x), \neg P(n)\}$.
- Unification cannot happen **inside** a clause, only across clauses!
- Original set was unsat, but no way to proceed from here and get the empty clause.
- **Takeaway**: Substitutions give you power; use it! Unify as much as possible in one go.

Resolution: Example

- Check if $\forall x. \ [P(x) \lor Q(x)] \models Q(m)$.

Resolution: Example

- Check if $\forall x.\ [P(x) \vee Q(x)] \vDash Q(m)$.
- Check if $\forall x.\ [P(x) \vee Q(x)] \cup \{\neg Q(m)\}$ is unsatisfiable.
- Clause for $\forall x.\ [P(x) \vee Q(x)]$ is $\{P(x), Q(x)\}$.
- Suppose $\delta = \{P(x), Q(x)\}$, and $\ell = \neg Q(m)$.
- Need to see if we can derive the empty clause from $\delta \cup \{\ell\}$.
- $Q(x)$ and $Q(m)$ unify (What's the mgu?)

Resolution: Example

- Check if $\forall x.\ [P(x) \vee Q(x)] \vDash Q(m)$.
- Check if $\forall x.\ [P(x) \vee Q(x)] \cup \{\neg Q(m)\}$ is unsatisfiable.
- Clause for $\forall x.\ [P(x) \vee Q(x)]$ is $\{P(x), Q(x)\}$.
- Suppose $\delta = \{P(x), Q(x)\}$, and $\ell = \neg Q(m)$.
- Need to see if we can derive the empty clause from $\delta \cup \{\ell\}$.
- $Q(x)$ and $Q(m)$ unify (What's the mgu?)
- So we can resolve, just as we did for propositional logic, but with unification thrown into the mix.

Resolution: Example

- Check if $\forall x.\ [P(x) \lor Q(x)] \models Q(m)$.
- Check if $\forall x.\ [P(x) \lor Q(x)] \cup \{\neg Q(m)\}$ is unsatisfiable.
- Clause for $\forall x.\ [P(x) \lor Q(x)]$ is $\{P(x), Q(x)\}$.
- Suppose $\delta = \{P(x), Q(x)\}$, and $\ell = \neg Q(m)$.
- Need to see if we can derive the empty clause from $\delta \cup \{\ell\}$.
- $Q(x)$ and $Q(m)$ unify (What's the mgu?)
- So we can resolve, just as we did for propositional logic, but with unification thrown into the mix.

$$\frac{\{P(x), Q(x)\} \qquad \{\neg Q(m)\}}{P(m)} \{m/x\}$$