CS440/ECE448: Intro to Artificial Intelligence

Lecture 10: Even more on predicate logic

Prof. Julia Hockenmaier
juliahmr@illinois.edu

http://cs.illinois.edu/fa11/cs440

Inference in predicate logic

All men are mortal.
Socrates is a man.
Socrates is mortal.

We need a new version of modus ponens:

$$\frac{\begin{array}{l}\forall x\ P(x) \longrightarrow Q(x) \\ \quad P(s')\end{array}}{Q(s')}$$

How do we deal with quantifiers and variables?

Solution 1: **Propositionalization**
Ground all the variables.

Solution 2: **Lifted inference**
Ground (skolemize) all the existentially quantified variables. All remaining variables are universally quantified.
Use unification.

Prerequisites for lifted inference: Skolemization and Unification

Skolemization: remove existentially quantified variables

Replace any existentially quantified variable $\exists x$ that is in the scope of universally quantified variables $\forall y_1 \ldots \forall y_n$ with a new function $F(y_1,\ldots,y_n)$ (a **Skolem function**)

Replace any existentially quantified variable $\exists x$ that is not in the scope of any universally quantified variables with a new constant c (a **Skolem term**)

The effect of Skolemization

$\forall x\ \forall y\ \exists w\ \forall z\ Q(x, y, w, z, G(w, x))$

is equivalent to

$\forall x\ \forall y\ \forall z\ Q(x, y, P(x, y), z, G(P(x, y), x))$

where P is the Skolem function for w.

NB: the Skolem function is a function, so this is not decidable anymore.

Universal quantifiers: Modus ponens

With propositionalization:

$$\frac{\dfrac{\forall x\ \text{human}(x) \longrightarrow \text{mortal}(x)}{\text{human}(s') \longrightarrow \text{mortal}(s')}\ (\text{UI}) \qquad \text{human}(s')}{\text{mortal}(s')}\ (\text{MP})$$

How can we match $\text{human}(s')$ and $\forall x\ \text{human}(x) \longrightarrow \text{mortal}(x)$ directly?

Substitutions

A *substitution* θ is a set of pairings of variables v_i with terms t_i:

$$\theta = \{v_1/t_1, v_2/t_2, v_3/t_3, \ldots, v_n/ t_n\}$$

- Each variable v_i is distinct
- t_i can be any term (variable, constant, function), as long as it does not contain v_i directly or indirectly

NB: the order of variables in θ doesn't matter

$\{x/y, y/f(a)\} = \{y/f(a), x/y\} = \{x/f(a), y/f(a)\}$

Unification

Two sentences φ and ψ **unify** to σ

(UNIFY(φ, ψ) = σ)

if σ is a substitution such that

SUBST(σ,φ) = SUBST (σ,ψ).

Example:

UNIFY(like(x, M'), like(C',y)) ={x/C', y/M'}

Unification

A set of sentences $\varphi_{1,}\ldots\varphi_n$ **unify** to σ
if for all $i \neq j$: $\textsc{Subst}(\sigma,\varphi_i) = \textsc{Subst}(\sigma, \varphi_j)$.

σ is the unifier of $\varphi_{1,}\ldots\varphi_n$
$\textsc{Subst}(\sigma,\varphi_i)$ is a unification instance.

Standardizing apart

Unification is not well-behaved if φ and ψ contain the same variable:

UNIFY(like(x, M'), like(C',x)): fail.

We need to *standardize* φ and ψ *apart* (rename this variable in one term):

UNIFY(like(x, M'), like(C',y)) = {x/C', y/M'}
to yield like(C',M')

Do these unify?
(Single lower case letters are variables)

UNIFY(P(x,y,z), P(w, w, Fred))

σ = {x=Fred, y=Fred, z=Fred, w=Fred}

Equivalently: σ' = {x=Fred, w=y, z=Fred , y=x}

Both yield P(Fred,Fred,Fred)

Are there others?

UNIFY(P(x,y,z), P(w, w, Fred))

σ = {x=Mary, y=Mary, z=Fred, w=Mary}

Equivalently: σ' = {x=Mary, w=y, z=Fred , y=x}

Both yield P(Mary, Mary, Fred)

Most General Unifier (MGU)

σ is the most general unifier (MGU) of φ and ψ if it imposes the fewest constraints.

The MGU of φ and ψ is unique.
(modulo alphabetic variants, i.e. different variable names)

Applying the MGU to an expression yields a *most general unification instance*.

We often define $\text{U}\textsc{nify}(\varphi, \psi)$ to return $\text{MGU}(\varphi, \psi)$

What is the MGU?

MGU(P(x,y,z), P(w,w,Fred))

σ ={x=w, y=w, z=Fred} yields P(w,w,Fred)

Equivalently, σ ={x=u, y=u, w=u, z=Fred} yields the alphabetic variant P(u,u,Fred)

What is the MGU?

MGU(m(Ann, x, Bob), m(Ann, x, Bob)):

m(Ann, x, Bob)

MGU(m(Ann, x, Bob), m(y, x, Chuck)):

fail.

MGU(m(Ann, x, Bob), m(y, x, Father-of(Chuck)):

fail.

MGU(p(w, w, Fred) , p(x, y, y)):

p(Fred, Fred, Fred)

MGU(q(r, r), q(x, F(x))):

fail

MGU(r(g(x,Bob),y, y), r(z,g(Fred,w),z)):

r(g(x,Bob), g(Fred,w), g(Fred(w)))

Lifted inference: Generalized Modus Ponens

Generalized modus ponens

If $p_1', \ldots, p_n'$, $p_1 \ldots p_n$ are atomic sentences with universally quantified variables, and there is a substitution θ such that $\text{SUBST}(\theta, p_i') = \text{SUBST}(\theta, p_i)$

$$\frac{p_1' \ \ldots \ p_n' \qquad (p_1 \wedge \ldots \wedge p_n) \rightarrow q}{\text{SUBST}(\theta, q)} \ \text{(GMP)}$$

Generalized modus ponens

Another way to look at GMP:

θ makes $p_1' \wedge \ldots \wedge p_n'$ and $p_1 \wedge \ldots \wedge p_n$ equal:

$$\underbrace{\text{SUBST}(\theta, p_1' \wedge \ldots \wedge p_n')}_{\varphi} = \underbrace{\text{SUBST}(\theta, p_1 \wedge \ldots \wedge p_n)}_{\varphi}$$

With a slight abuse of notation….

$$\frac{\textsc{Subst}(\theta, p_1' \wedge \ldots \wedge p_n') = \varphi \qquad \textsc{Subst}(\theta, p_1' \wedge \ldots \wedge p_n') \rightarrow \textsc{Subst}(\theta, q) = \varphi \rightarrow \textsc{Subst}(\theta, q)}{\textsc{Subst}(\theta, q)} \text{(MP)}$$

Generalized modus ponens

Knowledge base:

A person that sells drugs is a criminal.

$\forall x \forall y\ [s(x,y) \wedge p(x) \wedge d(y) \longrightarrow c(x)]$

Socrates is a person: $p(s')$

Socrates sells anything: $\forall z\ s(s',z)$

Cannabis is a drug: $d(c')$

Query:

Is Socrates a criminal? $c(s')$

Generalized modus ponens

$$\frac{\forall x \forall y[s(x,y) \wedge p(x) \wedge d(y) \longrightarrow c(x)] \quad p(s') \quad d(c') \quad \forall z s(s',z)}{\textsc{Subst}(\{x/s', y/c', z/c'\}, c(x)) \equiv c(s')} \;(\text{GMP})$$

$$\textsc{Subst}(\{x/s',y/c',z/c'\},\forall x \forall y[s(x,y) \wedge p(x) \wedge d(y) \longrightarrow c(x)])$$
$$\equiv \; s(s',c') \wedge p(s') \wedge d(t') \; \longrightarrow c(s')$$

Generalized modus ponens

This is a **lifted** version of modus ponens: it raises modus ponens from ground propositional logic to first-order logic.

Lifting is more efficient than propositionalization: only necessary substitutions are made.

Inference with GMP: Forward chaining for definite clauses

First order definite clauses

Definite clauses have exactly one positive literal.

Implications:

$[p_1(x_1,\ldots,x_n) \wedge \ldots \wedge p_m(x_1,\ldots,x_n)] \rightarrow q(x_1,\ldots,x_n)$

premise consequent

$\equiv \neg [p_1(x_1,\ldots,x_n) \wedge \ldots \wedge p_m(x_1,\ldots,x_n)] \vee q(x_1,\ldots,x_n)$

$\equiv \neg p_1(x_1,\ldots,x_n) \vee \ldots \vee \neg p_m(x_1,\ldots,x_n) \vee q(x_1,\ldots,x_n)$

Facts: $q(x_1,\ldots,x_n)$.

Generalized Modus Ponens in definite clause form

Given $(p_1 \wedge \ldots \wedge p_n) \rightarrow q$ and $p_1', \ldots, p_n'$
with $\textsc{Unify}(p_1 \wedge \ldots \wedge p_n, p_1' \wedge \ldots \wedge p_n') = \theta$,
prove q.

As def. clause: $(p_1 \wedge \ldots \wedge p_n) \rightarrow q \equiv \neg (p_1 \wedge \ldots \wedge p_n) \vee q$
$\equiv \neg p_1 \vee \ldots \vee \neg p_n \vee q$

$$\frac{p_1' \ldots \; p_n' \qquad \neg p_1 \vee \ldots \vee \neg p_n \vee q}{\textsc{Subst}(\theta, q)} \; (MP)$$

1. Americans who sell weapons to enemies are criminals

$\forall x \forall y \forall z\ [(a(x) \wedge w(y) \wedge e(z) \wedge sell(x,y,z)) \rightarrow c(x)]$

2. Nono has some weapons.

$\exists x[owns(N', x) \wedge w(x)].$

3. Its weapons were sold by West.

$\forall x[(owns(N', x) \wedge w(x)) \rightarrow sell(W', N', x)].$

4. West is an American.

$a(W')$

5. Nono is an enemy

$e(N')$

Query: is West a criminal? $c(W')$

In definite clause form

1. $\forall x \forall y \forall z\ [(a(x) \wedge w(y) \wedge e(z) \wedge sell(x,y,z)) \rightarrow c(x)]$
$\neg a(x) \vee \neg w(y) \vee \neg e\ (z) \vee \neg sell(x,y,z) \vee c(x)$

2. $\exists x[owns(N', x) \wedge w(x)]$.
2a) $owns(N', M')$ 2b) $w(M')$

3. $\forall x[(owns(N', x) \wedge w(x)) \rightarrow sell(W', N',x)]$.
$\neg owns(N', x) \vee \neg w(x) \vee sell(W', N',x)$.

4. $a(W')$ $a(W')$
5. $e(N')$ $e(N')$

Forward chaining:
apply GMP, starting from premises

owns(N,M) w(M) $\neg$owns(N,x)$\vee\neg$w(x)$\vee$sell(W,N,x).

——————————————————————— (GMP 2a, 2b, 3)

6. sell(W,N,M).

a(W) e(N) w(M) sell(W,N,M) $\neg$a(x)$\vee\neg$w(y)$\vee\neg$e(z)$\vee\neg$sell(x,y,z)$\vee$c(x)

——————————————————————— (GMP 4, 5, 2b, 6, 1)

7. c(W).

Yes, West is a criminal.

Inference with definite clauses: backward chaining

Two ways to use modus ponens

$$\frac{\varphi \longrightarrow \psi \qquad \varphi}{\psi}\ (\mathrm{MP})$$

Forward: I know that φ implies ψ. I also know φ. Hence, I can conclude that ψ is true as well.

Backward: I want to know whether ψ is true.
I know that φ implies ψ. Hence, if I can prove φ,
I can conclude that ψ is true as well.

Backward chaining

Goal: prove that the literal q' is true.

1. Find an implication clause $\neg p_1 \vee \ldots \vee \neg p_n \vee q$ such that goal q' unifies with consequent q.
$\text{UNIFY}(q', q) = \theta'$

2. Apply θ' to $\neg p_1 \vee \ldots \vee \neg p_n \vee q$.
$\text{SUBST}(\theta', \neg p_1 \vee \ldots \vee \neg p_n \vee q) = \neg p''_1 \vee \ldots \vee \neg p''_n \vee q''$

3. Find a unifier θ'' that allows you to prove that each literal p''_i is true. (Recursion!)

findImplications(*goal*, θ) returns a list of *implications* whose consequent unifies with *goal*, and the corresponding unifier θ'

backwardChain(literal *goal*, unifier θ)
    goal' = SUBST(*goal*, θ)
    foreach (clause *implication*, unifier θ')
        in findImplications(*goal*, θ):
        foreach p_i **in** *implication*.PREMISES:
            (boolean *retval*, unifier θ_i) =
                backwardChain(p_i, θ')
            if retval == false: **goto next** implication;
            θ' = θ_i
        if retval: **return** (true, θ');
    return (false, θ);

Logic programming with PROLOG

Horn clauses: *at most* one positive literal.

```
path(X,Z) :- path(X,Y), link(Y,Z).
consequent:- premise1, premise2.
```

Inference: uses backward chaining

Database semantics:

- Each constant refers to a unique object (no two names for the same object)
- Domain closure: domain consists only of those objects for which we have a name.
- Closed world assumption: if we don’t know that P is true, we assume it’s false.

Problem with backward chaining

Backward chaining is depth-first search.
It can go down infinite branches of the search tree.

This is fine (Prolog will try base case first):

```
path(X,Z) :- link(X,Z).
path(X,Z) :- path(X,Y), link(Y,Z).
```

This loops (Prolog will never get to the base case):

```
path(X,Z) :- path(X,Y), link(Y,Z).
path(X,Z) :- link(X,Z).
```

Inference in predicate logic: Resolution

Conjunctive normal form

CNF (in general): arbitrary number of positive literals.
Again, convert to prenex NF, skolemize and drop universal quantifiers.

The CNF of φ is inferentially equivalent to φ:
$\mathrm{CNF}(\varphi)$ is unsatisfiable iff φ is unsatisfiable.

Proof strategy: by contradiction (aka refutation).
To prove $\varphi \models \psi$, show $\varphi \wedge \neg\psi$ is unsatisfiable.

Inference rule: resolution

Translation to CNF

1. Eliminate implications: $(\varphi \rightarrow \psi) \equiv (\neg\varphi \vee \psi)$
2. Standardize variables apart
3. Translate to prenex NF: move quantifiers outwards (across negation, connectives)
4. Move ¬ negation inside connectives
   $\neg(\varphi \wedge \psi) \equiv (\neg\varphi \vee \neg\psi)$ $\neg(\varphi \vee \psi) \equiv (\neg\varphi \wedge \neg\psi)$
5. Skolemize $\exists x$
6. Drop universal quantifiers $\forall$
7. Distribute $\vee$ over $\wedge$

Resolution

A lifted version of propositional resolution:

If $\boldsymbol{p_i}$ unifies with $\neg \boldsymbol{q_j}$: $\textsc{Unify}(\boldsymbol{p_i}, \neg \boldsymbol{q_j}) = \theta$

$$\frac{p_1 \vee \ldots \vee \boldsymbol{p_i} \vee \ldots \vee p_n \qquad q_1 \vee \ldots \vee \boldsymbol{q_j} \vee \ldots \vee q_m}{\textsc{Subst}(\theta, p_1 \vee \ldots \vee p_{i-1} \vee p_{i+1} \vee \ldots \vee p_n \vee q_1 \vee \ldots \vee q_{j-1} \vee q_{j+1} \vee \ldots \vee q_m)}$$

Resolution is complete for FOL.
(again, we assume **factoring**: no duplicate literals
replace $\ldots \vee p \vee \ldots \vee p \vee \ldots.$ with $\ldots \vee p \vee \ldots.$)

Why $\text{UNIFY}(p_i, \neg q_j)$ and not $\text{UNIFY}(p_i, q_j)$?

Propositional resolution: $q_i \equiv \neg p_i$

$$\frac{p_1 \vee \ldots \vee p_i \vee \ldots \vee p_n \qquad q_1 \vee \ldots \vee \neg p_i \vee \ldots \vee q_m}{p_1 \vee \ldots \vee p_{i-1} \vee p_{i+1} \vee \ldots \vee p_n \vee q_1 \vee \ldots \vee q_{j-1} \vee q_{j+1} \vee \ldots \vee q_m}$$

Long answer: We know that $\text{UNIFY}(\boldsymbol{p}, \neg \boldsymbol{q}) = \theta$.

Apply the unifier θ to $\boldsymbol{p}$ and to $\boldsymbol{q}$:

$\text{SUBST}(\theta, \boldsymbol{p}) = \boldsymbol{p}'$ $\quad$ $\text{SUBST}(\theta, \boldsymbol{q}) = \boldsymbol{q}'$

How do $\boldsymbol{p}'$ and $\boldsymbol{q}'$ look like?

Answer: $\boldsymbol{q}' \equiv \neg \boldsymbol{p}'$

Short answer: $\text{UNIFY}(\boldsymbol{p}, \neg \boldsymbol{p}) = \text{FAIL}$.

Today's key concepts

Unification:

to deal with universal variables

Lifted inference:

Generalized modus ponens

forward chaining

backward chaining

first order resolution