G53KRR

Revision

Plan of the lecture

- exam format
- common mistakes
- resolution example
- Bayesian networks
- description logic
- circumscription
- defaults
- any other questions

Exam format

- 4 questions out of 4
- previous papers and answers on the web

Common mistakes 1

First order logic question: show that S1, S2 do not logically entail S3

- Correct answer: describe an interpretation which makes S1 and S2 true and S3 false
- Don't:
  - use truth tables for first order sentences
  - attempt a resolution derivation of [] from S1, S2 and $\neg$S3, and then stop and say 'see, it does not work, so S3 is not entailed'

Common mistakes 2

Resolution

- don't apply resolution to two literals at the same time:

$$MISTAKE: \quad \frac{[A, B],\ [\neg A, \neg B]}{[\,]}$$

  it is not sound! $A \vee B$ and $\neg A \vee \neg B$ should not derive false.

- only substitute for variables (not constants or functional terms) Don't do $a/f(x)$ or $f(x)/a$

$$MISTAKE: \frac{[(P(f(x))],\ [\neg P(a)]}{[\,]}$$

  (you can do x/a, with $f(x)$ becoming $f(a)$).

Resolution example

$KB = \{\forall x Plus(0, x, x), \forall x \forall y \forall z (Plus(x, y, z) \supset Plus(s(x), y, s(z))\}$

(Meaning: $Plus(x, y, z)$ is $x + y = z$,
$0 + x = x$,
$x + y = z \supset ((x + 1) + y = (z + 1)))$

Show that $KB \models \exists u\ Plus(s(s(0)), s(s(s(0))), u)$.
(Meaning: that $\exists u\ (2 + 3 = u)$)

Resolution example continued

1. $[Plus(0, x_1, x_1)]$ (KB) (I renamed x to x_1)
2. $[\neg Plus(x, y, z), Plus(s(x), y, s(z))]$ (KB)
3. $[\neg Plus(s(s(0)), s(s(s(0))), u)]$ (negation of $\exists u Plus(2, 3, u)$)
4. $[Plus(s(0), x_1, s(x_1))]$ from 1,2, $x/0, y/x_1, z/x_1$
5. $[Plus(s(s(0)), x_1, s(s(x_1)))]$ from 4, 2, $x/s(0), y/x_1, z/s(x_1)$
6. [ ] from 3,5, $x_1/s(s(s(0))), u/s(s(s(s(s(0)))))$

Horn clauses

- A Horn clause is a clause with at most one positive literal.
- A unit clause is a clause with one literal.
- A positive Horn clause contains one positive literal.
- A negative Horn clause has no positive literals.
- Examples:
  - $[\neg P(x), \neg Q(x), R(x)]$ (some negative and one positive literal; corresponds to a rule $\forall x(P(x) \wedge Q(x) \supset R(x))$
  - $[R(x)]$ (only one positive literal; corresponds to a fact)
  - $[\neg P(x), \neg Q(x)]$ (only negative literals; corresponds to a query)

SLD resolution

An SLD derivation of clause c from a set of clauses S is a sequence of steps $c_1, \ldots, c_n = c$ such that:

- $c_1 \in S$
- c_{i+1} is a resolvent of c_i and a clause in S.

Basic idea: in a proof, we use the clause we just derived plus some clause in S. We can also arrange things so that all the clauses in the derivation are negative.

SLD resolution is refutation-complete for Horn clauses (if [ ] is derivable at all, it is derivable by SLD resolution).

SLD resolution example

$KB = \{[p], [q], [\neg p, \neg q, r]\}$
Check if r follows, so we add $[\neg r]$ and try to derive $[\,]$ by SLD resolution.
$S = \{[p], [q], [\neg p, \neg q, r], [\neg r]\}$

SLD derivation of $[\,]$ from S:
$c_1 = [\neg r]$ (in S)
$c_2 = [\neg p, \neg q]$ (resolvent of c_1 and $[\neg p, \neg q, r]$)
$c_3 = [\neg q]$ (resolvent of c_2 and $[p]$)
$c_4 = [\,]$ (resolvent of c_3 and $[q]$).

Backward chaining

input: a finite set of atomic sentences $q_1, \ldots, q_n$
output: YES if KB entails all of q_i, NO otherwise
procedure: SOLVE[q1, ..., qn]
if n = 0 then return YES
for each clause c in KB do
.   if c = [not p1,...,not pm, q1] and SOLVE [p1,...,pm,q2,...,qn]
.   then return YES
end for
return NO

Example

Backward chaining corresponds to SLD resolution.
Given $KB = \{[p], [q], [\neg p, \neg q, r]\}$ and a query r:

SOLVE[r]	$[\neg r]$
SOLVE[p,q]	$[\neg p, \neg q]$
SOLVE[q]	$[\neg q]$
SOLVE[ ]	$[\ ]$
YES	

Forward chaining

input: an atomic sentence q
output: YES if KB entails q, NO otherwise
1. if q is marked as solved, return YES
2. if there is $[\neg p_1, \ldots, \neg p_n, q_1]$ in KB such that $p_1, \ldots, p_n$ are marked as solved and q_1 is not marked solved: mark q_1 as solved and go to 1;
else return NO.

Example

Given $KB = \{[p], [q], [\neg p, \neg q, r]\}$ and a query r:

$[p]$ is in KB and p is not marked solved; mark p as solved
another way of looking at it: add p to working memory
$[q]$ is in KB and q is not marked solved; mark q as solved
another way of looking at it: add q to working memory
$[\neg p, \neg q, r]$ is in KB, p, q are marked as solved, and r is not; mark r as solved
another way of looking at it: if p, q are in working memory, and there is a rule $p \wedge q \supset r$, then add r to working memory
YES

Bayesian networks

- Directed acyclic graph
- Nodes: propositional variables; a directed edge from p_i to p_j if the truth of p_i affects the truth of p_j. p_i parent of p_j.
- $$J(\langle P_1, \ldots, P_n \rangle) = Pr(P_1 \wedge \ldots \wedge P_n)$$
- Chain rule

$$Pr(P_1 \wedge \ldots \wedge P_n) = Pr(P_1) \cdot Pr(P_2|P_1) \cdots Pr(P_n|P_1 \wedge \ldots \wedge P_{n-1})$$

- Independence assumption *Each propositional variable in the belief network is conditionally independent from non-parent variables given its parent variables:*

$$Pr(P_i \mid P_1 \wedge \ldots \wedge P_{i-1}) = Pr(P_i \mid parents(P_i))$$

where $parents(P_i)$ is the conjunction of literals which correspond to parents of p_i in the network.

Mistake 3

- Mistake: suppose a network consists of two variables, p_1 and p_2, such that there is an edge from p_1 to p_2. The mistake is to say that $Pr(p_1 \mid p_2) = Pr(p_1)$ because p_2 is not a parent of p_1 (so apply the independence assumption 'in reverse order of indices')
- This is a much more subtle (and understandable given the way the independenc assumption is stated) mistake.
- The independence assumption statement assumes that in the state description, the variables are listed in topological sort order (if there is an edge from p_i to p_j, then p_i appears before p_j in the order). This is always possible since the graph is acyclic. So we never check probability of parent conditioned on a child or a set of descendants.

Circumscription

- The main idea: formalise common sense rules which admit exceptions.
- Rules like ‘Birds fly’ formalised as

$$\forall x(Bird(x) \wedge \neg Ab(x) \supset Flies(x))$$

- To check whether something is entailed by a knowledge base which contains such rules, we only check if it is entailed under the assumption that the set of exceptions $I(Ab)$ is as small as possible
- This is called circumscription or minimal entailment

Example

$$KB = \{Bird(tweety), \forall x(Bird(x) \wedge \neg Ab(x) \supset Flies(x))\}$$

- Classically, $KB \not\models Flies(tweety)$ because there are interpretations of KB where Tweety is an exceptional bird (it is in $I(Ab)$) and it does not fly
- But such interpretations do not minimise the set of exceptions: nothing which is said in KB forces us to think that Tweety is exceptional, so it does not have to be in $I(Ab)$
- If we only consider interpretation which satisfy KB and where the set of exceptions is as small as possible, Tweety is not in this set, so $Bird(tweety) \wedge \neg Ab(tweety)$ holds and hence $Flies(tweety)$ holds
- KB entails $Flies(tweety)$ on 'minimal' interpretations where $I(Ab)$ is circumscribed (made as small as possible)

Definition of minimal entailment

- Let $M_1 = (D, I_1)$ and $M_2 = (D, I_2)$ be two interpretations over the same domain such that every constant and function are interpreted the same way.
- $M_1 \leq M_2 \;\Leftrightarrow\; I_1(Ab) \subseteq I_2(Ab)$
- $M_1 < M_2$ if $M_1 \leq M_2$ but not $M_2 \leq M_1$. (There are strictly fewer abnormal things in M_1).
- *Minimal entailment:* $KB \models_{\leq} \alpha$ iff for all interpretations M which make KB true, either $M \models \alpha$ or M is not minimal (exists M' such that $M' < M$ and $M' \models KB$).

Back to the example

- $KB = \{Bird(tweety), \forall x(Bird(x) \wedge \neg Ab(x) \supset Flies(x))\}$
- $KB \models_{\leq} Flies(tweety)$ because for every interpretation M which makes KB true and $Flies(tweety)$ false, it has to be that $I(tweety) \in I(Ab)$.
- So for for every such interpretation there is an interpretation M' which is just like M, but $I'(tweety) \notin I'(Ab)$ and $I'(tweety) \in I'(Flies)$, and M' still makes KB true and $M' < M$.

Defaults

- A *default rule* consists of a *prerequisite* α, *justification* β, *conclusion* γ and says ‘if α holds and it is consistent to believe β, then believe γ’: $\dfrac{\alpha : \beta}{\gamma}$
- For example:

$$\frac{\mathit{Bird}(x) : \mathit{Flies}(x)}{\mathit{Flies}(x)}$$

- Default rules where justification and conclusion are the same are called *normal default rules* and are writted $\mathit{Bird}(x) \Rightarrow \mathit{Flies}(x)$.

Default theories and extensions

- A default theory *KB* consists of a normal first-order knowledge base *F* and a set of default rules *D*
- A set of reasonable beliefs given a default theory $KB = \{F, D\}$ is called an *extension* of *KB*
- *E* is an *extension* of (F, D) iff for every sentence π,

$$\pi \in E \;\Leftrightarrow\; F \cup \{\gamma \mid \frac{\alpha : \beta}{\gamma} \in D, \alpha \in E, \neg\beta \notin E\} \models \pi$$

How one could construct an extension

$$\pi \in E \;\Leftrightarrow\; F \cup \{\gamma \mid \frac{\alpha : \beta}{\gamma} \in D, \alpha \in E, \neg\beta \notin E\} \models \pi$$

1. $E := F$;
2. close E under classical entailment: $E := \{\pi : E \models \pi\}$
3. choose some (substitution instance of) $\frac{\alpha:\beta}{\gamma} \in D$
4. if $\alpha \in E$, and $\neg\beta \notin E$ (meaning, β is consistent with E), $E := E \cup \{\gamma\}$
5. go back to 2

Example

$$F = \{Bird(tweety)\}, D = \{\frac{Bird(x) : Flies(x)}{Flies(x)}\}$$

- $E := \{Bird(tweety)\}$
- close E under classical entailment: $E := \{\pi : Bird(tweety) \models \pi\}$
- $\frac{Bird(tweety):Flies(tweety)}{Flies(tweety)} \in D$
- $Bird(tweety) \in E$, and $\neg Flies(tweety) \notin E$
  $E := E \cup \{Flies(tweety)\}$
- $E := \{\pi : Bird(tweety), Flies(tweety) \models \pi\}$
- there are no more rules to apply

Example from 2008 exam, Q6e

$$F = \{Dutchman(peter),\ Dutchman(hans),\ Dutchman(johan),$$

$$peter \neq hans,\ hans \neq johan,\ peter \neq johan,$$

$$\neg Tall(peter) \vee \neg Tall(hans)\}$$

$$D = \{\frac{Dutchman(x) : Tall(x)}{Tall(x)}\}$$

Three instances of the default rule:

$$\frac{Dutchman(peter) : Tall(peter)}{Tall(peter)} \quad \frac{Dutchman(hans) : Tall(hans)}{Tall(hans)}$$

$$\frac{Dutchman(johan) : Tall(johan)}{Tall(johan)}$$

Exam 2008 example continued

- Suppose we start constructing E_1 with the first rule, for Peter. Since $\neg Tall(peter) \notin E_1$, we can add $Tall(peter)$ to E_1.
- After we close E_1 under consequence, from $Tall(peter)$ and $\neg Tall(peter) \vee \neg Tall(hans)$ we get $\neg Tall(hans) \in E_1$.
- So now the second rule for Hans is not applicable.
- The third rule is applicable, since $\neg Tall(johan) \notin E_1$, we can add $Tall(johan)$ to E_1
- Another possible extension is E_2: we use the second rule first, and add $Tall(hans)$ to E_2.
- Now the first rule is not applicable, because E_2 contains $\neg Tall(peter)$
- The third rule is applicable, since $\neg Tall(johan) \notin E_2$, we can add $Tall(johan)$ to E_2

Another example with two extensions

- Facts: $F = \{Republican(dick), Quaker(dick)\}$
- Default rules: $Republican(x) \Rightarrow \neg Pacifist(x)$, $Quaker(x) \Rightarrow Pacifist(x)$.
- Extension E_1 (pick the rule $Republican(x) \Rightarrow \neg Pacifist(x)$ first) is all consequences of $\{Republican(dick), Quaker(dick), \neg Pacifist(dick)\}$. Because we start with $E_1 = \{Republican(dick), Quaker(dick)\}$, $\neg\neg Pacifist(dick) \notin E_1$, so we can add $\neg Pacifist(dick)$ to E_1.
- Extension E_2 (pick the rule $Quaker(x) \Rightarrow Pacifist(x)$ first) is all consequences of $\{Republican(dick), Quaker(dick), Pacifist(dick)\}$. Because we start with $E_2 = \{Republican(dick), Quaker(dick)\}$, $\neg Pacifist(dick) \notin E_2$, so we can add $Pacifist(dick)$ to E_2.

Example with one extension

- Facts: $F = \{Republican(dick), Quaker(dick), \forall x(Republican(x) \supset MemberOfPoliticalParty(x))\}$
- Default rules: $Republican(x) \Rightarrow \neg Pacifist(x)$,

$$\frac{Quaker(x) : Pacifist(x) \wedge \neg MemberOfPoliticalParty(x)}{Pacifist(x)}$$

- Closure of F under consequence includes: $\{Republican(dick), Quaker(dick), \forall x(Republican(x) \supset MemberOfPoliticalParty(x)), MemberOfPoliticalParty(dick)\}$
- The second default rule is not applicable, because $\neg\neg MemberOfPoliticalParty(dick) \in E$
- only the first rule is applicable, since $\neg\neg Pacifist(dick) \notin E$, so $\neg Pacifist(dick)$ is added.

Any questions?