Natural Deduction and GMP

Lecture 5

Outline

1. Natural Deduction Systems
2. Reasoning with Generalized Modus Ponens

Outline

1. Natural Deduction Systems

2. Reasoning with Generalized Modus Ponens

Natural Deduction

- In a natural deduction syntactic inference system we typically have a large set of rules of inference:
  - Two rules for each connective: an introduction rule and an elimination rule.
- Inference rules are typically represented as

$$\frac{\phi_1, \phi_2, \ldots, \phi_n}{\psi}$$

  where ϕ_i and ψ are sentences.

$\wedge$-Rules

- $\wedge$-Introduction:

$$\frac{\phi, \psi}{\phi \wedge \psi}$$

- $\wedge$-Elimination: (two rules)

$$\frac{\phi \wedge \psi}{\phi \text{ (or } \psi)}$$

$\vee$-Rules

- $\vee$-Introduction:

$$\frac{\phi}{\phi \vee \psi} \qquad (or\ \frac{\phi}{\psi \vee \phi})$$

- $\vee$-Elimination: (two rules)

$$\frac{\phi \vee \psi, \neg\psi \text{ (or } \neg\phi)}{\phi \text{ (or } \psi)}$$

$\Leftrightarrow$-Rules

- $\Leftrightarrow$-Introduction:

$$\frac{\phi \Rightarrow \psi, \psi \Rightarrow \phi}{\phi \Leftrightarrow \psi}$$

- $\Leftrightarrow$-Elimination: (two rules)

$$\frac{\phi \Leftrightarrow \psi}{\phi \Rightarrow \psi \text{ (or } \psi \Rightarrow \phi)}$$

$\Rightarrow$ and $\neg$ Rules

- $\Rightarrow$-Elimination (Modus Ponens):

$$\frac{\phi \Rightarrow \psi, \phi}{\psi}$$

- $\neg$-Elimination:

$$\frac{\neg\neg\phi}{\phi}$$

∀ and ∃ Rules

$$\frac{\forall x(\phi)}{\textsc{Subst}(\{t/x\}, \phi)}$$

Subst({*a/x*, *b/y*}, *P(x,y,a)*)
= *P(a,b,a)*

$$\frac{\textsc{Subst}(\{t/x\}, \phi)}{\exists x(\phi)}$$

$$\frac{\exists x(\phi)}{\textsc{Subst}(\{c/x\}, \phi)}$$

- t is an arbitrary term.
- c has not been previously used in the derivation (a Skolem constant).
- c does not occur in the conclusion.

Proofs and Derivations

- A proof of $KB \vdash \phi$ is a proof by construction: construct a derivation of ϕ from KB.
- Such a derivation is a sequence of sentences ending with ϕ.
- Each sentence in the sequence is either in KB, or follows from earlier sentences by one of the inference rules.
- If $KB = \{\}$, then the derivation is a proof of the theorem ϕ.

Example

Prove that $\{A, (B \Rightarrow \neg C), ((A \wedge B) \Rightarrow (D \vee C)), B\} \vdash D$

Example

Prove that $\{A, (B \Rightarrow \neg C), ((A \wedge B) \Rightarrow (D \vee C)), B\} \vdash D$

1.A	(hypothesis)
2.$(B \Rightarrow \neg C)$	(hypothesis)
3.$(A \wedge B) \Rightarrow (D \vee C)$	(hypothesis)
4.B	(hypothesis)
5.$\neg C$	(2, 4, $\Rightarrow$-Elim)
6.$A \wedge B$	(1, 4, $\wedge$-Intro)
7.$D \vee C$	(3, 6, $\Rightarrow$-Elim)
8.D	(5, 7, $\vee$-Elim)

Another Example

1.$\forall x(P(x) \Rightarrow Q(x))$	(hypothesis)
2.$\exists y P(y)$	(hypothesis)
3.$P(a)$	(2, $\exists$-elim)
4.$P(a) \Rightarrow Q(a)$	(1, $\forall$-elim)
5.$Q(a)$	(3, 4, $\Rightarrow$-elim)
6.$\exists x Q(x)$	(5, $\exists$-intro)

Reasoning as Search

- Finding a proof is a search problem.
- A state is a set of sentences.
- The initial state is the initial KB.
- The operators are defined by the rules of inference and the sentences in the KB.
- The goal state is a set containing the query sentence.

Problems with Natural Deduction

- The number of rules is big.
- The branching factor increases with the size of the KB.
- Universal elimination can have a huge branching factor on its own.
- A lot of time is typically spent combining atomic sentences into conjunctions, instantiating universal rules to match, and then applying Modus Ponens.

Outline

1. Natural Deduction Systems
2. Reasoning with Generalized Modus Ponens

Generalized Modus Ponens

$$\frac{p_1', p_2', \ldots, p_n', (p_1 \wedge p_2 \wedge \cdots \wedge p_n \Rightarrow q)}{\textsc{Subst}(\theta, q)}$$

where $\textsc{Subst}(\theta, p_i') = \textsc{Subst}(\theta, p_i)$, for all i.

- It takes bigger steps.
- Uses substitutions that are guaranteed to work.
- It makes use of a precompilation step that puts sentences into a canonical form on which the rule can apply.

$$P(a, x) \land Q(b, x, y) \Longrightarrow R(x, a, y, b)$$
$$P(z, b)$$
$$P(b, a)$$
$$Q(u, b, b)$$
$$Q(b, a, a)$$

$$P(a, x) \wedge Q(b, x, y) \Longrightarrow R(x, a, y, b)$$
$$P(z, b)$$
$$P(b, a)$$
$$Q(u, b, b)$$
$$Q(b, a, a)$$

$\{a/z,\ b/x\}$

$$P(a, x) \wedge Q(b, x, y) \Longrightarrow R(x, a, y, b)$$
$$P(z, b)$$
$$P(b, a)$$
$$Q(u, b, b)$$
$$Q(b, a, a)$$

$\{a/z,\ b/x,\ b/u,\ b/y\}$

$P(a, x) \wedge Q(b, x, y) \Longrightarrow R(x, a, y, b)$
$P(z, b)$
$P(b, a)$
$Q(u, b, b)$
$Q(b, a, a)$

$\{a/z,\ b/x,\ b/u,\ b/y\}$

$\boxed{R(b, a, b, b)}$

$$\frac{p_1', p_2', \ldots, p_n', (p_1 \wedge p_2 \wedge \cdots \wedge p_n \Rightarrow q)}{\text{SUBST}(\theta, q)}$$

where $\text{SUBST}(\theta, p_i') = \text{SUBST}(\theta, p_i)$, for all i

Canonical Form

- Each sentence should be either an atom or an implication with a conjunction as the antecedent and an atom as a consequent.
- Sentences of this form are called Horn sentences.
- We typically convert a sentence into a set of Horn sentences using $\exists$-elimination and $\wedge$-elimination.

Applying GMP

- A major step in applying GMP is discovering the substitution θ.
  - There could be more than one.
- This involves a process that is at the heart of all first-order reasoning techniques—unification.

Unification

- To unify two FOPL expressions E_1 and E_2 is to find a substitution θ such that $\textsc{Subst}(\theta, E_1) = \textsc{Subst}(\theta, E_2)$.
- θ is a unifier and $\textsc{Subst}(\theta, E_1)$ (or $\textsc{Subst}(\theta, E_2)$) is a common instance of E_1 and E_2.
- Examples:

$\mathbf{E_1}$	$\mathbf{E_2}$	θ	Common Instance
$\textsc{SSet}(A, \mathbb{N})$	$\textsc{SSet}(x, \mathbb{N})$	$\{A/x\}$	$\textsc{SSet}(A, \mathbb{N})$
$\textsc{SSet}(A, y)$	$\textsc{SSet}(x, \mathbb{N})$	$\{A/x, \mathbb{N}/y\}$	$\textsc{SSet}(A, \mathbb{N})$
$\textsc{SSet}(\textsc{Int}(y), y)$	$\textsc{SSet}(x, \mathbb{N})$	$\{\textsc{Int}(\mathbb{N})/x, \mathbb{N}/y\}$	$\textsc{SSet}(\textsc{Int}(\mathbb{N}), \mathbb{N})$

The Most General Unifier

- Note that, in general, two expressions will have an infinite number of unifiers (if we have non-constant function symbols).
- Example For $\text{SSET}(y, z)$ and $\text{SSET}(x, \mathbb{N})$, we have
  - $\theta_1 = \{x/x, x/y, \mathbb{N}/z\}$
  - $\theta_2 = \{A/x, A/y, \mathbb{N}/z\}$
  - $\theta_3 = \{B/x, B/y, \mathbb{N}/z\}$
  - $\ldots$
- Looking ahead, always try to find a most general unifier (MGU)—a unifier that makes the least commitment about the bindings of variables.
- Formally, μ is an MGU of E_1 and E_2 if it is a unifier of E_1 and E_2, and for every unifier θ of E_1 and E_2, there is a substitution τ such that $\theta = \mu \circ \tau$.

The Unification Algorithm

```
UNIFY(E1, E2)
   return UNIFY1(LISTIFY(E1), LISTIFY(E2), {});
UNIFY1(E1, E2, μ)
   if μ = fail then
      return fail;
   if E1 = E2 then
      return μ;
   if VAR?(E1) then
      return UNIFYVAR(E1, E2, μ)
   if VAR?(E2) then
      return UNIFYVAR(E2, E1, μ)
   if ATOM?(E1) or ATOM?(E2) then
      return fail;
   if LENGTH(E1) ≠ LENGTH(E2) then
      return fail;
   return UNIFY1(REST(E1), REST(E2), UNIFY1(FIRST(E1), FIRST(E2), μ))
```

The Variable Unification Algorithm

$\text{UNIFYVAR}(x, e, \mu)$

if $t/x \in \mu$ **and** $t \neq x$ **then**
    return $\text{UNIFY1}(t, e, \mu)$;
$t = \text{SUBST}(\mu, e)$
if x occurs in t **then**
    return fail;
return $\mu \circ \{t/x\}$;

Example

Find the MGU (if it exists) of

1. $P(x, g(x), g(f(a)))$ and $P(f(u), v, v)$
2. $P(a, y, f(y))$ and $P(z, z, u)$
3. $f(x, g(x), x)$ and $f(g(u), g(g(z)), z)$

$P(x, g(x), g(f(a)))$ and $P(f(u), v, v)$

UNIFY(E_1, E_2)

return UNIFY1(LISTIFY(E_1), LISTIFY(E_2), {});

$(P\ x\ (g\ x)\ (g\ (f\ a)))$ $(P\ (f\ u)\ v\ v)$

$(P\ x\ (g\ x)\ (g\ (f\ a)))$ $(P\ (f\ u)\ v\ v)$ $\{\ \ \}$

UNIFY1(E_1, E_2, μ)

```
UNIFY1(E1, E2, μ)
  if μ = fail then
     return fail;
  if E1 = E2 then
     return μ;
  if VAR?(E1) then
     return UNIFYVAR(E1, E2, μ)
  if VAR?(E2) then
     return UNIFYVAR(E2, E1, μ)
  if ATOM?(E1) or ATOM?(E2) then
     return fail;
  if LENGTH(E1) ≠ LENGTH(E2) then
     return fail;
  return UNIFY1(REST(E1), REST(E2), UNIFY1(FIRST(E1), FIRST(E2), μ))
```

P P $\{\ \ \}$

$(x\ (g\ x)\ (g\ (f\ a)))$ $((f\ u)\ v\ v)$ $\{\quad\}$

```
UNIFY1(E1, E2, μ)
  if μ = fail then
    return fail;
  if E1 = E2 then
    return μ;
  if VAR?(E1) then
    return UNIFYVAR(E1, E2, μ)
  if VAR?(E2) then
    return UNIFYVAR(E2, E1, μ)
  if ATOM?(E1) or ATOM?(E2) then
    return fail;
  if LENGTH(E1) ≠ LENGTH(E2) then
    return fail;
  return UNIFY1(REST(E1), REST(E2), UNIFY1(FIRST(E1), FIRST(E2), μ))
```

x $(f\ u)$ $\{\quad\}$

$x \qquad (f\ u) \qquad \{\ \}$

$\textsc{UnifyVar}(x, e, \mu)$
if $t/x \in \mu$ **and** $t \neq x$ **then**
return $\textsc{Unify1}(t, e, \mu)$;
$t = \textsc{Subst}(\mu, e)$
if x occurs in t **then**
return fail;
return $\mu \circ \{t/x\}$;

$\{(f\ u)/x\}$

$((g\ x)\ (g\ (f\ a)))$ $(v\ v)$ $\{(f\ u)/x\}$

```
UNIFY1(E1, E2, μ)
  if μ = fail then
    return fail;
  if E1 = E2 then
    return μ;
  if VAR?(E1) then
    return UNIFYVAR(E1, E2, μ)
  if VAR?(E2) then
    return UNIFYVAR(E2, E1, μ)
  if ATOM?(E1) or ATOM?(E2) then
    return fail;
  if LENGTH(E1) ≠ LENGTH(E2) then
    return fail;
  return UNIFY1(REST(E1), REST(E2), UNIFY1(FIRST(E1), FIRST(E2), μ))
```

v $(g\ x)$ $\{(f\ u)/x\}$

$v \qquad (g\ x) \qquad \{(f\ u)/x\}$

$\textsc{UnifyVar}(x, e, \mu)$
  if $t/x \in \mu$ **and** $t \neq x$ **then**
    return $\textsc{Unify1}(t, e, \mu)$;
  $t = \textsc{Subst}(\mu, e)$
  if x occurs in t **then**
    return fail;
  return $\mu \circ \{t/x\}$;

$\{(f\ u)/x, (g\ (f\ u))/v\}$

$t = (g\ (f\ u))$

$((g\ (f\ a)))$ (v) $\{(f\ u)/x, (g\ (f\ u))/v\}$

```
UNIFY1(E1, E2, μ)
  if μ = fail then
    return fail;
  if E1 = E2 then
    return μ;
  if VAR?(E1) then
    return UNIFYVAR(E1, E2, μ)
  if VAR?(E2) then
    return UNIFYVAR(E2, E1, μ)
  if ATOM?(E1) or ATOM?(E2) then
    return fail;
  if LENGTH(E1) ≠ LENGTH(E2) then
    return fail;
  return UNIFY1(REST(E1), REST(E2), UNIFY1(FIRST(E1), FIRST(E2), μ))
```

v $(g\ (f\ a))$ $\{(f\ u)/x, (g\ (f\ u))/v\}$

$v \quad (g\ (f\ a)) \quad \{(f\ u)/x, (g\ (f\ u))/v\}$

UNIFYVAR(x, e, μ)

if $t/x \in \mu$ **and** $t \neq x$ **then**

return UNIFY1(t, e, μ);

$t = $ SUBST(μ, e)

if x occurs in t **then**

return fail;

return $\mu \circ \{t/x\}$;

$(g\ (f\ u)) \quad (g\ (f\ a)) \quad \{(f\ u)/x, (g\ (f\ u))/v\}$

$(g\ (f\ u))$ $(g\ (f\ a))$ $\{(f\ u)/x, (g\ (f\ u))/v\}$

```
UNIFY1(E1, E2, μ)
  if μ = fail then
    return fail;
  if E1 = E2 then
    return μ;
  if VAR?(E1) then
    return UNIFYVAR(E1, E2, μ)
  if VAR?(E2) then
    return UNIFYVAR(E2, E1, μ)
  if ATOM?(E1) or ATOM?(E2) then
    return fail;
  if LENGTH(E1) ≠ LENGTH(E2) then
    return fail;
  return UNIFY1(REST(E1), REST(E2), UNIFY1(FIRST(E1), FIRST(E2), μ))
```

u a $\{(f\ u)/x, (g\ (f\ u))/v\}$

$u \qquad a \qquad \{(f\ u)/x, (g\ (f\ u))/v\}$

UNIFYVAR(x, e, μ)

if $t/x \in \mu$ **and** $t \neq x$ **then**

return UNIFY1(t, e, μ);

$t =$ SUBST(μ, e)

if x occurs in t **then**

return fail;

return $\mu \circ \{t/x\}$;

$\{(f\ a)/x, a/u, (g\ (f\ a))/v\}$

Chaining Algorithms

- Systems based on generalized Modus Ponens typically use chaining algorithms for reasoning.
- Forward chaining:
  - Implemented as part of the TELL function.
  - Chains on antecedents of rules, deriving anything that follows from the added sentence.
- Backward chaining:
  - Implemented as part of the ASK function.
  - Chains backwards on the consequents of rules that match the queried sentence.

Forward chaining:

- Implemented as part of the TELL function.
- Chains on antecedents of rules, deriving anything that follows from the added sentence.

$$
\begin{array}{l} P \Rightarrow Q \\ Q \Rightarrow R \\ R \Rightarrow S \end{array}
\xrightarrow{\quad Tell(P) \quad}
\begin{array}{l} P \Rightarrow Q \\ Q \Rightarrow R \\ R \Rightarrow S \\ P \\ Q \\ R \\ S \end{array}
$$

Backward chaining:

- Implemented as part of the ASK function.
- Chains backwards on the consequents of rules that match the queried sentence.

$$P \Rightarrow Q$$
$$Q \Rightarrow R$$
$$R \Rightarrow S$$
$$P$$

$$\xrightarrow{Ask(S)}$$

$$P \Rightarrow Q$$
$$Q \Rightarrow R$$
$$R \Rightarrow S$$
$$P$$
$$Q$$
$$R$$
$$S$$

Problems with Generalized Modus Ponens

- Generalized Modus Ponens is not complete.
- That is, there are sentences ϕ such that $\models \phi$ and not $\vdash_{\text{GMP}} \phi$.
- The main reason is that some FOL sentences cannot be put in Horn normal form.
- For example, $\forall x(\neg P(x) \Rightarrow Q(x))$.
- Next time, we shall consider a complete system also based on a single rule of inference: resolution.

Natural Deduction and GMP

Lecture 5