Datasets

\[
p(a,b) \\
p(b,c) \\
p(c,d)
\]
Views

\[ g(a,c) \]
\[ g(b,d) \]

View

\[ p(a,b) \]
\[ p(b,c) \]
\[ p(c,d) \]
Operations

View

<table>
<thead>
<tr>
<th>p(a, b)</th>
<th>p(b, c)</th>
<th>p(c, d)</th>
</tr>
</thead>
<tbody>
<tr>
<td>t=1</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Operation

<table>
<thead>
<tr>
<th>p(b, a)</th>
<th>p(c, b)</th>
<th>p(d, c)</th>
</tr>
</thead>
<tbody>
<tr>
<td>t=2</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Operations

\[ g(a,c) \]
\[ g(b,d) \]

View

\[ p(a,b) \]
\[ p(b,c) \]
\[ p(c,d) \]

\[ t=1 \]

Operation

\[ g(c,a) \]
\[ g(d,b) \]

View

\[ p(b,a) \]
\[ p(c,b) \]
\[ p(d,c) \]

\[ t=2 \]
View Definitions

\[ r(X,Y) :\neg p(X,Y) \land \neg q(Y) \]
\[ s(X,Y) : r(X,Y) \land r(Y,Z) \]

Operation Definitions

\[ flip(X) :: p(X) \land \neg q(X) \implies \neg p(X) \land q(X) \]
\[ flop(X) :: r(X,Y) \implies flip(X) \land flop(Y) \]
AI Program Requirements

- Take at least 6 courses
- Take at most 3 courses per quarter

<table>
<thead>
<tr>
<th>Course</th>
<th>Units</th>
<th>Total: 13</th>
</tr>
</thead>
<tbody>
<tr>
<td>CS 124</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CS 224N</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CS 229</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CS 273B</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CS 131</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CS 225A</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CS 231N</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CS 276</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CS 157</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CS 227B</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CS 238</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CS 279</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CS 228</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CS 273A</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CS 331B</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Focus on AI Topics:
- 26.9% Computational Biology
- 26.9% Computer Vision
- 11.5% Information Retrieval
- 11.5% Logic
- 23.1% Machine Learning
- 11.5% Natural Language Processing
- Robotics

Professor:
- 2 courses
- 1 course
- 1 course

Demonstration
Demonstration
Trifecta

Demonstration
Solar System

Demonstration
Syntax
**Operation Constants**

**Operation constants** represent operations.
- **tick** - tick of the clock
- **click** - click a button on a web page
- **stack** - place one block on another
- **mark** - place a specific mark in a row and a column

Same spelling conventions as other constants. Like constructors, and predicates, each has a specific arity.

- **tick/0**
- **click/1**
- **stack/2**
- **mark/3**
An **action** is an application of an operation to objects.

In what follows, we denote actions using a syntax similar to that of compound terms, viz. an $n$-ary operation constant followed by $n$ terms enclosed in parentheses (as appropriate) and separated by commas.

**Examples:**
- `tick`
- `click(a)`
- `stack(a,b)`
- `mark(x,2,3)`

Syntactically, actions are treated as terms.
\[ c(a) :: p(a,b) \land q(a) \implies \neg q(a) \land c(b) \]

head                  conditions                  effects
(action)              (ordinary literals)          (base literals or actions)
Variables

c(X) :: p(X, Y) & q(X) ==> ~q(X) & c(Y)
A operation rule is **safe** if and only if every variable in every literal on the right hand side appears in the head or in a positive literal on the left hand side. Also, every variable in a negative literal on the left hand side appears in a prior positive literal.

**Safe Operation Rule**

\[ c(X) :: \]
\[ p(X,Y) \& \neg q(X) \Rightarrow \]
\[ \neg p(X,Y) \& q(X) \& c(Y) \]

**Unsafe Operation Rule**

\[ c(X) :: \]
\[ p(X,Y) \& \neg q(Z) \Rightarrow \]
\[ \neg p(X,Y) \& q(W) \& c(Y) \]
Degenerate Rule

\[ c(X) :: \text{true} \implies \neg p(X) \land q(X) \]

Shorthand

\[ c(X) :: \neg p(X) \land q(X) \]
An operation definition is a finite collection of operation rules with the same operation in the head.

**Example**

\[
\begin{align*}
c(X) & : : p(X) \land q(X) \\
c(X) & : : \neg r(X) \implies \neg p(X) \land r(X)
\end{align*}
\]

A dynamic logic program is a collection of view definitions and operation definitions.
Semantics
Given a dynamic logic program, the result of applying an action to a dataset is the dataset that results from

(1) deleting all of the negative effects of the action

and then

(2) adding in all of the positive effects.
Given a ruleset $\Omega$ with dataset $\Delta$ and a set $\Gamma$ of actions, an instance of an operation rule in $\Omega$ is active if and only if

(1) the head of the rule is in $\Gamma$

(2) the conditions of the rule are all true in $\Delta$.

Otherwise, the instance is inactive.
Data: $p(a), p(b), p(c), q(a), q(b), q(c), r(b)$

Rule:

$$u(X) :: p(X) \& q(X) \& \neg r(X) \implies \neg p(X) \& r(X)$$

Action: $u(a)$

Active Instance:

$$u(a) :: p(a) \& q(a) \& \neg r(a) \implies \neg p(a) \& r(a)$$

Inactive Instances:

$$u(b) :: p(b) \& q(b) \& \neg r(b) \implies \neg p(b) \& r(b)$$

$$u(c) :: p(c) \& q(c) \& \neg r(c) \implies \neg p(c) \& r(c)$$
The expansion* of an action set with respect to a rule set is the set of all effects in any active instance of any operation definition.

The positive updates of an action with respect to a rule set are the positive literals in the expansion.

The negative updates of an action with respect to a rule set are the negative literals in the expansion.

*Simple version
Data: $p(a), p(b), p(c), q(a), q(b), q(c), r(b)$

Rule:
$$u(X) :: p(X) \land q(X) \land \lnot r(X) \implies \lnot p(X) \land r(X)$$

Action: $u(a)$

Active Instance:
$$u(a) :: p(a) \land q(a) \land \lnot r(a) \implies \lnot p(a) \land r(a)$$

Expansion: $\lnot p(a), r(a)$

Negative Update: $p(a)$

Positive Update: $r(a)$
Given a rule set, the result of applying an action set to dataset $\Delta$ is the set consisting of all factoids in $\Delta$ minus the negative updates plus the positive updates.

$$\Delta - \text{negatives} \cup \text{positives}$$
Data: \( p(a), p(b), p(c), q(a), q(b), q(c), r(b) \)

Rule:
\[ u(X) :: p(X) \land q(X) \land \neg r(X) \implies \neg p(X) \land r(X) \]

Action: \( u(a) \)

Negative Updates: \( p(a) \)
Positive Updates: \( r(a) \)

Result: \( p(b), p(c), q(a), q(b), q(c), r(a), r(b) \)
Dataset: $p(a), p(b), p(c), q(a), q(b), q(c), r(b)$

Rule:
- $u(X) ::= p(X) \land q(X) \land \neg r(X) \implies \neg p(X)$
- $u(X) ::= p(X) \land q(X) \land \neg r(X) \implies r(X)$

Action: $u(a)$

Negative effects: $p(a)$
Positive effects: $r(a)$

Result: $p(b), p(c), q(a), q(b), q(c), r(a), r(b)$
Dataset: \{p(a), p(b), p(c), q(a), q(b), q(c)\}

Rule:
\[
\begin{align*}
    u(X) & : : p(X) \land q(X) \implies \neg r(X) \\
    u(X) & : : p(X) \land q(X) \implies r(X)
\end{align*}
\]

Action: \(u(a)\)

Negative effects: \(r(a)\)
Positive effects: \(r(a)\)

Result: \(p(a), p(b), p(c), q(a), q(b), q(c), r(a)\)
Simultaneous Actions

Data: \( p(a), p(b), p(c), \quad q(a), q(b), q(c), \quad r(b) \)

Rule:

\[
  u(X) :: p(X) \land q(X) \land \neg r(X) \implies \neg p(X) \land r(X)
\]

Actions: \( u(a), u(b), u(c) \)

Active Instances:

\[
  u(a) :: p(a) \land q(a) \land \neg r(a) \implies \neg p(a) \land r(a)
\]

\[
  u(c) :: p(c) \land q(c) \land \neg r(c) \implies \neg p(c) \land r(c)
\]

Inactive Instance:

\[
  u(b) :: p(b) \land q(b) \land \neg r(b) \implies \neg p(b) \land r(b)
\]
Simultaneous Actions

Data: \(p(a), p(b), p(c), \quad q(a), q(b), q(c), \quad r(b)\)

Rule:
\[
u(X) :: p(X) \land q(X) \land \neg r(X) \implies \neg p(X) \land r(X)
\]

Actions: \(u(a), u(b), u(c)\)

Expansion: \(\neg p(a), \neg p(c), r(a), r(c)\)

Negative Updates: \(p(a), p(c)\)

Positive Updates: \(r(a), r(c)\)

Result: \(p(b), \quad q(a), q(b), q(c), \quad r(a), r(b), r(c)\)
Data: \( p(a), p(b), p(c), q(a), q(b), q(c), r(b) \)

Rule:

\[
 u(X) \iff p(X) \land q(X) \implies \neg p(X) \land r(X) \land u(c)
\]

Input Action: \( u(a) \) \hspace{1cm} Derived action: \( u(c) \)

Expansion: \( \neg p(a), \neg p(c), r(a), r(c), u(a), u(c) \)

Negative Updates: \( \{p(a), p(c)\} \)

Positive Updates: \( \{r(a), r(c)\} \)

Result: \( p(b), q(a), q(b), q(c), r(a), r(b), r(c) \)
Given a rule set $\Omega$ and a dataset $\Delta$ a set $\Gamma$ of actions, consider the following series.

$\Gamma_0 = \Gamma$

$\Gamma_{n+1} = \text{the set of all effects of } \Gamma \text{ in any active rule instance}$

The expansion* of $\Gamma$ with respect to $\Omega$ and $\Delta$ is the fixpoint of this series.

The positive updates of an action with respect to a rule set are the positive literals in the full expansion.

The negative updates of an action with respect to a rule set are the negative literals in the full expansion.

*Exact version
function interchange ()
    {x = y;
     y = x}

[x, y]
[3, 4]

interchange()

[x, y]
[4, 4]

function interchange ()
    {var z = x;
     x = y;
     y = z}
interchange ::
  \( \text{val}(x,X) \& \text{val}(y,Y) \implies \)
  \( \neg\text{val}(x,X) \& \neg\text{val}(y,Y) \& \)
  \( \text{val}(x,Y) \& \text{val}(y,X) \)

\( \text{val}(x,3) \)
\( \text{val}(y,4) \)

**Execute:** interchange
\( \text{val}(x,4) \)
\( \text{val}(y,3) \)
A production system is a set of condition-action rules. On each step in the execution of a production system, an active rule is chosen and the actions are performed. The cycle then repeats on the new state.

\[
\begin{align*}
\text{if } p(X), & \text{ then del } p(X) \text{ and add } q(X) \\
\text{if } q(X), & \text{ then del } q(X) \text{ and add } p(X)
\end{align*}
\]

Before: \{p(a),q(b)\}
Step 1: \{q(a),q(b)\}
Step 2: \{p(a),q(b)\} or \{p(b),q(a)\}

When do we stop?
Dynamic logic programs differ from production systems in that all active transition rules “fire” at the same time. (1) All updates are computed before any changes are made, and (2) all changes are made simultaneously.

\[
\begin{align*}
tick :: p(X) & \Rightarrow \neg p(X) \& q(X) \\
tick :: q(X) & \Rightarrow \neg q(X) \& p(X)
\end{align*}
\]

Before: \{p(a), q(b)\}
After: \{p(b), q(a)\}
Blocks World
Blocks World
External Actions

\[ u(a,b) \]

\[ u(d,e) \]
Describing States

\[
\begin{align*}
&\text{clear}(a) \\
&\text{on}(a,b) \\
&\text{on}(b,c) \\
&\text{on}(d,e) \\
&\ldots \\
&\text{clear}(a) \\
&\text{table}(a) \\
&\text{clear}(b) \\
&\text{on}(b,c) \\
&\text{on}(d,e) \\
&\ldots
\end{align*}
\]
Operations:
\( u(x, y) \) means that \( x \) is moved from \( y \) to the table.
\( s(x, y) \) means that \( x \) is moved from the table to \( y \).

Operation Definitions:
\[
\begin{align*}
\text{u}(X,Y) &::= \text{clear}(X) \land \text{on}(X,Y) \\
&\implies \neg \text{on}(X,Y) \land \text{table}(X) \land \text{clear}(Y)
\end{align*}
\]
Operations:

\( u(x, y) \) means that \( x \) is moved from \( y \) to the table.

\( s(x, y) \) means that \( x \) is moved from the table to \( y \).

Operation Definitions:

\[ u(X,Y) :: \]
\[ \text{clear}(X) \land \text{on}(X,Y) \rightarrow \neg \text{on}(X,Y) \land \text{table}(X) \land \text{clear}(Y) \]

\[ s(X,Y) :: \]
\[ \text{table}(X) \land \text{clear}(X) \land \text{clear}(Y) \rightarrow \neg \text{table}(X) \land \neg \text{clear}(Y) \land \text{on}(X,Y) \]
The Game of Life
(1) Any \textit{live} cell with \textit{two or three} live neighbors lives on to the next generation.

(2) Any \textit{live} cell with \textit{fewer than two} live neighbors dies (as if caused by underpopulation).

(3) Any \textit{live} cell with \textit{more than three} live neighbors dies (as if by overpopulation).

(4) Any \textit{dead} cell with \textit{exactly three} live neighbors becomes a live cell (as if by reproduction).
Symbols: \( c_{11}, c_{12}, \ldots \)

Unary Predicates:
- \( \text{on} \) - cell is live
- \( \text{cell} \) - true of cells

Binary Predicates:
- \( \text{neighbor} \) - cells are neighbors
Any live cell with fewer than two live neighbors dies.

tick ::
  on(Y) & countofall(X,neighbor(X,Y)&on(X),0)
  ==> ~on(Y)

tick ::
  on(Y) & countofall(X,neighbor(X,Y)&on(X),1)
  ==> ~on(Y)
Any live cell with more than three live neighbors dies.

\textbf{tick} ::
\begin{align*}
on(Y) & \quad \& \\
countofall(X, \text{neighbor}(X, Y) & \& \text{on}(X), N) \quad \& \\
\text{leq}(4, N) & \quad ==\Rightarrow \quad \neg \text{on}(Y)
\end{align*}
Any *dead* cell with *exactly three* live neighbors becomes live.

\[
\text{tick} :: \\
\text{cell}(Y) \ & \sim \text{on}(Y) \ & \\
\text{countofall}(X, \text{neighbor}(X,Y) \& \text{on}(X), 3) \\
\Longrightarrow \text{on}(Y)
\]
Example

Demonstration
Tic Tac Toe
<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td>O</td>
<td>X</td>
</tr>
</tbody>
</table>

- cell(1,1,x)
- cell(1,2,b)
- cell(1,3,b)
- cell(2,1,b)
- cell(2,2,o)
- cell(2,3,b)
- cell(3,1,b)
- cell(3,2,b)
- cell(3,3,x)
- control(o)
legal(M,N) :- cell(M,N,b)

State:
cell(1,1,x)
cell(1,2,b)
cell(1,3,b)
cell(2,1,b)
cell(2,2,o)
cell(2,3,b)
cell(3,1,b)
cell(3,2,b)
cell(3,3,x)
control(o)

Legal Moves:
mark(1,2)
mark(1,3)
mark(2,1)
mark(2,3)
mark(3,1)
mark(3,2)
\[
\text{mark}(M,N) :: \\
\quad \text{control}(Z) \implies \neg \text{cell}(M,N,b) \land \text{cell}(M,N,Z)
\]

\[
\text{mark}(M,N) :: \\
\quad \text{control}(x) \implies \neg \text{control}(x) \land \text{control}(o)
\]

\[
\text{mark}(M,N) :: \\
\quad \text{control}(o) \implies \neg \text{control}(o) \land \text{control}(x)
\]
row(M,Z) :- cell(M,1,Z) & cell(M,2,Z) & cell(M,3,Z)
col(M,Z) :- cell(1,N,Z) & cell(2,N,Z) & cell(3,N,Z)
diag(Z) :- cell(1,1,Z) & cell(2,2,Z) & cell(3,3,Z)
diag(Z) :- cell(1,3,Z) & cell(2,2,Z) & cell(3,1,Z)

line(Z) :- row(M,Z)
line(Z) :- col(M,Z)
line(Z) :- diag(Z)

win(x) :- line(x)
win(o) :- line(o)

terminal :- win(Z)
terminal :-
    evaluate(countofall([M,N],cell(M,N,b)),0)
Example

Tic Tac Toe

Click in a clear square to mark that square.

Player: x

Demonstration
Assignments
The goal of this exercise is for you to familiarize yourself with the Sierra capabilities for editing and using action definitions. Go to http://epilog.stanford.edu and click on the Sierra link.

In a separate window, open the documentation for Sierra. To access the documentation, go to http://epilog.stanford.edu, click on Documentation, and then click on the Sierra item on the resulting drop-down menu.

Read though Sections 7 and 8 of the documentation and reproduce the examples in the Sierra window you opened earlier. Once you have done this, experiment on your own. Try different data and different actions.
http://logicprogramming.stanford.edu/assignments/nineboard/index.html
Pelican Hunters

http://logicprogramming.stanford.edu/assignments/pelicanhunters/index.html
<table>
<thead>
<tr>
<th>Requirements:</th>
<th>Theory Courses:</th>
<th>Prerequisites:</th>
</tr>
</thead>
<tbody>
<tr>
<td>CS 103 required.</td>
<td>CS 154</td>
<td>CS 109 is a prerequisite for CS 229</td>
</tr>
<tr>
<td>One theoretical course.</td>
<td>CS 157</td>
<td>CS 145 is a prerequisite for CS 345</td>
</tr>
<tr>
<td>CS 109 or CS 157.</td>
<td>CS 161</td>
<td>CS 154 is a prerequisite for CS 254</td>
</tr>
<tr>
<td>Prerequisites satisfied.</td>
<td>CS 254</td>
<td>CS 157 is a prerequisite for CS 227</td>
</tr>
<tr>
<td>At least five courses.</td>
<td></td>
<td>CS 157 is a prerequisite for CS 345</td>
</tr>
</tbody>
</table>

STANFORD UNIVERSITY
Computer Science Department
Program Sheet

- CS 103
- CS 109
- CS 145
- CS 154
- CS 157
- CS 161
- CS 172
- CS 173
- CS 188
- CS 221
- CS 223
- CS 227
- CS 228
- CS 229
- CS 261
- CS 264
- CS 272
- CS 273
- CS 277
- CS 289
- CS 321
- CS 329
- CS 345
- CS 361
- CS 399

http://logicprogramming.stanford.edu/assignments/programsheets/index.html
### Schedule

<table>
<thead>
<tr>
<th>Course</th>
<th>Room</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>cs151</td>
<td></td>
<td></td>
</tr>
<tr>
<td>cs157</td>
<td></td>
<td></td>
</tr>
<tr>
<td>cs161</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Schedule</th>
<th>g100</th>
<th>g200</th>
<th>g300</th>
</tr>
</thead>
<tbody>
<tr>
<td>morning</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>afternoon</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>evening</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Term Project Proposal

Term Project