Part III – Knowledge and Reasoning

- **6 Agents that Reason Logically**
  - Knowledge-based Agents. – Representations.
  - Propositional Logic. – The Wumpus World.
- **7 First-Order Logic**
  - Syntax and Semantics. – Using First-Order Logic.
  - Logical Agents. – Representing Changes.
  - Deducing Properties of the World.
  - Goal-based Agents.
- **8 Building a Knowledge Base**
  - Knowledge Engineering. – General Ontology.

Recap

- Inherits bulk of inference (tautolgies) from proposition logics.
- Allows descriptions of relations and properties.
- Allows the descriptions of relations and properties in a compounded manner.
- Permits the use of constant, variables and placeholders – skolem constant and functions.
- Allows general statements to be made.
- Separation of inference from the representation of knowledge.
- Sound Logic – deductive logic

Recap 2

- **Normal Forms – CNF and INF**

  P=>Q can be rewritten as ¬ P v Q

- **Auto documentation with predicate naming**
  - Reading from left to right
  - Eg 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 ^

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":
      $\forall x, \forall y \;\; Parent(x, y) \Rightarrow Child(y, x)$
      "every person has a parent": $\forall x \; Person(x) \Rightarrow \exists y \; Parent(y, x)$
  - Semantics depends on quantifiers ordering
    - e.g. $\exists y, \forall x \;\; Parent(y, x)$
      "there is someone who is everybody's parent" ?!?
  - Choosing variables to avoid confusion
    - e.g. $\forall x \; King(x) \vee \exists x \; Brother(Richard, x)$ is better written:
      $\forall x \; King(x) \vee \exists z \; Brother(Richard, z)$
- **Well-formed formula (WFF)**
  - Sentences with all variables properly quantified

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)
      ”every person has a parent”: $\forall$ x Person(x) $\Rightarrow$ $\exists$ y Parent(y, x)

Nesting and Mixing Quantifiers

- **Combining ∀ and ∃**
  - Express more complex sentences”:
  - Semantics depends on quantifiers ordering
    - e.g. ∃ y, ∀ x Parent(y, x)

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

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

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

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

Example:

- Leftleg(John) -> function symbol
- John-leftleg -> constant

- Leftleg(Peter) -> function symbol
- Peter-leftleg -> constant

- Leftleg(Jane) -> function symbol
- Jane-leftleg -> constant

Equality Predicate Symbol

- **Need for equality**
  - State that two terms refer to the same object
    - e.g. Father(John) **=** Henry, or **=**(Father(John), Henry)
  - Predicate symbol with fixed semantics
    - Identity relation, i.e. the set of pairs (2-tuples) of objects where both elements of a pair are the same object.

      e.g. { ⟨Henry, Henry⟩,
      ⟨KingJohn, KingJohn⟩,
      ⟨RichardLionheart, RichardLionheart⟩,
      ... }
  - Useful to define properties
    - e.g. "King John has two brothers":

      $\exists x,y$ Brother(x, KingJohn) $\wedge$ Brother(y, KingJohn) $\wedge$ $\neg(x=y)$

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	$\rightarrow$	$\land$ | $\lor$ | $\Leftrightarrow$ | $\Rightarrow$
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() | …

Extensions to First-Order Logic

- **Higher-order logics**
  - First-order logic: quantifiers over objects
    - e.g. $\forall$ x,y Equal(x, y) $\Leftrightarrow$ $\forall$ x,y (x=y)
  - Second-order logic: quantifiers over relations
    - e.g. "2 objects are equal iff all properties are equivalent":
      $\forall$ x,y Equal(x, y) $\Leftrightarrow$ ($\forall$ p p(x)=p(y))
      or "2 functions are equal iff they have the same value for all args":
      $\forall$ f,g (f=g) $\Leftrightarrow$ ($\forall$x f(x)=g(x))
    - Problem: inference procedures not well understood.
- **λ-expressions**
  - "Macros" to construct complex predicates and functions
    - e.g.Definition: λx,y -> King(x) $\wedge$ Brother(x,y)
      Usage: (λx,y -> King(x) $\wedge$ Brother(x,y)) (Richard, John)

Using First-Order Logic

- **Knowledge domain**
  - A part of the world we want to express knowledge about
- **Example of the kinship domain (<u>The Royals</u>)**
  - 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 <u>facts</u> e.g., Charles is a male
    and <u>rules</u> 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) $\Leftrightarrow$ ¬x=y $\wedge$ $\exists$ z, Parent(z,x) $\wedge$ Parent(z,y)

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

  – How about grandmother relation? Grandmother(x, y)?

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

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
  - Yes/no 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}
    - <u>Example 2</u>

Summary

- **First-order logic …**
  - Is a general-purpose knowledge representation language.
  - Is based on the ontological commitment that the world is composed of objects, with properties and relations.
- **The syntax of first-order logic …**
  - Constant symbols name objects.
  - Predicate symbols name relations.
  - Complex terms name objects using function symbols.
  - Atomic sentences consist of predicates applied to terms.
  - Complex sentences use connectives.
  - Quantified sentences allow to express general rules.

A Logical Agent for the Wumpus World

- **TELLing the KB**
  - Interface: percepts + actions
  - Percept sentences
    - e.g. Percept([Stench, Breeze, Glitter, None, None], t)
  - Action sentences
    - e.g. Turn(Right), Turn(Left), Forward, Shoot, Grab, Climb
- **ASKing the KB**
  - Queries
    - e.g. $\exists a$ Action(a, t+1)
      returning a binding list, such as {a / Grab}

Summary

- **Knowledge-based agents can …**
  - Be designed using first-order logic.
  - Reason using first-order logic.
- **Knowledge-based agents need to …**
  - React to what they perceive.
  - Abstract descriptions of states from percepts.
  - Maintain an internal model of the relevant aspects of the world not directly available from percepts.
  - Express and use information about the desirability of their actions.
  - Use goals in conjunction with knowledge to make plans.

References

– Bell, J. L. and Machover, M. (1977). *A Course in Mathematical Logic*. Elsevier/North-Holland, Amsterdam, London, New York.

– Enderton, H. B. (1972). *A Mathematical Introduction to Logic*. Academic Press, New York.

– Kowalski, R. (1979b). *Logic for Problem Solving*. Elsevier/North-Holland, Amsterdam, London, New York.

– Quine, W. V. (1982). *Methods of Logic*. Harvard University Press, Cambridge, Massachusetts, fourth edition.

– Reiter, R. (1980). A logic for default reasoning. *Artificial Intelligence*, 13(1-2):81-132.

– Clocksin W.F. and Mellish C.S. (2003). Programming in Prolog, 3rd Edition, Springer.

Family Tree

- **Basic ground relationships.**
- **Derived relationships based on the basic.**
  - Avoid nested and circular definitions
- **Inference may generate on facts from the defined relations and exiting ground relationships.**
- **Derived from derived from ground definition.**
- **Organisation:**
  - Facts, derived relations, derived-derived relations.
  - Do not have derived relation not based on ground relations or other derived based on ground.

Part III – Knowledge and Reasoning

- **9 Inference in First-Order Logic**
  - Inference Rules. – Generalised Modus Ponens.
  - Forward and Backward Chaining. – Resolution.
- **10 Logical Reasoning Systems**
  - Indexing, Retrieval and Unification. – Logic Programming / Prolog. – Production Systems.
  - Frames and Semantic / Conceptual Networks.
  - Managing Retractions, Assumptions, and Explanations.

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
    - $$\frac{\alpha \vee \beta,\ \neg\beta \vee \gamma}{\alpha \vee \gamma}$$

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)
- **Inference rules**
  - Universal Elimination
    - $$\frac{\forall\ x\ \ \alpha}{\text{SUBST(\{x/g\}, } \alpha)}$$

$\forall$ x Dog(x) $\Rightarrow$ Friendly(x)

|– Dog(Snoopy) $\Rightarrow$ Friendly(Snoopy)

  - Existential Introduction
    - $$\frac{\alpha}{\exists\ \text{x SUBST(\{g/v\}, } \alpha)}$$
  - Existential Elimination
    - $$\frac{\exists\ \ x\ \ \alpha}{\text{SUBST(\{x/ K\}, } \alpha)}$$

(Skolemization)

$\exists$x Dog (x) Λ Owns(John, x)

|– Dog (Lassie), Owns(John,Lassie)

An Example of Logical Proof

- **Proof procedure**
  - Analysis of the problem description (Natural Language)
  - Translation from NL to first-order logic
  - Application of inference rules (proof)

- **Problem statement**
  - *“It is a crime for an American to sell weapons to hostile nations. The country Nono, an enemy of America, has some missiles, and all of its missiles were sold by Col. West, who is American.”*

end

Temp slides -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
    - $\forall x \, P \Leftrightarrow \neg(\exists x \, (\neg P))$ $\quad P \wedge Q \Leftrightarrow \neg(\neg P \vee \neg Q)$

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

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

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