A Representation and Reasoning System (RRS) is made up of:
- Formal language (syntax):
  - Specifies the legal sentences (the range of things that can be said)
- Semantics:
  - Specifies the meaning of the symbols (for your domain)
  - Specifies what is a correct conclusion
- Reasoning theory or proof procedure:
  - Specifies how an answer can be produced
  - Can be nondeterministic

Implementation of an RRS:
- Reasoning procedure (deductive)
- Reasoning procedure (inductive)
- Reasoning theory (for your domain)
- Reasoning theory (for your domain) (explored)

A Representation and Reasoning System (RRS) is made up of:
- Complete reasoning with the facts and rules to make new conclusions
  - Facts about the world
  - Rules about the world
- Symbolic methods to build a knowledge base that computers can do better
- Symbols have meaning in the knowledge engineer
  - Introduced the symbolic approach
  - Introduced a last domain: robot delivery and waiting

Previous Class:
- Two Views of Semantics
  - Logical consequences
  - Models
- Adding variables to Semantics
  - Semantics of databases
  - Syntax of databases

Representation and Reasoning System
Overview

• Representation and Reasoning System
⇒ Syntax of Datalog

• Semantics of Datalog
⇒ Adding Variables to Semantics
⇒ Semantics of Datalog
⇒ Syntax of Datalog

• Models
• Logical Consequence

• Two Views of Semantics
⇒ Logical Consequence
⇒ Models

Simplifying Assumptions of Initial RRS

• An agent's knowledge can be usefully described in terms of individuals and relations among individuals
• The environment is static
• Only a finite number of individuals of interest in the domain
• An agent's knowledge can be usefully described in terms of individuals and relations among individuals
• Each individual can be given a unique name
• Only a finite number of individuals of interest in the domain

Different RRS's

• Different RRS's are good for different domains
• Or with different semantics for connectives
• With different variables
• Very simple different connectives: ways to build complex expressions
⇒ Choose the simplest RRS possible for your application

The richer the syntax, the more difficult the reasoning procedure

⇒ Different RRS's are good for different domains
Example

Knowledge Base

- \[ \text{male}(\text{william}) \]
- \[ \text{male}(\text{george}) \]
- \[ \text{female}(\text{sally}) \]
- \[ \text{father}(\text{william}, \text{george}) \]
- \[ \text{father}(\text{george}, \text{sally}) \]
- \[ \text{person}(X) \leftarrow \text{female}(X) \]
- \[ \text{person}(X) \leftarrow \text{male}(X) \]
- \[ \text{parent}(X, Y) \leftarrow \text{father}(X, Y) \]
- \[ \text{grandfather}(Z, X) \leftarrow \text{father}(Z, Y) \land \text{parent}(Y, X) \]

What are the constants?
- What are the predicate symbols?
- What are the variables?

Whether knowledge base is correct depends on semantics

More Syntax of Datalog

- Definite Clause
  - either an atomic symbol (a fact) or of the form
  \[ a \leftarrow b_1 \land \ldots \land b_m \]

- Query
  - of the form \(? b_1 \land \ldots \land b_m \)

- Knowledge Base
  - set of definite clauses

- Syntax allows us to write sentences about the world
  - Whether sentences are true or not depends on what the symbols mean.

- Semantics
  - When symbols are used, it is not specified what the symbols mean.

Define Datalog

- Syntax of Datalog
  - Variable
    - starts with upper-case letter
  - Constant
    - starts with lower-case letter or is a sequence of digits (numeric)
  - Predicate symbol
    - starts with lower-case letter
  - Atomic symbol
    - of the form \( p(t_1, \ldots, t_n) \) where \( p \) is a predicate symbol and \( t_i \) are terms
  - Term
    - either a variable or a constant
  - Atomic symbol (atom)

Syntac of Datalog

- Atomic symbol (atom)
- Term
- Syntax with lower case letters
- Predicates symbol
- Sentence with upper case letter
- Constant
- Terms with upper case letters
- Variable
**Interpretation**

An interpretation is a triple $I = (D, \phi, \pi)$ where:

- $D$ is the domain, a non-empty set.
- $\phi$ maps each constant to an element of $D$.
- $\pi$ maps each n-ary predicate symbol to a subset of $D^n$.

**NOTE:** It does not map to a subset of constants.

- Alternatively, $\phi$ can think of $c$ as mapping each tuple $D^n$ to true or false.
- $\pi$ assigns a boolean value to tuples in $D^n$.
- $\phi(c)$ denotes the individual in $D$.
- $\pi(\phi(c), \ldots)$ denotes the elements of $D$.

**Semantics**

Semantics concerns two things:

- A domain, and a mapping from the syntax to the domain.

We call this an interpretation:

- How constants and predicate symbols in the syntax correspond to the domain.
- How individuals and relations in the domain correspond to the syntax.

Semantics concerns two things:

- Two views of semantics.
- Logical consequence.
- Models.
- Adding variables to semantics.
- Semantics of Datalog.
- Syntax of Datalog.
- Representation and Reasoning Systems.
Second Example

• Example:
  - Language with constants $a$ and $b$ and 1-ary predicate $\text{female}(x)$
  - Domain with $D = \{x, y, z\}$
  - How many different $\phi$'s?

- $\phi_1(a)$
- $\phi_2(b)$
- $\phi_3$ (example)
- $\phi_4$ (example)
- $\phi_5$ (example)
- $\phi_6$ (example)
- $\phi_7$ (example)
- $\phi_8$ (example)
- $\phi_9$ (example)

Example Continued

• William and George are male, Sally is female
  - Let's have $\pi$ map male to $\{<\text{William}>, <\text{George}>\}$
  - female to $\{<\text{Sally}>\}$

- Knowledge Engineer decides on mapping of predicates
  - Must decide on the mapping for all predicates
  - Hence, must do mapping for male, even if no facts in KB about male

• This is an example of an intended interpretation:
  - The interpretation the knowledge engineer has in mind when coming up with the language and knowledge base

Example Interpretation: Robot

• $D$ is the set of people
  - $\phi$ maps constants of syntax to objects in the domain
  - Williams = William
  - Knowledge Engineer decides $D$ and mapping of all constants to $D$

Example Interpretation: Robot
Semantics of Connectives

Still need to specify what ‘∧’ and ‘←’ mean

<table>
<thead>
<tr>
<th>p</th>
<th>q</th>
<th>p ∧ q</th>
</tr>
</thead>
<tbody>
<tr>
<td>true</td>
<td>true</td>
<td>true</td>
</tr>
<tr>
<td>true</td>
<td>false</td>
<td>false</td>
</tr>
<tr>
<td>false</td>
<td>true</td>
<td>false</td>
</tr>
<tr>
<td>false</td>
<td>false</td>
<td>false</td>
</tr>
</tbody>
</table>

Nota bene!

- p ← q is true when both p and q are false
- p ∧ q doesn’t always correspond to English meaning

Thus h ← b₁ ∧ ... ∧ bₘ is false in interpretation I if h is false in I and each bᵢ is true in I

⇒ Semantics of ‘∧’ and ‘←’ part of Datalog

Determining Truth of Ground Atoms

• Ground atom has no variables
  - p(t₁, ..., tₙ) maps to true if (φ(t₁), ..., φ(tₙ)) ∈ π(p)
  - otherwise to false

What does male(george) map to?

- φ(george) = George
- π(male) = {<William>, <George>}
- <George> ∈ {<William>, <George>}
- So it maps to true

• For predicates without arguments
  - π(p) is either the set with the empty tuple {<>}
  - or it is empty {}

⇒ Semantics of Ground Atoms comes from interpretation

Example Continued

How many π’s?

x ∈ π(female) y ∈ π(female) z ∈ π(female)

π₁ π₂ π₃ π₄ π₅ π₆ π₇ π₈

⇒ How many different interpretations are there altogether?

$(n \times ... \times n)$ for m = 3

How many ≠ φ?
Semantics & Variables

- How do we interpret clauses such as `person(X) ← female(X)`?
  - Clause is true if it is true for all values of `X`.
  - `person(X)` must be true whenever `female(X)` is true.
  - Remember, knowledge engineer had to specify mapping for all predicates.

- It really has a universal quantifier:
  - `∀X (female(X) ⊆ person(X))`.

- So, variables have an implicit universal quantifier over the clause:
  - `∀X female(X) ← person(X)`.
  - For all `X` true.

Semantics & Variables

- Two Views of Semantics:
  - Logical Consequence
  - Models

- Models:
  - `π(female) ⊆ π(person)`.

- Adding Variables to Semantics:
  - `Semantics of Datalog`

- Syntactic of Datalog

- Representation and Reasoning System

Limitations of Datalog

- More expressive formalisms can handle this (negation knowledge).
- More expressive formalisms can handle this.
- Datalog does not include an operator that means negation.
- Datalog does not include negation.
- Even if every object is male or female, both predicates needed.
- Some variables have an implicit universal quantifier over the clause:
  - `∀X female(X) ← person(X)`.
  - For all `X` true.

Overview

- Representation and Reasoning System
  - Models
  - Logical Consequence
  - More expressive formalisms can handle this (negation knowledge).

- Datalog does not include an operator that means negation.
- Even if every object is male or female, both predicates needed.
- Some variables have an implicit universal quantifier over the clause:
  - `∀X female(X) ← person(X)`.
  - For all `X` true.
Sets of Clauses

- A set of clauses is true in an interpretation if each clause is true in the interpretation.

- Note that we universally quantify over the variables over each clause.

- In other words, if two clauses use the same variables, it is the same as if they used different variables.

\[
\begin{align*}
\text{person}(X) & \leftarrow \text{male}(X) \\
\text{parent}(X,Y) & \leftarrow \text{father}(X,Y) \\
\text{grandfather}(Z,X) & \leftarrow \text{father}(Z,Y) \land \text{parent}(Y,X)
\end{align*}
\]
Two Views of Semantics

• Logical Consequence
• Models

Adding Variables to Semantics

Semantics of Datalog

Syntax of Datalog

Models

⇒ Logical Consequence

Two Views of Semantics

• Logical Consequence
• Models

Example with constants

• Example:
  
  \[ \text{Language with constants } a \text{ and } b \text{ and 1-ary predicate } \text{girl}(x) \]
  
  \[ \text{Domain with } D = \{ x, y, z \} \]
  
  \[ \text{9 } \phi \text{'s and 8 } \pi \text{'s, so 72 interpretations} \]

• How many models of \( KB = \{ \text{girl}(a), \text{girl}(b) \} \)?

  \[ \text{(Checking each would take too long, so lets break down into cases)} \]

  - Case 1:
    \[ \phi_i(a) = \phi_i(b) \]
    
    How many of the 9 \( \phi_i \)'s have \( \phi_i(a) = \phi_i(b) \)?
    
    When \( \phi_i(a) = x \) and \( \phi_i(b) = y \), which \( \pi_i \)'s make the \( KB \) true?
    
    So how many models with \( \phi_i(a) = \phi_i(b) \)?

  - Case 2:
    \[ \phi_i(a) \neq \phi_i(b) \]
    
    How many of the 9 \( \phi_i \)'s have \( \phi_i(a) \neq \phi_i(b) \)?
    
    \[ \text{Example: (focus on all interpretations, not just intended one)} \]

Models

\[ \begin{array}{c|c|c|c}
\text{Model of } KB & \text{FALSE} & \text{TRUE} & \text{FALSE} \\
\hline
(\text{b } \rightarrow \text{d}) & \text{FALSE} & \text{TRUE} & \text{FALSE} \\
\text{b } \rightarrow \text{d} & \text{TRUE} & \text{TRUE} & \text{TRUE} \\
\end{array} \]

- A model of a set of clauses is an interpretation in which all the clauses are true.

Models
Overview

- Logical Consequence
- Models
- Adding Variables to Semantics
- Semantics of Disjunctive
- Syntax of Disjunctive
- Representation and Reasoning System

Example Revisited

- $KB$: $p \leftarrow q$
  - $\pi(p)$
  - $\pi(q)$
  - $\pi(p \leftarrow q)$

Does $KB \models p$?

- $I_1$: $\text{TRUE} \text{ TRUE}$
- $I_2$: $\text{TRUE} \text{ FALSE}$
- $I_3$: $\text{FALSE} \text{ TRUE}$
- $I_4$: $\text{FALSE} \text{ FALSE}$

Logical Consequence

- If $KB$ is a set of clauses and $g$ is a conjunction of atoms, $g$ is a logical consequence of $KB$, written $KB \models g$, if
- $g$ is true in every model of $KB$.
- Other terms that mean the same thing:
  - $g$ logically follows from $KB$
  - $KB$ entails $g$

- That is, $KB \models g$ if there is no interpretation in which $KB$ is true and $g$ is false.
- $KB \not\models g$ if $g$ is not a logical consequence of $KB$. $g$ is a set of clauses and $g'$ is a conjunction of atoms.
Summary of Semantics

- User has intended interpretation
  - But just tells the computer a small set of facts that hopefully adequately captures the user's intended interpretation
  - But this limits what the computer can infer

Computer's view of semantics

- Computer given the knowledge base
  - Computer doesn't have access to the intended interpretation
- User asks it a question
  - Computer should answer true if $\text{KB} \models \phi$ + $\phi$ is true in all models, so is true in user's intended interpretation
  - Otherwise, computer should answer "I don't know"
  - There is at least one model in which $\phi$ is false
  - Note $\phi$ might have been true in user's intended interpretation. In this case, user doesn't have enough knowledge in the KB to adequately constrain the models

User's View of Semantics

- Choose a task domain: intended interpretation
- Associate constants with individuals you want to name
- For each relation you want to represent, associate a predicate symbol in the language
- Tell the system clauses that are true in the intended interpretation of the domain
  - Hopefully you tell it enough knowledge about the domain so that it can conclude everything you want it to