

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.

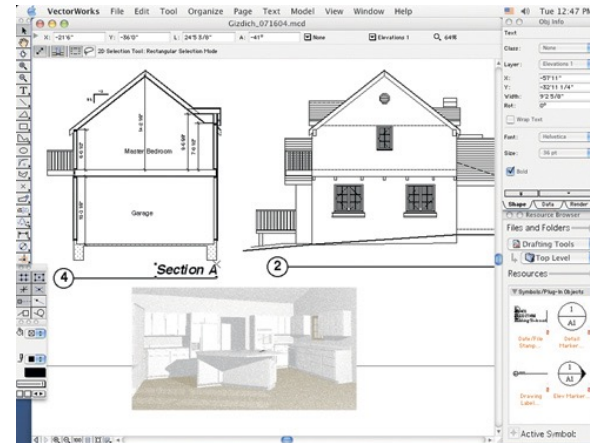
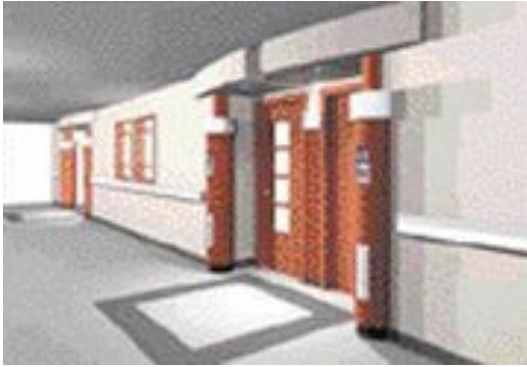
Required:	Equivalent elsewhere (course number/title/institution)	Approval	Grade	Units
Logic, Automata and Complexity (<input checked="" type="checkbox"/> CS 103)				4
Probability (<input type="checkbox"/> CS 109, <input type="checkbox"/> STATS 116, <input type="checkbox"/> CME 106, or <input type="checkbox"/> 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**

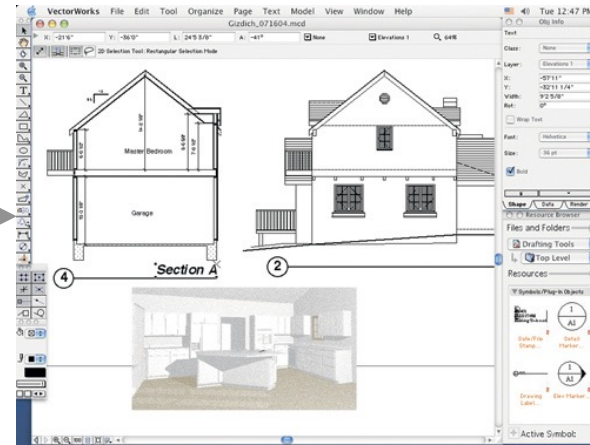
Note: This total may not exceed 10 units.

7 Requirements Left Total Units: 10 Status: Draft

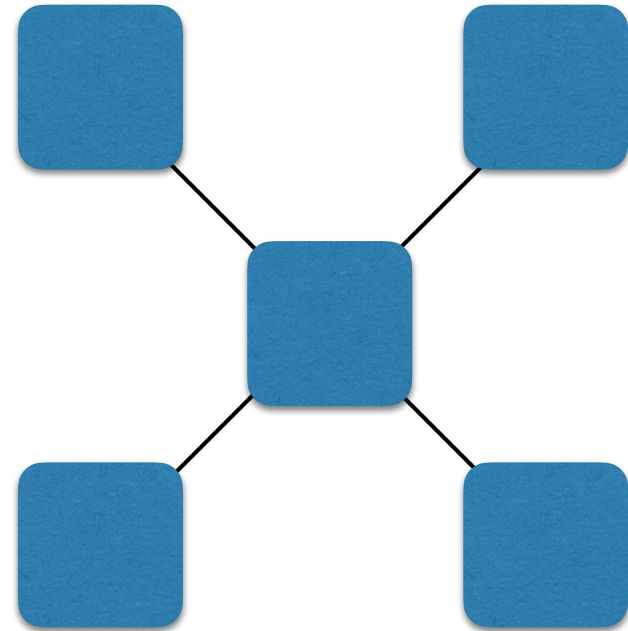
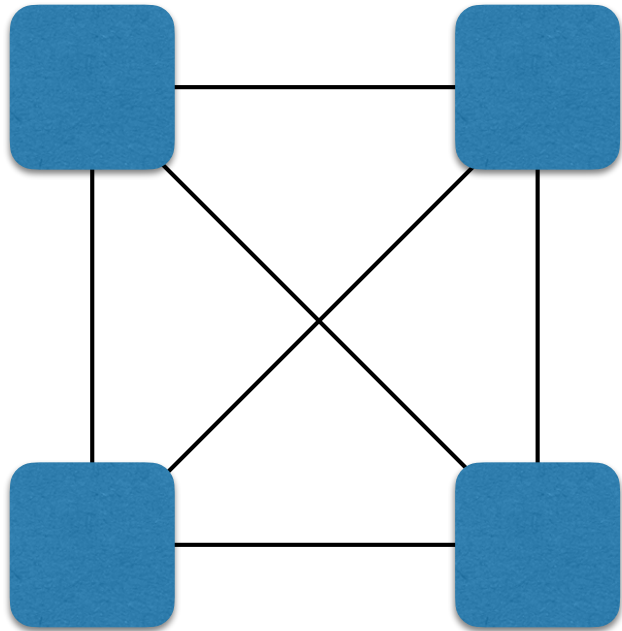
Heterogeneous Worksheets



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

<http://logicprogramming.stanford.edu/examples/courses/index.html>

Schedule 1

	Course 1	Course 2	Course 3	Course 4
When	<div style="border: 1px solid grey; padding: 5px;"><p>Autumn Winter Spring Summer</p></div>	<div style="border: 1px solid grey; padding: 5px;"><p>Autumn Winter Spring Summer</p></div>	<div style="border: 1px solid grey; padding: 5px;"><p>Autumn Winter Spring Summer</p></div>	<div style="border: 1px solid grey; padding: 5px;"><p>Autumn Winter Spring Summer</p></div>

```
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
What	<div style="border: 1px solid grey; padding: 5px;"><div style="background-color: grey; color: white; padding: 2px;">Course 1</div>Course 2 Course 3 Course 4</div>	<div style="border: 1px solid grey; padding: 5px;">Course 1 Course 2 Course 3 Course 4</div>	<div style="border: 1px solid grey; padding: 5px;">Course 1 Course 2 Course 3 Course 4</div>	<div style="border: 1px solid grey; padding: 5px;">Course 1 Course 2 Course 3 Course 4</div>

```
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

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)`

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

Lambda:

`offered(course1, autumn)`

`offered(course2, autumn)`

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



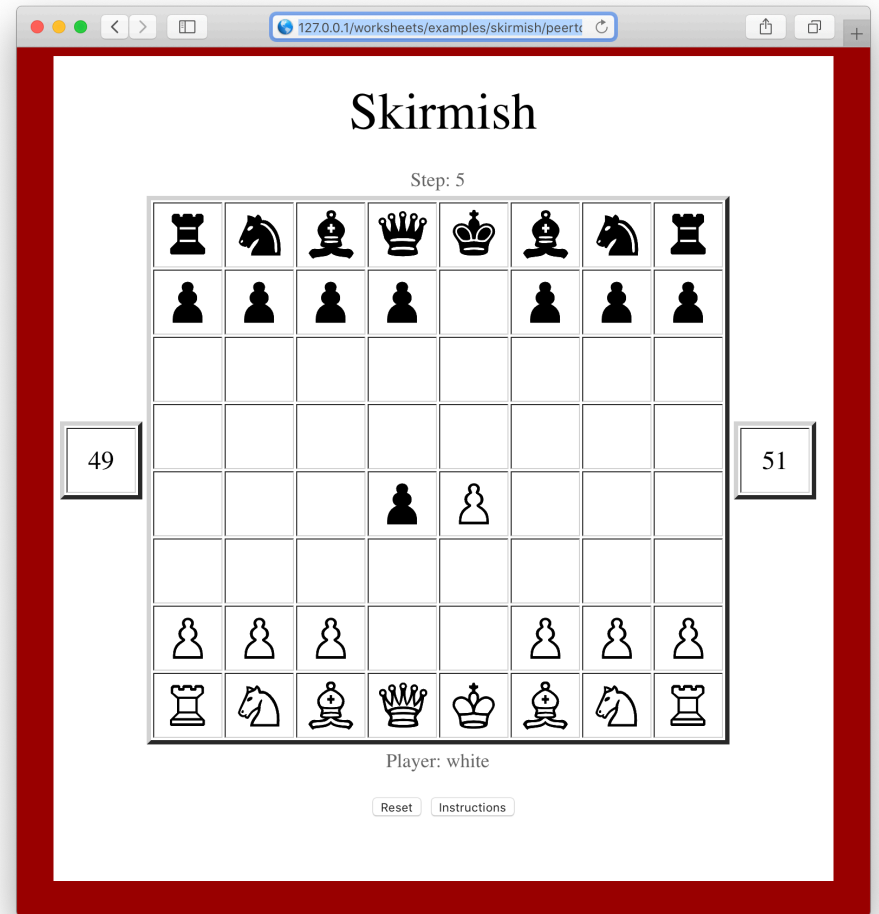
Google
Sheets

Skirmish



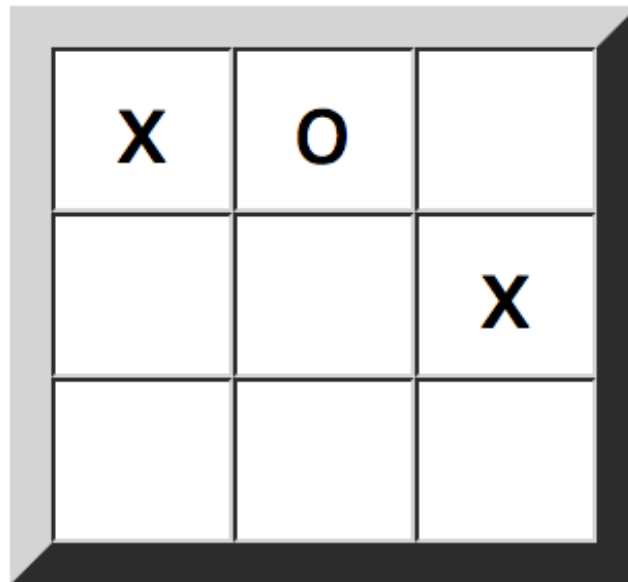
<http://worksheets.stanford.edu/examples/skirmish/peertopeer.html?room=skirmish>

Collaborative Skirmish



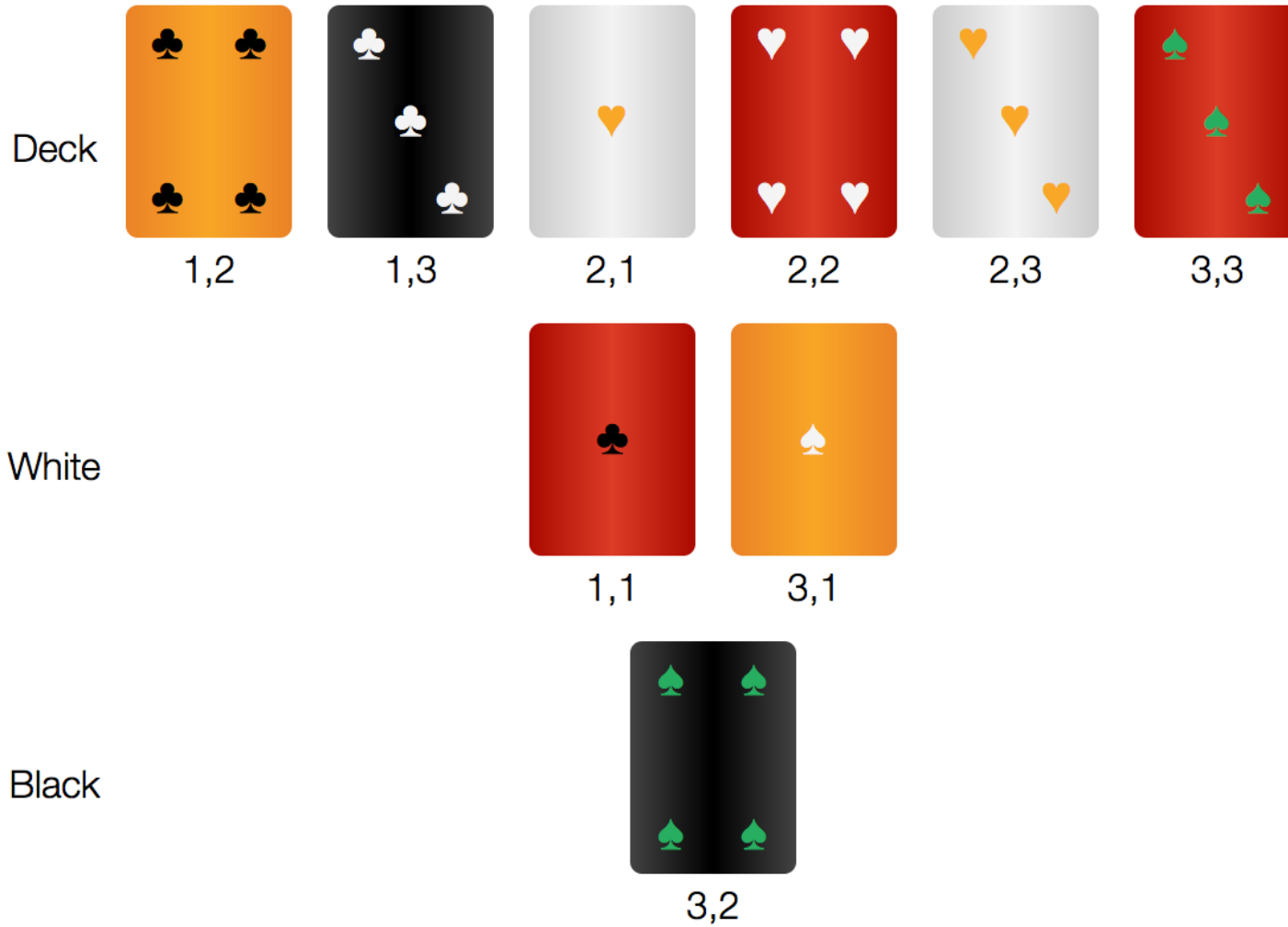
<http://worksheets.stanford.edu/examples/skirmish/peertopeer.html?room=skirmish>

Tic Tac Toe



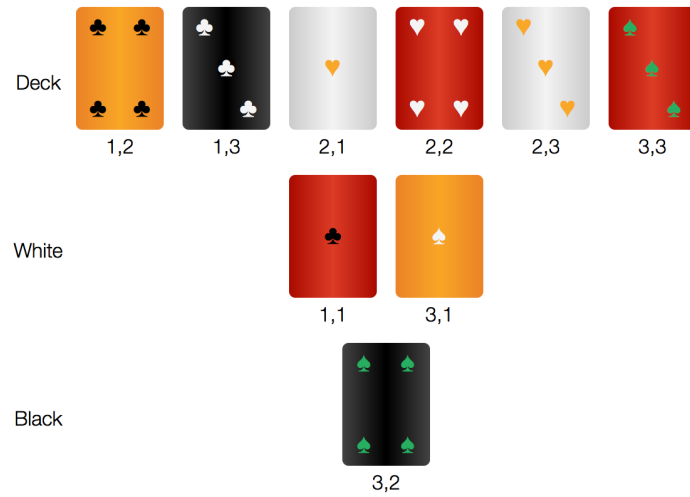
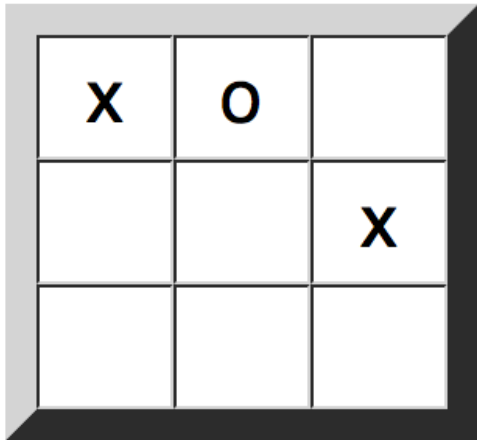
<http://worksheets.stanford.edu/examples/tictactoe/peertopeer.html?room=cs151>

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 main window is titled "Sierra" and contains a menu bar with "Files", "Tools", and "Worksheets". Below the menu bar, there is a text area with the following text:

Sierra is a browser-based interactive development environment (IDE) for Epilog. It allows users to view and edit datasets and rulesets. It provides a variety of tools for querying and modifying datasets and rulesets. As changes are made, it automatically updates visible datasets in spreadsheet-like fashion in accordance with the user's rules. It also provides tools for analyzing datasets and rulesets, tools for tracing program execution, and tools for saving and loading files.

The command bar across the top provides access to menus concerning files, tools, and worksheets. Click [here](#) for an introduction to the system and its capabilities.

This version of Sierra utilizes different windows for different tools. It replaces a previous version in which all tools are accessed in the same window. Click [here](#) to switch to the old version.

Overlaid on the Sierra window are several other windows:

- Lambda**: A window with a text area containing a list of function definitions:

```
cell(1,1,x)
cell(2,1,o)
cell(2,2,b)
cell(2,3,b)
cell(3,1,b)
cell(3,2,b)
cell(3,3,b)
cell(1,3,x)
cell(1,2,o)
control(x)
```
- Evaluate**: A window with a "Query" input field containing `countofall(pair(M,N),cell(M,N,b))` and a "Recursion Depth" field set to `10000`. The output area shows the number `5`.
- Query**: A window with a "Query" input field containing `(mark(M,N))` and a "Results" field set to `100`. The "Unification Limit" is set to `100000`. The output area shows `6 unification(s)` and a list of results:

```
k(2,2)
k(2,3)
k(3,1)
k(3,2)
k(3,3)
```
- Execute**: A window with an "Execute" button and an "Expansion Depth" field set to `1000`. The output area shows `(3,1)`.
- Tic Tac Toe**: A window with a 3x3 grid. The top row contains 'X', 'O', 'X'. The bottom-left cell contains 'O'. Below the grid, it says "Player: x". There is a "Reset" button and a table with columns 'x' and 'o', and rows with the value '50'.
- Deck**: A window showing a deck of cards. The cards are: a purple diamond, a red spade, a grey diamond, a black heart, a red club, a red diamond, a purple spade, a black heart, and a grey club. Below the cards, there is a table with columns "Player" and "Score", and rows with the value "50".

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>
```



