Logic Programming
General Game Playing

Michael Genesereth
Computer Science Department
Stanford University

Game Playing

Human Game Playing

- Intellectual Activity
- Skill Comparison

Computer Game Playing

- Testbed for AI
- Limitations

Limitations of Game Playing for AI

Narrowness

- Good at one game, not so good at others
- Cannot do anything else

Not really testing intelligence of machine

- Programmer does all the interesting analysis / design
- Machine simply follows the recipe

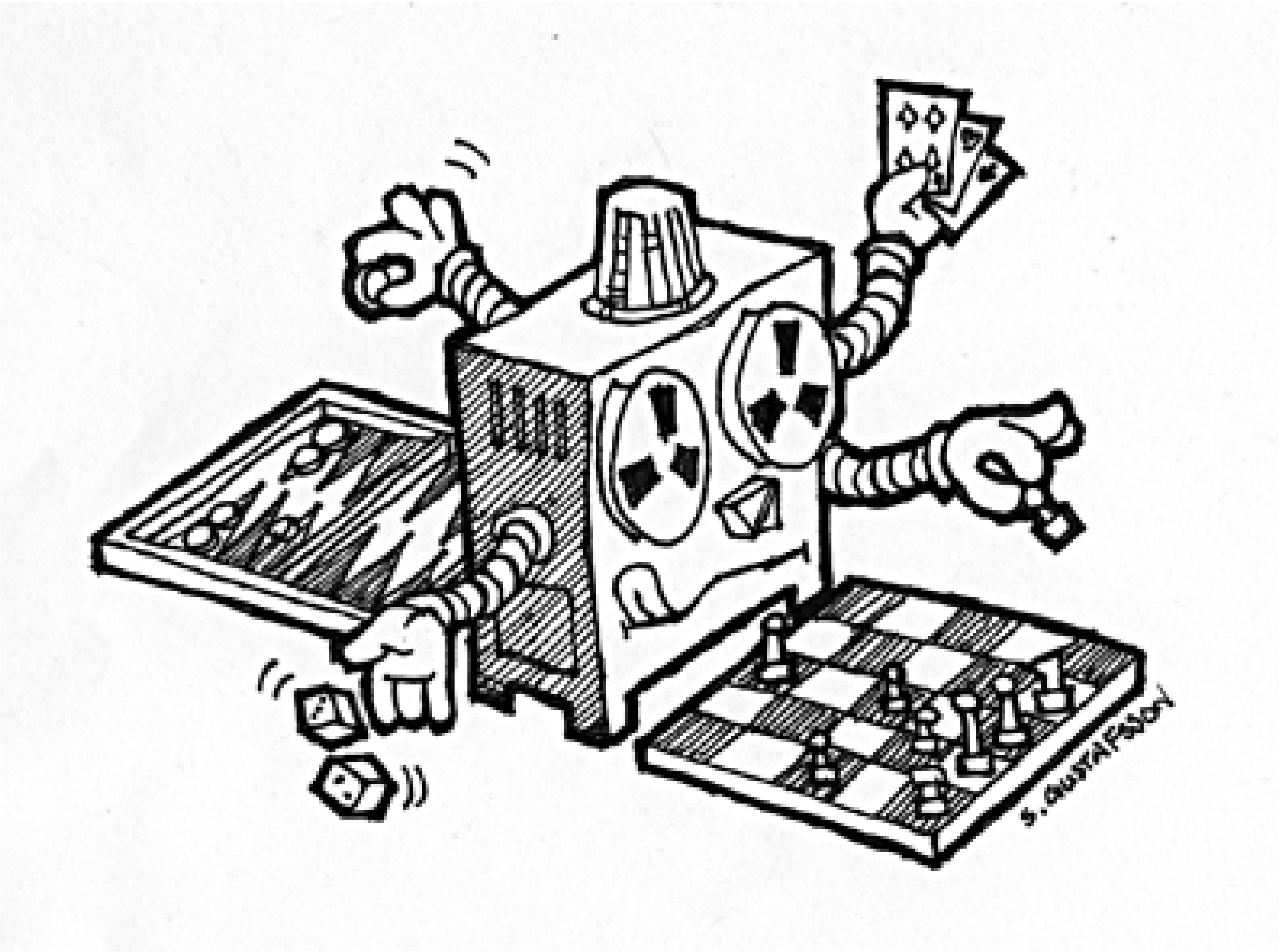
General Game Playing

General Game Players are systems able to play arbitrary games effectively based solely on formal descriptions supplied at “runtime”.

Translation: They don’t know the rules until the game starts.

Must figure out for themselves:
legal moves, winning strategy
in the face of incomplete info and resource bounds

Variety of Games



Novelty

International GGP Competition

Annual GGP Competition

Annual GGP Competition

- Held at AAAI or IJCAI conference
- Administered by Stanford University
- (Stanford folks not eligible to participate)

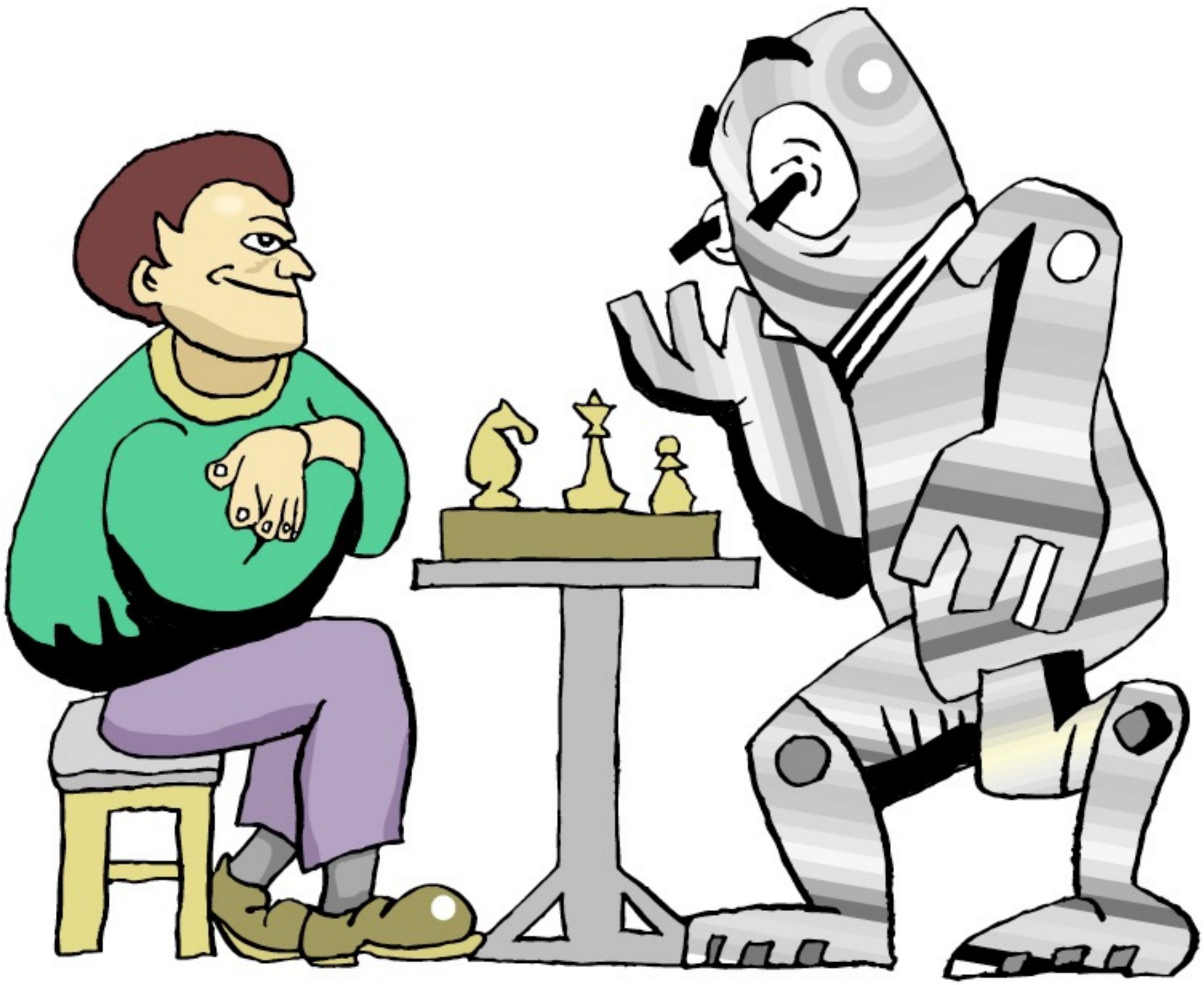
GGP-05 Winner Jim Clune

International GGP Competition

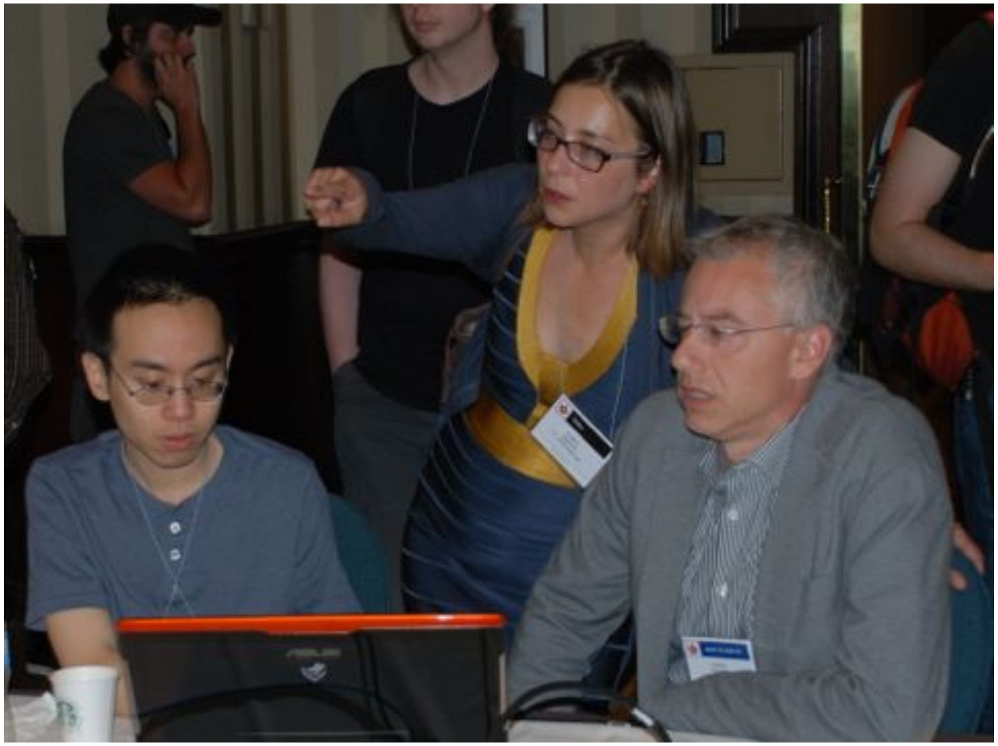
Winners Over the Years

ClunePlayer - Jim Clune (USA)
FluxPlayer - Schiffel, Thielscher (Germany)
CadiaPlayer - Bjornsson, Finsson (Iceland)
CadiaPlayer - Bjornsson, Finsson (Iceland)
Ary - Mehat (France)
TurboTurtle - Schreiber (USA)
CadiaPlayer - Bjornsson, Finsson (Iceland)
TurboTurtle - Schreiber (USA)
Sancho - Draper (USA), Rose (UK)
Galvanise - Emslie
WoodStock - Piette (France)

Carbon versus Silicon



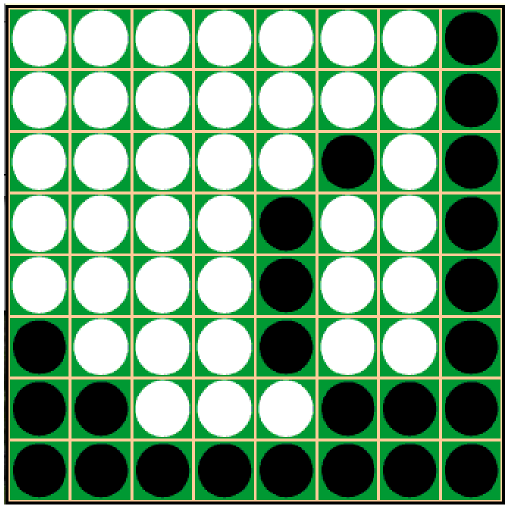
Human Race Being Defeated

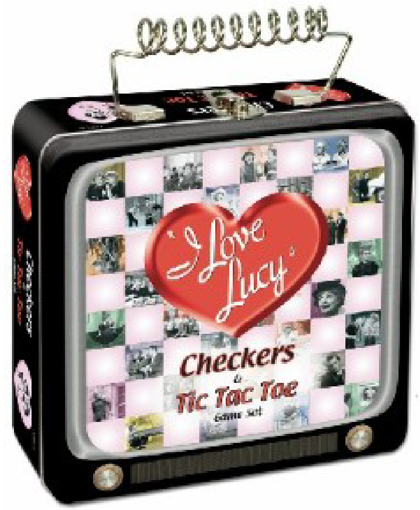


Result

Game Description

Multiplicity of Games





Finite Synchronous Games

Environment

- Environment with finitely many states
- One initial state and one or more terminal states
- Each state has a unique goal value for each player

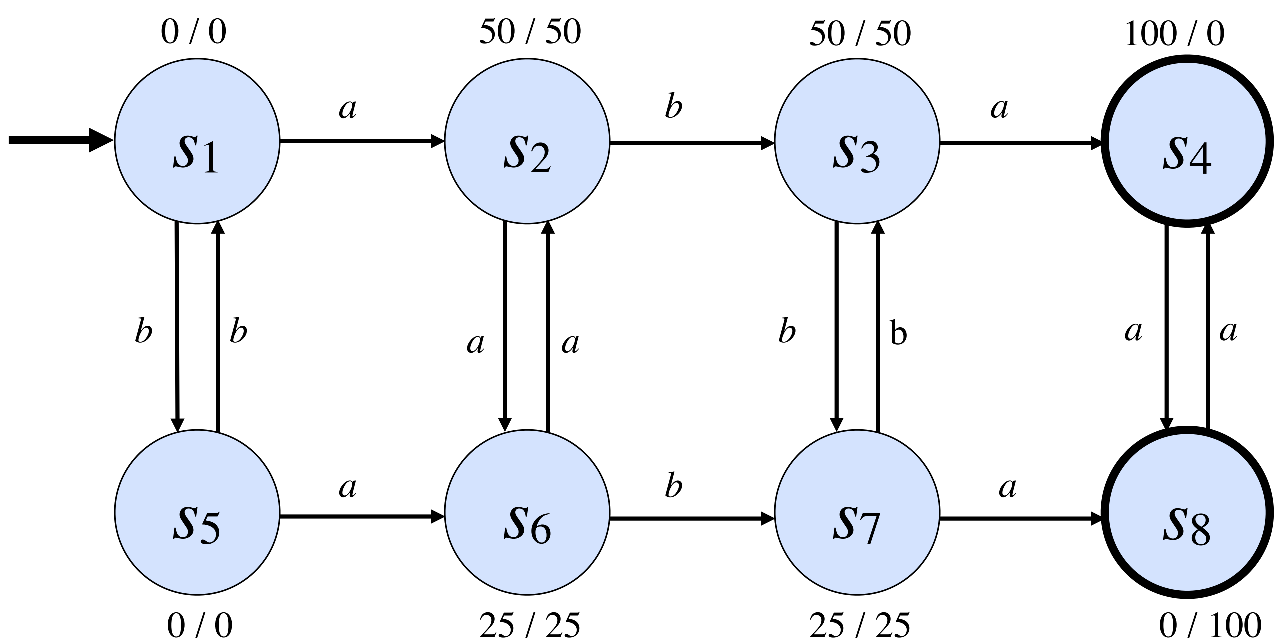
Players

- Fixed, finite number of players
- Each with finitely many moves

Dynamics

- Finitely many steps
- All players move on all steps (some no ops)
- Environment changes only in response to moves

Finite State Machine



Direct Description

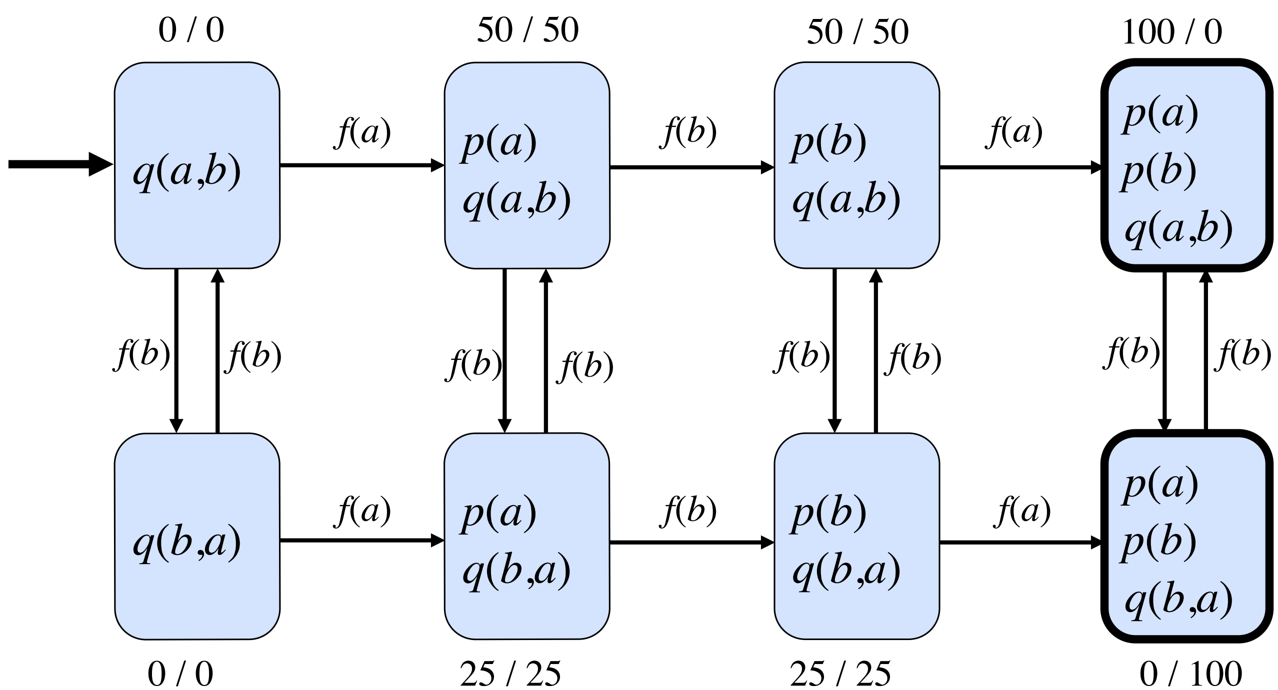
Good News: Since all of the games that we are considering are finite, it is possible in principle to communicate game information in the form of state graphs.

Problem: Size of description. Even though everything is finite, these sets can be large.

Solution:

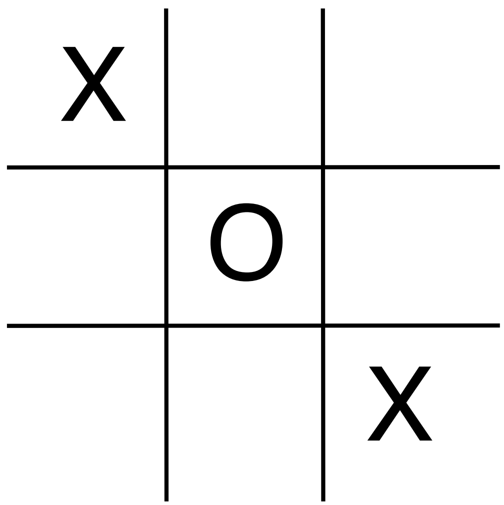
Exploit regularities / structure in state graphs to produce compact encoding

Structured State Machine



States Represented as Datasets

We use datasets to characterize states of the world. The *facts in a dataset are assumed to be true* and those that are *not in the dataset are assumed to be false*.




```
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)
```

Actions Transform Datasets

```
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)
```

mark(1,3) →

```
cell(1,1,x)
cell(1,2,b)
cell(1,3,o)
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(x)
```

Game Description Language

Game Description Language (or GDL) is a formal language for encoding the rules of games conceptualized as structured sate machines.

Game rules written as sentences in Epilog.

GDL is widely used in the research literature and is used in virtually all General Game Playing competitions.

GDL extensions are applicable in real-world applications such as Enterprise Management and Computational Law.

Rules of Tic-Tac-Toe

```
role(white)
role(black)

base(cell(M,N,Z)) :-
  index(M) &
  index(N) &
  filler(Z)

base(control(W)) :- role(W)

input(W,mark(X,Y)) :-
  role(W) &
  index(X) &
  index(Y)

input(W,noop) :- role(W)

init(cell(X,Y,b)) :-
  index(X) &
  index(Y)

init(control(white))

legal(P,mark(X,Y)) :-
  true(cell(X,Y,b)) &
  true(control(P))

legal(x,noop) :-
  true(control(black))

legal(o,noop) :-
  true(control(white))
```

```
next(cell(M,N,x)) :-
    does(white,mark(M,N))

next(cell(M,N,0)) :-
    does(black,mark(M,N))

next(cell(M,N,Z)) :-
    does(P,mark(M,N)) &
    true(cell(M,N,Z)) & Z!=b

next(cell(M,N,b)) :-
    does(P,mark(J,K)) &
    true(cell(M,N,b)) &
    distinct(M,J)

next(cell(M,N,b)) :-
    does(P,mark(J,K)) &
    true(cell(M,N,b)) &
    distinct(N,K)

next(control(white)) :-
    true(control(black))

next(control(black)) :-
    true(control(white))

goal(white,100) :- line(x) & ~line(o)
goal(white,50) :- ~line(x) & ~line(o)
goal(white,0) :- ~line(x) & line(o)
goal(black,100) :- ~line(x) & line(o)
goal(black,50) :- ~line(x) & ~line(o)
goal(black,0) :- line(x) & ~line(o)
```

```
terminal :- line(P)
terminal :- ~open

row(M,P) :-
    true(cell(M,1,P)) &
    true(cell(M,2,P)) &
    true(cell(M,3,P))

column(N,P) :-
    true(cell(1,N,P)) &
    true(cell(2,N,P)) &
    true(cell(3,N,P))

diagonal(P) :-
    true(cell(1,1,P)) &
    true(cell(2,2,P)) &
    true(cell(3,3,P))

diagonal(P) :-
    true(cell(1,3,P)) &
    true(cell(2,2,P)) &
    true(cell(3,1,P))

line(P) :- row(M,P)
line(P) :- column(N,P)
line(P) :- diagonal(P)

open :- true(cell(M,N,b))

index(1)      filler(x)
index(2)      filler(o)
index(3)      filler(b)
```

Game Specific Constants

Objects:

`x, o` - *roles*
`1, 2, 3` - *indices of rows and columns*
`b` - *blank*

Relations:

- `cell(index,index,mark)`
- `row(index,mark)`
- `column(index,mark)`
- `diagonal(mark)`
- `line(mark)`
- `open`

Operation:

- `mark(index,index)`

Game-Independent Vocabulary

Object Constants:

0, 1, 2, 3, … , 100 - *numbers (i.e. entities)*

Relation Constants:

- **control**(*role*)
- **legal**(*role,action*)
- **goal**(*role,number*)
- **terminal**

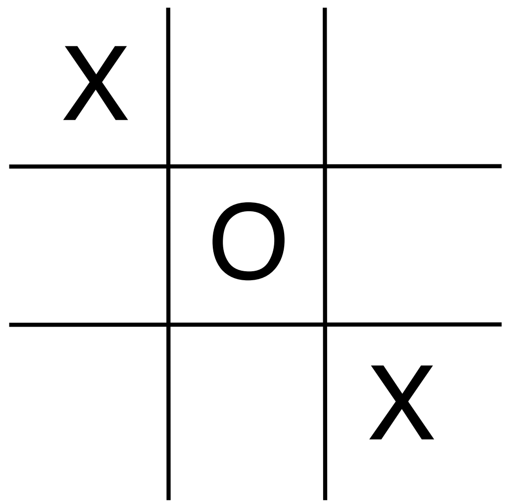
- **init**(*proposition*)
- **role**(*role*)
- **base**(*proposition*)
- **action**(*role,action*)

Legal Moves

```
legal(mark(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)
```

Physics

```
mark(M,N) ::
   control(Z) ==> ~cell(M,N,b) & cell(M,N,Z)
mark(M,N) ::
   control(x) ==> ~control(x) & control(o)
mark(M,N) ::
   control(o) ==> ~control(o) & control(x)
```

State		Next state
cell(1,1,x)		cell(1,1,x)
cell(1,2,b)		cell(1,2,b)
cell(1,3,b)		cell(1,3,o)
cell(2,1,b)		cell(2,1,b)
cell(2,2,o)	mark(1,3) →	cell(2,2,o)
cell(2,3,b)		cell(2,3,b)
cell(3,1,b)		cell(3,1,b)
cell(3,2,b)		cell(3,2,b)
cell(3,3,x)		cell(3,3,x)
control(o)		control(x)

Supporting Concepts

```
row(M,Z) :- cell(M,1,Z) & cell(M,2,Z) & cell(M,3,Z)
col(N,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)
```

Goals and Termination

```
goal(x,100) :- line(x)
goal(x,50) :- ~line(x) & ~line(o)
goal(x,0) :- line(o)

goal(o,100) :- line(o)
goal(o,50) :- ~line(x) & ~line(o)
goal(o,0) :- line(x)


terminal :- line(W)
terminal :- ~open

open :- cell(X,Y,b)
```

Initial State

```
init(cell(1,1,b))          cell(1,1,b)
init(cell(1,2,b))          cell(1,2,b)
init(cell(1,3,b))          cell(1,3,b)
init(cell(2,1,b))          cell(2,1,b)
init(cell(2,2,b))          cell(2,2,b)
init(cell(2,3,b))          cell(2,3,b)
init(cell(3,1,b))          cell(3,1,b)
init(cell(3,2,b))          cell(3,2,b)
init(cell(3,3,b))          cell(3,3,b)
init(control(x))           control(x)
```

Metadata

```
role(x)
role(o)

base(cell(X,Y,x)) :- index(X) & index(Y)
base(cell(X,Y,o)) :- index(X) & index(Y)
base(cell(X,Y,b)) :- index(X) & index(Y)
base(control(W)) :- role(W)

action(mark(X,Y)) :- index(X) & index(Y)

index(1)
index(2)
index(3)
```

Game Simulation

Sierra

Not Secure — epilog.stanford.edu

Epilog

What versus How

Tutorial Vocabulary Sierra Examples Games EpilogJS

Sierra is a collection of related browser-based applications of use in developing rulesets and datasets encoded in Epilog. Each application allows users to view and edit datasets and rulesets and provides tools of various sorts for processing those rule sets and data sets.

Sierrabase provides elementary tools for viewing and editing Epilog datasets. It includes tools for evaluating relational and functional queries on data and applying transformations to that data. Click here for a brief tour of Sierrabase.

Sierralite provides elementary tools for viewing and editing rulesets as well as datasets. It includes tools for computing defined relations, evaluating defined functions, and applying defined operations. *It is the most commonly used Sierra application*. Click here for a brief tour of Sierralite.

Sierraplus provides elementary tools for satisfying relational constraints. Experimental.

Sierraplan provides elementary tools for creating plans in dynamic, single-agent environments. Experimental.

Sierraplay provides elementary tools for selecting actions in dynamic, multi-agent environments. Experimental.

Sierrarule provides tools for analyzing and transforming rules as well as data. Experimental.

Sierrameta provides tools for proof extraction, tracing, and error checking. Experimental.

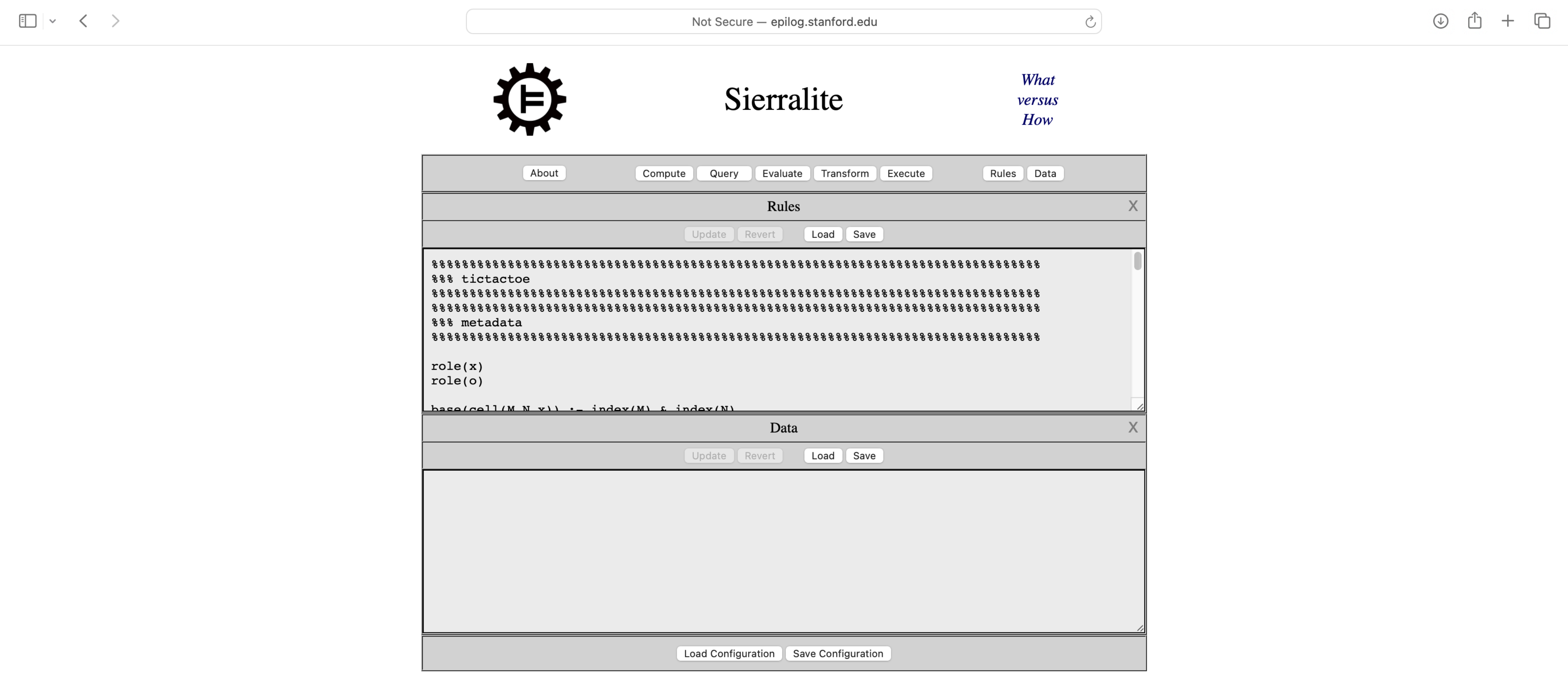
Sierrapro is an integrated development environment (IDE) that combines the capabilities of these more specific applications operating in integrated fashion in separate, communicating windows. Sierrapro is preferred by many advanced users. However, it can be confusing for those who are not accustomed to working with multiple, interacting windows, and such users are encouraged to begin with Sierralite. Click here for a brief tour of Sierrapro.

Feedback

Sierra


Not Secure — epilog.stanford.edu
Sierralite
What
versus
How
About
Compute
Query
Evaluate
Transform
Execute
Rules
Data
Rules
X
Update
Revert
Load
Save
%%%
%%% tictactoe
%%%
%%%
%%% metadata
%%%
role(x)
role(o)
Data
X
Update
Revert
Load
Save
Load Configuration
Save Configuration

Sierra



Not Secure — epilog.stanford.edu
Sierralite
What
versus
How
About
Compute
Query
Evaluate
Transform
Execute
Rules
Data
Rules
X
Update
Revert
Load
Save
%%
%%% tictactoe
%%
%%
%%% metadata
%%
role(x)
role(o)
Data
X
Update
Revert
Load
Save
Load Configuration
Save Configuration

Sierra

Not Secure — epilog.stanford.edu

Sierralite

What versus How

About | Compute | Query | Evaluate | Transform | Execute | Rules | Data

Query

Pattern P

Query init(P)

Show Results 100 Unification Limit 1000000 Autorefresh

14 unification(s)

```
cell(1,1,b)
cell(1,2,b)
cell(1,3,b)
cell(2,1,b)
cell(2,2,b)
cell(2,3,b)
cell(3,1,b)
cell(3,2,b)
cell(3,3,b)
control(x)
```

Rules

Update Revert Load Save

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% tictactoe
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% metadata
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

role(x)
role(o)

base(cell(M,N,x)) :- index(M) & index(N)
```

Data

Update Revert Load Save

Sierra

Not Secure — epilog.stanford.edu

Sierralite

What versus How

About | Compute | Query | Evaluate | Transform | Execute | Rules | Data

Query

Pattern: A

Query: legal(A)

Show Results 100 Unification Limit 1000000 Autorefresh

10 unification(s)

```
mark(1,1)
mark(1,2)
mark(1,3)
mark(2,1)
mark(2,2)
mark(2,3)
mark(3,1)
mark(3,2)
mark(3,3)
```

Rules

Update Revert Load Save

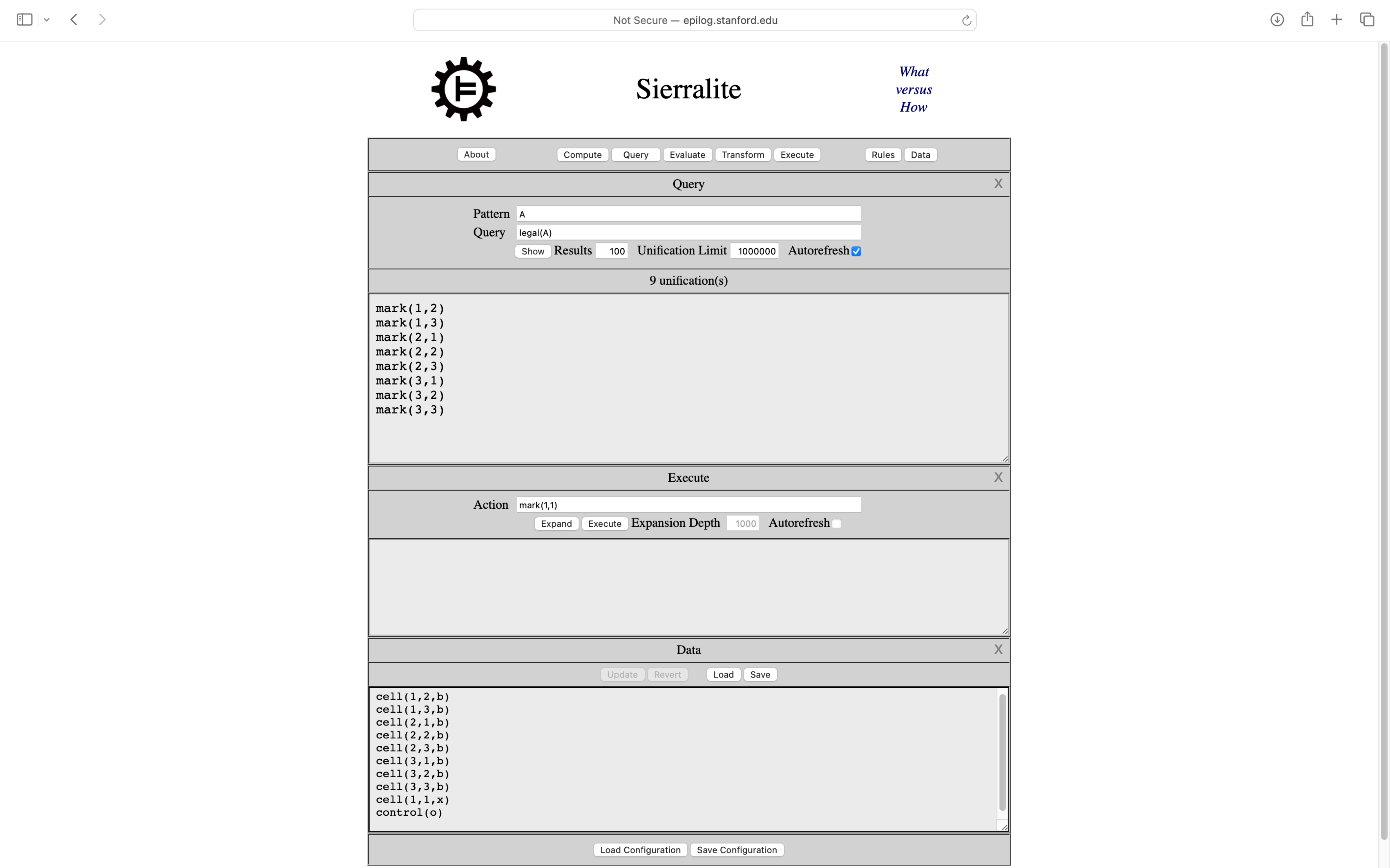
```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% tictactoe
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% metadata
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

role(x)
role(o)

base(cell(M,N,x)) :- index(M) & index(N)
```

Data

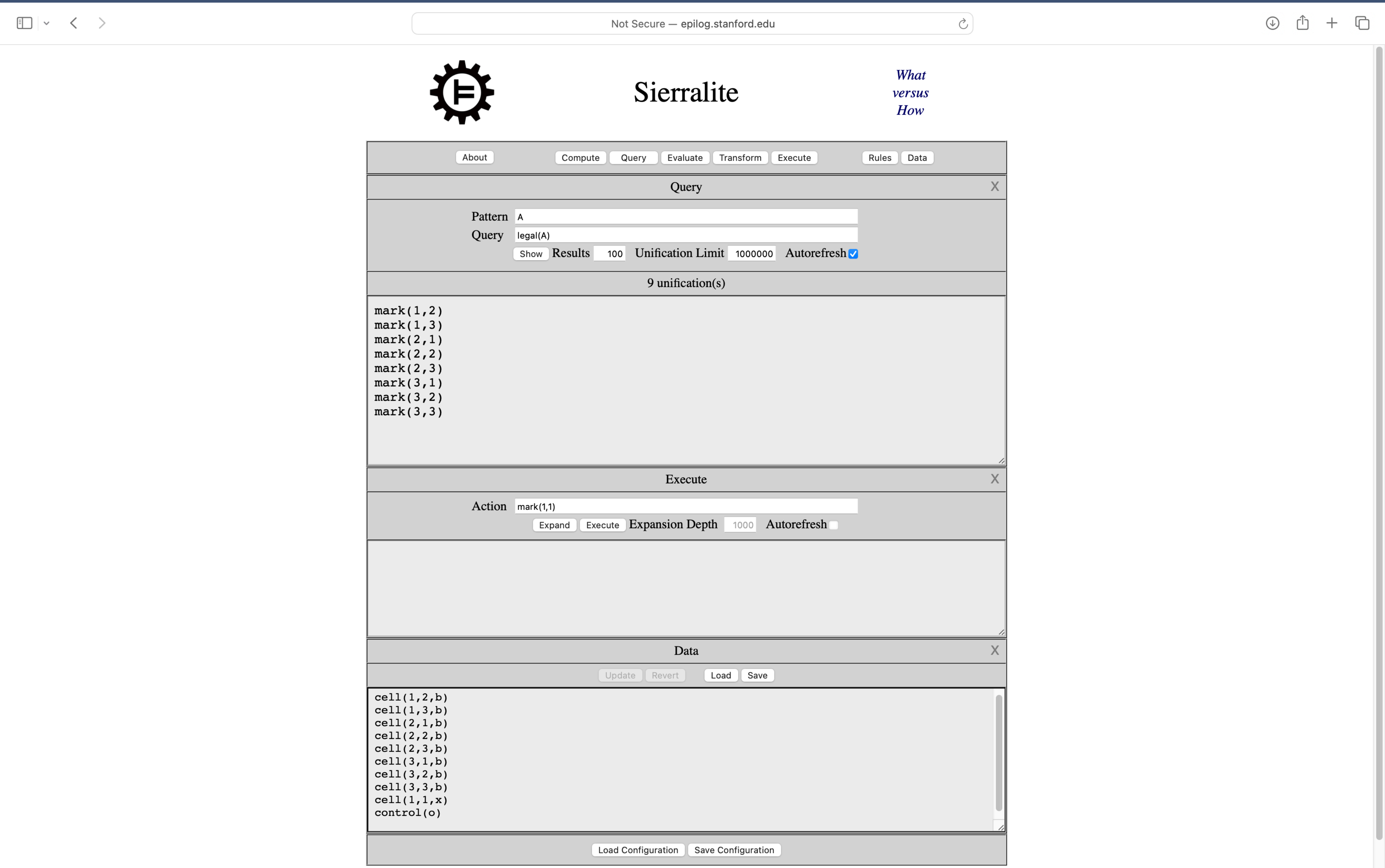
Update Revert Load Save

```
cell(1,1,b)
cell(1,2,b)
cell(1,3,b)
cell(2,1,b)
cell(2,2,b)
cell(2,3,b)
cell(3,1,b)
cell(3,2,b)
cell(3,3,b)
control(x)
```

Sierra


Not Secure — epilog.stanford.edu
Sierralite
What
versus
How
About
Compute
Query
Evaluate
Transform
Execute
Rules
Data
Query
X
Pattern
A
Query
legal(A)
Show
Results
100
Unification Limit
1000000
Autorefresh
9 unification(s)
mark(1,2)
mark(1,3)
mark(2,1)
mark(2,2)
mark(2,3)
mark(3,1)
mark(3,2)
mark(3,3)
Execute
X
Action
mark(1,1)
Expand
Execute
Expansion Depth
1000
Autorefresh
Data
X
Update
Revert
Load
Save
cell(1,2,b)
cell(1,3,b)
cell(2,1,b)
cell(2,2,b)
cell(2,3,b)
cell(3,1,b)
cell(3,2,b)
cell(3,3,b)
cell(1,1,x)
control(o)
Load Configuration
Save Configuration

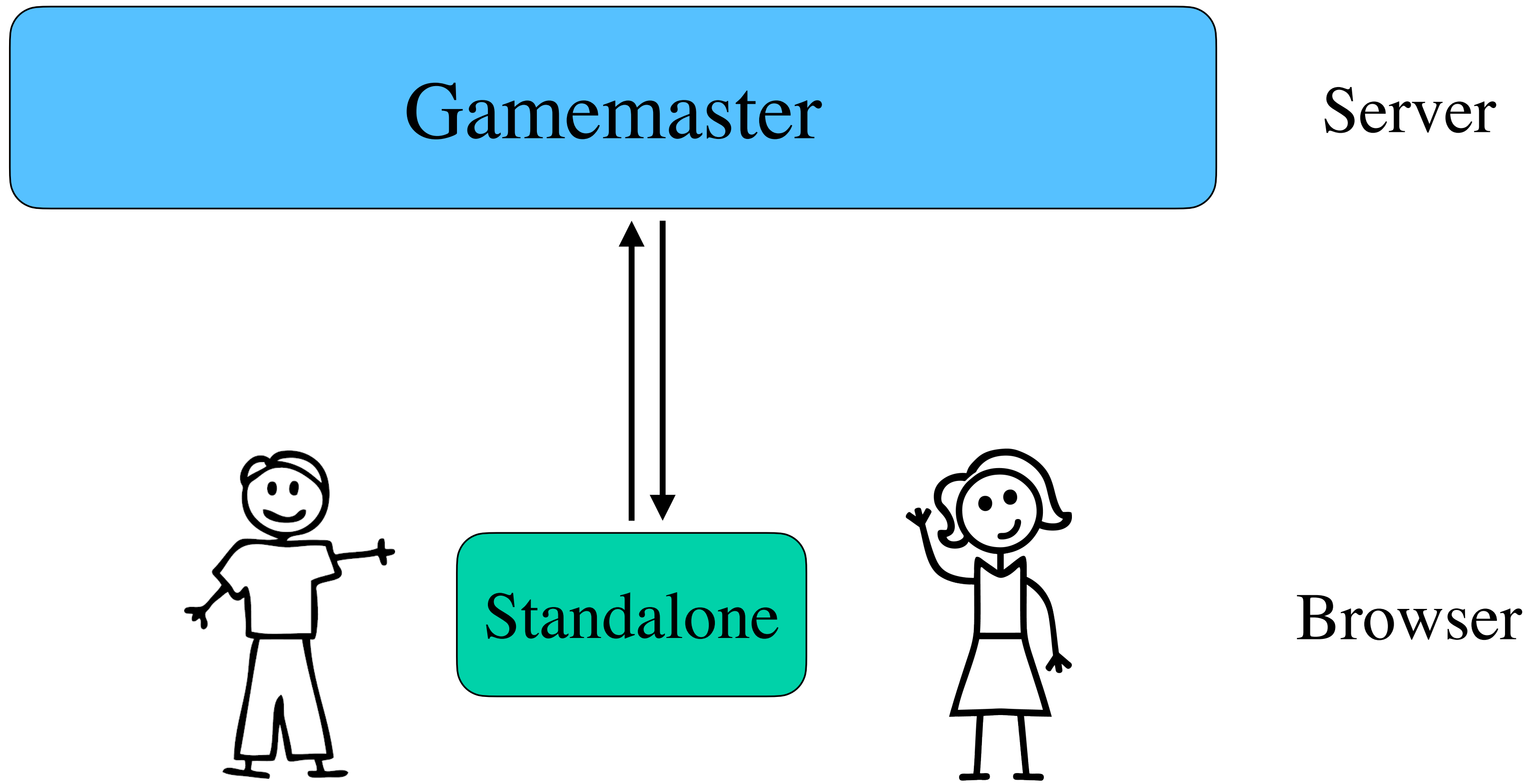
Sierra



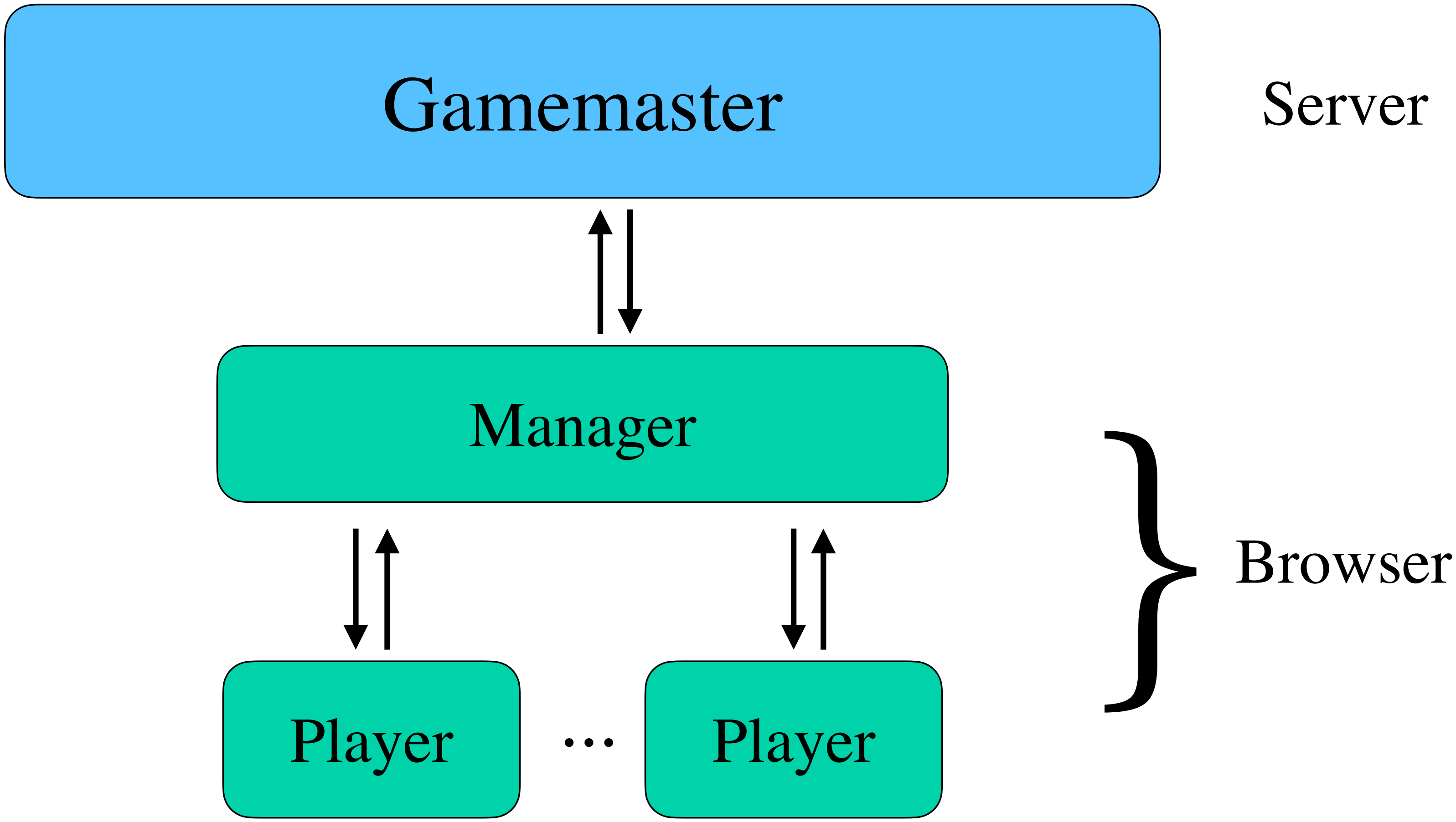
Not Secure — epilog.stanford.edu
Sierralite
What versus How
About
Compute
Query
Evaluate
Transform
Execute
Rules
Data
Query
Pattern A
Query legal(A)
Show
Results 100
Unification Limit 1000000
Autorefresh
9 unification(s)
mark(1,2)
mark(1,3)
mark(2,1)
mark(2,2)
mark(2,3)
mark(3,1)
mark(3,2)
mark(3,3)
Execute
Action mark(1,1)
Expand
Execute
Expansion Depth 1000
Autorefresh
Data
Update
Revert
Load
Save
cell(1,2,b)
cell(1,3,b)
cell(2,1,b)
cell(2,2,b)
cell(2,3,b)
cell(3,1,b)
cell(3,2,b)
cell(3,3,b)
cell(1,1,x)
control(o)
Load Configuration
Save Configuration

Game Management

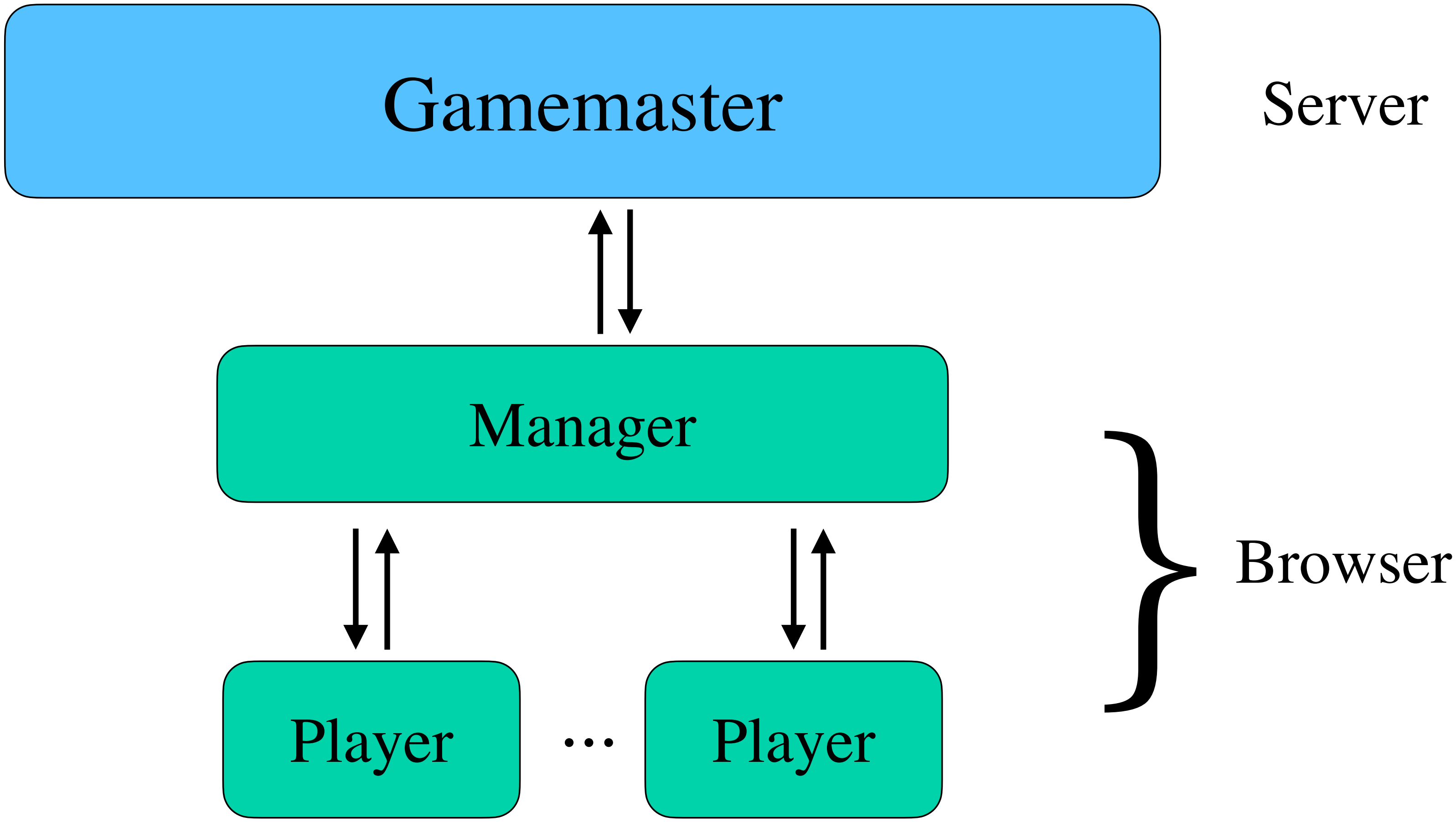
Standalone Game Playing



Automatic Game Playing

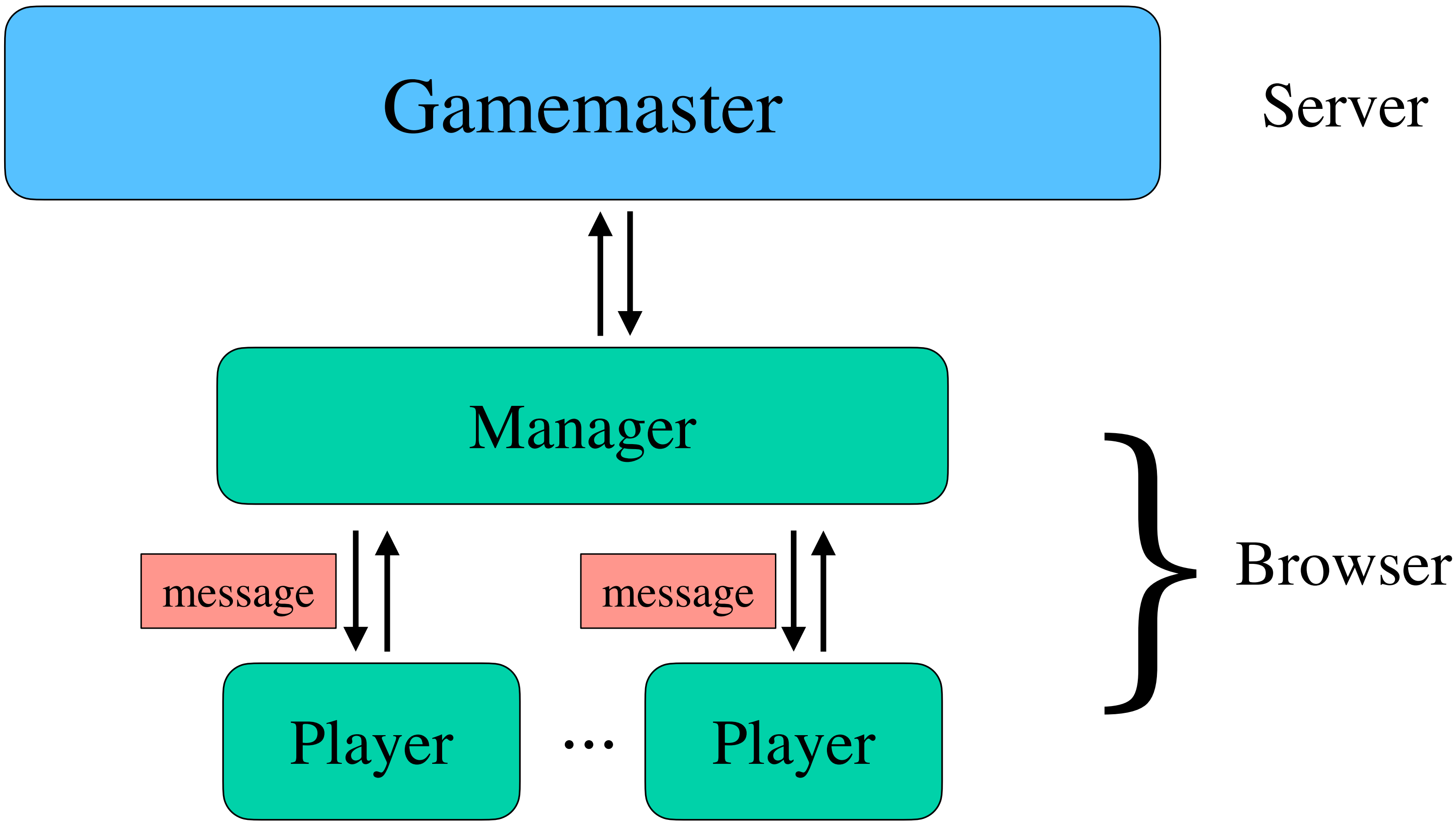


Automatic Game Playing



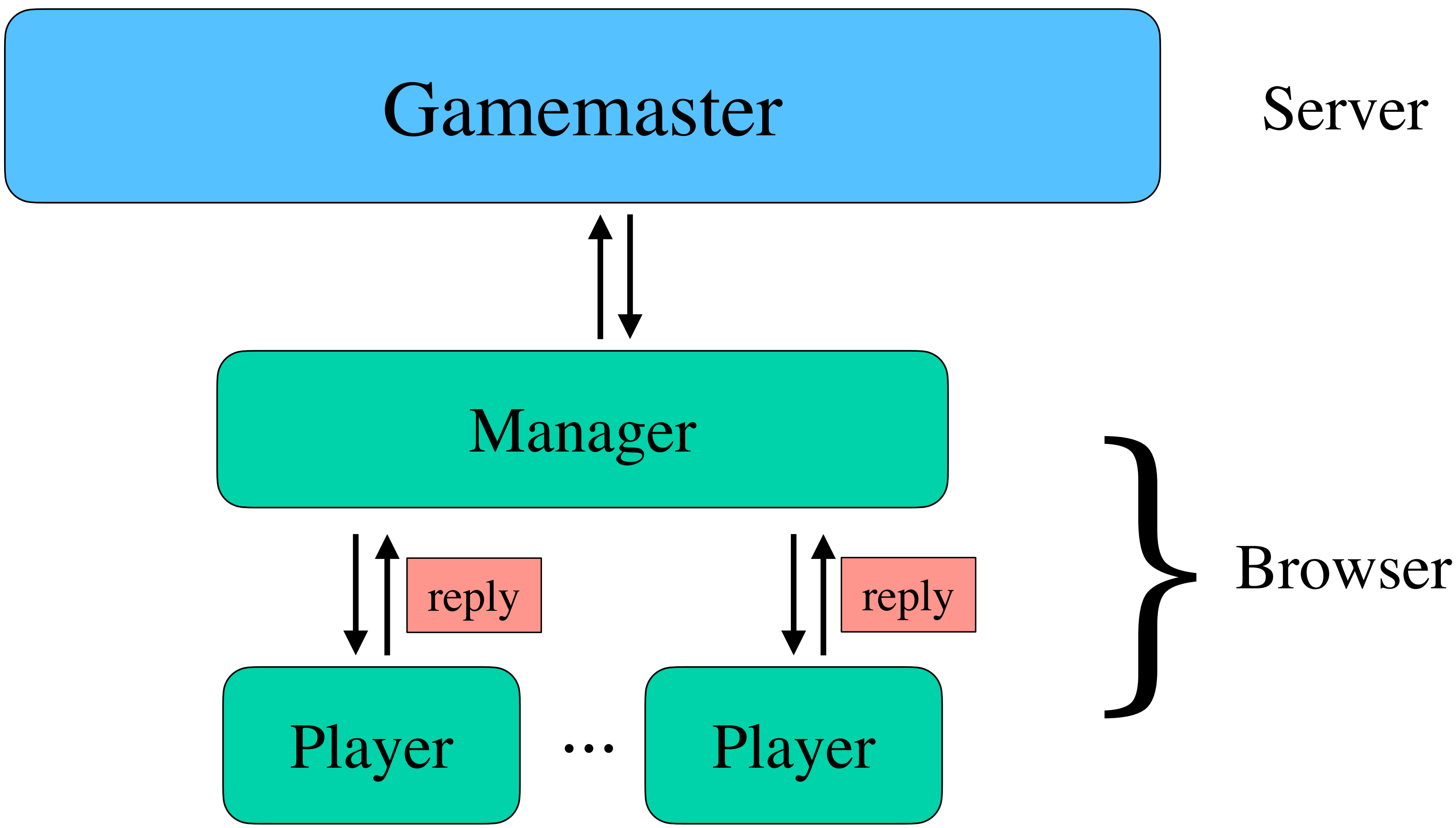
Communication by message passing between manager and players.

Automatic Game Playing



Manager writes messages into browser's local storage.

Automatic Game Playing



Players receive messages, process, write replies into local storage.

Gamemaster

Gamemaster

Not Secure — gamemaster.stanford.edu

Gamemaster

Games Players Metagamers

Matches Leaderboard

Developer About

Games

Not Secure — gamemaster.stanford.edu

Gamemaster

General Game Playing

Game	Information	Standalone	Human	Manager
albuquerque	Information	Standalone	Human	Manager
alquerque	Information	Standalone	Human	Manager
badconnectfour	Information	Standalone	Human	Manager
badqueens	Information	Standalone	Human	Manager
badqueens6x6	Information	Standalone	Human	Manager
badtictactoe	Information	Standalone	Human	Manager
battleofnumbers	Information	Standalone	Human	Manager
bestbuttonsandlights	Information	Standalone	Human	Manager
breakthrough	Information	Standalone	Human	Manager
buttonsandlights	Information	Standalone	Human	Manager
capturetheflag	Information	Standalone	Human	Manager
cbnk	Information	Standalone	Human	Manager
chinesecheckers1	Information	Standalone	Human	Manager
chinesecheckers3	Information	Standalone	Human	Manager
chinook	Information	Standalone	Human	Manager
connectfour	Information	Standalone	Human	Manager
crusade	Information	Standalone	Human	Manager
cryptarithmetic	Information	Standalone	Human	Manager
donttouch	Information	Standalone	Human	Manager
hamilton	Information	Standalone	Human	Manager
hex7x7	Information	Standalone	Human	Manager
hunter	Information	Standalone	Human	Manager
jointbuttonsandlights	Information	Standalone	Human	Manager
knights	Information	Standalone	Human	Manager
knightstour	Information	Standalone	Human	Manager
knightthrough	Information	Standalone	Human	Manager

Description

Not Secure — gamemaster.stanford.edu

Gamemaster

General Game Playing

tictactoe	
Description	Tic Tac Toe (or Noughts and Crosses, Xs and Os) is a game for two players who take turns placing their marks in a 3x3 grid. The first player to place three of his marks in a horizontal, vertical, or diagonal row wins the game.
Rulesheet	(see below)
Stylesheet	(see below)
Ownership	genesereth

Rulesheet:

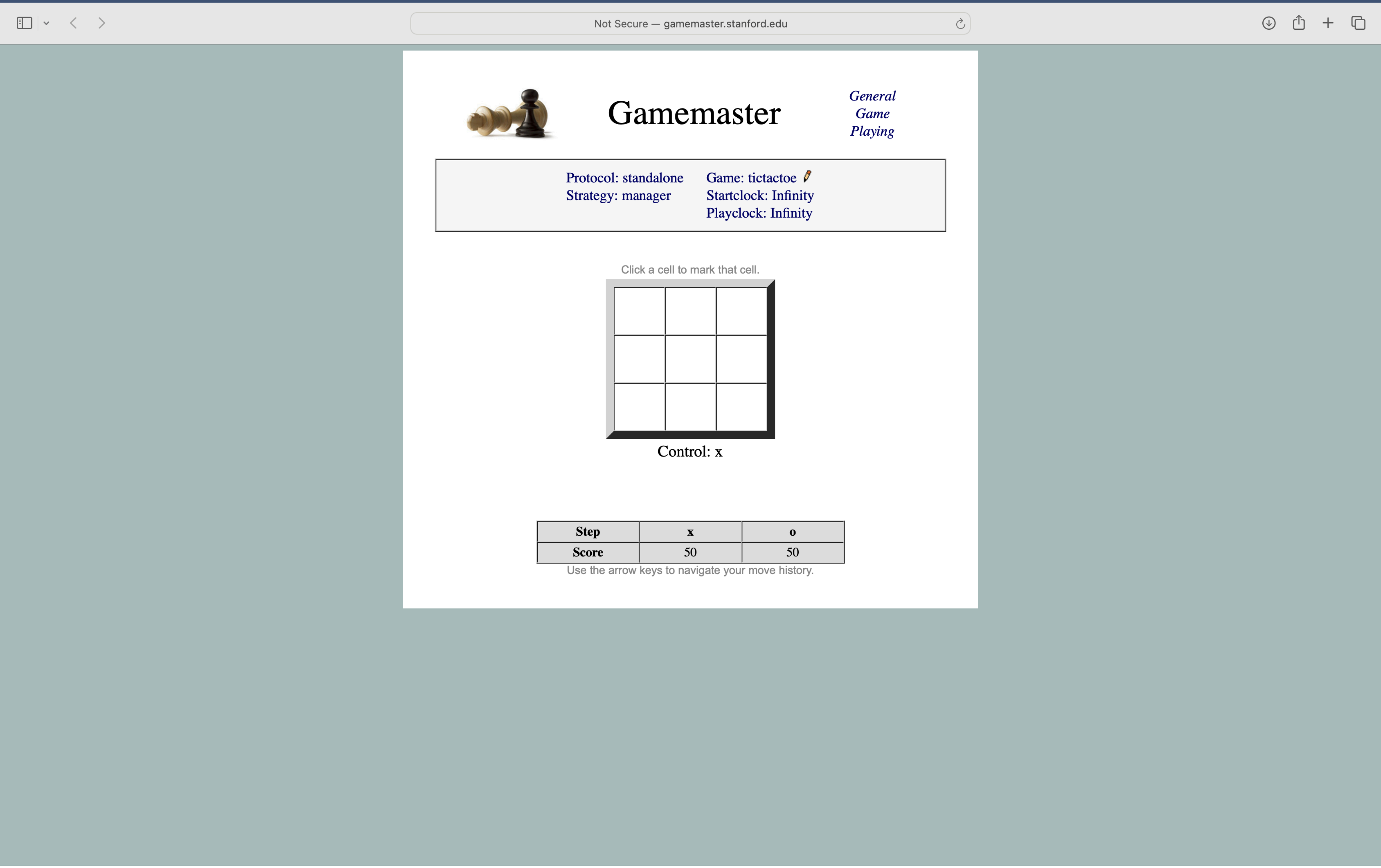
```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% tictactoe
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% metadata
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

role(x)
role(o)

base(cell(M,N,x)) :- index(M) & index(N)
base(cell(M,N,o)) :- index(M) & index(N)
base(cell(M,N,b)) :- index(M) & index(N)
base(control(R))  :- role(R)

action(mark(M,N)) :- index(M) & index(N)

index(1)
index(2)
index(3)
```

Stylesheet:

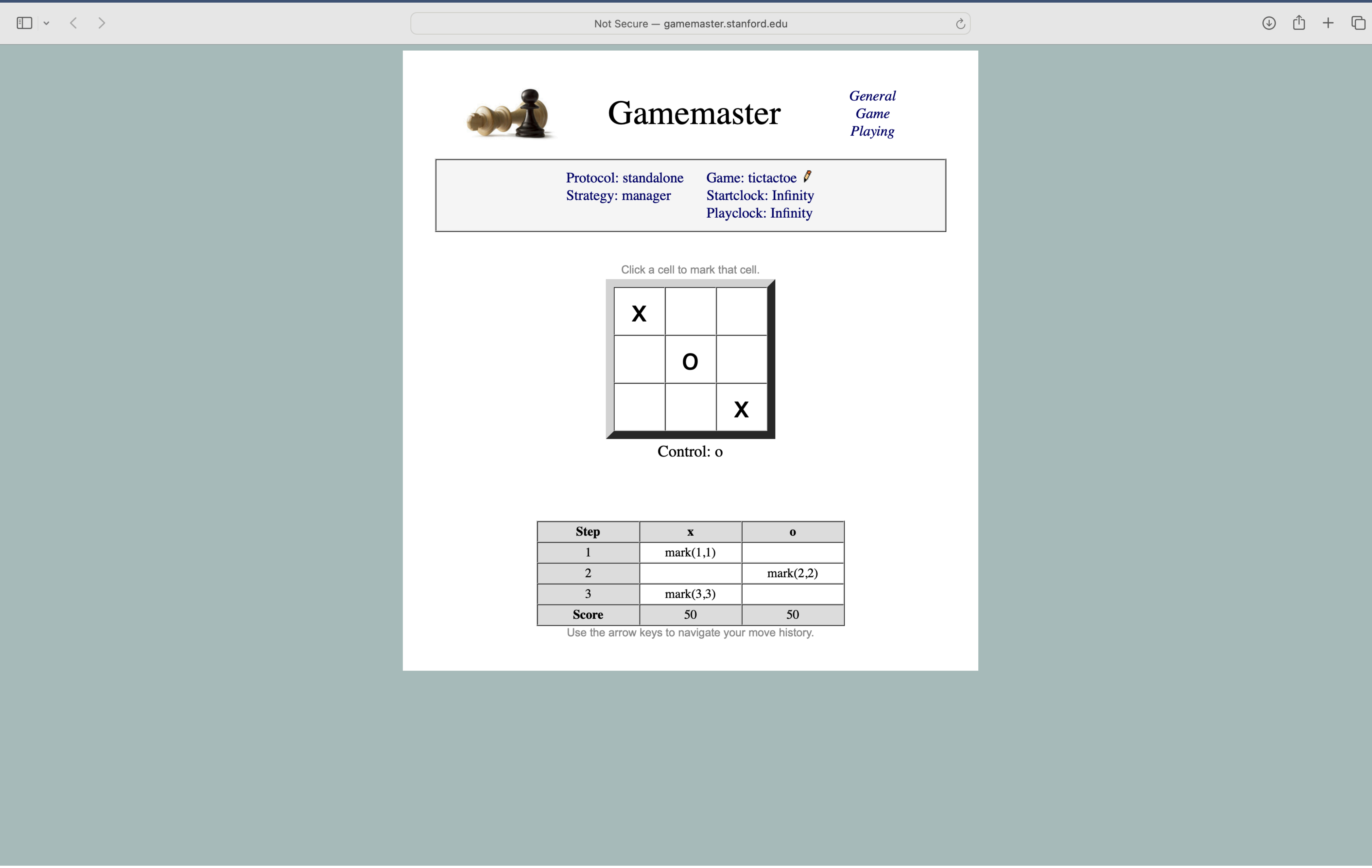
```
//=================================================================================
// tictactoe
//=================================================================================

function rendersituation (state)
 {var role = findcontrol(state,library);
  var table = document.createElement('table');
  table.setAttribute('border','0');
  var row = table.insertRow(table.rows.length);
  var cell = row.insertCell(0);
  cell.setAttribute('align','center');
  cell.setAttribute('style',"font-size:14px;font-family:arial;color:#888888");
  cell.innerHTML = "Click a cell to mark that cell.";
  var row = table.insertRow(table.rows.length);
  var cell = row.insertCell(0);
  cell.setAttribute('align','center');
  cell.appendChild(renderactiveboard(state));
  row = table.insertRow(table.rows.length);
  var cell = row.insertCell(0);
  cell.setAttribute('align','center');
```

Standalone | Human | Manager

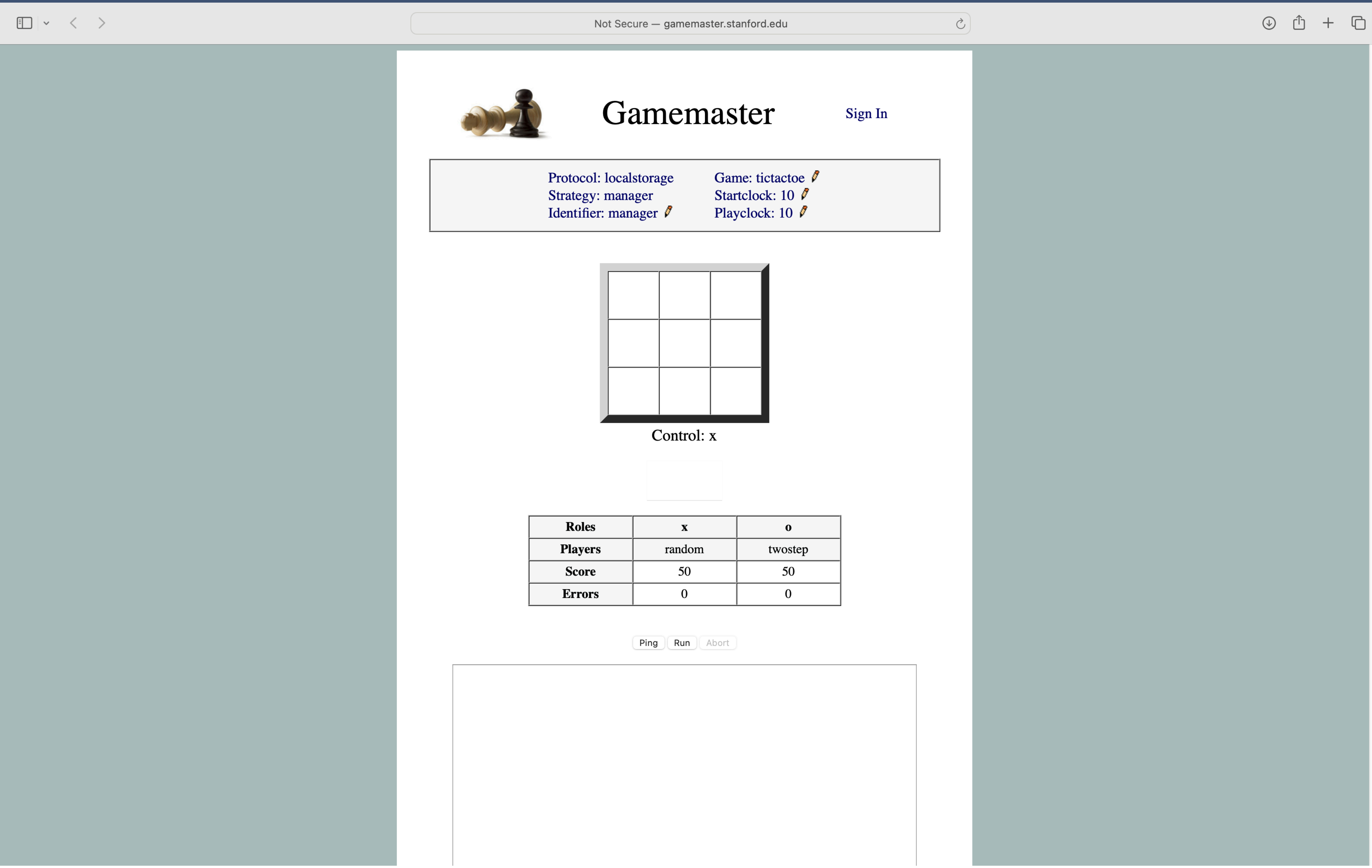
Standalone


Not Secure — gamemaster.stanford.edu
Gamemaster
General Game Playing
Protocol: standalone
Strategy: manager
Game: tictactoe
Startclock: Infinity
Playclock: Infinity
Click a cell to mark that cell.
Control: x
Step
x
o
Score
50
50
Use the arrow keys to navigate your move history.

Standalone


Not Secure — gamemaster.stanford.edu
Gamemaster
General Game Playing
Protocol: standalone
Strategy: manager
Game: tictactoe
Startclock: Infinity
Playclock: Infinity
Click a cell to mark that cell.
X
O
X
Control: o
Step	x	o
1	mark(1,1)	
2		mark(2,2)
3	mark(3,3)	
Score	50	50
Use the arrow keys to navigate your move history.

Manager


Not Secure — gamemaster.stanford.edu
Gamemaster
Sign In
Protocol: localstorage
Strategy: manager
Identifier: manager
Game: tictactoe
Startclock: 10
Playclock: 10
Control: x
Roles x o
Players random twostep
Score 50 50
Errors 0 0
Ping
Run
Abort

Players

Not Secure — gamemaster.stanford.edu

Gamemaster

General Game Playing

Player	Information	Action
bombini_gusini	Information	Launch
busch_light	Information	Launch
caseyhn	Information	Launch
cyclone	Information	Launch
cypress	Information	Launch
dumbo	Information	Launch
egghead	Information	Launch
greedy	Information	Launch
hangman	Information	Launch
hans_magnus	Information	Launch
indy	Information	Launch
jester	Information	Launch
lara	Information	Launch
legal	Information	Launch
maverick	Information	Launch
maximax	Information	Launch
mcs	Information	Launch
mcsid	Information	Launch
mcts	Information	Launch
minimax	Information	Launch
minimaxdepth	Information	Launch
minimaxid	Information	Launch
newplayer	Information	Launch
onestep	Information	Launch
pts	Information	Launch
random	Information	Launch

Manager

Not Secure — ggp.stanford.edu

Gamemaster

egghead
Sign Out

Protocol: manager
Identifier: manager

Game: tictactoe
Startclock: 10
Playclock: 10

X	X	X
	O	
	O	

Game over

Roles	x	o
Players	indy	lara
Score	100	0
Errors	0	0

Clear Begin Pause Resume End

```
message(1649703514,manager,indy,start(1,x,ruleset(role(x),role(o),base(cell(M,N,x))
:- index(M) & index(N),base(cell(M,N,o)) :- index(M) & index(N),base(cell(M,N,b)) :-
index(M) & index(N),base(control(R)) :- role(R),action(mark(M,N)) :- index(M) &
index(N),index(1),index(2),index(3),init(cell(M,N,b)) :- index(M) &
index(N),init(control(x)),legal(mark(M,N)) :- cell(M,N,b),mark(M,N) :: control(R)
==> cell(M,N,R) & ~cell(M,N,b),mark(M,N) :: control(x) ==> ~control(x) &
control(o),mark(M,N) :: control(o) ==> ~control(o) & control(x),goal(x,100) :-
line(x) & ~line(o),goal(x,50) :- line(x) & line(o),goal(x,50) :- ~line(x) &
~line(o),goal(x,0) :- ~line(x) & line(o),goal(o,100) :- ~line(x) &
```

Manager

Not Secure — ggp.stanford.edu

Gamemaster

egghead
Sign Out

Protocol: manager
Identifier: manager

Game: tictactoe
Startclock: 10
Playclock: 10

X	X	X
	O	
	O	

Game over

Roles	x	o
Players	indy	lara
Score	100	0
Errors	0	0

Clear Begin Pause Resume End

```
message(1649703514,manager,indy,start(1,x,ruleset(role(x),role(o),base(cell(M,N,x))
:- index(M) & index(N),base(cell(M,N,o)) :- index(M) & index(N),base(cell(M,N,b)) :-
index(M) & index(N),base(control(R)) :- role(R),action(mark(M,N)) :- index(M) &
index(N),index(1),index(2),index(3),init(cell(M,N,b)) :- index(M) &
index(N),init(control(x)),legal(mark(M,N)) :- cell(M,N,b),mark(M,N) :: control(R)
==> cell(M,N,R) & ~cell(M,N,b),mark(M,N) :: control(x) ==> ~control(x) &
control(o),mark(M,N) :: control(o) ==> ~control(o) & control(x),goal(x,100) :-
line(x) & ~line(o),goal(x,50) :- line(x) & line(o),goal(x,50) :- ~line(x) &
~line(o),goal(x,0) :- ~line(x) & line(o),goal(o,100) :- ~line(x) &
```

Gamemaster

Not Secure — gamemaster.stanford.edu

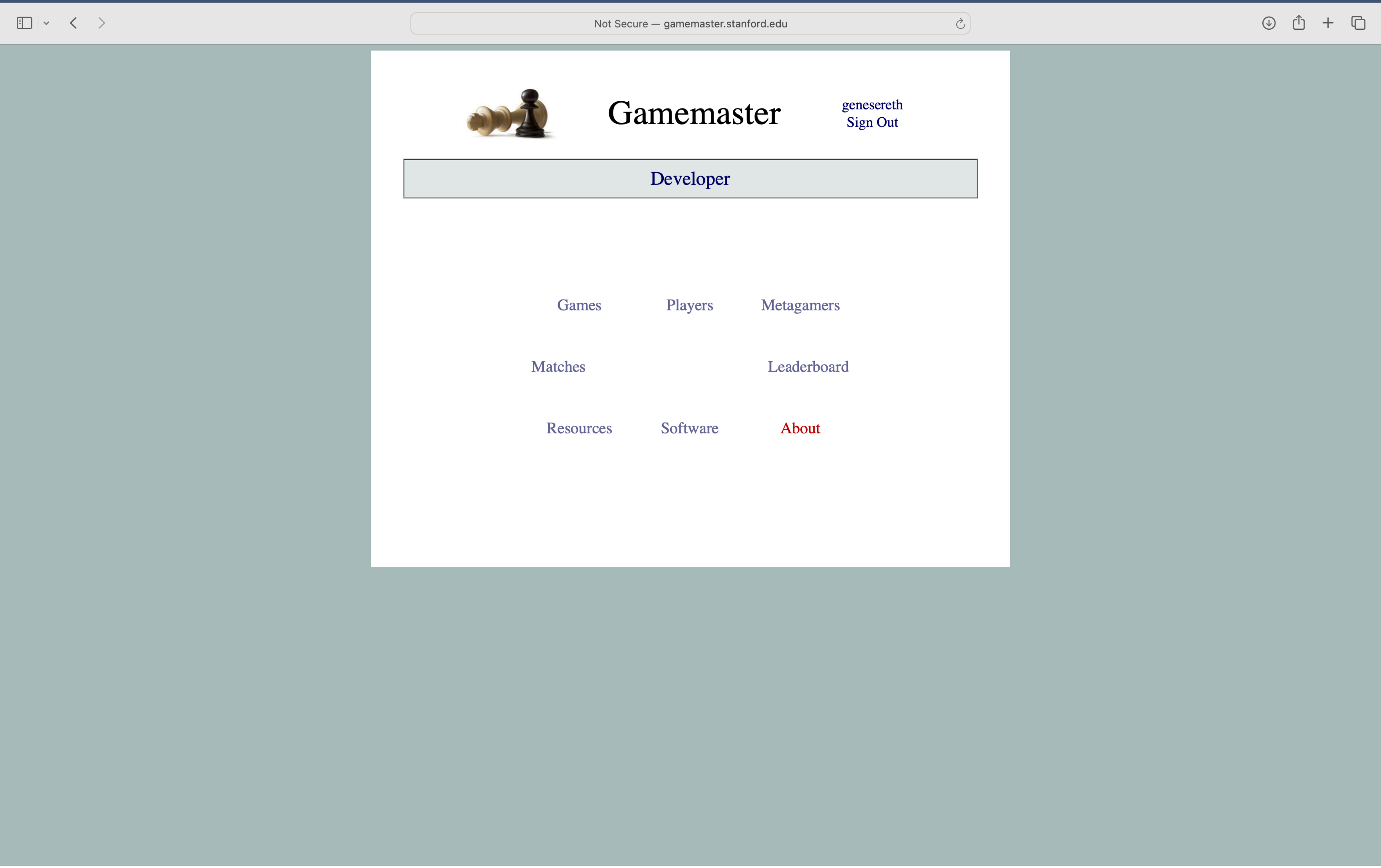
Gamemaster

Games Players Metagamers

Matches Leaderboard

Developer About

Resources


Not Secure — gamemaster.stanford.edu
Gamemaster
genesereth
Sign Out
Developer
Games
Players
Metagamers
Matches
Leaderboard
Resources
Software
About

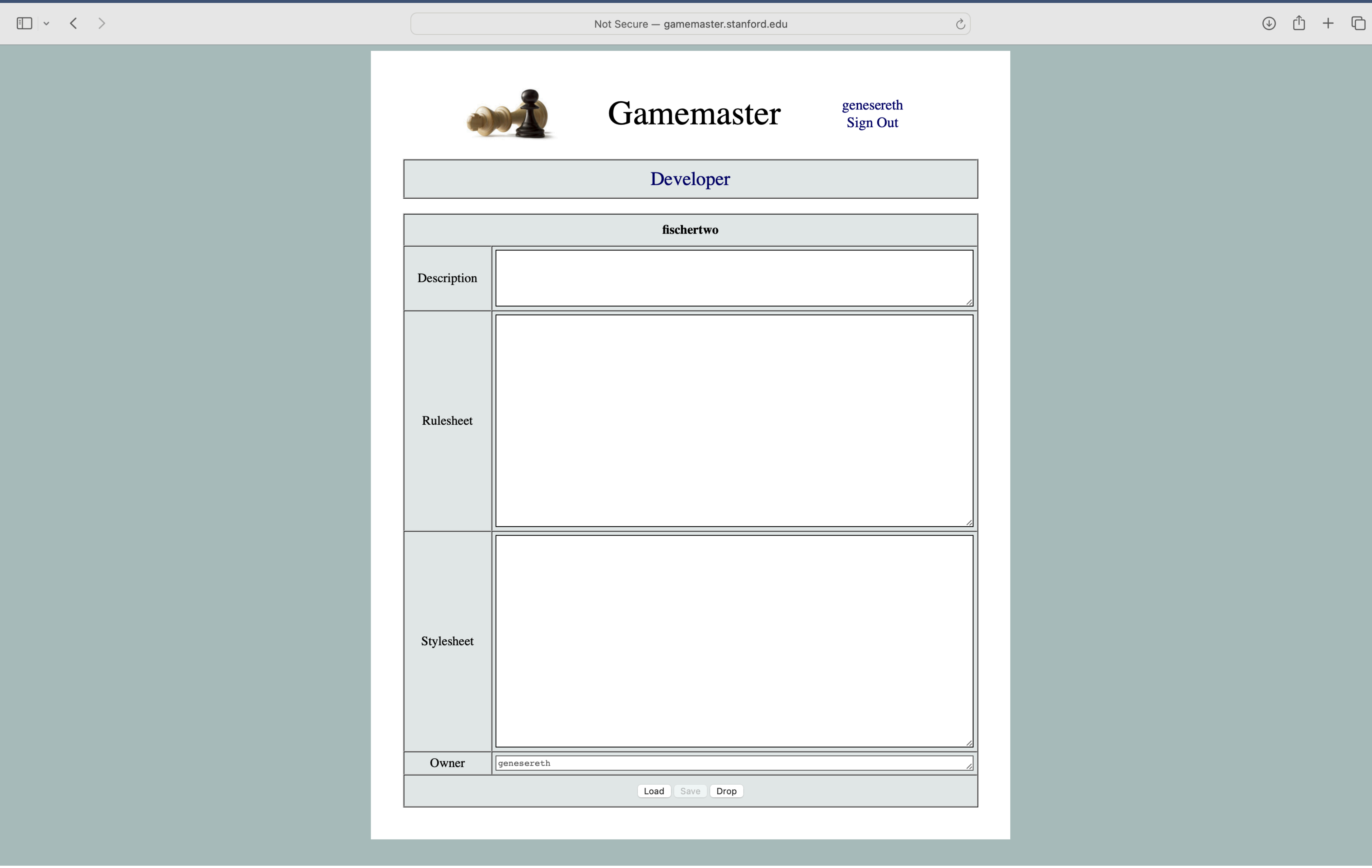
Resources

Not Secure — gamemaster.stanford.edu

multiplesuicide	Standalone	Human	Manager		
multipletictactoe	Standalone	Human	Manager		
newergame	Standalone	Human	Manager		
newestgame	Standalone	Human	Manager		
newgame	Standalone	Human	Manager		
nineboardtictactoe	Standalone	Human	Manager		
parallelbuttonsandlights	Standalone	Human	Manager		
parallelhunter	Standalone	Human	Manager		
parallelknightthrough	Standalone	Human	Manager		
pentago	Standalone	Human	Manager		
pentagosuicide	Standalone	Human	Manager		
queens	Standalone	Human	Manager		
queens6x6	Standalone	Human	Manager		
rainbow	Standalone	Human	Manager		
reversi	Standalone	Human	Manager		
reversum	Standalone	Human	Manager		
simplebuttonsandlights	Standalone	Human	Manager		
simplelightboard	Standalone	Human	Manager		
simpleswitches	Standalone	Human	Manager		
skirmish	Standalone	Human	Manager		
soldiers	Standalone	Human	Manager		
suicide	Standalone	Human	Manager		
sukoshi	Standalone	Human	Manager		
switches	Standalone	Human	Manager		
threepuzzle	Standalone	Human	Manager		
tictactoe	Standalone	Human	Manager		
tictactoe3	Standalone	Human	Manager		
tictactoe4x4	Standalone	Human	Manager		
tictactoe5	Standalone	Human	Manager		
tictactoe7	Standalone	Human	Manager		
trifecta	Standalone	Human	Manager		
ttcc4	Standalone	Human	Manager		
ttt3tp	Standalone	Human	Manager		
utopia	Standalone	Human	Manager		

Add Game

Resources



Not Secure — gamemaster.stanford.edu
Gamemaster
genesereth
Sign Out
Developer
fischertwo
Description
Rulesheet
Stylesheet
Owner
genesereth
Load
Save
Drop

Gardner Project

Martin Gardner in Scientific American

MATHEMATICAL GAMES

Some old and new versions of ticktacktoe, plus the answers to last month's puzzles

by Martin Gardner

Who has not as a child played ticktacktoe, that most ancient and universal struggle of wits of which Wordsworth wrote:

At evening, when with pencil, and smooth slate
In square divisions parcelled out and all
With crosses and with cyphers scribbled o'er,
We schemed and puzzled, head opposed to head
In strife too humble to be named in verse.

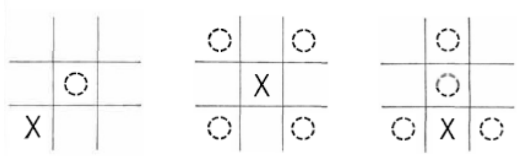
Forms of ticktacktoe were popular in ancient China, Greece and Rome—Ovid mentioned it in his *Art of Love*. At first sight it is not easy to understand the enduring appeal of a game which seems no more than child's play. While it is true that even in the simplest version of the game the number of possible moves is very large—15,120 ($9 \times 8 \times 7 \times 6 \times 5$) different sequences for the first five moves alone—there are really only a few basic patterns, and any astute youngster can become an unbeatable player with only an hour or so of analysis of the game. But ticktacktoe also has its more complex variations and strategic aspects.

In the lingo of game theory, ticktacktoe is a two-person contest which is "finite" (comes to a definite end), has no element of chance and is played with "perfect information," all moves being known to both players. If played "rationally" by both sides, the game must end in a draw. The only chance of winning is to catch an imperfect opponent in a "trap" where a row can be scored on the next move in two ways, only one of which can be blocked.

Of the three possible opening plays—a corner, the center or a side box—the strongest opening is the corner, because the opponent can avoid being trapped at the next move only by one of the eight possible choices: the center. The center opening can be met by seizing one of the four corners. The side opening is in many ways the most interesting, because of its richness in traps on both sides [*see diagrams below*].

A very ancient variant of the game gives each player three counters (one player may use pennies, the other dimes). The two players take turns placing a counter on the board until all six are down. If neither player has won by getting three in a row, each then is allowed to move one counter at a turn to an adjacent empty square, but he can move only vertically or horizontally, not diagonally [*see diagram on page 162*]. The first player has a sure win by placing his first counter in the center box, so this opening is usually barred. After any other opening, the second player must immediately take the center to avoid defeat. This game also ends in a draw with perfect play, but it swarms with potential traps on both sides.

There are variations of the game which permit diagonal moves (one of them attributed to early American Indians). A free-wheeling French version called "*les pendus*" (the hanged) allows any piece to be moved to any vacant



Three openings in ticktacktoe: corner (left), center (middle) and side (right)

Gardner Project

Devised by Vinay Chaudhri

Goal: Encode Gardner's games as rules in GDL

Simple Approach:
  Manual game description

Advanced Approach:
  Develop means to generate the games automatically from some sort of meta-description

Applications of advanced approach:
  software testing
  educational

Variations of Tic Tac Toe

1. Baseline version
2. Each player has three counters that they can place on an empty square. If neither has won by getting three in a row, each is allowed to move one counter at a turn to an adjacent empty square either vertically or horizontally, but not diagonally. The first move cannot be in the center.
3. Same as (2) but allow diagonal moves
4. Les Pendus: Same as (2), but any piece can be moved to any empty cell
5. Same as (2), but the grid size is 4x4, and each player has 4 counters
6. Teeko: Same as (2), but the gride size is 5x5, and each player has 4 counters. The players can move in any direction. The objective is to get four in a row, or assemble them in square formation

Variations of Tic Tac Toe

7. The first person to get three in a row loses
8. Same as 1, but on a 3x3x3 cube
9. Same as 1, but on a 4x4x4 cube
10. Same as 1, but on a 4x4x4x4 hypercube with the objective of getting four marks in a straight-line
11. Same as 1, but on a 5x5x5x5 hypercube
12. Gobang: Game board is 19x19, players have unlimited counters, the objective is to get five in a line horizontal, vertical or diagonal. The pieces cannot be moved once placed
13. Any of the variations when the opponent is an inexperienced player (i.e., not a rational player).

Dimensions of Variation

- Number of cells that can be occupied (3, 4, 5 or unlimited)
- Allow changes to the cells once occupied (3 variations)
- Change the grid size (3, 4, 5 or 19)
- Change the win condition (2 variations)
- Introduce multiple dimensions (3, 4 or 5 dimensions)
- Rational vs irrational play (2 variations)

4x3x4x2x3x2 => 576 variations in total

- Initial state of the board can be non-empty
- Change turn rules (e.g., allow skipping a turn, two moves at a time)

Project

localhost/logicprogramming/stanford/gardner.php

Logic Programming

Profile
Sign Out

Lessons Epilog Sierra Examples Games Forum

Over the years, Martin Gardner wrote dozens of Scientific American articles describing games of various sorts. The link below provides access to Gardner's articles.

Gardner Articles

The goal of the Gardner Project is to formalize the rules of the games described in these articles. You can help out by formalizing the games described in one or more of these articles. For example, you might take the first article (on the game of Hex) and encode some or all of the variants described therein.

Your encodings should be expressed as rule sets in Game Description Language, a specialized version of Epilog. Click the link below to learn more about GDL.

Game Description Language

Once you have encoded the rules for a game, you can publish your encoding in the Gamemaster gaming environment linked below. Gamemaster is an online competition environment for General Game Playing. The homepage has links to games, players, metagamers, matches, a leaderboard, and the Gamemaster developers site. Click on About to learn more about the system.

Gamemaster

To publish your games, use the Gamemaster Developer portal. From the Gamemaster home page, click on Developer, create an account (if necessary), and sign in. Click on Games to see currently available games, click the Add Game link at the bottom of the resulting table, enter information about your game, and Press Save.

Once you have done this, you can use the Gamemaster Development portal to play your games yourselves using the Standalone tool; or you can have automated players play the games using the Manager tool and some of the game players available in Gamemaster, e.g. Random, Twostep, Minimax, Minimaxid, Persistent Tree Search (PTS), or Monte Carlo Tree Search (MCTS).

Feedback

Lessons

Project (Weeks 8 and 9 and 10)

Extra - General Game Playing
Extra - General Game Playing: Killer App for Logic Programming
Extra - Gardner Project
Extra - Gamemaster

Lecture 15 - General Game Playing
Session 16 - No class - Project Consultations
Session 17 - No class - Project Consultations
Session 18 - No class - Project Consultations
Session 19 - No class - Project Evaluations
Session 20 - No class - Project Evaluations

Project Presentation
Project Evaluations
Project Report


GENERAL
GAME
PLAYING