Overview of Inference in First-Order Logic

Chapter 9

Outline

- Reducing first-order inference to propositional inference
- Lifting inference in propositional logic to first-order logic.
  - Unification
  - Resolution

Two Approaches for Inference in FOL

Propositionalisation:

- Treat a first-order sentences as a template.
- Instantiating all variables with all possible constants gives a set of ground propositional clauses.
- Apply efficient propositional solver, e.g. SAT.

Two Approaches for Inference in FOL

Propositionalisation:

- Treat a first-order sentences as a template.
- Instantiating all variables with all possible constants gives a set of ground propositional clauses.
- Apply efficient propositional solver, e.g. SAT.

Lifted Inference:

- Generalize propositional methods to 1^{st}-order methods.
- Issue: dealing with variables and quantifiers
- Rule of inference: resolution
- Unification: instantiate variables where necessary.

Propositionalisation

- Easy case: A finite world in which all individuals have names
  - E.g. the wumpus world
  - But also many planning, scheduling, etc. problems

Propositionalisation

- Easy case: A finite world in which all individuals have names
  - E.g. the wumpus world
  - But also many planning, scheduling, etc. problems
- Idea:
  - Replace a universally-quantified sentence with all of its instances
  - Replace an existentially-quantified sentence with a disjunction of its instances

Propositionalisation

- Easy case: A finite world in which all individuals have names
  - E.g. the wumpus world
  - But also many planning, scheduling, etc. problems
- Idea:
  - Replace a universally-quantified sentence with all of its instances
  - Replace an existentially-quantified sentence with a disjunction of its instances
- A formula (KB, etc.) with no variables is called *ground*
- *Inference procedure:*
  - Ground the KB and the query, and
  - run an inference procedure for propositional logic.

Universals

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

  yields

$$\begin{array}{l} King(John) \wedge Greedy(John) \Rightarrow Evil(John) \\ King(Richard) \wedge Greedy(Richard) \Rightarrow Evil(Richard) \\ King(car_{54}) \wedge Greedy(car_{54}) \Rightarrow Evil(car_{54}) \\ \quad \ldots \end{array}$$

Existentials

- E.g., $\exists x \; Likes(John, x)$

  yields

$$\begin{aligned} &Likes(John, John) \vee Likes(John, Richard) \vee \cdots \vee \\ &\qquad Likes(John, car_{54}) \vee \ldots \end{aligned}$$

Existentials

- E.g., $\exists x \; Likes(John, x)$

  yields

  $$Likes(John, John) \vee Likes(John, Richard) \vee \cdots \vee \\ Likes(John, car_{54}) \vee \ldots$$

Q: What does "Everyone likes someone" look like?

Reduction to propositional inference

- Suppose the KB contains just the following:
  $\forall x\ King(x) \wedge Greedy(x) \Rightarrow Evil(x)$
  $King(John), \quad Greedy(John), \quad Brother(Richard, John)$
- Instantiating the universal sentence in *all possible* ways, we get
  $King(John) \wedge Greedy(John) \Rightarrow Evil(John)$
  $King(Richard) \wedge Greedy(Richard) \Rightarrow Evil(Richard)$
  $King(John), \quad Greedy(John), \quad Brother(Richard, John)$
- The new KB is *propositionalized*.
- Proposition symbols are
  $King(John)$,
  $Greedy(John)$,
  $Brother(John, Richard)$,
  $Brother(John, John)$, etc.

Problems with propositionalization

- Usually generates lots of irrelevant sentences.
- E.g., consider:

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

  - For query $Evil(John)$, propositionalization produces lots of facts (like $Greedy(Richard)$) that are irrelevant
- k-ary predicate and n constants $\Rightarrow$ n^k instances

Problems with propositionalization

- Usually generates lots of irrelevant sentences.
- E.g., consider:

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

  - For query $Evil(John)$, propositionalization produces lots of facts (like $Greedy(Richard)$) that are irrelevant
- k-ary predicate and n constants $\Rightarrow$ n^k instances
- However, many recent AI applications use propositionalization for FO KBs over a finite domain.
  - Has led to work in *intelligent grounding*.
- Can make propositionalization work for *arbitrary* FO theories
  ☞ See text for more

General FOL: Dealing with Variables

Consider the KB:

$$\{ \forall x(Grad(x) \Rightarrow Student(x)),$$
$$\forall y(Student(y) \Rightarrow Happy(y)),$$
$$Grad(ZeNian),$$
$$UGrad(Andrei) \}$$

- Intuitively $Happy(ZeNian)$ is inferrable.
    - This requires *instantiating* x and y to $ZeNian$.
- For such a deduction $Andrei$ is irrelevant.

Idea: Try to limit instantiation of variables to *useful* instances.

Unification

- If two formulas can be made the same by substitutions of variables, they are said to be *unified*
- Unification is the process of making 2 formulas (terms, etc) the same by finding an appropriate substitution for variables.

Unification

- If two formulas can be made the same by substitutions of variables, they are said to be *unified*
- Unification is the process of making 2 formulas (terms, etc) the same by finding an appropriate substitution for variables.
- Consider:
  $\forall x(Grad(x) \Rightarrow Student(x)), \qquad Grad(ZeNian)$

Unification

- If two formulas can be made the same by substitutions of variables, they are said to be *unified*
- Unification is the process of making 2 formulas (terms, etc) the same by finding an appropriate substitution for variables.
- Consider:
  $\forall x(Grad(x) \Rightarrow Student(x)), \qquad Grad(ZeNian)$
- To obtain $Student(ZeNian)$ we have the following steps:

Unification

- If two formulas can be made the same by substitutions of variables, they are said to be *unified*
- Unification is the process of making 2 formulas (terms, etc) the same by finding an appropriate substitution for variables.
- Consider:

  $\forall x(Grad(x) \Rightarrow Student(x)), \qquad Grad(ZeNian)$
- To obtain $Student(ZeNian)$ we have the following steps:
  - Figure out how to make $Grad(x)$ and $Grad(ZeNian)$ the same.
    - This is easy: Bind x to $ZeNian$.

Unification

- If two formulas can be made the same by substitutions of variables, they are said to be *unified*
- Unification is the process of making 2 formulas (terms, etc) the same by finding an appropriate substitution for variables.
- Consider:
  $\forall x(Grad(x) \Rightarrow Student(x)), \qquad Grad(ZeNian)$
- To obtain $Student(ZeNian)$ we have the following steps:
  - Figure out how to make $Grad(x)$ and $Grad(ZeNian)$ the same.
    - This is easy: Bind x to $ZeNian$.
  - Substituting, we get the rule instance:
    $Grad(ZeNian) \Rightarrow Student(ZeNian)$.

Unification

- If two formulas can be made the same by substitutions of variables, they are said to be *unified*
- Unification is the process of making 2 formulas (terms, etc) the same by finding an appropriate substitution for variables.
- Consider:
  $\forall x(Grad(x) \Rightarrow Student(x)), \qquad Grad(ZeNian)$
- To obtain $Student(ZeNian)$ we have the following steps:
  - Figure out how to make $Grad(x)$ and $Grad(ZeNian)$ the same.
    - This is easy: Bind x to $ZeNian$.
  - Substituting, we get the rule instance:
    $Grad(ZeNian) \Rightarrow Student(ZeNian)$.
  - Can now derive $Student(ZeNian)$.

Unification Examples

Look for substitution θ such that $\alpha\theta = \beta\theta$

α	β	θ
$Knows(John, x)$	$Knows(John, Jane)$	
$Knows(John, x)$	$Knows(y, OJ)$	
$Knows(John, x)$	$Knows(y, Mother(y))$	
$Knows(John, x)$	$Knows(x, OJ)$	

Unification Examples

Look for substitution θ such that $\alpha\theta = \beta\theta$

α	β	θ
$Knows(John, x)$	$Knows(John, Jane)$	$\{x/Jane\}$
$Knows(John, x)$	$Knows(y, OJ)$	
$Knows(John, x)$	$Knows(y, Mother(y))$	
$Knows(John, x)$	$Knows(x, OJ)$	

Unification Examples

Look for substitution θ such that $\alpha\theta = \beta\theta$

α	β	θ
$Knows(John, x)$	$Knows(John, Jane)$	$\{x/Jane\}$
$Knows(John, x)$	$Knows(y, OJ)$	$\{x/OJ, y/John\}$
$Knows(John, x)$	$Knows(y, Mother(y))$	
$Knows(John, x)$	$Knows(x, OJ)$	

Unification Examples

Look for substitution θ such that $\alpha\theta = \beta\theta$

α	β	θ
$Knows(John, x)$	$Knows(John, Jane)$	$\{x/Jane\}$
$Knows(John, x)$	$Knows(y, OJ)$	$\{x/OJ, y/John\}$
$Knows(John, x)$	$Knows(y, Mother(y))$	$\{y/John,$ $x/Mother(John)\}$
$Knows(John, x)$	$Knows(x, OJ)$	

Unification Examples

Look for substitution θ such that $\alpha\theta = \beta\theta$

α	β	θ
$Knows(John, x)$	$Knows(John, Jane)$	$\{x/Jane\}$
$Knows(John, x)$	$Knows(y, OJ)$	$\{x/OJ, y/John\}$
$Knows(John, x)$	$Knows(y, Mother(y))$	$\{y/John,$ $x/Mother(John)\}$
$Knows(John, x)$	$Knows(x, OJ)$	*fail*

Problem: Can't substitute both *John* and *OJ* for x at the same time.

Solution: Standardize variables apart:

- Replace $Knows(x, OJ)$ with $Knows(y, OJ)$

Reasoning and Unification

- Unification lets us work with both universally quantified variables and arbitrary terms.
- We can use unification in rules such as:
  $$Parent(x, y) \wedge Parent(y, z) \Rightarrow GrandParent(x, z)$$
  where the variables are taken as being universally quantified.
- Then forward chaining and backward chaining with unification can be defined for such rules.

☞ For backward chaining, following one line of development, one ends up with the programming language Prolog.

Resolution: Brief summary

- Resolution can be used in the first-order case (where it forms the basis for much of theorem proving)
- Full first-order version:

$$\frac{\ell_1 \vee C_1, \qquad \ell_2 \vee C_2}{(C_1 \vee C_2)\theta} \qquad \text{where } \ell_1\theta = \neg\ell_2\theta.$$

- For example,

$$\frac{\begin{array}{l}\neg Rich(x) \vee Unhappy(x) \\ Rich(Ken)\end{array}}{Unhappy(Ken)} \qquad \text{with } \theta = \{x/Ken\}$$

- For details see the text or CMPT 411.

Inference in FOL

For KB and query α:

- Convert $KB \wedge \neg\alpha$ to CNF.
  - This is trickier than in propositional logic, since we have to deal with variables and quantifiers.
- Apply resolution steps to $CNF(KB \wedge \neg\alpha)$
  - No longer guaranteed to terminate if satisfiable
  - FOL is *undecidable*

☞ Complete for FOL

Summary

- Propositionalization
  - Grounding approach: reduce all sentences to PL and apply propositional inference techniques.
- FOL/Lifted inference techniques
  - Propositional techniques + Unification.
  - Generalized Modus Ponens
  - Resolution-based inference.
- Many other aspects of FOL inference not discussed in class