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 what is a correct conclusion.
  - Specifies the meaning of the symbols (for your domain).
  - Specifies the knowledge (signifies):
    - Formal Language (syntax).

Implementation of an RRS:
- Can be nondeterministic.
- Specification of how an answer can be produced.
- Reasoning theory or proof procedure:
  - Specifies the procedure.
  - Specifies the meaning of the symbols (for your domain).
  - Specifies the knowledge (signifies):
    - Formal Language (syntax).

Implementation of an RRS:
- Reasoning procedure:
  - Resolves nondeterminism of reasoning theory.
  - Produces knowledge.
  - Produces the facts, rules, and new conclusions.

Previous Class:
- Introduced a task domains: robot delivery and wiring.
- Introduced the symbolic approach.
- Facts about the world.
- Rules about the world.
- Symbols used to build a knowledge base that the computer is told about.
- Symbols are meaningful to the knowledge engineer.
- Two Views of Semantics:
  - Logical Consequence
  - Models
  - Validates whether a sentence is a model.

Overview:
- Representations and Reasoning System.
Overview

• Representation and Reasoning System
  ⇒ Syntax of Datalog

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

Simplifying Assumptions of Initial RRS

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

Different RRS’s

• Different RRS’s good for different domains
• Different RRS’s good for different domains
• Different RRS’s good for different domains

Choose the simplest RRS possible for your application.

The richer the syntax, the more difficult the reasoning procedure.

Different RRS’s good for different domains

- Of different semantics for composition
- With different syntaxes
- With different completeness
- With different expressiveness

Different RRS’s
Example

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

• 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 (atom)
  - of the form p(t₁, ..., tₙ) where p is a predicate symbol and tᵢ are terms

• Term
  - either a variable or a constant

• Syntax with lowercase letters
  - variables with lowercase letter

• Syntax with uppercase letters
  - constants

• Variable
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 $D^n$.

The domain is a non-empty set of elements of $D$.

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

$\forall c \in D \exists \phi(c)$

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

We call this an interpretation.

**Semantics**

- The domain and relations in the domain.
- How constants and predicate symbols in the syntax correspond to the domain and relations.
- How individuals and relations in the domain correspond to the syntax.
- What individuals in the domain are.
- Relations between them.

Two Views of Semantics:

- Logical Consequence
- Models
- Syntax of Datalog
- Semantics of Datalog
- Representation and Reasoning Systems

**Overview**
Second Example

Example: (focus on all interpretations, not just intended one)

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

\[
\phi_1(a) \quad \phi_2(b) \quad \phi_3 \quad \phi_4 \quad \phi_5 \quad \phi_6 \quad \phi_7 \quad \phi_8 \quad \phi_9
\]

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 mapping for all predicates
  - Hence, must do mapping for \(\text{male}\) even if no facts in KB about male

Example Interpretation

- \(D\) is the set of people \(\text{William}, \text{George}, \text{Sally}\)
- Knowledge Engineer decides \(D\) and mapping of all constants to \(D\)

Example Interpretation continued

- Knowledge Engineer decides on mapping of predicates
  - \(\phi(\text{william}) = \text{William}\)
  - \(\phi(\text{female}) = \text{Sally}\)
  - D is the set of people
  - \(D\) maps constants of syntax \(\phi\) to objects in the domain
  - \(\phi\) maps constants of syntax to objects in the domain
Still need to specify what '∧' and '←' mean

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>P</td>
<td>Q</td>
</tr>
<tr>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
</tr>
<tr>
<td>F</td>
<td>F</td>
</tr>
</tbody>
</table>

Not a 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 Connectives

Semantics of Ground Atoms comes from interpretation

{ } is empty set with the empty tuple ∈ if set is empty

- So it maps to
- φ(William) ∈ George
- φ(William) ∈ George
- George ∈ φ(William)
- Who does head(cook)(?) map to?
- Otherwise false

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>b</td>
<td>d</td>
</tr>
<tr>
<td>¬</td>
<td>d</td>
</tr>
<tr>
<td>v</td>
<td>d</td>
</tr>
<tr>
<td>∨</td>
<td>d</td>
</tr>
<tr>
<td>b</td>
<td>d</td>
</tr>
<tr>
<td>b</td>
<td>d</td>
</tr>
</tbody>
</table>

Determining Truth of Ground Atoms in I

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
</tr>
<tr>
<td>F</td>
<td>F</td>
</tr>
</tbody>
</table>

Example Continued

(different combinations of φ and ψ)

How many different interpretations are there altogether

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>x ∈ female</td>
<td>x ∈ female</td>
</tr>
<tr>
<td>y ∈ female</td>
<td>y ∈ female</td>
</tr>
<tr>
<td>z ∈ female</td>
<td>z ∈ female</td>
</tr>
<tr>
<td>w ∈ female</td>
<td>w ∈ female</td>
</tr>
</tbody>
</table>

How many Φ, ψ?
Overview

• Representation and Reasoning System
• Logical Consequence
• Models
⇒ Adding Variables to Semantics
• Semantics of Datalog
• Syntax of Datalog
• Models
• Logical Consequence
• Two Views of Semantics

Limitations of Datalog

- More expressive formalisms can handle the (negative knowledge)
- Datalog does not include an operator that means negation
- Cannot write a rule that ensures just one of male and female is true for any person
- Even if every object is male or female: both predicates needed
- Datalog does not include an operator that means negation

Is male(george) ∧ female(sally) true in I?
Is male(george) ← female(sally) true in I?
Is male(george) ← female(william) true in I?
Is female(george) ← male(william) true in I?
Is female(george) ∨ female(sally) true in I?

Example
Example

\( \pi(\text{male}) = \{ <\text{William}>, <\text{George}> \} \)

\( \pi(\text{female}) = \{ <\text{Sally}> \} \)

\( \pi(\text{person}) = \{ <\text{William}>, <\text{George}>, <\text{Sally}> \} \)

- Are the following true?
  
  \( \text{person}(X) \leftarrow \text{male}(X) \)
  
  \( \text{person}(X) \leftarrow \text{female}(X) \)

  \( \text{male}(X) \land \text{female}(X) \)

  \( \text{male}(X) \lor \text{female}(X) \)

  \( \text{person}(X) \leftarrow \text{female}(X) \land \text{male}(\text{William}) \)

### Variable Assignment: Formal Definition

- Define a variable assignment \( \rho \):
  
  \( \rho \) maps each variable to some object in the domain.

- Together, \( \rho \) and \( \phi \) assign each term to some object in the domain.

- Together, \( \rho \) and interpretation \( I \) map every clause to true or false.

- Even ungrounded ones.

- Even ungrounded ones.

- Define a variable assignment \( \phi \).

- \( \phi \) assigns each term to some object in the domain.

### Semantics & Variables

- How do we interpret clauses such as:
  
  \( \text{person}(X) \leftarrow \text{female}(X) \)

  \( \text{female}(X) \rightarrow \text{person}(X) \)

- For all \( X \), \( \text{female}(X) \rightarrow \text{person}(X) \)

- \( \text{female}(X) \rightarrow \text{person}(X) \)

- Many other clauses such as:

- \( \text{female}(X) \rightarrow \text{person}(X) \)

- \( \text{male}(X) \rightarrow \text{person}(X) \)

- \( \text{person}(X) \leftarrow \text{male}(X) \)

- \( \text{person}(X) \leftarrow \text{female}(X) \)

- \( \text{male}(X) \land \text{female}(X) \)

- \( \text{male}(X) \lor \text{female}(X) \)

- \( \text{female}(X) \land \text{male}(\text{William}) \)
Models

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

Example KB:

\[
\begin{array}{|c|c|}
\hline
\text{Model of KB?} & \text{Value} \\
\hline
\text{I}_1 & \text{TRUE TRUE} \\
\text{I}_2 & \text{TRUE FALSE} \\
\text{I}_3 & \text{FALSE TRUE} \\
\text{I}_4 & \text{FALSE FALSE} \\
\hline
\end{array}
\]

Sets of Clauses

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

In other words, if two clauses use the same variables, then the variables of each clause are bound to the same value.

If each clause is true in the interpretation, then all clauses are true.

Two Views of Semantics

- Logical Consequence
- Models
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 $KB$.

This tells us that our $KB$, by its definition, always forces $g$ to be true.

Other terms that mean the same thing:

- $g$ logically follows from $KB$.
- $KB$ entails $g$.

If $g$ is not a logical consequence of $KB$, written $KB \not\models g$, if there is no interpretation in which $KB$ is true and $g$ is false.

Example with constants

- Example: (focus on all interpretations, not just intended one)
- Language with constants $a$ and $b$ and 1-ary predicate $\text{girl}()$
- Domain with $D = \{x, y, z\}$
- $\phi$'s and $\pi$'s, so 72 interpretations

- How many models of $KB = \{\text{girl}(a), \text{girl}(b)\}$?
  - (Checking each would take too long, so lets break down into subcases)
  - Case 1: $\phi_i(a) = \phi_i(b)$
    - How many of the 9 $\phi_i$'s have $\phi_i(a) = \phi_i(b)$?
    - When $\phi_i(a) = \phi_i(b) = x$, which $\pi_i$'s make $KB$ 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 $\phi_i(a) = x$ and $\phi_i(b) = y$?
    - Which $\pi_i$'s make $KB$ true?
    - So how many models with $\phi_i(a) \neq \phi_i(b)$?

Two Views of Semantics

- Logical Consequence
- Models
- Adding Variables to Semantics
- Semantics of Datalog
- Syntax of Datalog

Representation and Reasoning System
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:
  - axiomatizing the domain
  - hopefully you tell it enough knowledge about the domain so that it can conclude everything you want it to
- Ask questions about your domain

Overview

- Representation and Reasoning System
- Syntax of Datalog
- Semantics of Datalog
- Models
- Logical Consequence

Example Revised

<table>
<thead>
<tr>
<th>b</th>
<th>d</th>
</tr>
</thead>
<tbody>
<tr>
<td>:</td>
<td>:</td>
</tr>
<tr>
<td>( b \rightarrow d )</td>
<td>( b )</td>
</tr>
<tr>
<td>:</td>
<td>:</td>
</tr>
<tr>
<td>model of KB</td>
<td>KB</td>
</tr>
</tbody>
</table>

\[ d \models B \]
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
  - Computer answers true if all interpretations that make KB true (models) make the question true
  - Now we have specs for the computer's reasoning algorithm
  - Computer answers true if all interpretations that make KB true
  - Adequately captures the user's intended interpretation
  - But just tells the computer a small set of facts that hopefully
    - User has intended interpretation

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 $\text{KB} | g = g$, otherwise answer "I don't know"
  - $g$ is true in all models, so is true in user's intended interpretation
  - Otherwise, computer should answer "I don't know"
  - Computer can answer the question by enumerating over all possible interpretations (model checking)
  - But number of interpretations grows quickly!

• Aside: computer could answer the question by enumerating over all possible interpretations (model checking)
  - But number of interpretations grows quickly!