Outline

- First-order inference rules: Modus Ponens, Modus Tollens
- Resolution

Modus Ponens

- Assume you are given the following two statements:
  - "you are in this class"
  - "if you are in this class, you will get a grade"

- Let p = "you are in this class"
- Let q = "you will get a grade"

- By Modus Ponens, you can conclude that you will get a grade

$$\begin{array}{ll} & p \\ & \underline{p \rightarrow q} \\ \therefore & q \end{array}$$

Modus Ponens

- Consider $(p \wedge (p \rightarrow q)) \rightarrow q$

p	q	$p \rightarrow q$	$p \wedge (p \rightarrow q))$	$(p \wedge (p \rightarrow q)) \rightarrow q$
T	T	T	T	T
T	F	F	F	T
F	T	T	F	T
F	F	T	F	T

$$\begin{array}{ll} & p \\ & \underline{p \rightarrow q} \\ \therefore & q \end{array}$$

Chain Rule

From $p \to q$, and $q \to r$, we can infer $p \to r$

$$\begin{array}{rl} & p \to q \\ & \underline{q \to r} \\ \therefore & p \to r \end{array}$$

Modus Tollens

- Assume that we know: $\neg q$ and $p \rightarrow q$
  - Recall that $p \rightarrow q = \neg q \rightarrow \neg p$
- Thus, we know $\neg q$ and $\neg q \rightarrow \neg p$
- We can conclude $\neg p$

$$\begin{array}{l} \neg q \\ \underline{p \rightarrow q} \\ \therefore \neg p \end{array}$$

Modus Tollens

- Assume you are given the following two statements:
  - "you will not get a grade"
  - "if you are in this class, you will get a grade"

- Let p = "you are in this class"
- Let q = "you will get a grade"

- By Modus Tollens, you can conclude that you are not in this class

Addition and Simplification

- Addition: If you know that p is true, then $p \vee q$ will ALWAYS be true

$$\begin{array}{l} \quad \underline{p} \\ \therefore\ p \vee q \end{array}$$

- Simplification: If $p \wedge q$ is true, then p will ALWAYS be true

$$\begin{array}{l} \quad \underline{p \wedge q} \\ \therefore\ p \end{array}$$

Rules of inference for the universal quantifier

- Assume that we know that $\forall x\, P(x)$ is true
  - Then we can conclude that $P(c)$ is true
    - Here c stands for some specific constant
  - This is called "universal instantiation"

- Assume that we know that $P(c)$ is true for any value of c
  - Then we can conclude that $\forall x\, P(x)$ is true
  - This is called "universal generalization"

Rules of inference for the existential quantifier

- Assume that we know that $\exists x\ P(x)$ is true
  - Then we can conclude that $P(c)$ is true for some value of c
  - This is called "existential instantiation"

- Assume that we know that $P(c)$ is true for some value of c
  - Then we can conclude that $\exists x\ P(x)$ is true
  - This is called "existential generalization"

Anatomy of a propositional function

$$P(x) = \underbrace{x + 5 > x}_{\text{predicate}}$$

variable

Universal instantiation (UI)

- A predicate that has no variables is called a ground atom.
- Every instantiation of a universally quantified sentence is entailed by it:

$$\frac{\forall v\ \alpha}{\text{Subst}(\{v/g\}, \alpha)}$$

for any variable v and ground term g (*Subst(x,y) = substitution of y by x*)

- E.g., $\forall x\ King(x) \wedge Greedy(x) \Rightarrow Evil(x)$ yields:

$King(John) \wedge Greedy(John) \Rightarrow Evil(John)$

$King(Richard) \wedge Greedy(Richard) \Rightarrow Evil(Richard)$

$King(Father(John)) \wedge Greedy(Father(John)) \Rightarrow Evil(Father(John))$

.

.

.

Existential instantiation (EI)

- For any sentence α, variable v, and constant symbol *k that does not appear elsewhere in the knowledge base*:

$$\frac{\exists v\ \alpha}{\text{Subst}(\{v/k\}, \alpha)}$$

- E.g., $\exists x\ Crown(x) \land OnHead(x,John)$ yields:

$$Crown(C_1) \land OnHead(C_1,John)$$

provided C_1 is a new constant symbol, called a Skolem constant

EI versus UI

- UI can be applied several times to *add* new sentences; the new KB is logically equivalent to the old.
- EI can be applied once to replace the existential sentence; the new KB is not equivalent to the old but is satisfiable if the old KB was satisfiable.

Reduction to propositional inference

- Suppose the KB contains just the following:

  $\forall x$ King(x) $\wedge$ Greedy(x) $\Rightarrow$ Evil(x)
  King(John)
  Greedy(John)
  Brother(Richard,John)

- Instantiating the universal sentence in all possible ways, we have:

  King(John) $\wedge$ Greedy(John) $\Rightarrow$ Evil(John)
  King(Richard) $\wedge$ Greedy(Richard) $\Rightarrow$ Evil(Richard)
  King(John)
  Greedy(John)
  Brother(Richard,John)

- The new KB is propositionalized: proposition symbols are
  John, Richard and also King(John), Greedy(John), Evil(John), King(Richard), etc.

Making modus ponens complete

- Modus ponens is incomplete in a general KB
  - Need other inference rules
  - E.g. KB = $\{A \wedge B\}$, we cannot infer anything with MP
- There is a (finite) set of inference rules such that repeatedly applying them forms a complete inference procedure
  - All valid sentences can be inferred in this way
- For each inference rule, we can associate the infinite set of logical axioms (implications) that corresponds to each possible instantiation of the rule
  - E.g. and-elimination corresponds to all axioms of the form $A \wedge B \Rightarrow A$
  - Note that the axioms are in our FOL language, whereas inference rules are in a “meta-language”
- Modus ponens becomes complete if we add to our knowledge bases all the logical axioms corresponding to the other inference rules

Inference in FOL

- **Entailment:** KB$\models\alpha$ whenever every model of KB is also a model of α.

Inference procedures:

- o propositionalization (Universal and Existential elimination; convert to Propositional Logic; apply prop logic inference)
- o lifted inference rules, and in particular refutation/resolution proof for FOL
- o forward/backward chaining for definite clauses

Propositionalization

- Problem: with function symbols, there are infinitely many ground terms,
  - e.g., *Father*(*Father*(*Father*(*John*)))
- Theorem: Herbrand (1930). If a sentence α is entailed by an FOL KB, it is entailed by a finite subset of the propositionalized KB
- Idea: For $n = 0$ to ∞ do
  - create a propositional KB by instantiating with depth-n terms
  - see if α is entailed in this KB (e.g. using refutation)
- Problem: works if α is entailed, loops if α is not entailed
- Theorem: Turing (1936), Church (1936) Entailment for FOL is semi-decidable (algorithms exist that say yes to every entailed sentence, but no algorithm exists that also says no to every non-entailed sentence.)

Propositionalization: Example

∀ x King(x) ∧ Greedy(x) => Evil(x)
King(John)
Hassan
Giro
Lukasz
Jacomo
John
∀ y Greedy(y)
Would like to conclude Evil(John).

By Propositionalization, write out:
King(John) ∧ Greedy(John) => Evil(John)
King(Hassan) ∧ Greedy(Hassan) => Evil(Hassan)
King(Giro) ∧ Greedy(Giro) => Evil(Giro)
King(Jacomo) ∧ Greedy(Jacomo) => Evil(Jacomo)
King(Lukasz) ∧ Greedy(Lukasz) => Evil(Lukasz)
Greedy(John)
Greedy(Hassan) ….

finally, conclude
Evil(John)

Problems with propositionalization

- Propositionalization seems to generate lots of irrelevant sentences
- E.g., from:

  $\forall x$ King(x) $\wedge$ Greedy(x) $\Rightarrow$ Evil(x)

  King(John)

  $\forall y$ Greedy(y)

  Brother(Richard,John)
- It seems obvious that *Evil*(*John*), but propositionalization produces lots of facts such as *Greedy*(*Richard*) that are irrelevant
- With p k-ary predicates and n constants, there are $p \cdot n^k$ instantiations

Generalized Modus Ponens

- Lifted version of the propositional Modus Ponens
- For atomic sentences p_i, p_i', and q, where there is a substitution θ that satisfies $\text{Subst}(\theta,p'_i)=\text{Subst}(\theta,p_i)$ for all *i*

$$\frac{p'_1,\ p'_2,\ \ldots,\ p'_n,\ (p_1 \wedge p_2 \wedge \ldots \wedge p_n \Rightarrow q)}{\text{Subst}(\theta,\ q)}$$

•Example:

p_1' is King(John) p_1 is King(x)

p_2' is Greedy(y) p_2 is Greedy(x)

θ is {x/ John, y/ John} q is Evil(x)

Subst(θ,q) is Evil(John)

we say θ "unifies" p_1' and p_1; and p_2' and p_2.

Soundness of GMP

- For any sentence p with universally quantified variables and for any substitution θ, $p \models Subst(\theta,p)$

- So, from $(p_1',\ldots,p_n')$ we can infer:

  $Subst(\theta,p_1') \wedge \ldots \wedge Subst(\theta,p_n')$ (1)

- From the implication $p_1 \wedge \ldots \wedge p_n \Rightarrow q$ we can infer:

  $Subst(\theta,p_1) \wedge \ldots \wedge Subst(\theta,p_n) \Rightarrow Subst(\theta,q)$ (2)

By the GMP rule, when the first sentence (1) matches the premise of (2) exactly we can infer $Subst(\theta,q)$.

This follows immediately from Modus Ponens.

Unification

- To apply GMP we need a substitution that *unifies* a sentences in the KB with the premises, i.e. to find Subst(θ,p'_i)=Subst(θ,p_i) for all *i*

$$\frac{p'_1, p'_2, \ldots, p'_n, \ (p_1 \wedge p_2 \wedge \ldots \wedge p_n)\ q)}{\text{Subst}(\theta, q)}$$

•**Unification:** find a substitution of variables for terms that makes two sentences equivalent.

Example:

Unify(Knows(John,x); Knows(John,Jane)) ={x/Jane}

Unify(Knows(John,x); Knows(y,Bill)) = {x/Bill, y/John}

Write: unify(α,β) = θ , to denote the unifier θ, e.g. θ=unify(α,β}={x/Jane}

- Unifiers θ=unify(α,β), can:
  - replace a variable by a constant term, e.g. {x/John}
  - replace a variable with a variable, e.g. {x/y}
  - replace a variable by a function expression, e.g. {x / Mother(y)}
    - CAREFUL: need to check variable (e.g. x) does not appear inside the complex term
    - "Occur check"
    - makes complexity quadratic

Examples

Find a unifier for the following sentences:

variable term

1 – unify(Knows(John,x), Knows(John,Jane)) $\theta = \{x/Jane\}$

2 – unify(Knows(John,x), Knows(y,Bill)) $\theta = \{x/Bill, y/John\}$

Generalized Resolution

- Lifted version of resolution

$$\frac{p_1 \vee \ldots \vee p_m,\ \neg q_1 \vee \ldots \vee q_n\,, \quad \mathit{Unify}(p_1, \neg q_1) = \theta}{\mathrm{Subst}(\theta,\ p_2 \vee \ldots \vee p_m \vee q_2 \vee \ldots \vee q_n)}$$

Example:

$Animal(F(x)) \vee Loves(G(x),x)$ and $\neg Loves(u,v) \vee \neg Kills(u,v)$

take unifier $\theta=\{u/G(x), v/x\}$, and get resolvent clause:

$Animal(F(x)) \vee \neg Kills(G(x),x)$

Resolution Algorithm

- start with KB, add $\neg\, \alpha$
- while False $\notin$ KB {
  - find two sentences α, $\beta \in$KB that unify
  - add resolvent(α, β) to KB

}

Resolution is "refutation-complete:" will report "yes" (find the empty clause) if sentence is entailed.

Resolution Strategies

- Complete
  - Breadth-first (slow)
  - Set of Support
  - Linear resolution
  - Subsumption: remove all sentences in your KB that are subsumed. (e.g. remove $A \lor B(k)$ if $B(x)$ is in KB)
- Incomplete
  - Unit resolution
  - Input resolution