CZ3005 Artificial Intelligence

Week 10b – First-Order Logic

Yu Han

han.yu@ntu.edu.sg

Nanyang Assistant Professor
School of Computer Science and Engineering
Nanyang Technological University

Recap

- **First-Order Logic:**
  - Allows descriptions of relations and properties.
  - Allows the descriptions of relations and properties in a compounded manner.
  - Permits the use of constant and variables.
  - Allows general statements to be made.
  - Separation of inference from the representation of knowledge.

Recap

- **Normal Forms**
  - P=>Q can be rewritten as ¬ P v Q
- **Auto documentation with predicate naming**
  - Reading from left to right
  - e.g., grandfather(Philip, William)
- **For all** quantifier (universal for all models) is used with implication connective
  - ∀ **with =>**
- **Existential** quantifier (satisfiable for at least one model) is used with and connective
  - ∃ with ^

Recap

Generation of complex sentences

Composition via

Connectives – expressiveness

i.e. complex relations

Generalization – universal statement

– existential statement

Nesting and Mixing Quantifiers

- **Combining $\forall$ and $\exists$**
  - Express more complex sentences
    - e.g., "if x is the parent of y, then y is the child of x":
    - $\forall$ x, $\forall$ y Parent(x, y) $\Rightarrow$ Child(y, x)
    - "everyone has a parent": $\forall$ x, $\exists$ y Parent(y, x)
  - Semantics depends on quantifiers ordering
    - e.g., $\exists$ y, $\forall$ x Parent(y, x)
    - "there is someone who is everybody's parent" ?
- **Well-formed formula (WFF)**
  - Sentences with all variables properly quantified

Connections between Quantifiers

- **Equivalences**
  - Using the negation *(hence only one quantifier is needed)*

    $$\forall x \; P(x) \Leftrightarrow \neg \exists x \neg P(x)$$

    - e.g. "everyone is mortal":

      $$\forall x \; Mortal(x) \Leftrightarrow \neg \exists x \neg Mortal(x)$$

  - De Morgan's Laws
    - $\forall x\, P \Leftrightarrow \neg\exists x\, \neg P$ $\qquad$ $P \wedge Q \Leftrightarrow \neg(\neg P \vee \neg Q)$

      $\forall x\, \neg P \Leftrightarrow \neg\exists x\, P$ $\qquad$ $\neg P \wedge \neg Q \Leftrightarrow \neg(P \vee Q)$

      $\neg\forall x\, P \Leftrightarrow \exists x\, \neg P$ $\qquad$ $\neg (P \wedge Q) \Leftrightarrow \neg P \vee \neg Q$

      $\neg\forall x\, \neg P \Leftrightarrow \exists x\, P$ $\qquad$ $\neg(\neg P \wedge \neg Q) \Leftrightarrow P \vee Q$

Equality Predicate Symbol

- **Need for equality**
  - State that two terms refer to the same object
    - e.g., Father(John) **=** Henry, or
    - **=**(Father(John), Henry)
  - Useful to define properties
    - e.g. “King John has two brothers”:
    - $\exists$ x, y Brother(x, KingJohn) Λ Brother(y, KingJohn) Λ ¬(x=y)

Grammar of First-Order Logic

(Backus-Naur Form)

Sentence	$\rightarrow$	AtomicSentence | (Sentence)
		| Sentence Connective Sentence
		| ¬Sentence
		| Quantifier Variable, … Sentence
AtomicSentence	$\rightarrow$	Predicate(Term, …) | Term = Term
Term	$\rightarrow$	Function(Term, …) | Constant | Variable
Connective	$\rightarrow$	$\wedge$ | $\vee$ | $\Leftrightarrow$ | $\Rightarrow$
Quantifier	$\rightarrow$	$\forall$ | $\exists$
Constant	$\rightarrow$	A | X_1 | John | …
Variable	$\rightarrow$	a | x | person | …
Predicate	$\rightarrow$	P() | Colour() | Before() | …
Function	$\rightarrow$	F() | MotherOf() | SquareRootOf() | …

Using First-Order Logic

- **Knowledge domain**
  - A part of the world we want to express knowledge about
- **Example of the kinship domain**
  - Objects: people e.g., Elizabeth, Charles, William, etc.
  - Properties: gender i.e., male, female
    Unary predicates: Male() and Female()
  - Relations: kinship e.g., motherhood, brotherhood, etc.
    Binary predicates: Parent(), Sibling(), Brother(), Child(), etc.
    Functions: MotherOf(), FatherOf()
  - > Express facts e.g., Charles is a male
    and rules e.g., the mother of a parent is a grandmother

Sample Functions and Predicates

- **Functions**

  $\forall$ x,y FatherOf(x)=y $\Leftrightarrow$ Parent(y,x) $\wedge$ Male(y)

  $\forall$ x,y MotherOf(x)=y $\Leftrightarrow$ Parent(y,x) $\wedge$ Female(y)

- **Predicates**

  $\forall$ x,y Parent(x,y) $\Leftrightarrow$ Child(y,x)

  $\forall$ x,y Grandparent(x,y) $\Leftrightarrow$ $\exists$ z, Parent(x,z) $\wedge$ Parent(z,y)

  $\forall$ x,y Sibling(x,y) $\Leftrightarrow$ $\neg$x=y $\wedge$ $\exists$ z, Parent(z,x) $\wedge$ Parent(z,y)

  $\forall$ x Male(x) $\Leftrightarrow$ $\neg$Female(x)

- **Potential problems**
  - Self-definition (causes infinite recursion)
    - e.g., $\forall$ x,y Child(x,y) $\Leftrightarrow$ Parent(y,x) <u>following the above</u>

TELLing and ASKing

- **TELLing the KB**
  - Assertion: add a sentence to the knowledge base
    - e.g.
      TELL(KB, $\forall$x,y MotherOf(x)=y $\Leftrightarrow$ Parent(y,x) $\wedge$ Female(y))
      and so on, then
      TELL(KB, Female(Elizabeth) $\wedge$ Parent(Elizabeth,Charles) $\wedge$ Parent(Charles,William))
- **ASKing the KB**
  - Query: retrieve/infer a sentence from the knowledge base
  - <u>Yes/No</u> answer
    - e.g. ASK(KB, Grandparent(Elizabeth, William))
  - Binding list, or <u>substitution</u>
    - e.g. ASK(KB, $\exists$x Child(William, x)) yields {x / Charles}

Inferences Rules for FOL

- **Inference rules from Propositional Logic**
  - *Modus Ponens*
    - $$\frac{\alpha \Rightarrow \beta,\ \alpha}{\beta}$$
  - Double-Negation-Elimination
    - $$\frac{\neg\neg\alpha}{\alpha}$$
  - And-Elimination
    - $$\frac{\alpha_1 \wedge \alpha_2 \wedge \ldots \wedge \alpha_n}{\alpha_i}$$
  - And-Introduction
    - $$\frac{\alpha_1, \alpha_2, \ldots, \alpha_n}{\alpha_1 \wedge \alpha_2 \wedge \ldots \wedge \alpha_n}$$
  - Or-Introduction
    - $$\frac{\alpha_i}{\alpha_1 \vee \alpha_2 \vee \ldots \vee \alpha_n}$$
  - Resolution
    - $$\frac{\alpha \vee \beta,\ \neg\beta \vee \gamma}{\alpha \vee \gamma}$$

Universal Elimination

- ∀ x, Likes(x, flower)
- Substituting x by Shirin gives
- Likes(Shirin, flower)
- The substitution should be done by a constant term.
- In this way, the ∀ quantifier can be eliminated.

Existential Elimination/Introduction

- **Existential Elimination**
  - ∃x, likes(x, flower)
  - Can be changed to:
  - likes(Person, flower)
  - As long as the person is not in the knowledge base.
- **Existential Introduction**
  - Likes(Marry, flower)
  - Can be written as:
  - ∃x, likes(x, flower)

Inferences Rules with Quantifiers

- **Substitutions**
  - SUBST(θ , α):
  - binding list θ applied to a sentence α
    - e.g., SUBST({x / John, y / Richard}, Brother(x, y)) = Brother(John, Richard)

Working Example with Prolog

- SWI-Prolog offers a comprehensive free Prolog environment.
- Since its start in 1987, SWI-Prolog development has been driven by the needs of real world applications.
- SWI-Prolog is widely used in research and education as well as commercial applications.
- Download it here: https://www.swi-prolog.org/
- Let’s see the “royal family” example together

Working Example with Prolog

SWI-Prolog -- d:/Documents/PhD/Courses/CZ3005/Lab Week 10/family.pl

File Edit Settings Run Debug Help

```
            stack_guard(Guard)
        ),
        (   current_prolog_flag(backtrace_depth, Depth)
        ->  Depth>0
        ;   Depth=20
        ),
        get_prolog_backtrace(Depth,
                             Stack0,
                             [frame(Fr), guard(Guard)]),
        debug(backtrace, 'Stack = ~p', [Stack0]),
        clean_stack(Stack0, Stack1),
        join_stacks(Ctx0, Stack1, Stack)
    ).

:- dynamic goal_expansion/2.
:- multifile goal_expansion/2.


:- thread_local thread_message_hook/3.
:- dynamic thread_message_hook/3.
:- volatile thread_message_hook/3.


parent_of(warren, jerry).
parent_of(maryalice, jerry).
parent_of(warren, kather).
parent_of(A, B) :-
    brother(B, C),
    parent_of(A, C).

male(jerry).
male(stuart).
male(warren).
male(peter).
true.

?-
```

Thank you!