Announcement

• Final exam April 11
  – Last class April 6

• Practice exercise 8 (Logics: Syntax) available on course website & on WebCT
  – More practice exercises in this 2nd part of the course: new exercise roughly every second class
  – Please use them
Lecture Overview

Recap: CSP planning

• Intro to Logic
• Propositional Definite Clause Logic: Syntax
• Propositional Definite Clause Logic: Semantics
What is the difference between CSP and Planning?

• CSP: static
  – Find a single variable assignment that satisfies all constraints

• Planning: sequential
  – Find a sequence of actions to get from start to goal
    • CSPs don’t even have a concept of actions

  – Some similarities to CSP:
    • Use of variables/values
    • Can solve planning as CSP.
      But the CSP corresponding to a planning instance can be very large
      – Make CSP variables for STRIPS variables at each time step
      – Make CSP variables for STRIPS actions at each time step
CSP Planning: Solving the problem

Map STRIPS Representation into CSP for horizon 0, 1, 2, 3, …

Solve CSP for horizon 0, 1, 2, 3, … until solution found at the lowest possible horizon

K = 0
Is there a solution for this horizon?
If yes, DONE!
If no, continue …
CSP Planning: Solving the problem

Map STRIPS Representation into CSP for horizon 0, 1, 2, 3, …

Solve CSP for horizon 0, 1, 2, 3, … until solution found at the lowest possible horizon

K = 1
Is there a solution for this horizon?
If yes, DONE!
If no, continue …
CSP Planning: Solving the problem

Map STRIPS Representation into CSP for horizon 0, 1, 2, 3, …

Solve CSP for horizon 0, 1, 2, 3, … until solution found at the lowest possible horizon

K = 2: Is there a solution for this horizon?
   If yes, DONE!
   If no….continue
Solving Planning as CSP: pseudo code

solved = false
for horizon h=0,1,2,...
    map STRIPS into a CSP csp with horizon h
    solve that csp
    if solution to the csp exists then
        return solution
    else
        horizon = horizon + 1
end

Solve each of the CSPs based on systematic search
- Not SLS! SLS cannot determine that no solution exists!
Learning Goals for Planning

• STRIPS
  • Represent a planning problem with the STRIPS representation
  • Explain the STRIPS assumption

• Forward planning
  • Solve a planning problem by search (forward planning). Specify states, successor function, goal test and solution.
  • Construct and justify a heuristic function for forward planning

• CSP planning
  • Translate a planning problem represented in STRIPS into a corresponding CSP problem (and vice versa)
  • Solve a planning problem with CSP by expanding the horizon
Lecture Overview

• Recap: CSP planning

Intro to Logic

• Propositional Definite Clause Logic: Syntax
• Propositional Definite Clause Logic: Semantics
Course Overview

Environment

Deterministic

Stochastic

Problem Type

Static

Sequential

Constraint Satisfaction

Variables + Constraints

Arc Consistency

Search

Logics

Search

STRIPS

As CSP (using arc consistency)

Bayesian Networks

Variable Elimination

Decision Networks

Variable Elimination

Markov Processes

Value Iteration

Course Module

Representation

Reasoning Technique

Uncertainty

Decision Theory

Back to static problems, but with richer representation
Logics in AI: Similar slide to the one for planning

Propositional Definite Clause Logics

Semantics and Proof Theory

Description Logics

First-Order Logics

Satisfiability Testing (SAT)

Ontologies

Propositional Logics

Production Systems

Hardware Verification

Cognitive Architectures

Software Verification

Semantic Web

Video Games

Product Configuration

Information Extraction

Summarization

Tutoring Systems

Some Applications
Logics in AI: Similar slide to the one for planning

Propositional Definite Clause Logics

Semantics and Proof Theory

First-Order Logics

Satisfiability Testing (SAT)

Description Logics

Production Systems

Hardware Verification

Ontologies

Cognitive Architectures

Software Verification

Semantic Web

Summarization

Product Configuration

Video Games

Tutoring Systems

Information Extraction

you will know a little

Some Applications
What you already know about logic...

- From programming: Some logical operators
  - If ((amount > 0) && (amount < 1000)) || !(age < 30)
  - ...
  - You know what they mean in a “procedural” way

Logic is the language of Mathematics. To define formal structures (e.g., sets, graphs) and to prove statements about those

We use logic as a Representation and Reasoning System that can be used to formalize a domain and to reason about it
Why Logics?

• “Natural” to express **knowledge** about the world
  • (more natural than a “flat” set of variables & constraints)

• **E.g. “Every 322 student who works hard passes the course”**
  • \(\text{student}(s) \land \text{registered}(s, c) \land \text{course\_name}(c, 322) \land \text{works\_hard}(s) \implies \text{passes}(s,c)\)
  • \(\text{student}(\text{sam})\)
  • \(\text{registered}(\text{sam}, c1)\)
  • \(\text{course\_name}(c1, 322)\)
  • **Query: \(\text{passes}(\text{sam}, c1)\)?**

• **Compact representation**
  - Compared to, e.g., a CSP with a variable for each student
  - It is easy to Incrementally add knowledge
  - It is easy to check and debug knowledge
  - Provides language for asking complex queries
  - Well understood formal properties
Logic: A general framework for reasoning

• Let's think about how to represent a world about which we have only partial (but certain) information

• Our tool: propositional logic

• General problem:
  – tell the computer how the world works
  – tell the computer some facts about the world
  – ask a yes/no question about whether other facts must be true
We have seen several representations and reasoning procedures:

- State space graph + search
- CSP + search/arc consistency
- STRIPS + search/arc consistency

Definition (RRS)
A Representation and Reasoning System (RRS) consists of:

- syntax: specifies the symbols used, and how they can be combined to form legal sentences
- semantics: specifies the meaning of the symbols
- reasoning theory or proof procedure: a (possibly nondeterministic) specification of how an answer can be produced.

We have seen several representations and reasoning procedures:
Using a Representation and Reasoning System

1. Begin with a task domain.
2. Distinguish those things you want to talk about (the ontology)
3. Choose symbols in the computer to denote propositions
4. Tell the system knowledge about the domain
5. Ask the system whether new statements about the domain are true or false
Example: Electrical Circuit
light_l1.
light_l2.
ok_l1.
ok_l2.
ok_cb1.
ok_cb2.
live_outside.

live_l1 ← live_w0.
live_w0 ← live_w1 ∧ up_s2.
live_w0 ← live_w2 ∧ down_s2.
live_w1 ← live_w3 ∧ up_s1.
live_w2 ← live_w3 ∧ down_s1.
live_l2 ← live_w4.
live_w4 ← live_w3 ∧ up_s3.
live_p1 ← live_w3.
live_w3 ← live_w5 ∧ ok_cb1.
live_p2 ← live_w6.
live_w6 ← live_w5 ∧ ok_cb2.
live_w5 ← live_outside.
lit_l1 ← light_l1 ∧ live_l1 ∧ ok_l1.
lit_l2 ← light_l2 ∧ live_l2 ∧ ok_l2.
Propositional Definite Clauses

• A simple representation and reasoning system
• Two kinds of statements:
  – that a proposition is true
  – that a proposition is true if one or more other propositions are true

• Why only propositions?
  – We can exploit the Boolean nature for efficient reasoning
  – Starting point for more complex logics

• To define this RSS, we'll need to specify:
  – syntax
  – semantics
  – proof procedure
Lecture Overview

- Recap: CSP planning
- Intro to Logic
  - Propositional Definite Clause (PDC) Logic: Syntax
  - Propositional Definite Clause (PDC) Logic: Semantics
**Propositional Definite Clauses: Syntax**

<table>
<thead>
<tr>
<th>Definition (atom)</th>
</tr>
</thead>
<tbody>
<tr>
<td>An <strong>atom</strong> is a symbol starting with a lower case letter</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Definition (body)</th>
</tr>
</thead>
<tbody>
<tr>
<td>A <strong>body</strong> is an atom or is of the form $b_1 \land b_2$ where $b_1$ and $b_2$ are bodies.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Definition (definite clause)</th>
</tr>
</thead>
<tbody>
<tr>
<td>A <strong>definite clause</strong> is an atom or is a rule of the form $h \leftarrow b$ where $h$ is an atom (“head”) and $b$ is a body. (Read this as $``h$ if $b.&quot;$)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Definition (KB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>A <strong>knowledge base (KB)</strong> is a set of definite clauses</td>
</tr>
</tbody>
</table>

Examples:
- **atom**: $p_1, \text{live}_1$
- **body**: $p_1 \land p_2, \text{ok}_1 \land \text{live}_0$
- **definite clause**: $p_1 \leftarrow p_2, \text{live}_0 \leftarrow \text{live}_1 \land \text{up}_2$
- **KB**: $\{p_1 \leftarrow p_2, \text{live}_0 \leftarrow \text{live}_1 \land \text{up}_2\}$
light_l1.
light_l2.
ok_l1.
ok_l2.
ok_cb1.
ok_cb2.
live_outside.

live_l1 ← live_w0.
live_w0 ← live_w1 ∧ up_s2.
live_w1 ← live_w3 ∧ up_s1.
live_w2 ← live_w3 ∧ down_s1.
live_l2 ← live_w4.
live_w4 ← live_w3 ∧ up_s3.
live_p1 ← live_w3.
live_w3 ← live_w5 ∧ ok_cb1.
live_p2 ← live_w6.
live_w6 ← live_w5 ∧ ok_cb2.
live_w5 ← live_outside.
lit_l1 ← light_l1 ∧ live_l1 ∧ ok_l1.
lit_l2 ← light_l2 ∧ live_l2 ∧ ok_l2.
Definition (definite clause)
A **definite clause** is an atom or is a rule of the form \( h \leftarrow b \)
where \( h \) is an atom ("head") and \( b \) is a body.
(Read this as `\`h if b.`")

- **Legal PDC clause**
  - a) \( ai\_is\_fun \)
  - b) \( ai\_is\_fun \lor ai\_is\_boring \)
  - c) \( ai\_is\_fun \leftarrow learn\_useful\_techniques \)
  - d) \( ai\_is\_fun \leftarrow learn\_useful\_techniques \land \neg TooMuch\_work \)
  - e) \( ai\_is\_fun \leftarrow learn\_useful\_techniques \land \neg TooMuch\_work \)
  - f) \( ai\_is\_fun \leftarrow f(time\_spent, material\_learned) \)
  - g) \( srtsyj \leftarrow errt \land gffdgdg \)
PDC Syntax: more examples

Legal PDC clause  Not a legal PDC clause

a) ai_is_fun

b) ai_is_fun ∨ ai_is_boring

c) ai_is_fun ← learn_useful_techniques

d) ai_is_fun ← learn_useful_techniques ∧ notTooMuch_work

e) ai_is_fun ← learn_useful_techniques ∧ ¬ TooMuch_work

f) ai_is_fun ← f(time_spent, material_learned)

g) srtsyj ← errt ∧ gffdgddgd

Do any of these statements mean anything? Syntax doesn't answer this question!
Lecture Overview

• Recap: CSP planning
• Intro to Logic
• Propositional Definite Clause (PDC) Logic: Syntax
  Propositional Definite Clause (PDC) Logic: Semantics
Propositional Definite Clauses: Semantics

- Semantics allows you to relate the symbols in the logic to the domain you're trying to model.

**Definition (interpretation)**

An interpretation $I$ assigns a truth value to each atom.

- If our domain has 5 atoms, how many interpretations are there?

\[
5+2, \quad 5\times2, \quad 5^2, \quad 2^5
\]
Propositional Definite Clauses: Semantics

- Semantics allows you to relate the symbols in the logic to the domain you're trying to model.

**Definition (interpretation)**

An *interpretation* $I$ assigns a truth value to each atom.

- If our domain has 5 atoms, how many interpretations are there?
  - 2 values for each atom, so $2^5$ combinations
  - Similar to possible worlds in CSPs
Propositional Definite Clauses: Semantics

Semantics allows you to relate the symbols in the logic to the domain you're trying to model.

**Definition (interpretation)**
An *interpretation* $I$ assigns a truth value to each atom.

We can use the interpretation to determine the truth value of clauses.

**Definition (truth values of statements)**
- A *body* $b_1 \land b_2$ is true in $I$ if and only if $b_1$ is true in $I$ and $b_2$ is true in $I$.
- A *rule* $h \leftarrow b$ is false in $I$ if and only if $b$ is true in $I$ and $h$ is false in $I$. 
PDC Semantics: Example

Truth values under different interpretations
F=false, T=true

<table>
<thead>
<tr>
<th>I_1</th>
<th>a_1</th>
<th>a_2</th>
<th>a_1 \land a_2</th>
</tr>
</thead>
<tbody>
<tr>
<td>I_1</td>
<td>F</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>I_2</td>
<td>F</td>
<td>T</td>
<td>F</td>
</tr>
<tr>
<td>I_3</td>
<td>T</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>I_4</td>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>h</th>
<th>b</th>
<th>h \leftarrow b</th>
</tr>
</thead>
<tbody>
<tr>
<td>I_1</td>
<td>F</td>
<td>F T T T</td>
</tr>
<tr>
<td>I_2</td>
<td>F</td>
<td>F F F T</td>
</tr>
<tr>
<td>I_3</td>
<td>T</td>
<td>T F T F</td>
</tr>
<tr>
<td>I_4</td>
<td>T</td>
<td>T T T T</td>
</tr>
</tbody>
</table>
### PDC Semantics: Example

#### Truth values under different interpretations

<table>
<thead>
<tr>
<th>$F$=false, $T$=true</th>
<th>$h$</th>
<th>$a_1$</th>
<th>$a_2$</th>
<th>$h \leftarrow a_1 \land a_2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$l_1$</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>$l_2$</td>
<td>F</td>
<td>F</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>$l_3$</td>
<td>F</td>
<td>T</td>
<td>F</td>
<td>T</td>
</tr>
<tr>
<td>$l_4$</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>$l_5$</td>
<td>T</td>
<td>F</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>$l_6$</td>
<td>T</td>
<td>F</td>
<td>T</td>
<td>F</td>
</tr>
<tr>
<td>$l_7$</td>
<td>T</td>
<td>T</td>
<td>F</td>
<td>T</td>
</tr>
<tr>
<td>$l_8$</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>F</td>
</tr>
</tbody>
</table>

$h \leftarrow b$ is only false if $b$ is true and $h$ is false.
PDC Semantics: Example for truth values

Truth values under different interpretations

<table>
<thead>
<tr>
<th></th>
<th>h</th>
<th>b</th>
<th>h ← b</th>
</tr>
</thead>
<tbody>
<tr>
<td>I₁</td>
<td>F</td>
<td>F</td>
<td>T</td>
</tr>
<tr>
<td>I₂</td>
<td>F</td>
<td>T</td>
<td>F</td>
</tr>
<tr>
<td>I₃</td>
<td>T</td>
<td>F</td>
<td>T</td>
</tr>
<tr>
<td>I₄</td>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
</tbody>
</table>

h ← a₁ ∧ a₂

Body of the clause: a₁ ∧ a₂

Body is only true if both a₁ and a₂ are true in I

<table>
<thead>
<tr>
<th></th>
<th>h</th>
<th>a₁</th>
<th>a₂</th>
<th>h ← a₁ ∧ a₂</th>
</tr>
</thead>
<tbody>
<tr>
<td>I₁</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>I₂</td>
<td>F</td>
<td>F</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>I₃</td>
<td>F</td>
<td>T</td>
<td>F</td>
<td>T</td>
</tr>
<tr>
<td>I₄</td>
<td>F</td>
<td>T</td>
<td>T</td>
<td>F</td>
</tr>
<tr>
<td>I₅</td>
<td>T</td>
<td>F</td>
<td>F</td>
<td>T</td>
</tr>
<tr>
<td>I₆</td>
<td>T</td>
<td>F</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>I₇</td>
<td>T</td>
<td>T</td>
<td>F</td>
<td>T</td>
</tr>
<tr>
<td>I₈</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
</tbody>
</table>
Propositional Definite Clauses: Semantics

Semantics allows you to relate the symbols in the logic to the domain you're trying to model.

**Definition (interpretation)**

An *interpretation* $I$ assigns a truth value to each atom.

We can use the interpretation to determine the truth value of clauses and knowledge bases:

**Definition (truth values of statements)**

- A **body** $b_1 \land b_2$ is true in $I$ if and only if $b_1$ is true in $I$ and $b_2$ is true in $I$.
- A **rule** $h \leftarrow b$ is false in $I$ if and only if $b$ is true in $I$ and $h$ is false in $I$.
- A **knowledge base** $KB$ is true in $I$ if and only if every clause in $KB$ is true in $I$. 
## Propositional Definite Clauses: Semantics

### Definition (interpretation)
An **interpretation** \( I \) assigns a truth value to each atom.

### Definition (truth values of statements)
- A **body** \( b_1 \land b_2 \) is true in \( I \) if and only if \( b_1 \) is true in \( I \) and \( b_2 \) is true in \( I \).
- A **rule** \( h \leftarrow b \) is false in \( I \) if and only if \( b \) is true in \( I \) and \( h \) is false in \( I \).
- A **knowledge base** \( KB \) is true in \( I \) if and only if every clause in \( KB \) is true in \( I \).

### Definition (model)
A **model** of a knowledge base \( KB \) is an interpretation in which \( KB \) is true.

---

**Similar to CSPs:** a **model** of a set of clauses is an interpretation that makes all of the clauses true.
Definition (model)
A **model** of a knowledge base KB is an interpretation in which every clause in KB is true.

KB = \[
\begin{align*}
p &\leftarrow q \\
q &\quad \\
r &\leftarrow s \\
\end{align*}
\]

Which of the interpretations below are models of KB?

- $I_1$, $I_3$
- $I_1$, $I_3$, $I_4$
- All of them
- $I_3$

<table>
<thead>
<tr>
<th></th>
<th>p</th>
<th>q</th>
<th>r</th>
<th>s</th>
</tr>
</thead>
<tbody>
<tr>
<td>$I_1$</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>$I_2$</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>$I_3$</td>
<td>T</td>
<td>T</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>$I_4$</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>F</td>
</tr>
<tr>
<td>$I_5$</td>
<td>F</td>
<td>T</td>
<td>F</td>
<td>T</td>
</tr>
</tbody>
</table>
PDC Semantics: Example for models

**Definition (model)**
A **model** of a knowledge base KB is an interpretation in which every clause in KB is true.

KB = \[
\begin{cases}
p \leftarrow q \\
q \\
r \leftarrow s
\end{cases}
\]

Which of the interpretations below are models of KB?

<table>
<thead>
<tr>
<th></th>
<th>p</th>
<th>q</th>
<th>r</th>
<th>s</th>
<th>p \leftarrow q</th>
<th>q</th>
<th>r \leftarrow s</th>
<th>KB</th>
</tr>
</thead>
<tbody>
<tr>
<td>I₁</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td></td>
</tr>
<tr>
<td>I₂</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>T</td>
<td>F</td>
<td></td>
</tr>
<tr>
<td>I₃</td>
<td>T</td>
<td>T</td>
<td>F</td>
<td>F</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td></td>
</tr>
<tr>
<td>I₄</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>F</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td></td>
</tr>
<tr>
<td>I₅</td>
<td>F</td>
<td>T</td>
<td>F</td>
<td>F</td>
<td>T</td>
<td>T</td>
<td>F</td>
<td></td>
</tr>
</tbody>
</table>
Definition (model)
A model of a knowledge base KB is an interpretation in which every clause in KB is true.

KB = \[
\begin{cases}
  p \leftarrow q \\
  q \\
  r \leftarrow s
\end{cases}
\]

Which of the interpretations below are models of KB?
All interpretations where KB is true: \( I_1, I_3, \) and \( I_4 \)
Next class

• We’ll start using all these definitions for automated proofs!