Knowledge Representation

• This material is covered in chapters 7—9 and 12 of the text.
• Chapter 7 provides a useful motivation for logic, and an introduction to some basic ideas. It also introduces propositional logic, which is a good background for first-order logic.
• What we cover here is mainly covered in Chapters 8 and 9. However, Chapter 8 contains some additional useful examples of how first-order knowledge bases can be constructed. Chapter 9 covers forward and backward chaining mechanisms for inference, while here we concentrate on resolution.
• Chapter 12 covers some of the additional notions that have to be dealt with when using knowledge representation in AI.
Knowledge Representation

• What is knowledge?
• Information we have about the world we inhabit
  • Both the physical and mental world.
  • We have knowledge about many abstract mental constructs and ideas
• Besides knowledge we have various other mental attitudes and feelings about our environment.
  • John knows “…”
  • John fears “…”
  • Then things get complex: John knows that he fears “…”
  • So knowledge can take a variety of forms, some quite complex
Knowledge Representation

- What is Representation?
- Symbols standing for things in the world

CSC 384

Words we use in language

Symbols we use in mathematics
Knowledge Representation

• Can all knowledge be symbolically represented?
• No - we do not symbolically represent the “pixels” that we perceive at the back of our retina.
• So intelligent agents also perform a great deal of low level “non-symbolic reasoning” over their perceptual inputs.
• But higher level “symbolically represented” knowledge also seems to be essential
  • This is the kind of knowledge that we learn in school, by reading, etc.
• In this module we study symbolically represented knowledge
Reasoning

• What is reasoning (in the context of symbolically represented knowledge)?
  • Manipulating our symbols to produce new symbols that represent new knowledge.
    Deriving a new sentence
  • Typically “symbols” are sequences of symbols, e.g., words in language sequenced together to form sentences.
  • So we will develop methods for manipulating “sentences” to produce new “sentences”
Reasoning

- In language we can make up a huge variety of sentences.
- Each of these sentences makes some sort of claim or assertion about our world (mental or physical).
- These claims could be true or false.
  - I am anxious, so the sentence “I feel calm and relaxed.” Is false
- Reasoning aims to be **TRUTH PRESERVING**.
- If we use reasoning to manipulate a collection of **TRUE** sentences, we want the newly derived sentences to also be **TRUE**

- If our reasoning is truth preserving we say that is is **SOUND**
Reasoning

• A more subtle idea is **COMPLETENESS**.

• Completeness says that our reasoning system is powerful enough to produce **ALL** sentences that must be true given one current collection of true sentences.

• Completeness requires a formal characterization of “sentence” in order to answer the question of if we have produced **ALL** true sentences.
Knowledge Representation

• Consider the task of understanding a simple story.

• How do we test understanding?

• Not easy, but understanding at least entails some ability to answer simple questions about the story.
Example.

- Three little pigs
Example.

- Three little pigs
Example.

• Why couldn’t the wolf blow down the house made of bricks?

• What background knowledge are we applying to come to that conclusion?
  • Brick structures are stronger than straw and stick structures.
  • Objects, like the wolf, have physical limitations. The wolf can only blow so hard.
Why Knowledge Representation?

- Large amounts of knowledge are used to understand the world around us, and to communicate with others.
- We also have to be able to reason with that knowledge.
  - Our knowledge won’t be about the blowing ability of wolfs in particular, it is about physical limits of objects in general.
  - We have to employ reasoning to make conclusions about the wolf.
  - More generally, reasoning provides an exponential or more compression in the knowledge we need to store. I.e., without reasoning we would have to store an infeasible amount of information: e.g., Elephants can’t fit into teacups.
Logical Representations

- AI typically employs logical representations of knowledge.

- Logical representations useful for a number of reasons:
Logical Representations

• They are mathematically precise, thus we can analyze their limitations, their properties, the complexity of inference etc.

• They are formal languages, thus computer programs can manipulate sentences in the language.

• They come with both a formal syntax and a formal semantics.

• Typically, have well developed proof theories: formal procedures for reasoning at the syntactic level (achieved by manipulating sentences).

• In this module we will study First-Order logic, and a reasoning mechanism called resolution that operates on First-Order logic.
First Order Logic (FOL)

- Two components: Syntax and Semantics.
  - In a programming language we have a syntax for an if statement: “if <boolean condition>:<expressions>”
  - The if statement also has semantics: if <boolean condition> evaluates to TRUE then we execute <expressions>.
- Syntax gives the grammar or rules for forming proper sentences.
- Semantics gives the meaning.
Basic Semantic entities of FOL

• We have a set of objects \( D \). These are objects in the world that are important for our application.
  • Often we will want to form tuples of objects, e.g., \((d_1, d_2)\) where \( d_1 \in D \) and \( d_2 \in D \) are a pair of objects
  • A k-ary tuple is a subset of \( D^k = D \times D \times \ldots \times D \) the k-wise Cartesian product of \( D \)
• We can identify special sets of objects (subsets of \( D \)) that have some property in common. These sets are called properties.
  • E.g., female, male, children, adult could each need subsets that we identify as being useful in our application. If an object \( d \) is in the set male, we can say that \( d \) has the property male: \( \text{male}(d) \).
Basic Semantic entities of FOL

• Sometimes individual objects are not sufficient, we want to identify special groups (tuples) of objects that are related to each other. We call these sets relations.
  • E.g. \textit{married} might be a special subset of pairs that we wish to keep track of in our application.
• Finally, we might want to keep track of functions over our objects. \( f: D \rightarrow D \)
  • E.g. for \( d \in \text{student} \), we might want a function \( \text{faculty}(d) \) that gives the faculty the student is registered in.
  • More generally we might want \( f: D^k \rightarrow D \), i.e., a function of many arguments mapping \( D \).
Basic Syntactic symbols of FOL

• The syntax starts off with a different symbol for each basic semantic entity (objects, functions, predicates, relations) that we have decided to utilize.
• We get to decide what symbols we use (but of course want to use symbols that are easy to understand)
• These user specified symbols are called the **primitive symbols**

<table>
<thead>
<tr>
<th>Syntax</th>
<th>Semantics</th>
</tr>
</thead>
<tbody>
<tr>
<td>Constant symbols</td>
<td>A particular object (d \in D)</td>
</tr>
<tr>
<td>Function symbols</td>
<td>Some function (f: D^k \rightarrow D)</td>
</tr>
<tr>
<td>Predicate symbols</td>
<td>Some subset of (D)</td>
</tr>
<tr>
<td>Relation symbols</td>
<td>Some subset of (D^k)</td>
</tr>
</tbody>
</table>
### Basic Syntactic symbols of FOL

- In addition we introduce some additional symbols that we will use to connect our basic symbols into sentences.

<table>
<thead>
<tr>
<th>Syntax</th>
<th>Semantics</th>
</tr>
</thead>
<tbody>
<tr>
<td>Constant symbols</td>
<td>A particular object ( d \in D )</td>
</tr>
<tr>
<td>Function symbols</td>
<td>Some function ( f: D^k \rightarrow D )</td>
</tr>
<tr>
<td>Predicate symbols</td>
<td>Some subset of ( D )</td>
</tr>
<tr>
<td>Relation symbols</td>
<td>Some subset of ( D^k )</td>
</tr>
<tr>
<td>Equality (commonly used relation)</td>
<td>Subset of ( D^2 = { (d,d) \mid d \in D } )</td>
</tr>
<tr>
<td>Variables (as many as we need)</td>
<td>An object ( d \in D ) (which particular object can vary)</td>
</tr>
<tr>
<td>Logical connectives: ( \land,\lor,\neg,\rightarrow )</td>
<td>...defined below...</td>
</tr>
<tr>
<td>Quantifiers: ( \forall,\exists )</td>
<td>...defined below...</td>
</tr>
</tbody>
</table>
Example

• Teaching CSC384, want to represent knowledge that would be useful for making the course a successful learning experience. So we might choose syntactic symbols like

• Objects:
  • Students, subjects, assignments, numbers.

• Predicates:
  • difficult(subject), CSMajor(student)

• Relations:
  • handedIn(student, assignment)

• Functions:
  • Grade(student, assignment) → number
First Order Syntax (the grammar)

• We start with our basic syntactic symbols constants, functions, predicates, relations, and variables.
  • Note: the function and relation symbols each have specific arities (the number of arguments it takes)

• From these we can build upon terms and sentences (formulas). Terms are ways of applying functions to build up new “names” for objects. Formulas, are
First Order Syntax - Terms

• Terms are used as names (perhaps complex nested names) for objects in the domain.

<table>
<thead>
<tr>
<th>Terms</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Constants</td>
<td>c, john, mary</td>
</tr>
<tr>
<td>Variables</td>
<td>x, y, z, ...</td>
</tr>
<tr>
<td>Function application</td>
<td>f(t₁, t₂, ..., tₖ)</td>
</tr>
<tr>
<td></td>
<td>tᵢ are already constructed terms</td>
</tr>
</tbody>
</table>

• 5 is a constant term: a symbol representing the number 5.
• john is a term — a symbol representing the person John.
• +(5,5) is a function application term — a new symbol representing the number 10.
First Order Syntax - Terms

• **Note**: constants are the same as functions taking zero arguments.

• Terms are names for objects (things in the world):
  • Constants denote specific objects
  • Functions map tuples of objects to other objects
    • bill, jane, father(jane), father(father(jane))
    • $X$, father($X$), hotel7, rating(hotel7), cost(hotel7)
  • Variables like $X$ are not yet determined, but they will eventually denote particular objects.
First Order Syntax - Sentences.

• Once we have terms we can build up sentences (formulas)
  Terms represent objects, formulas represent true/false assertions about these objects
First Order Syntax - Sentences.

<table>
<thead>
<tr>
<th>Formula</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Atomic formula</td>
<td>( p(t) ) or ( r(t_1, t_2, \ldots, t_k) )</td>
</tr>
<tr>
<td></td>
<td>( p ) is a predicate symbol, ( r ) is a ( k )-ary relation symbol, ( t_i ) are terms</td>
</tr>
<tr>
<td>Negation</td>
<td>( \neg f )</td>
</tr>
<tr>
<td></td>
<td>( F ) is a formula</td>
</tr>
<tr>
<td>Conjunction</td>
<td>( f_1 \land f_2 \land \ldots \land f_k )</td>
</tr>
<tr>
<td></td>
<td>( f_i ) are formulas</td>
</tr>
<tr>
<td>Disjunction</td>
<td>( f_1 \lor f_2 \lor \ldots \lor f_k )</td>
</tr>
<tr>
<td>Implication</td>
<td>( f_1 \rightarrow f_2 )</td>
</tr>
<tr>
<td></td>
<td>( f_1 ) and ( f_2 ) are formulas</td>
</tr>
<tr>
<td></td>
<td>( f_1 ) often called the antecedent, ( f_2 ) the consequence</td>
</tr>
<tr>
<td>Existential</td>
<td>( \exists X. f )</td>
</tr>
<tr>
<td></td>
<td>( f ) is a formula ( X ) is a variable</td>
</tr>
<tr>
<td>Universal</td>
<td>( \forall X. f )</td>
</tr>
</tbody>
</table>

Poole & Allin, CSC384, University of Toronto, Winter 2019
Intuition (formalized later).

- Atoms denote facts that can be true or false about the world
  - father_of(jane,bill), female(jane), system_down()
  - satisfied(client15), satisfied(C)
  - desires(client15, rome, week29), desires(X,Y,Z)
  - rating(hotel7, 4), cost(hotel7, 125)

- Other formulas generate more complex assertions by composing these atomic formulas.
  - Their truth is dependent on the truth of the atomic formulas in them
Semantics

• Formulas (syntax) can be built up recursively, and can become arbitrarily complex

• Intuitively, there are various distinct formulas (viewed as strings) that really are asserting the same thing
  • $\forall X, Y. \, \text{elephant}(X) \land \text{teacup}(Y) \to \text{largerThan}(X, Y)$
  • $\forall X, Y. \, \text{teacup}(Y) \land \text{elephant}(X) \to \text{largerThan}(X, Y)$

• To capture this equivalence and to make sense of complex formulas we utilize the semantics
Semantics

• A formal mapping from formulas to true/false assertions about our semantic entities (individuals, sets and relations over individuals, functions over individuals).

• The mapping mirrors the recursive structure of the syntax, so we can map any formula to a composition of assertions about the semantic entities.
Semantics - The language

• First, we must fix the particular first-order language we are going to provide semantics for. The **primitive** symbols included in the syntax defines the particular language.

$L(F,P,V)$

$F = \text{set of function (and constant symbols)}$

$\text{Each symbol } f \text{ in } F \text{ has a particular arity.}$

$P = \text{set of predicate and relation symbols.}$

$\text{Each relation symbol } r \in P \text{ has a particular arity. (The predicate symbols always have arity 1)}$

$V = \text{an infinite set of variables.}$
Semantics - Primitive Symbols

• An interpretation (model) specifies the mapping from the primitive symbols to semantic entities. It is a tuple \( \langle D, \Phi, \Psi, V \rangle \)
  • \( D \) is a non-empty set of objects (domain of discourse)
  • \( \Phi \) specifies the meaning of each primitive function symbol
    • Also handles the primitive constant symbols (these can be viewed as being zero-arity functions.
  • \( \Psi \) specifies the meaning of each primitive predicate and relation symbol.
  • \( V \) specifies the meaning of the variables.

• Note, the semantic entities that a syntactic symbol maps to is often called the meaning of the symbol or the denotation of the symbol
# Semantics - Primitive Symbols

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Semantics</th>
</tr>
</thead>
<tbody>
<tr>
<td>Constant Symbol $c$</td>
<td>$\Phi(c) \in D$ (some particular object)</td>
</tr>
</tbody>
</table>
| K-ary function symbol $f$     | $\Phi(f)$  
Some particular function $D^k \rightarrow D$                        |
| Predicate symbol $p$          | $\Psi(p)$  
Some particular subset of $D$                                           |
| K-ary relation symbol $r$     | $\Psi(r)$  
Some particular subset of $D^k$                                         |
| Variable $x$                  | $V(x) \in D$ (some particular object)                                    |
Intuitions: Domain

• Domain D: d ∈ D is an *individual*

• E.g. \{craig, jane, grandhotel, le-fleabag, rome, protofino, 100, 110, 120 …\}

• We use underlined symbols to talk about domain individuals (syntactic symbols of the first-order language are not underlined)

• Domains often infinite, but we’ll use finite models to prime our intuitions
Intuitions: $\Phi$

• Given $k$-ary function $f$ and $k$ individuals $d_1 \ldots d_k$, what individual does $f(d_1, \ldots, d_k)$ denote

  • Constants (0-ary functions) are mapped to individuals in $D$.
    • $\Phi(\text{client17}) = \text{craig}$, $\Phi(\text{hotel5}) = \text{le-fleabag}$, $\Phi(\text{rome}) = \text{rome}$
  • 1-ary functions are mapped to particular functions in $D \rightarrow D$
    • $\Phi(\text{rating}) = f_{\text{rating}}$:
      • $f_{\text{rating}}(\text{grandhotel}) = 5\text{stars}$
  • 2-ary functions are mapped to functions from $D^2 \rightarrow D$
    • $\Phi(\text{distance}) = f_{\text{distance}}$:
      • $f_{\text{distance}}(\text{toronto}, \text{sienna}) = 3256$
  • N-ary functions are mapped similarly
Intuitions: $\Psi$

- Given k-ary relation $r$, what does $r$ denote
- 0-ary predicates are mapped to true or false.
  $\Psi(\text{rainy}) = \text{True} \quad \Psi(\text{sunny}) = \text{False}$
- 1-ary predicates are mapped to subsets of $D$.
  - $\Psi(\text{privatebeach}) = p_{\text{privatebeach}}$: (the subset of hotels that have a private beach)
    
    e.g. $p_{\text{privatebeach}} = \{\text{grandhotel, fourseasons}\}$
  - 2-ary predicates are mapped to subsets of $D^2$ (sets of pairs of individuals)
    - $\Psi(\text{location}) = p_{\text{location}}: p_{\text{location}}(\text{grandhotel, rome}) = \text{True} \quad p_{\text{location}}(\text{grandhotel, sienna}) = \text{False}$
  
    $\Psi(\text{available}) = p_{\text{available}}: p_{\text{available}}(\text{grandhotel, week29}) = \text{True}$
- n-ary predicates..subsets of $D^n$
**Intuitions: v**

- V exists to take care of quantification. As we will see the exact mapping it specifies will not matter.

- Notation: V[X/d] is a new variable assignment function.
  - Exactly like V, except that it maps the variable X to the individual d.
  - So for Y \neq X:
    \[ V[X/d](Y) = V(Y) \]
  - For X:
    \[ V[X/d](X) = d \]
Semantics — Terms

• Given language $L(F,P,V)$, and an interpretation $I = \langle D,\Phi,\Psi,V \rangle$ and a term $t$. $I(t)$ is the denotation of $t$ under $I$. 

<table>
<thead>
<tr>
<th>Term</th>
<th>Semantics</th>
</tr>
</thead>
<tbody>
<tr>
<td>Constant Symbol $c$</td>
<td>$I(c) = \Phi(c) \in D$ (some particular object)</td>
</tr>
<tr>
<td>Variable $x$</td>
<td>$I(x) = V(x) \in D$ (some particular object)</td>
</tr>
</tbody>
</table>
| Function application $f(t_1,t_2,\ldots,t_k)$ | $I(f(t_1,t_2,\ldots,t_k)) = \Phi(f)( I(t_1), I(t_2),\ldots,I(t_k)) $  
First we obtain the denotation of each argument under $I$, then we apply the function $\Phi(f)$ to these interpreted terms |

• Hence the terms always denote individuals under interpretation $I$
Semantics — Formulas

• Formulas will always be True or False under any interpretation \( I \).

<table>
<thead>
<tr>
<th>Formula</th>
<th>Semantics</th>
</tr>
</thead>
<tbody>
<tr>
<td>Atomic formula</td>
<td>( l(r(t_1, t_2, \ldots, t_k)) = )</td>
</tr>
<tr>
<td>( r(t_1, t_2, \ldots, t_k) )</td>
<td>True if ((l(t_1), l(t_2), \ldots, l(t_k)) \in \Psi(r))</td>
</tr>
<tr>
<td></td>
<td>False otherwise</td>
</tr>
<tr>
<td></td>
<td>First we obtain the denotation of each argument under ( I ). Then we check if this tuple of interpreted</td>
</tr>
<tr>
<td></td>
<td>terms is in the set of tuples ( \Psi(r) )</td>
</tr>
</tbody>
</table>

• \( \Psi \) Maps \( r \) to a subset of \( D^k \) (a subset of \( k \)-ary tuples of individuals). So the atomic formula is true if its arguments are in the stated relation.
Semantics — Formulas

<table>
<thead>
<tr>
<th>Formula</th>
<th>Semantics</th>
</tr>
</thead>
</table>
| \( \neg f \) | \( I(\neg f) = \)
|           | True if \( I(f) = \text{False} \)
|           | False otherwise |

| \( f_1 \land f_2 \land \ldots \land f_k \) | \( I(f_1 \land f_2 \land \ldots \land f_k) = \)
|           | True if \( I(f_i) = \text{True} \) for every \( i \)
|           | False otherwise |

| \( f_1 \lor f_2 \lor \ldots \lor f_k \) | \( I(f_1 \lor f_2 \lor \ldots \lor f_k) = \)
|           | True if \( I(f_i) = \text{True} \) for any \( i \)
|           | False otherwise |

| \( f_1 \rightarrow f_2 \) | \( I(f_1 \rightarrow f_2) = \)
|           | True if \( I(f_1) = \text{False} \) or \( I(f_2) = \text{True} \)
|           | False otherwise |

- Standard rules for proposition logic that you would have seen before (check chap 7 if not)
Semantics — Formulas

<table>
<thead>
<tr>
<th>Formula</th>
<th>Semantics</th>
</tr>
</thead>
</table>
| $\exists X. f$ | $I(f) = \begin{cases} 
\text{True} & \text{if for some } d \in D, \ I'(f) = \text{True} \\
\text{False} & \text{otherwise} 
\end{cases}$ \\
$I' = \langle D, \Phi, \Psi, V'[X/d] \rangle$ |
| $\forall X. f$ | $I(f) = \begin{cases} 
\text{True} & \text{if for all } d \in D, \ I'(f) = \text{True} \\
\text{False} & \text{otherwise} 
\end{cases}$ \\
$I' = \langle D, \Phi, \Psi, V'[X/d] \rangle$ |

- Quantifiers. Exists checks if $f$ is true under some different variable mapping for the variable $X$. Forall checks if $f$ is true under all possible mappings of the variable $X$. 
Example

\[ D = \{ \text{bob, jack, fred} \} \]
\[ I(\text{happy} = \{ \text{bob, jack, fred} \}) \]
\[ I(\forall X. \text{happy}(X)) \]

1. \( \Psi(\text{happy})(v[X/\text{bob}](X)) = \Psi(\text{happy})(\text{bob}) = \text{True} \)

2. \( \Psi(\text{happy})(v[X/\text{jack}](X)) = \Psi(\text{happy})(\text{jack}) = \text{True} \)

3. \( \Psi(\text{happy})(v[X/\text{fred}](X)) = \Psi(\text{happy})(\text{fred}) = \text{True} \)

Therefore \( I(\forall X. \text{happy}(X)) = \text{True} \).