

Logic Programming

Semantic Worksheets

Michael Genesereth
Computer Science Department
Stanford University

Simple Worksheets

DEPARTMENT OF COMPUTER SCIENCE MSCS Program Sheet (2010-11)

Artificial Intelligence Primary Specialization

Name: **Charles Parnell Naut** Advisor: Proposed date for degree conferral: Date: 10/8/2010
 Student ID #: Email: **cnaut@stanford.edu** HCP? Coterm?

GENERAL INSTRUCTIONS

Before the end of your first quarter, you should complete the following steps. Detailed instructions are included in the **Guide to the MSCS Program Sheet** in your orientation packet (an online version is available at cs.stanford.edu/degrees/mscs/programsheets/):

- Complete this program sheet by filling in the number, name and units of each course you intend to use for your degree.
- Create a course schedule showing the year and quarter in which you intend to take each course in your program sheet.
- Meet with your advisor and secure the necessary signatures on the program sheet.

FOUNDATIONS REQUIREMENT

You must satisfy the requirements listed in each of the following areas; all courses taken elsewhere must be approved by your advisor on a foundation course waiver form. Required documents for waiving a course include course descriptions, syllabi, and textbook lists. These documents can be organized here: cs.stanford.edu/degrees/mscs/waivers/. Do not enter anything in the "Units" column for courses taken elsewhere.

Note: If you are amending an old program sheet, enter "**on file**" in the approval column for courses that have already been approved.

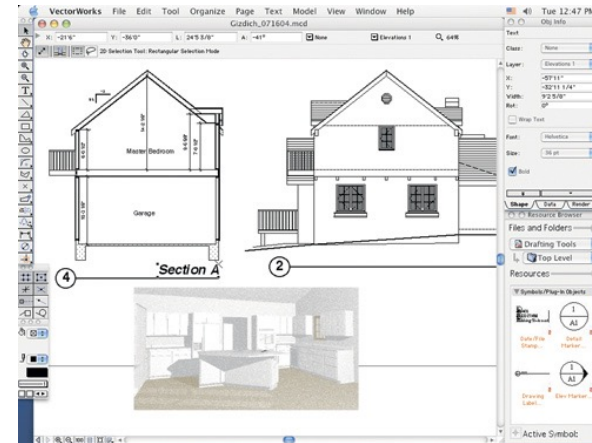
Required:	Equivalent elsewhere (course number/title/institution)	Approval	Grade	Units
Logic, Automata and Complexity (<input checked="" type="checkbox"/> CS 103)				4
Probability (<input type="checkbox"/> CS 109, <input type="checkbox"/> STATS 116, <input type="checkbox"/> CME 106, or <input type="checkbox"/> MS&E 220)				
Algorithmic Analysis (<input checked="" type="checkbox"/> CS 161)				5
Computer Organization and Systems (<input checked="" type="checkbox"/> CS 107)				5
Principles of Computer Systems (<input checked="" type="checkbox"/> CS 110)				5

TOTAL UNITS USED TO SATISFY FOUNDATIONS REQUIREMENT: **10**

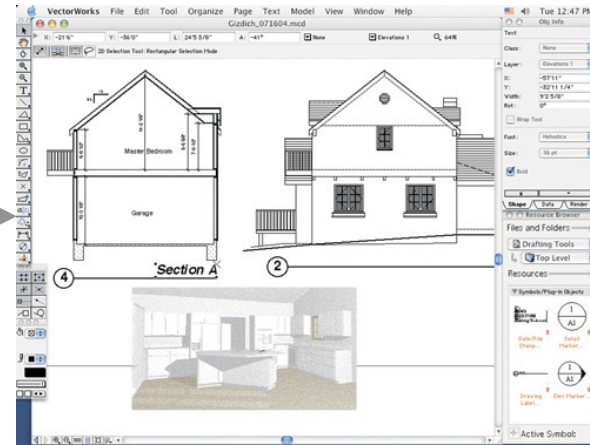
Note: This total may not exceed 10 units.

 7 Requirements Left Total Units: 10 Status: Draft

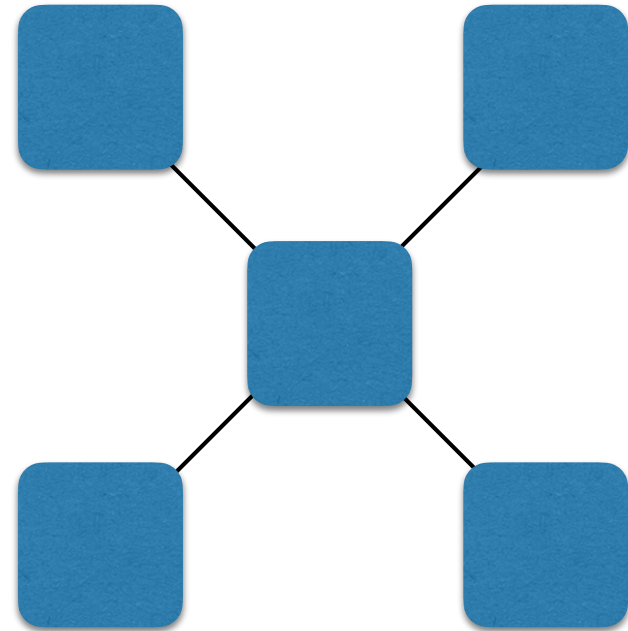
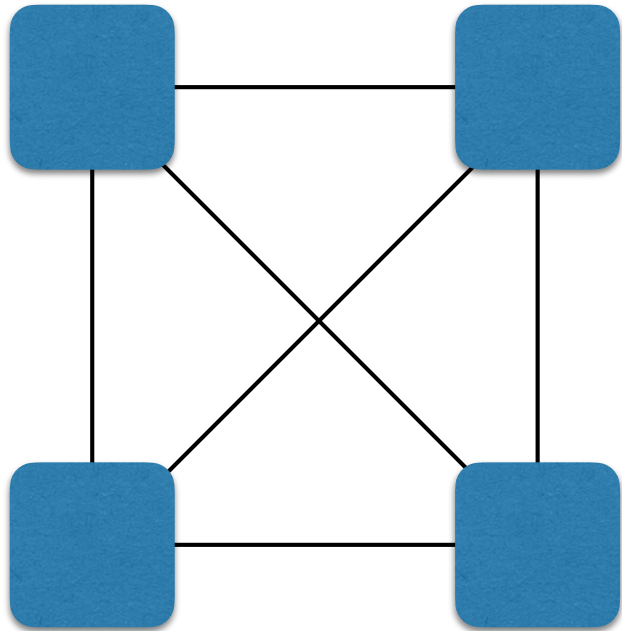
Heterogeneous Worksheets



Collaborative Heterogeneous Worksheets



Architectural Alternatives



Syntactic vs Semantic Worksheets

Syntactic Worksheets

User gestures (e.g. clicking buttons) change *widget state*

Widget state (e.g. values of selectors) stored in lambda

Page state (e.g. colors of text) affects the display

Semantic Worksheets

User gestures translated to *application operations*

Application operations view and change *application state*

Application state (e.g. courses student has taken) stored

Page state defined as views of *application state*

Page state (e.g. colors of text) affects the display

Multiple Perspectives Example

Course Scheduling Perspectives

Course 1

Autumn
Winter
Spring
Summer

Course 2

Autumn
Winter
Spring
Summer

Course 3

Autumn
Winter
Spring
Summer

Course 4

Autumn
Winter
Spring
Summer

Autumn

Course 1
Course 2
Course 3
Course 4

Winter

Course 1
Course 2
Course 3
Course 4

Spring

Course 1
Course 2
Course 3
Course 4

Summer

Course 1
Course 2
Course 3
Course 4

Schedule 1

Course 1	Course 2	Course 3	Course 4
Autumn Winter Spring Summer	Autumn Winter Spring Summer	Autumn Winter Spring Summer	Autumn Winter Spring Summer

```
click(when(C,Q)) :: style(when(C,Q), "background-color", white)
==> ~style(when(C,Q), "background-color", white) &
style(when(C,Q), "background-color", grey)
```

```
click(when(C,Q)) :: style(when(C,Q), "background-color", grey)
==> ~style(when(C,Q), "background-color", grey) &
style(when(C,Q), "background-color", white)
```

Schedule 2

Autumn	Winter	Spring	Summer
Course 1	Course 1	Course 1	Course 1
Course 2	Course 2	Course 2	Course 2
Course 3	Course 3	Course 3	Course 3
Course 4	Course 4	Course 4	Course 4

```
click(what(Q,C)) :: style(what(Q,C), "background-color", grey)
==> ~style(what(Q,C), "background-color", grey) &
     style(what(Q,C), "background-color", white)
```

```
click(what(Q,C)) :: style(what(Q,C), "background-color", white)
==> ~style(what(Q,C), "background-color", white) &
     style(what(Q,C), "background-color", grey)
```

Syntactic Mapping Rules

```
click(when(C,Q)) :: style(when(C,Q), "background-color", white)
==> ~style(when(C,Q), "background-color", white) &
     style(when(C,Q), "background-color", grey)
```

```
click(when(C,Q)) :: style(when(C,Q), "background-color", grey)
==> ~style(when(C,Q), "background-color", grey) &
     style(when(C,Q), "background-color", white)
```

```
click(when(C,Q)) :: style(what(Q,C), "background-color", grey)
==> ~style(what(Q,C), "background-color", grey) &
     style(what(Q,C), "background-color", white)
```

```
click(when(C,Q)) :: style(what(Q,C), "background-color", white)
==> ~style(what(Q,C), "background-color", white) &
     style(what(Q,C), "background-color", grey)
```

+ 4 analogous rules for what(Q,C)

Semantic Version

Data:

```
offered(course1, autumn)
offered(course2, autumn)
```

Operations (similar to previous operation definitions):

```
click(when(C,Q)) :: offered(C,Q) ==> ~offered(C,Q)
click(when(C,Q)) :: ~offered(C,Q) ==> offered(C,Q)
```

```
click(what(Q,C)) :: offered(C,Q) ==> ~offered(C,Q)
style(when(Q,C)) :: ~offered(C,Q) ==> offered(C,Q)
```

Views (in place of mapping rules):

```
style(when(C,Q), "background-color", grey) :- offered(C,Q)
style(when(C,Q), "background-color", white) :- ~offered(C,Q)
```

```
style(what(Q,C), "background-color", grey) :- offered(C,Q)
style(what(Q,C), "background-color", white) :- ~offered(C,Q)
```

Schedule

Course	Room	Time
cs151	<input type="text"/>	<input type="text"/>
cs157	<input type="text"/>	<input type="text"/>
cs161	<input type="text"/>	<input type="text"/>

Schedule	g100	g200	g300
morning	<input type="text"/>	<input type="text"/>	<input type="text"/>
afternoon	<input type="text"/>	<input type="text"/>	<input type="text"/>
evening	<input type="text"/>	<input type="text"/>	<input type="text"/>

Schedule Problem

Schedule

Course	Room	Time
cs151	<input type="text"/>	<input type="text"/>
cs157	<input type="text"/>	<input type="text"/>
cs161	<input type="text"/>	<input type="text"/>

Schedule	g100	g200	g300
morning	<input type="text"/>	<input type="text"/>	<input type="text"/>
afternoon	<input type="text"/>	<input type="text"/>	<input type="text"/>
evening	<input type="text"/>	<input type="text"/>	<input type="text"/>

Schedule Problem

Collaborative Worksheets



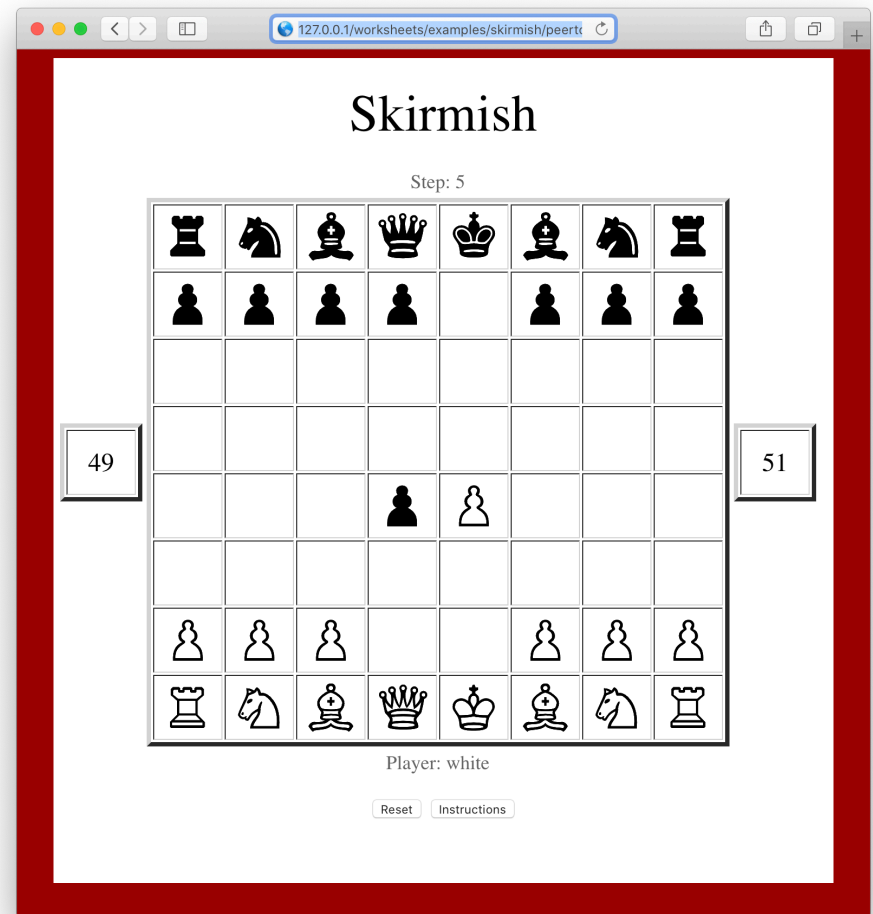
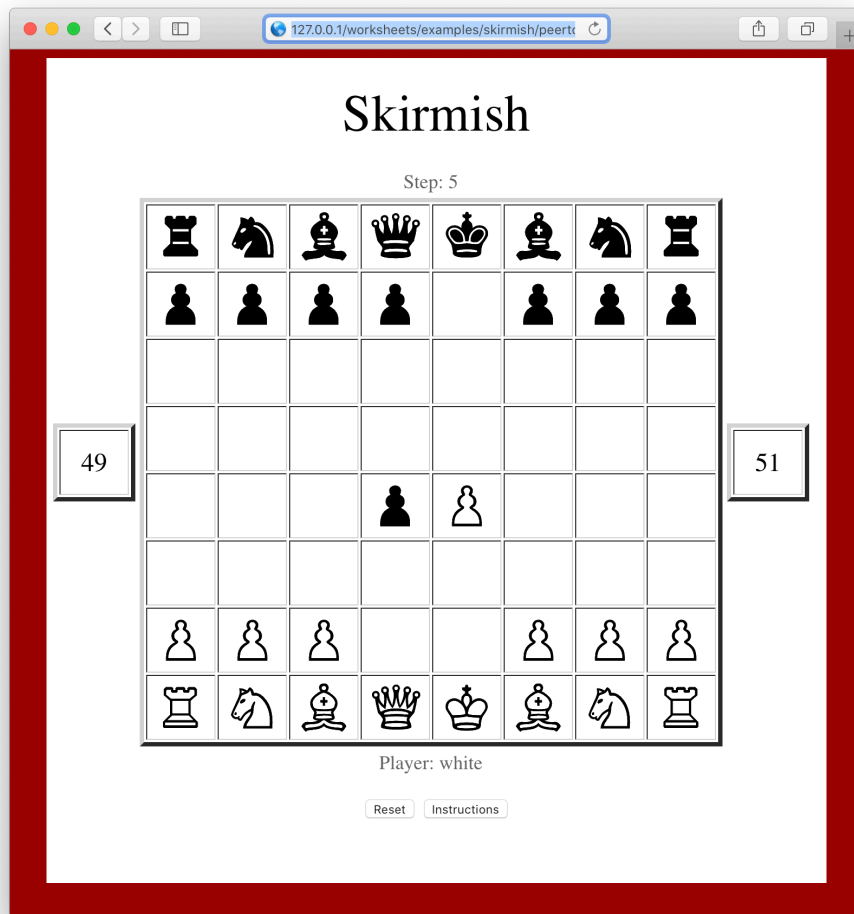
Google
Sheets

Skirmish



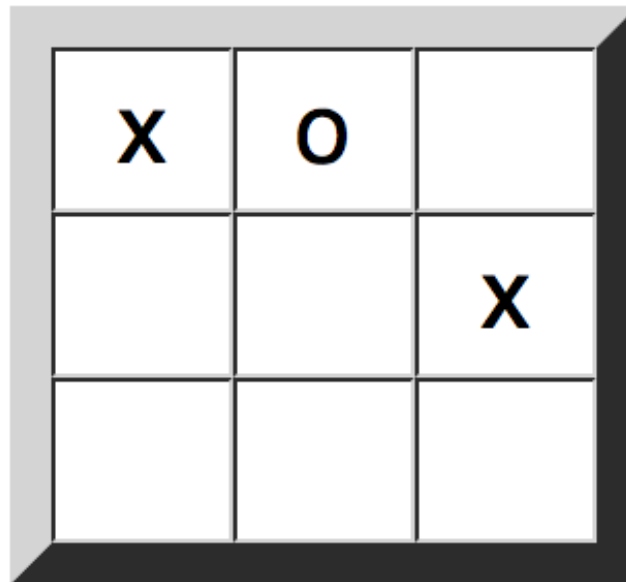
<http://worksheets.stanford.edu/examples/skirmish/peertopeer.html?room=skirmish>

Collaborative Skirmish



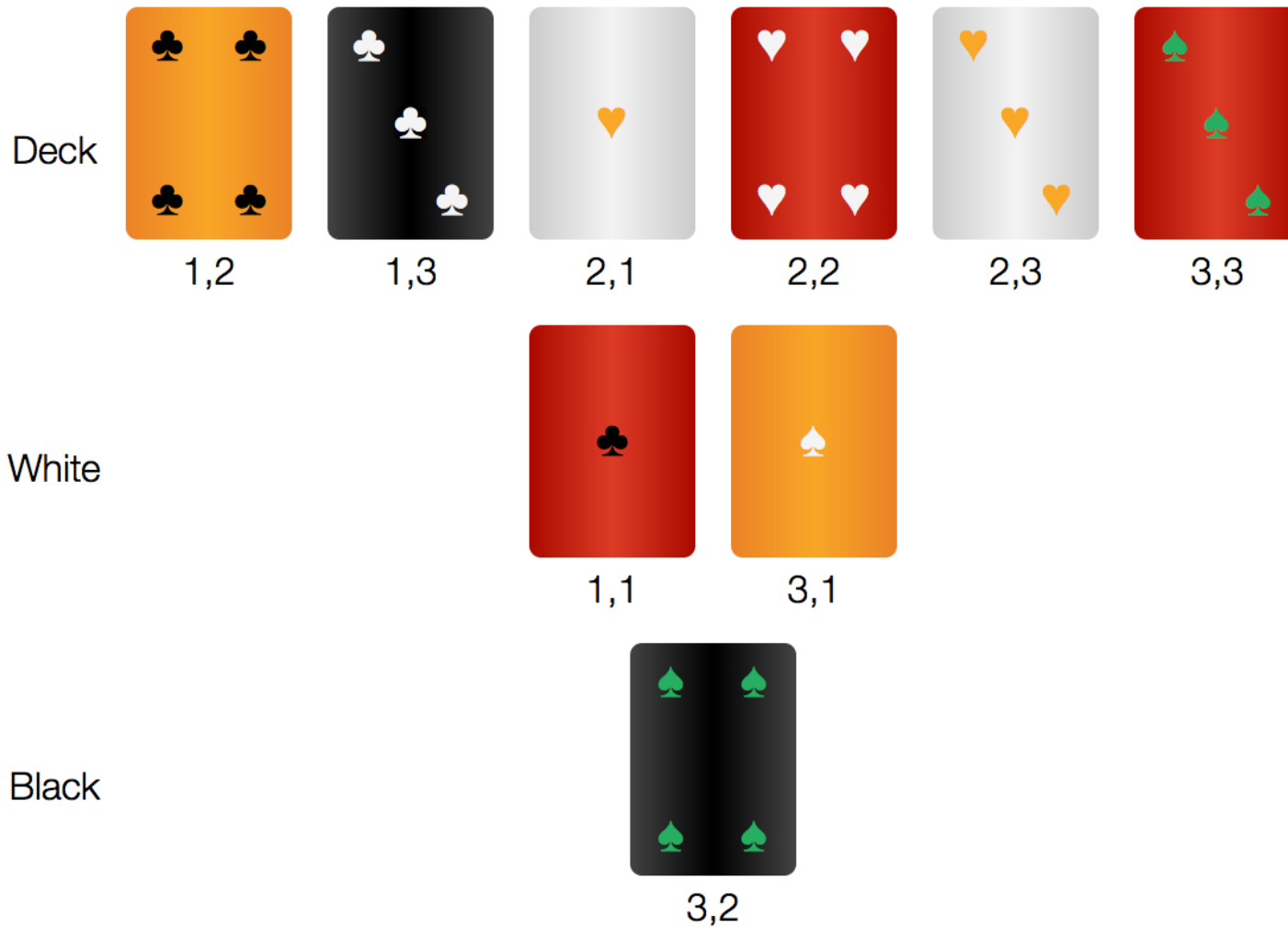
<http://worksheets.stanford.edu/examples/skirmish/peertopeer.html?room=skirmish>

Tic Tac Toe



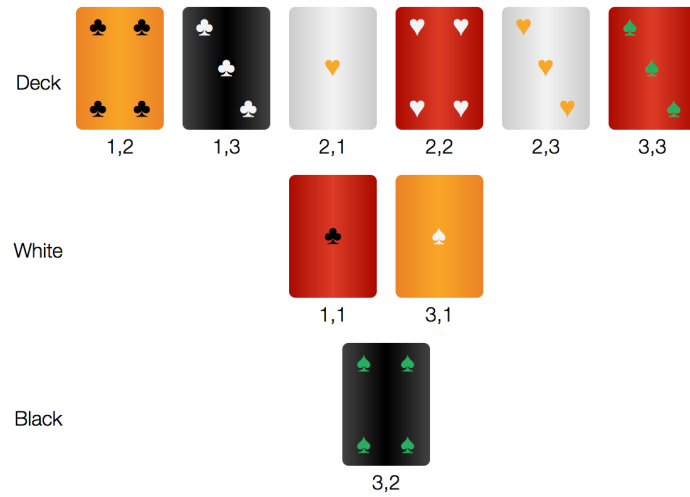
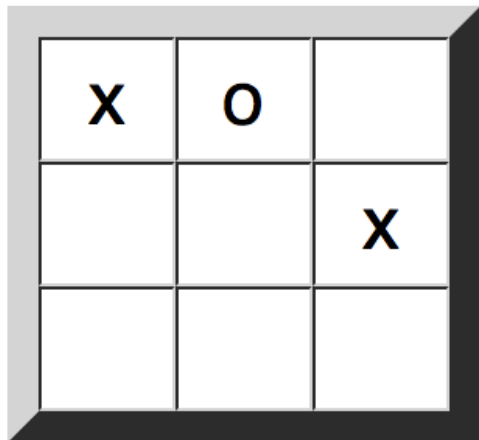
<http://worksheets.stanford.edu/examples/tictactoe/peertopeer.html?room=cs151>

Trifecta



<http://worksheets.stanford.edu/examples/trifecta/peertopeer.html?room=cs151>

Tic Tac Toe - Trifecta



Remote Collaboration

Dataset Sharing

Easy to implement and debug

May move lots of data

Allows all users to see and modify all data

Message Passing (Communication Channels)

Difficult to implement and debug

Moves minimal data

Privacy and security assured

Backend Server (MySQL, PHP, etc.)

Moderate effort to implement and debug

Development and maintenance of backend infrastructure

Moves minimal data

Privacy and security assured

Collaboration Code

```
<script src='http://epilog.stanford.edu/javascript/  
epilog.js'></script>
```

```
<script src='http://worksheets.stanford.edu/javascript/  
worksheets.js'></script>
```

```
<script src='http://worksheets.stanford.edu/javascript/  
warehouse.js'></script>
```



Collaboration Control

```
<textarea id='lambda' type='text/hrf'  
          broadcast='true' reception='true'  
          style='display:none'>
```

```
location(cell(a,1),piece(white,rook,1))  
location(cell(b,1),piece(white,knight,1))
```

```
...
```

```
location(cell(g,8),piece(black,knight,2))  
location(cell(h,8),piece(black,rook,2))
```

```
white(50)
```

```
black(50)
```

```
control(white)
```

```
step(1)
```

```
</textarea>
```


Worksheet Editing

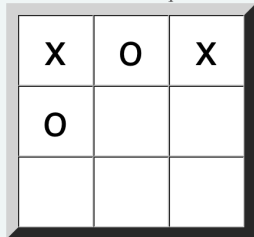
The image displays a web browser window with the Sierra IDE interface. The browser's address bar shows the URL 127.0.0.1. The Sierra IDE is divided into several windows:

- Lambda:** A window for editing code, showing a list of `cell` and `control` functions.
- Sierra:** The main IDE window, displaying a description of the system and its capabilities.
- Evaluate:** A window for running queries, showing a query `countofall(pair(M,N),cell(M,N,b))` and a recursion depth of 10000.
- Query:** A window showing the results of a query, including a unification limit of 100000 and 6 unification(s).
- Tic Tac Toe:** A window for playing a game of Tic Tac Toe, showing a 3x3 grid and a score table.
- Deck:** A window for playing a card game, showing a deck of cards and a score table.

The browser's top menu bar includes Safari, File, Edit, View, History, Bookmarks, Develop, Window, and Help. The top status bar shows the date and time as Sun 12:24 PM, along with various system icons and a 100% zoom level.

Tic Tac Toe

Click in a clear square to mark that square.



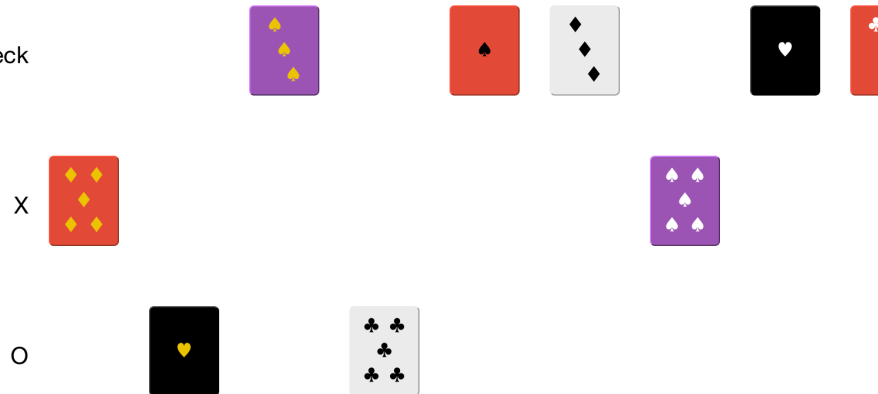
Player: x

Reset

x	o
50	50

Player	Score
50	50

Deck



(mark(M,N))
(mark(M,N))
Results 100 Unification Limit 100000
6 unification(s)
k(2,2)
k(2,3)
k(3,1)
k(3,2)
k(3,3)

127.0.0.1

Execute

(3,1)
Expand Execute Expansion Depth 1000

Editing Code

```
<script src='http://epilog.stanford.edu/javascript/  
epilog.js'></script>
```

```
<script src='http://worksheets.stanford.edu/javascript/  
worksheets.js'></script>
```

```
<script src='http://worksheets.stanford.edu/javascript/  
debugger.js'></script>
```



