# Proofs in Predicate Logic: Extending Resolution

# Resolution Reprise

Recall resolution for propositional logic:

- Convert the premises and the negation of the conclusion into Conjunctive Normal Form.
- Use the resolution rule $\dfrac{\alpha \vee p \quad \neg p \vee \beta}{\alpha \vee \beta}$ repeatedly.
- One of the following will happen.
  - The empty clause is derived. The conclusion is proved.
  - No new clauses can be formed. A satisfying valuation exists.

Enhancements for predicate logic:

- In conversion to CNF, allow for quantifiers on variables.
- Resolve on atomic formulas, containing arbitrary terms.
- Re-assess the possible outcomes.

# Literals in Predicate Logic

In predicate logic, a literal is an atomic formula or the negation of one.

When do literals comflict?

- $R(c)$ and $\neg R(c)$ definitely conflict.
- $Q(f(c))$ and $\neg Q(c)$ are compatible, in general.
- $P(f(c))$ and $\neg P(x)$ conflict when $x$ has value $f(c)$.

*Convention*:

> All variables are treated as universally quantified (by $\forall$).

We deal with possible values of variables by substitution.

# Substitution: Example

Given: two clauses $P(f(c)) \vee Q(y)$ and $\neg P(x) \vee R(g(x))$.

Replace $x$ by $f(c)$ in the second clause: $\neg P(f(c)) \vee R(g(f(c)))$.

Now apply the resolution rule, yielding the new clause $Q(y) \vee R(g(f(c)))$.

In some cases, we must make several substitutions in order to apply the resolution rule. E.g., if we have $Q(x, c)$ and $\neg Q(a, y)$, then we must substitute for both $x$ and $y$.

# Unification of Terms

Goal: given terms $t_1$ and $t_2$, determine whether there are substitutions for the variables that make the terms the same.

Finding such a substitution is called *unification* of the terms.

The substitution itself is called a *unifier*.

# Examplecises

Unify the literals $P(f(x), g(y))$ and $\neg P(f(f(a)), g(z))$.

# Examplecises

Unify the literals $P(f(x), g(y))$ and $\neg P(f(f(a)), g(z))$.

Substitution: $[f(a)/x]\,[z/y]$
(or $[f(a)/x]\,[y/z]$, or $[f(a)/x]\,[u/y]\,[u/z]$)

What clause results from resolving the clauses

$P(f(x), g(y)) \vee Q(x, y)$ and $\neg P(f(f(a)), g(z)) \vee Q(f(a), z)$ ?

# Examplecises

Unify the literals $P(f(x), g(y))$ and $\neg P(f(f(a)), g(z))$.

Substitution: $[f(a)/x]\,[z/y]$
(or $[f(a)/x]\,[y/z]$, or $[f(a)/x]\,[u/y]\,[u/z]$)

What clause results from resolving the clauses

$P(f(x), g(y)) \vee Q(x, y)$ and $\neg P(f(f(a)), g(z)) \vee Q(f(a), z)$ ?

$Q(f(a), z) \vee Q(f(a), z)$, which is equivalent to simply $Q(f(a), z)$.

# Examplecises

Unify the literals $P(f(x), g(y))$ and $\neg P(f(f(a)), g(z))$.

Substitution: $[f(a)/x]\,[z/y]$
(or $[f(a)/x]\,[y/z]$, or $[f(a)/x]\,[u/y]\,[u/z]$)

What clause results from resolving the clauses

$P(f(x), g(y)) \vee Q(x, y)$ and $\neg P(f(f(a)), g(z)) \vee Q(f(a), z)$ ?

$Q(f(a), z) \vee Q(f(a), z)$, which is equivalent to simply $Q(f(a), z)$.

Literals $P(g(y), f(x, h(x), y))$ and $P(u, f(g(z), w, z))$.

# Examplecises

Unify the literals $P(f(x), g(y))$ and $\neg P(f(f(a)), g(z))$.

Substitution: $[f(a)/x]\,[z/y]$
(or $[f(a)/x]\,[y/z]$, or $[f(a)/x]\,[u/y]\,[u/z]$)

What clause results from resolving the clauses

$P(f(x), g(y)) \vee Q(x, y)$ and $\neg P(f(f(a)), g(z)) \vee Q(f(a), z)$ ?

$Q(f(a), z) \vee Q(f(a), z)$, which is equivalent to simply $Q(f(a), z)$.

Literals $P(g(y), f(x, h(x), y))$ and $P(u, f(g(z), w, z))$.

Substitution: $[g(y)/u]\,[g(z)/x]\,[h(g(z))/w]\,[z/y]$
(or variations).

# A Full Refutation

| | | |
|---|---|---|
| 1. | $\neg P_1(x) \vee Q(x) \vee R_1(x, f(x))$ | Premise |
| 2. | $\neg P_1(x) \vee Q(x) \vee R_2(f(x))$ | Premise |
| 3. | $P_2(a)$ | Premise |
| 4. | $P_1(a)$ | Premise |
| 5. | $\neg R_1(a, y) \vee P_2(y)$ | Premise |
| 6. | $\neg P_2(x) \vee \neg Q(x)$ | Premise |
| 7. | $\neg P_2(x) \vee \neg R_2(x)$ | Premise |
| 8. | $\neg Q(a)$ | 3, 6; $[a/x]$ |
| 9. | $Q(a) \vee R_2(f(a))$ | 2, 4; $[a/x]$ |
| 10. | $R_2(f(a))$ | 8, 9 |
| 11. | $Q(a) \vee R_1(a, f(a))$ | 1, 4; $[a/x]$ |
| 12. | $R_1(a, f(a))$ | 8, 11 |
| 13. | $P_2(f(a))$ | 5, 12; $[f(a)/y]$ |
| 14. | $\neg R_2(f(a))$ | 7, 13; $[f(a)/x]$ |
| 15. | $\perp$ | 10, 14 |

# The Resolution Rule, with Unification

Consider an atomic formula and a negated atomic formula with the same predicate:

$$R(t_1, \ldots, t_k) \text{ and } \neg R(s_1, \ldots s_k) \ .$$

where the free variables are $x_1, \ldots, x_i$.

Suppose there are terms $r_1, \ldots, r_i$ such that the substitutions

$$[r_1/x_1] \quad \ldots \quad [r_i/x_i]$$

yield $t_j[r_1/x_1]\ldots[r_i/x_i] = s_j[r_1/x_1]\ldots[r_i/x_i]$, for each $1 \leq j \leq k$. Then

$$\frac{\alpha \vee R(t_1, \ldots, t_k) \qquad \neg R(s_1, \ldots s_k) \vee \beta}{(\alpha \vee \beta)[r_1/x_1]\ldots[r_i/x_i]}$$

is an instance of the resolution rule.

# Why Does the Rule Work?

*Lemma*: If some interpretation $\mathcal{I}$ and environment $E$ satisfy

$$\big(\forall x_1 \cdot \ldots \forall x_i \cdot R(t_1, \ldots, t_k) \vee \alpha\big) \wedge \big(\forall x_1 \cdot \ldots \forall x_i \cdot \neg R(s_1, \ldots, s_k) \vee \beta\big) \ ,$$

and for every $j$ we have $t_j[r_1/x_1]\ldots[r_i/x_i] = s_j[r_1/x_1]\ldots[r_i/x_i]$, then $\mathcal{I}$ and $E$ also satisfy

$$(\alpha \vee \beta)[r_1/x_1]\ldots[r_i/x_i] \ .$$

*Proof* (outline): The assumption implies that $\mathcal{I}$ and $E$ satisfy

$$(R(t_1, \ldots, t_k) \vee \alpha)[r_1/x_1]\ldots[r_i/x_i] \wedge (\neg R(s_1, \ldots, s_k) \vee \beta)[r_1/x_1]\ldots[r_i/x_i].$$

The simple resolution rule gives the result.

# Handling Quantifiers

Recall that we have assumed that all variables have universal quantifiers.

- If a given formula contains a free variable, we replace it by a fresh constant.
  (Equivalently, simply declare the symbol to be a constant.)
- If a given formula contains "$\forall$", then we will simply remove it.
- But what about "$\exists$"?

If we have only $\exists$ quantifiers, we can replace the variables by fresh constants. If $\exists x \cdot \alpha$ is satisfiable, and constant symbol $c$ does not occur in $\alpha$, then $\alpha[c/x]$ is satisfiable.

But this fails if $\forall$ quantifiers are present. . . .

# Skolem Functions

In general, a quantifier $\exists x$ is replaced by a "Skolem function".*

Suppose that a formula has an existential quantifier inside of $k$ universal quantifiers:

$$\forall y_1 \cdot \ldots \forall y_k \cdot \exists x \cdot \alpha \ .$$

Let "$f^{(k)}$" be a fresh function symbol, not used elsewhere. The Skolem form of the formula is

$$\forall y_1 \cdot \ldots \forall y_k \cdot \alpha[f(y_1, \ldots, y_k)/x] \ .$$

Example: The Skolem form of

$$\forall y \cdot \exists x \cdot \big(P(y) \rightarrow Q(x, y)\big)$$

is

$$\forall y \cdot \big(P(y) \rightarrow Q(f(y), y)\big) \ .$$

* *Named for Thoralf Skolem, who invented this technique.*

# Why Do Skolem Functions Work?

*Lemma*: If $\mathcal{I} \models_E \forall y_1 \cdot \ldots \forall y_k \cdot \exists x \cdot \alpha$, and $f$ is a fresh function symbol, then $\mathcal{I}$ can be augmented to $\mathcal{I}'$ (with a definition of $f$) such that

$$\mathcal{I}' \models_E \forall y_1 \cdot \ldots \forall y_k \cdot \alpha[f(y_1, \ldots, y_k)/x] \ .$$

*"Proof by example"*: Let $k = 1$ and $\alpha$ is $(P(y) \rightarrow Q(x, y))$. Suppose that

$P^{\mathcal{I}}$ is the predicate "is even", and

$Q^{\mathcal{I}}$ is "twice the first number equals the second",

then a suitable function for $f^{\mathcal{I}'}$ is "half-of":

$$f(z) = \begin{cases} z/2 & \text{if } z \text{ is even} \\ \langle \textit{arbitrary} \rangle & \text{otherwise.} \end{cases}$$

# Conversion to CNF

1. Re-name bound variables, so that each quantifier uses a different variable.
2. Eliminate $\rightarrow$ and $\leftrightarrow$ as before.
3. Push negation to literals.
   Replace $\neg(\exists x \cdot \alpha)$ by $\forall x \cdot \neg\alpha$.
   Replace $\neg(\forall x \cdot \alpha)$ by $\exists x \cdot \neg\alpha$.
4. Replace existentially quantified variables by their Skolem functions. Then remove universal quantifiers.
5. Transform into a conjunction of clauses, and simplify.

# Possible Outcomes of Resolution

When doing a resolution proof, there are three things that might happen.

1. A contradiction is reached.
2. No new formulas can be derived by resolution.
3. Neither of the above — the process simply continues.

In the first case, the original set of formulas was inconsistent. (Proof from the two lemmas.)

# When Resolution "Gets Stuck"

As in the propositional case, when no further formulas can be derived, and no contradition has been found, we can find a satisfying interpretation and environment.

Domain: the set of ground terms that appear.
(A "ground" term is one that has no variables.)

Values of constants: themselves.

Values of functions: themselves.

Values of relations: as required.

When a choice exists, any one can work.

# Resolution May Not Terminate

Since there are function symbols, the number of possible terms (and atomic formulas) is infinite. Sometimes, applying resolution may never end—there are always new formulas to create.

*Example*: Assume PA1 and PA2. In CNF, these are

1. $\neg\big(s(x) = 0\big)$
2. $\neg\big(s(y) = s(z)\big) \vee y = z$.

Steps of resolution:

| | |
|---|---|
| 3. $\neg\big(s(s(x)) = s(0)\big)$ | 1, 2: $[s(x)/y][0/z]$ |
| 4. $\neg\big(s(s(s(x))) = s(s(0))\big)$ | 2, 3: $[s(s(x))/y][s(0)/z]$ |
| 5. $\neg\big(s(s(s(s(x)))) = s(s(s(0)))\big)$ | 2, 4: $[s(s(s(x)))/y][s(s(0))/z]$ |
| $\vdots$ *etc.* | |

# Resolution for Predicate Logic Is Not an Algorithm

By definition, an "algorithm" must always produce an answer eventually.

The resolution method sometimes continues forever.
It is not an algorithm.

We shall see later:

> No algorithm exists, that can always determine whether or not $\Sigma \models \varphi$, for given assumptions $\Sigma$ and conclusion $\varphi$.

In other words, entailment of formulas in Predicate Logic is *undecidable*.