EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

MATHEMATISCH-
NATURWISSENSCHAFTLICHE FAKULTÄT

Artificial Intelligence
Chapter 8: First-Order Logic

Andreas Zell

After the Textbook: Artificial Intelligence,
A Modern Approach
by Stuart Russel and Peter Norvig (3rd Edition)

8.1 Representation Revisited

- One drawback of propositional logic is that it can’t describe environments with many objects efficiently
- For example, in the Wumpus world, for each square there needs to be a rule for pits, such as
  - $B_{1,1} \Leftrightarrow (P_{1,2} \vee P_{2,1})$
- In natural languages, it is easy to say “Squares adjacent to pits are breezy”, even for a grid of infinite size

8.1 Representation Revisited

- Natural languages like English or German:
  - are far more expressive than propositional logic
  - serve as medium for communication
  - depend heavily on the context
  - can be ambiguous
- We can adapt propositional logic (PL) by borrowing advantages from natural languages, while keeping the declarative, compositional semantics that is context-independent and unambiguous.

8.1 Representation Revisited

- Obvious elements of natural language are:
  - nouns and noun phrases that refer to objects
  - verbs and verb phrases that refer to relations
  - some relations that are functions
- Examples are:
  - Objects: people, houses, colors, time, …
  - Relations:
    - unary relations or properties: red, round, prime
    - n-ary relations: brother of, bigger than, part of, owns, …
  - Functions: father of, best friend, one more than, …

8.1 Representation Revisited

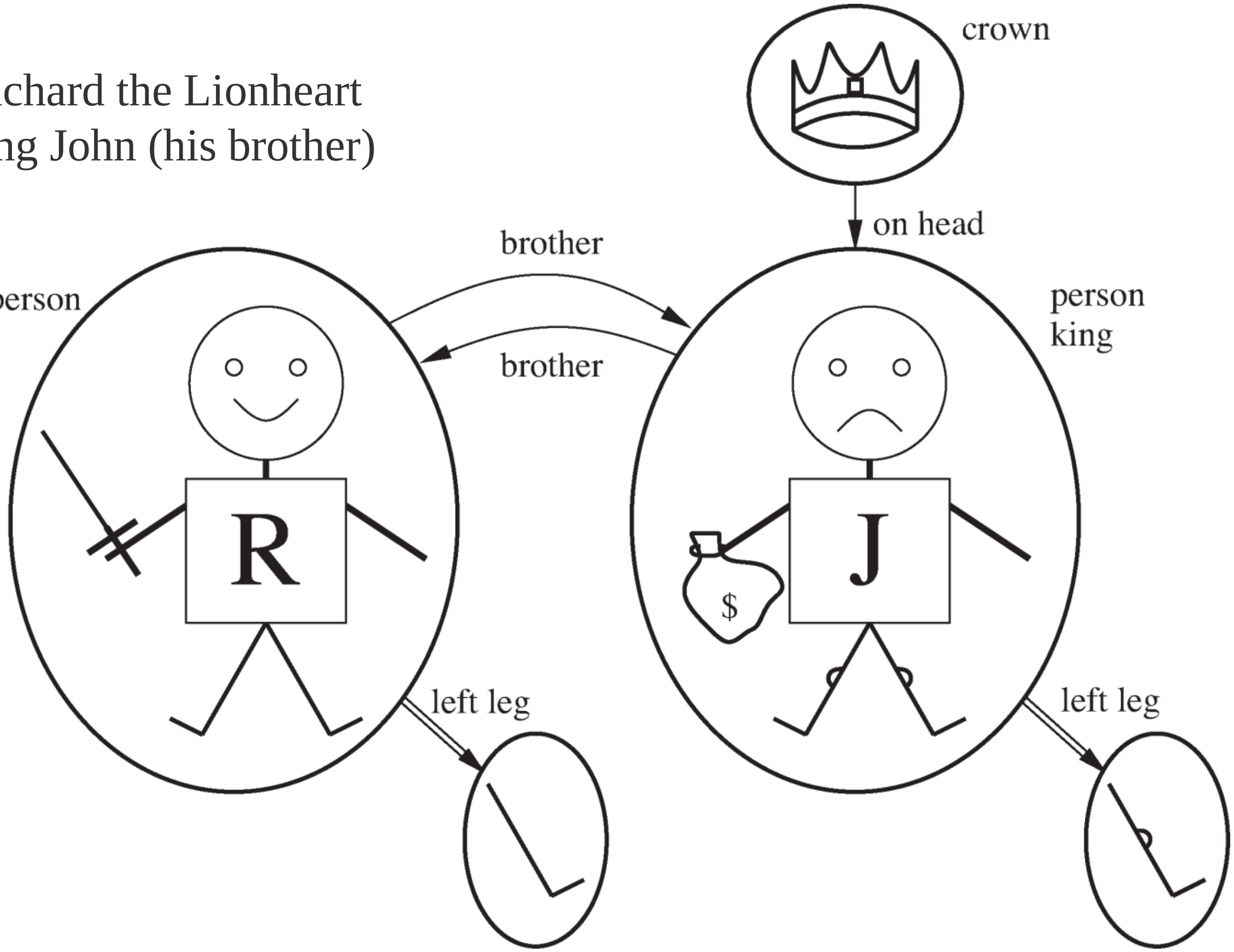
- First-order logic (FOL) is built around objects and relations
- It can also express facts about some or all objects in the universe
- Difference in ontological commitment
  - PL assumes that there are facts that can either be true or false, and each model assigns true or false to each fact via its propositional symbol
  - FOL also assumes that there are objects with certain relations among them that do or do not hold. Its formal models are more complicated.

8.2 Syntax and Semantics of First-Order Logic

- The domain of a model in FOL is the set of objects or domain elements that it contains
- It must be nonempty, every possible world must contain at least one object
- It doesn't matter what these objects are
- In the following examples, we use named objects, i.e., we refer to objects using their names
- We could also use pictures for it

8.2 Syntax and Semantics of First-Order Logic

R: Richard the Lionheart
J: King John (his brother)



8.2 Syntax and Semantics of First-Order Logic

- Formally, a relation is the set of tuples of objects that are related
- Tuples are always ordered
- The “brother” relation in this model is the set
  {<Richard the Lionheart, King John>,
  <King John, Richard the Lionheart>}
- The “on head” relation in this model is the set
  {<the crown, King John>}
- The “person” property (unary relation) is the set
  {<Richard the Lionheart>, <King John>}

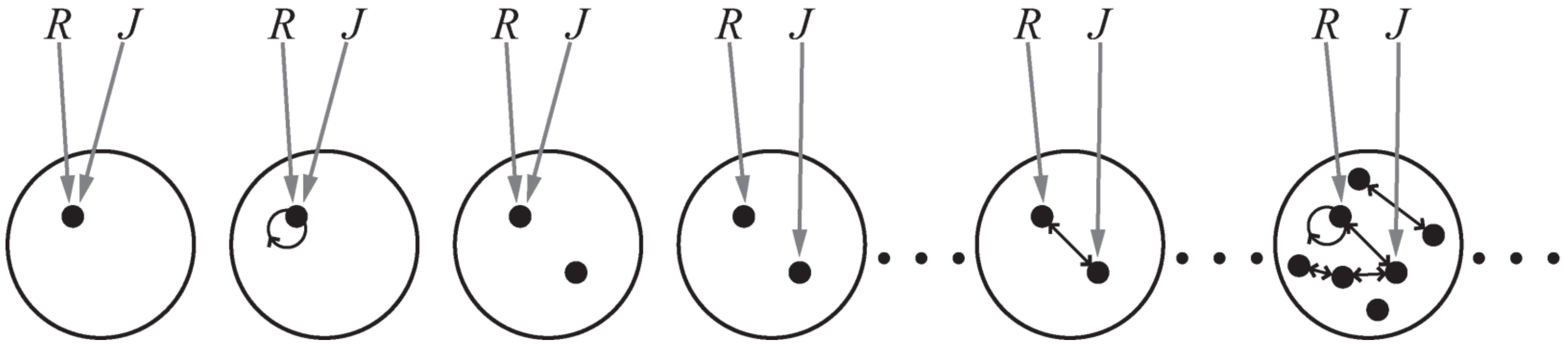
8.2 Syntax and Semantics of First-Order Logic

- Functions relate tuples to exactly one object
- The (unary) “left leg” function includes:
  - <Richard the Lionheart> → Richard’s left leg
  - <King John> → King John’s left leg
- All functions in FOL must be total functions, i.e., there must be a value for every input tuple
- The crown also must have a left leg and so must each of the left legs.
- We introduce an “invisible” object that is the left leg of everything that has no left leg.
  - as long as we don’t use it, there is no problem

- The basic syntactic elements of FOL are the symbols for objects, relations and functions
  - constant symbols for objects
    here: *Richard*, *John*
  - predicate symbols for relations
    here: *Brother*, *OnHead*, *Person*, *King*, *Crown*
  - functions symbols for functions
    here: *LeftLeg*
- Convention: Symbols start with uppercase letters

- Each model must have an interpretation that specifies exactly which objects, relations and functions are referred to by the symbols
- One possible (intended) interpretation would be
  - *Richard* refers to Richard the Lionheart
  - *John* refers to the evil King John
  - …
- Another possible interpretation could be
  - *Richard* refers to the crown
  - *John* refers to King John's left leg

- Note that there are objects without a name, e.g., the intended interpretation does not name the crown or the legs.
- It is also possible that one object has two names



  - Some members of the set of all models with two constant symbols, R and J. The interpretation of each constant symbol is shown by a gray arrow.

- A term is a logical expression that refers to an object, e.g., *LeftLeg(John)*
- “King John’s left leg” does not name the leg, but refers to it
- Note that this is not a subroutine call that returns a value, it is just like a complicated name
- We can reason about left legs, without ever providing a definition of *LeftLeg*

- Formal semantics:
  Consider a term $f(t_1, \ldots, t_n)$
  - The function symbol f refers to some function in the model (call if F)
  - The argument terms refer to objects in the domain (call them $d_1, \ldots, d_n$)
  - The term as a whole refers to the object that is the value of the function F applied to $d_1, \ldots, d_n$.
- If *LeftLeg* refers to the “left leg” function and *John* refers to King John, *LeftLeg(John)* refers to King John’s left leg.

8.2 Syntax and Semantics of First-Order Logic

- An atomic sentence (or atom) is formed from a predicate symbol optionally followed by a list of terms, such as
  - $Brother(Richard, John)$
  - $Married(Father(Richard), Mother(John))$
- We can use logical connectives to construct more complex sentences
  - $\neg Brother(LeftLeg(Richard), John)$
  - $Brother(Richard, John) \wedge Brother(John, Richard)$
  - $King(Richard) \vee King(John)$
  - $\neg King(Richard) \Rightarrow King(John)$

8.2 Syntax and Semantics of First-Order Logic

- Quantifiers let us express properties of entire collections of objects, instead of enumerating the objects by name
- Universal quantification: $\forall x$
- Existential quantification: $\exists x$
- x is called a variable
- variables are terms by itself
- A term with no variables is called ground term

8.2 Syntax and Semantics of First-Order Logic

- For all x, if x is a king, then x is a person

$$\forall x \;\; King(x) \Rightarrow Person(x)$$

- If there are five objects, this universally quantified sentence is equivalent to asserting the following five sentences:
  - Richard is a king $\Rightarrow$ Richard is a person
  - John is a king $\Rightarrow$ John is a person
  - Richard’s left leg is a king $\Rightarrow$ Richard’s left leg is a person
  - John’s left leg is a king $\Rightarrow$ John’s left leg is a person
  - The Crown is a king $\Rightarrow$ the crown is a person

- A common mistake is to use conjunction instead of implication

$$\forall x \;\; King(x) \wedge Person(x)$$

- This sentence would also state that Richard's left leg is a king and Richard's left leg is a person, which is not what we wanted to express.

8.2 Syntax and Semantics of First-Order Logic

- There exists an x such that x is a crown and x is on the head of King John

$$\exists x \;\; Crown(x) \wedge OnHead(x, John)$$

- Similar to universal quantification, this expressed that of all the objects in the model, at least one makes this assertion true.
- A common mistake here is to use the implication instead of the conjunction:

$$\exists x \;\; Crown(x) \Rightarrow OnHead(x, John)$$

- If x is not a crown, the assertion is true, which is not what we wanted to express.

8.2 Syntax and Semantics of First-Order Logic

EBERHARD KARLS UNIVERSITÄT TÜBINGEN

- Quantifiers can be nested

$$\forall x\ \forall y\ \ Brother(x, y) \Rightarrow Sibling(x, y)$$

- But the order is important:

$$\forall x\ \exists y\ \ Loves(x, y)$$

(Everybody loves somebody)

is not the same as

$$\exists y\ \forall x\ \ Loves(x, y)$$

(There is someone who is loved by everyone)

- Because ∀ is a conjunction over the universe of objects and ∃ is a disjunction, they obey De Morgan's rules:

$$
\begin{aligned}
\forall x \ \ \neg P &\equiv \neg \exists x \ \ P \\
\neg \forall x \ \ P &\equiv \exists x \ \ \neg P \\
\forall x \ \ P &\equiv \neg \exists x \ \ \neg P \\
\exists x \ \ P &\equiv \neg \forall x \ \ \neg P
\end{aligned}
$$

8.2 Syntax and Semantics of First-Order Logic

- We can use the equality symbol to signify that two terms refer to the same object

  $$Father(John) = Henry$$

- In the same way, we can express that two symbols refer to different objects

  $$\exists x\ \exists y\ \ Brother(x, Richard) \land Brother(y, Richard) \land \neg(x = y)$$

- The above sentence states that Richard has at least two brothers.
- Sometimes, $x \neq y$ is used for $\neg(x = y)$

8.2 Syntax and Semantics of First-Order Logic

- There are alternative semantics, such as the database semantics, which has:
  - the unique-names assumption:
    We assume that different symbols refer to different objects.
  - the closed-world assumption:
    We assume that all atomic sentences not known to be true are in fact false.
  - domain closure:
    We assume that each model contains no more domain elements than those named by the constant symbols

8.3 Using First-Order Logic

- TELL adds knowledge to the database:

  $TELL(KB, King(John))$

  $TELL(KB, Person(Richard))$

  $TELL(KB, \forall x\ King(x) \Rightarrow Person(x))$

- ASK queries it (exactly like in PL):

  $ASK(KB, King(John))$ returns true

  $ASK(KB, \exists x\ Person(x))$ returns true

- The last query is like asking “Can you tell me the time?” and getting the answer “Yes.”

  $ASK(KB, \exists x\ Person(x))$ returns $\{x/John\}$ and $\{x/Richard\}$

8.3 Using First-Order Logic

- The kinship domain
  - $\forall m,c \;\; Mother(c) = m \;\Leftrightarrow\; Female(m) \wedge Parent(m,c)$
  - $\forall w,h \;\; Husband(h,w) \;\Leftrightarrow\; Male(h) \wedge Spouse(h,w)$
  - $\forall x \;\; Male(x) \;\Leftrightarrow\; \neg Female(x)$
  - $\forall p,c \;\; Parent(p,c) \;\Leftrightarrow\; Child(c,p)$
  - $\forall g,c \;\; Grandparent(g,c) \;\Leftrightarrow\; \exists p \;\; Parent(g,p) \wedge Parent(p,c)$
  - $\forall x,y \;\; Sibling(x,y) \;\Leftrightarrow\; x \neq y \wedge \exists p \;\; Parent(p,x) \wedge Parent(p,y)$
  - …

8.3 Using First-Order Logic

- Each of these sentences is an axiom, as they provide basic factual information
- They are definitions, since they have the form:
  - $\forall x,y \; P(x,y) \Leftrightarrow \ldots$
- Some logical sentences are theorems
  - $\forall x,y \; Sibling(x,y) \Leftrightarrow Sibling(y,x)$
- Theorems provide no new information, as they are entailed by the axioms
- But they reduce the computational cost of deriving new sentences

8.3 Using First-Order Logic

- Not all axioms are definitions
- For example, we do not know enough to fully define what a person is
  - $\forall x \ Person(x) \Leftrightarrow \ \ldots$
- But we can still have axioms that are partial specifications of properties
  - $\forall x \ Person(x) \Rightarrow \ \ldots$
  - $\forall x \ \ldots \ \Rightarrow Person(x)$
- Axioms can also be plain facts:
  - $Male(Jim)$
  - $Spouse(Jim, Laura)$

8.3 Using First-Order Logic

- The Peano axioms define natural numbers
  - $NatNum(0)$
  - $\forall n \;\; NatNum(n) \Rightarrow NatNum(S(n))$
- With this definition, successors ($S(n)$) of natural numbers are also natural numbers
  - Natural numbers are: $0, S(0), S(S(0)), S(S(S(0))), \ldots$
- We need to constrain the successor function
  - $\forall n \;\; 0 \neq S(n)$
  - $\forall m,n \;\; m \neq n \Rightarrow S(m) \neq S(n)$

8.3 Using First-Order Logic

- Now we can define addition
  - $\forall m \;\; NatNum(m) \;\Rightarrow\; +(0,m) = m$
  - $\forall m,n \;\; NatNum(m) \wedge NatNum(n) \;\Rightarrow\; +(S(m),n) = S(+(m,n))$
- For better readability, we can use the infix notation ($m + n$) instead of prefix ($+(m,n)$)
  - $\forall m,n \;\; NatNum(m) \wedge NatNum(n) \;\Rightarrow\; (m+1)+n = (m+n)+1$
- The infix notation is syntactic sugar, that is, an extension to or abbreviation of the standard syntax, that does not change semantics.

8.3 Using First-Order Logic

- Set theory
  - The empty set is a constant written as { }
  - Binary predicates are $x \in s$ and $s_1 \subseteq s_2$
  - Binary functions are $s_1 \cap s_2$, $s_1 \cup s_2$, $\{x|s\}$ (the set resulting from adjoining element x to set s)
- Lists are similar to sets, but are ordered
  - The empty list is a constant written as [] or *Nil*
  - The function *Cons(x,Nil)* is written as [*x*]
  - and *Cons(x,y)*, with *y* being a list, as [*x*|*y*]
  - A list of several elements like [*A*, *B*, *C*] corresponds to *Cons(A, Cons(B, Cons(C, Nil)))*

The wumpus world:

- the agent receives a percept vector with five elements
- we must also include the time at which a percept occurred, for which we use integers
  - *Percept([Stench, Breeze, Glitter, None, None], 5)*
- The actions can be represented as logical terms
  - *Turn(Right), Turn(Left), Forward, Shoot, Grab, Climb*
- To determine which is best, the agents asks
  - *AskVars(∃a BestAction(a,5))*
- This returns a binding list such as *{a/Grab}*

- The raw percept data implies certain facts about the current state, for example:
  - $\forall t,s,g,m,c\;\; Percept([s,Breeze,g,m,c],t) \Rightarrow Breeze(t)$
  - $\forall t,s,b,m,c\;\; Percept([s,b,Glitter,m,c],t) \Rightarrow Glitter(t)$
- Simple reflex behavior can be represented as
  - $\forall t\;\; Glitter(t) \Rightarrow BestAction(Grab,t)$
- Instead of naming squares, we identify them by their row and column
  - $\forall x,y,a,b\;\; Adjacent([x,y],[a,b]) \Leftrightarrow$
    $(x = a \wedge (y = b - 1 \vee y = b + 1)) \vee$
    $(y = b \wedge (x = a - 1 \vee x = a + 1))$

8.3 Using First-Order Logic

- The agent's location changes over time, so
  - $At(Agent,s,t)$ means the agent is at square s at time t
- The wumpus can't move, so we fix its position
  - $\forall t\ At(Wumpus,[2,2],t)$
- Objects can only be at one location at a time
  - $\forall x,s_1,s_2,t\ \ At(x,s_1,t) \wedge At(x,s_2,t) \Rightarrow s_1 = s_2$
- Possible inference rules are
  - $\forall s,t\ \ At(Agent,s,t) \wedge Breeze(t) \Rightarrow Breezy(s)$
  - $\forall s\ \ Breezy(s) \Leftrightarrow \exists r\ \ Adjacent(r,s) \wedge Pit(r)$
  - $\forall t\ \ HaveArrow(t+1) \Leftrightarrow (HaveArrow(t) \wedge \neg Action(Shoot,t))$

8.4 Knowledge Engineering in First-Order Logic

- The process of constructing a knowledge-base is called knowledge engineering
- It can vary widely in content, scope, and difficulty, but it includes the following steps:
  1. Identify the task
  2. Assemble the relevant knowledge
  3. Decide on a vocabulary
  4. Encode general knowledge about the domain
  5. Encode a description of the specific problem instance
  6. Pose queries to the inference procedure
  7. Debug the knowledge base

8.5 Summary

EBERHARD KARLS UNIVERSITÄT TÜBINGEN

- First-order logic is far more powerful than propositional logic
- Knowledge representation should be declarative, compositional, expressive, context independent and unambiguous
- Logics differ in their ontological and epistemological commitments
- The syntax of FOL builds on that of PL
- A possible world, or model, for FOL includes a set of objects and an interpretation that maps symbols to objects, predicates and functions