Logical Inference

Stephen G. Ware

CSCI 4525 / 5525

Narrative Intelligence LAB

THE UNIVERSITY of NEW ORLEANS

Outline

- Unification
- Resolution theorem proving

Unification

A **substitution** is a list of variable equivalencies in the form: $\{x/t\}$, where x is a variables and t is a term.

Remember, a term can be:

- A constant (i.e. this variable has this value)
- A function (i.e. this variable has this value)
- A variables (i.e. two variables have the same value)

Unification

A **unifier** is a substitution which makes two predicate logic expressions the same.

Expression 1: $pit(x) \wedge adjacent(x, y) \rightarrow breeze(y)$

Expression 2: $pit(C1) \wedge adjacent(C1, B1) \rightarrow breeze(B1)$

Unifier: $\{x/C1, y/B1\}$

If we substitute the values in the unifier into Expression 1:

Expression 3: $pit(x) \wedge adjacent(x, y) \rightarrow breeze(y)$

(the same as Expression 1)

Unification

A **unifier** is a substitution which makes two predicate logic expressions the same.

Expression 1: $pit(x) \wedge adjacent(x, y) \rightarrow breeze(y)$

Expression 2: $pit(C1) \wedge adjacent(C1, B1) \rightarrow breeze(B1)$

Unifier: $\{x/C1, y/B1\}$

If we substitute the values in the unifier into Expression 1:

Expression 3: $pit(C1) \wedge adjacent(C1, y) \rightarrow breeze(y)$

Unification

A **unifier** is a substitution which makes two predicate logic expressions the same.

Expression 1: $pit(x) \land adjacent(x, y) \rightarrow breeze(y)$

Expression 2: $pit(C1) \land adjacent(C1, B1) \rightarrow breeze(B1)$

Unifier: $\{x/C1, y/B1\}$

If we substitute the values in the unifier into Expression 1:

Expression 3: $pit(C1) \land adjacent(C1, B1) \rightarrow breeze(B1)$

(the same as Expression 2)

Unification

If a unifier can be found, we say the two expressions **unify**.

The process of finding a unifier for two expressions is called **unification**.

Unifying Expressions

```
To unify two expressions X and Y:
    If the expressions have different predicates, fail.
    If the expressions have different arity, fail.
    Unify the terms of the expressions.
```

Unifying Terms

```
To unify two terms X and Y:
    If either X or Y is a variable:
        Let V be the variable and U be the other.
        If V has a value W, unify W and U.
        Else if U is a function and V occurs in U, fail.
        Else set V = U.
    Else if X and Y are constants:
        If X and Y are the same constant, succeed.
        If X and Y are different constants, fail.
    Else if X and Y are functions of the same name and arity:
        Unify the parameters of X and Y.
    Else fail.
```

Unification Example 1

Exp 1: $pit(x)$

Exp 2: $breeze(y)$

Unifier: $\{\}$

Unification Example 1

Exp 1: $pit(x)$

Different predicates.

Exp 2: $breeze(y)$

Unifier: fail

Unification Example 2

Exp 1: $adjacent(x)$

Exp 2: $adjacent(y, z)$

Unifier: $\{\}$

Unification Example 2

Exp 1: $adjacent(x)$ arity = 1

Different arity.

Exp 2: $adjacent(y, z)$ arity = 2

Unifier: fail

Unification Example 3

Exp 1: $adjacent(x, C1)$

Exp 2: $adjacent(B1, y)$

Unifier: {}

Unification Example 3

Exp 1: $adjacent(x, C1)$

Set $x = B1$

Exp 2: $adjacent(B1, y)$

Unifier: $\{x/B1\}$

Unification Example 3

Exp 1: $adjacent(x, C1)$

Set $y = C1$

Exp 2: $adjacent(B1, y)$

Unifier: $\{x/B1, y/C1\}$

Unification Example 3

Exp 1: $adjacent(x, C1)$

Exp 2: $adjacent(B1, y)$

Unifier: $\{x/B1, y/C1\}$

Success!

Unification Example 4

Exp 1: $adjacent(x, C1)$

Exp 2: $adjacent(y, B1)$

Unifier: {}

Unification Example 4

Exp 1: $adjacent(x, C1)$

Set $x = y$

Exp 2: $adjacent(y, B1)$

Unifier: $\{x/y\}$

Unification Example 4

Exp 1: $adjacent(x, C1)$

$C1$ and $B1$ are different constants.

Exp 2: $adjacent(y, B1)$

Unifier: fail

Unification Example 5

Exp 1: $adjacent(above(x), x)$

Exp 2: $adjacent(y, y)$

Unifier: {}

NIL

Unification Example 5

Exp 1: $adjacent(above(x), x)$

Set $y = above(x)$.

Exp 2: $adjacent(y, y)$

Unifier: $\{y/above(x)\}$

Unification Example 5

Exp 1: $adjacent(above(x), x)$

Set $x = y$.

Exp 2: $adjacent(y, y)$

Unifier: $\{y/above(x), x/y\}$

NIL

Unification Example 5

Exp 1: $adjacent(above(x), x)$

Set $x = above(x)$.
x occurs in $above(x)$!

Exp 2: $adjacent(y, y)$

Unifier: fail

NIL

Resolution

The process of **resolution** provides a simple way to perform logical inference with any logical statements in conjunctive normal form (i.e. all logical statements).

$$\frac{\overbrace{(X_1 \vee \cdots \vee X_i \vee Z)}^{\text{first clause}} \wedge \overbrace{(Y_1 \vee \cdots \vee Y_j \vee \neg Z)}^{\text{second clause}}}{(X_1 \vee \cdots \vee X_i \vee Y_1 \vee \cdots \vee Y_j)}$$

Resolution

The process of **resolution** provides a simple way to perform logical inference with any logical statements in conjunctive normal form (i.e. all logical statements).

complimentary literals

$$\frac{(X_1 \vee \cdots \vee X_i \vee Z) \wedge \left(Y_1 \vee \cdots \vee Y_j \vee \neg Z\right)}{\left(X_1 \vee \cdots \vee X_i \vee Y_1 \vee \cdots \vee Y_j\right)}$$

Resolution

The process of **resolution** provides a simple way to perform logical inference with any logical statements in conjunctive normal form (i.e. all logical statements).

$$\frac{(X_1 \vee \cdots \vee X_i \vee Z) \wedge \left(Y_1 \vee \cdots \vee Y_j \vee \neg Z\right)}{\underbrace{\left(X_1 \vee \cdots \vee X_i \vee Y_1 \vee \cdots \vee Y_j\right)}_{\text{result}}}$$

Resolution

Why does resolution work?

- If Z is *true*, then the first clause is *true*.
- If Z is *true*, $\neg Z$ is *false*, so one of $Y_1 \vee \cdots \vee Y_j$ must be *true*.
- Since one of $Y_1 \vee \cdots \vee Y_j$ must be *true*, the result must be *true*.

$$\frac{(X_1 \vee \cdots \vee X_i \vee Z) \quad (Y_1 \vee \cdots \vee Y_j \vee \neg Z)}{(X_1 \vee \cdots \vee X_i \vee Y_1 \vee \cdots \vee Y_j)}$$

Resolution

Why does resolution work?

- If Z is *false*, then $\neg Z$ is *true*, so the second clause is *true*.
- If Z is *false*, one of $X_1 \vee \cdots \vee X_i$ must be *true*.
- Since one of $X_1 \vee \cdots \vee X_i$ must be *true*, the result must be *true*.

$$\frac{(X_1 \vee \cdots \vee X_i \vee Z) \quad (Y_1 \vee \cdots \vee Y_j \vee \neg Z)}{(X_1 \vee \cdots \vee X_i \vee Y_1 \vee \cdots \vee Y_j)}$$

Proof by Resolution

Again, we use proof by contradiction.

To prove something true, we add its negation to the knowledge base and demonstrate that this is unsatisfiable.

Then we apply resolution until we derive the empty clause (which we know must be *false*).

Proof by Resolution

Knolwedge Base:

1. $\forall x\, \neg breeze(x) \rightarrow (\forall y\, adjacent(x, y) \rightarrow \neg pit(y))$
2. $\forall x\, \neg stench(x) \rightarrow (\forall y\, adjacent(x, y) \rightarrow \neg wumpus(y))$
3. $\forall x\, \neg pit(x) \wedge \neg wumpus(x) \rightarrow safe(x)$
4. $breeze(B1)$
5. $\neg stench(B1)$
6. $stench(A2)$
7. $\neg breeze(A2)$

Query: $safe(B2)$

Proof by Resolution

Knolwedge Base:

1. $\forall x\ \neg breeze(x) \rightarrow \left(\forall y\ adjacent(x,y) \rightarrow \neg pit(y)\right)$
2. $\forall x\ \neg stench(x) \rightarrow \left(\forall y\ adjacent(x,y) \rightarrow \neg wumpus(y)\right)$
3. $\forall x\ \neg pit(x) \land \neg wumpus(x) \rightarrow safe(x)$
4. $breeze(B1)$
5. $\neg stench(B1)$
6. $stench(A2)$
7. $\neg breeze(A2)$

Query: $safe(B2)$

Not shown:

- $adjacent(A1, A2)$
- $adjacent(A2, A3)$
- $adjacent(A3, A4)$
- etc…

Proof by Resolution

Knowledge Base: (in CNF)

1. $breeze(a) \lor \neg adjacent(a,b) \lor \neg pit(b)$
2. $stench(c) \lor \neg adjacent(c,d) \lor \neg wumpus(d)$
3. $pit(e) \lor wumpus(e) \lor safe(e)$
4. $breeze(B1)$
5. $\neg stench(B1)$
6. $stench(A2)$
7. $\neg breeze(A2)$

Query: $safe(B2)$

Proof by Resolution

Knowledge Base:

1. $breeze(a) \lor \neg adjacent(a,b) \lor \neg pit(b)$
2. $stench(c) \lor \neg adjacent(c,d) \lor \neg wumpus(d)$
3. $pit(e) \lor wumpus(e) \lor safe(e)$
4. $breeze(B1)$
5. $\neg stench(B1)$
6. $stench(A2)$
7. $\neg breeze(A2)$
8. $\neg safe(B2)$

Proof by Resolution

Clause 3: $pit(e) \vee wumpus(e) \vee safe(e)$

Clause 8: $\neg safe(B2)$

Clause 9:

Unifier = {}

Proof by Resolution

Clause 3: $pit(e) \vee wumpus(e) \vee safe(e)$

Clause 8: $\neg safe(B2)$

Clause 9:

Unifier = $\{e/B2\}$

Proof by Resolution

Clause 3: $pit(e) \lor wumpus(e) \lor safe(e)$

Clause 8: $\neg safe(B2)$

Clause 9: $pit(B2) \lor wumpus(B2)$

Unifier = $\{e/B2\}$

Proof by Resolution

Clause 3: $pit(e) \vee wumpus(e) \vee safe(e)$

Clause 8: $\neg safe(B2)$

Clause 9: $pit(B2) \vee wumpus(B2)$

Unifier = $\{e/B2\}$

Notice in the resulting clause that the unifier has been substituted.

Proof by Resolution

Clause 9: $pit(B2) \vee wumpus(B2)$

Clause 1: $breeze(a) \vee \neg adjacent(a, b) \vee \neg pit(b)$

Clause 10:

Unifier = {}

Proof by Resolution

Clause 9: $pit(B2) \vee wumpus(B2)$

Clause 1: $breeze(a) \vee \neg adjacent(a,b) \vee \neg pit(b)$

Clause 10:

Unifier = $\{b/B2\}$

Proof by Resolution

Clause 9: $pit(B2) \lor wumpus(B2)$

Clause 1: $breeze(a) \lor \neg adjacent(a, b) \lor \neg pit(b)$

Clause 10: $wumpus(B2) \lor breeze(a) \lor \neg adjacent(a, B2)$

Unifier = $\{b/B2\}$

Proof by Resolution

Clause 10: $wumpus(B2) \vee breeze(a) \vee \neg adjacent(a, B2)$

Clause 2: $stench(c) \vee \neg adjacent(c, d) \vee \neg wumpus(d)$

Clause: 11:

Unifier = {}

Proof by Resolution

Clause 10: $wumpus(B2) \vee breeze(a) \vee \neg adjacent(a, B2)$

Clause 2: $stench(c) \vee \neg adjacent(c, d) \vee \neg wumpus(d)$

Clause: 11:

Unifier = $\{d/B2\}$

Proof by Resolution

Clause 10: $wumpus(B2) \lor breeze(a) \lor \neg adjacent(a, B2)$

Clause 2: $stench(c) \lor \neg adjacent(c, d) \lor \neg wumpus(d)$

Clause: 11: $breeze(a) \lor \neg adjacent(a, B2) \lor stench(c) \lor \neg adjacent(c, B2)$

Unifier = $\{d/B2\}$

Proof by Resolution

Clause: 11: $breeze(a) \vee \neg adjacent(a, B2) \vee stench(c) \vee \neg adjacent(c, B2)$

Clause 7: $\neg breeze(A2)$

Clause 12:

Unifier = {}

Proof by Resolution

Clause: 11: $breeze(a) \vee \neg adjacent(a, B2) \vee stench(c) \vee \neg adjacent(c, d)$

Clause 7: $\neg breeze(A2)$

Clause 12:

Unifier = $\{a/A2\}$

Proof by Resolution

Clause: 11: $breeze(a) \lor \neg adjacent(a, B2) \lor stench(c) \lor \neg adjacent(c, d)$

Clause 7: $\neg breeze(A2)$

Clause 12: $\neg adjacent(A2, B2) \lor stench(c) \lor \neg adjacent(c, B2)$

Unifier = $\{a/A2\}$

Proof by Resolution

Clause 12: $\neg adjacent(A2, B2) \vee stench(c) \vee \neg adjacent(c, B2)$

Clause 5: $\neg stench(B1)$

Clause 13:

Unifier = {}

Proof by Resolution

Clause 12: $\neg adjacent(A2, B2) \vee stench(c) \vee \neg adjacent(c, B2)$

Clause 5: $\neg stench(B1)$

Clause 13:

Unifier = $\{c/B1\}$

NIL

Proof by Resolution

Clause 12: $\neg adjacent(A2, B2) \lor stench(c) \lor \neg adjacent(c, B2)$

Clause 5: $\neg stench(B1)$

Clause 13: $\neg adjacent(A2, B2) \lor \neg adjacent(B1, B2)$

Unifier = $\{c/B1\}$

NIL

Proof by Resolution

Clause 13: $\neg adjacent(A2, B2) \vee \neg adjacent(B1, B2)$

Clause not shown: $adjacent(A2, B2)$

Clause 14:

Unifier = {}

Proof by Resolution

Clause 13: $\neg adjacent(A2, B2) \vee \neg adjacent(B1, B2)$

Clause not shown: $adjacent(A2, B2)$

Clause 14:

Unifier = {}

Proof by Resolution

Clause 13: $\neg adjacent(A2, B2) \vee \neg adjacent(B1, B2)$

Clause not shown: $adjacent(A2, B2)$

Clause 14: $\neg adjacent(B1, B2)$

Unifier = {}

Proof by Resolution

Clause 14: $\neg adjacent(B1, B2)$

Clause not show: $adjacent(B1, B2)$

Clause 15:

Unifier = {}

Proof by Resolution

Clause 14: $\neg adjacent(B1, B2)$

Clause not show: $adjacent(B1, B2)$

Clause 15:

Unifier = {}

Proof by Resolution

Clause 14: $\neg adjacent(B1, B2)$

Clause not show: $adjacent(B1, B2)$

Clause 15: $\square$

Unifier = {}

NIL

Proof by Resolution

Clause 14: $\neg adjacent(B1, B2)$

Clause not show: $adjacent(B1, B2)$

Clause 15: $\square$ Q.E.D.

Unifier = {}

Value of Resolution

Unlike logic programming…

Resolution is **sound**, meaning it only deduces true statements.

When paired with any complete search algorithm, resolution is also **complete**, meaning it will eventually deduce a statement that is logically equivalent to any entailed statement.

Decidability

Given a first order predicate logic knowledge base and a query, can we prove whether or not the query is entailed?

If it is entailed, resolution will succeed and answer "yes."

If it is not entailed, resolution may run forever and return no answer.

The question of entailment for FOPL is **semidecidable**.

Semidecidability

Consider this knowledge base:

- $child(y, x) \rightarrow ancestor(x, y)$
- $ancestor(x, y) \wedge ancestor(y, z) \rightarrow ancestor(x, z)$

Suppose we have the function $father(x)$, which represents a person's father.

We want to know if Adam is your ancestor.

Semidecidability

We want to know if Adam is your ancestor.

Adam might be $father(x)$

or he might be $father(father(x))$

or he might be $father(father(father(x)))$...

If he is your ancestor, eventually we will find him.

If not, we will keep nesting forever.