General Game Playing

Game Description

Michael Genesereth
Computer Science Department
Stanford University
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.
Game Description Language (or GDL) is a formal language for encoding the rules of games.

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.
Today = Firehose
Datasets
Conceptualization

**Objects** - e.g. people, companies, cities
- concrete (*person*) or abstract (*number, set, justice*)
- primitive (*computer chip*) or composite (*circuit*)
- real (*earth*) or fictitious (*Sherlock Holmes*)

**Relationships**
- properties of objects or relationships among objects
  - e.g. Joe *is a person*
  - e.g. Joe *is the parent of* Bill
  - e.g. Joe *likes* Bill *more than* Harry
parent(art,bob)
parent(art,bud)
parent(bob,cal)
parent(bob,cam)
parent(bea,cat)
parent(bea,coe)
Constants are strings of lower case letters, digits, underscores, and periods or strings of arbitrary ascii characters within double quotes.

Examples:

joe, bill, cs151, 3.14159
person, worksfor, office
the_house_that_jack_built,
“Mind your p’s & q’s!”

Non-examples:

Art, p&q, the-house-that-jack-built

A set of constants is called a vocabulary.
Types of Constants

Symbols / object constants represent objects.
  joe, bill, harry, a23, 3.14159
  the_house_that_jack_built
  “Mind your p’s & q’s!”

Constructors / function constants represent functions.
  pair, triple, set

Predicates / relation constants represent relations.
  person, parent, prefers
The **arity** of a predicate is the number of arguments that can be associated with the constructor or predicate in writing complex expressions in the language.

**Unary** predicate (1 argument): \( \text{person}(joe) \)

**Binary** predicate (2 arguments): \( \text{parent}(\text{art}, \text{bob}) \)

**Ternary** predicate (3 arguments): \( \text{prefers}(\text{art}, \text{bob}, \text{bea}) \)

In defining vocabulary, we sometimes notate the arity of a constructor or predicate by annotating with a slash and the arity, e.g. \texttt{male/1}, \texttt{parent/2}, and \texttt{prefers/3}. 
A **ground term** is either a *symbol* or a *compound name*.

A **compound name** is an expression formed from an *n*-ary constructor and *n* ground terms enclosed in parentheses and separated by commas.

Symbols:  $a, b$
Constructor:  $f/1, g/1$
Ground terms:  $f(a), f(a), g(a), g(b)$
Ground terms:  $f(f(a)), f(g(a)), g(f(a)), g(g(a))$

The adjective “ground” here means that the term does not contain any “variables” (which we discuss in later lessons).
A **datum / factoid / fact** is an expression formed from an \( n \)-ary predicate and \( n \) ground terms enclosed in parentheses and separated by commas.

Symbols: \( a, b \)

Predicate: \( p/2, q/1 \)

Sample Datum: \( p(a, b) \)
Sample Datum: \( q(a) \)

The **Herbrand base** for a vocabulary is the set of all factoids that can be formed from the vocabulary.
A **dataset** is any set of factoids that can be formed from a vocabulary, i.e. a subset of the Herbrand base.

**Symbols:** $a, b$

**Predicates:** $p/2, q/1$

**Dataset:** \{p(a, b), p(b, a), q(a)\}

**Dataset:** {}  

**Dataset:** \{p(a, a), p(a, b), p(b, a), p(b, b), q(a), q(b)\}

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.
Vocabulary

Symbols: a, b
Predicates: p/2, q/1

Questions

How many elements in the Herbrand base?
How many possible datasets?
Vocabulary

Symbols: $a, b$
Constructors: $f, g$
Predicates: $p/2, q/1$

Questions

How many elements in the Herbrand base?
How many possible datasets?
Logic Programs
A logic program is a collection of logical rules that define relations on data.

\[ \begin{align*}
  p(a,b) \\
  p(b,c) \\
  p(c,d) \\
  p(d,c) \\
  + \\
  r(X,Y) & :- p(X,Y) \& p(Y,X) \\
  = \\
  r(c,d) \\
  r(d,c)
\end{align*} \]
A **constant** is a string of lower case letters, digits, underscores, and periods *or* a string of ascii characters within double quotes.

```
joe, bill, cs151, 3.14159
person, worksfor, office
the_house_that_jack_built,
“Mind your p’s & q’s!”
```

A **variable** is either a lone underscore or a string of letters, digits, underscores beginning with an upper case letter.

```
X   Y23   Somebody   _
```
Terms

Symbols
  art
  bob

Variables
  X
  Y23

Compound Terms
  pair(art,bob)
  pair(X,Y23)
  pair(pair(art,bob),pair(X,Y23))
Atoms

\( p(a, b) \)
\( p(a, x) \)
\( p(Y, c) \)

Negations

\( \neg p(a, b) \)

Literals (atoms or negations of atoms)

\( p(a, Y) \)
\( \neg p(a, Y) \)

An atom is a *positive literal*.
A negations is a *negative literal*. 
$r(X,Y) :- p(X,Y) \& \neg q(Y)$
An **instance of a rule** is a rule in which all variables have been consistently replaced by ground terms.

Rule

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

Ground Terms

\{a, b\}

Instances

\[ r(a,a) :- p(a,a) \land \neg q(a) \]
\[ r(a,b) :- p(a,b) \land \neg q(b) \]
\[ r(b,a) :- p(b,a) \land \neg q(a) \]
\[ r(b,b) :- p(b,b) \land \neg q(b) \]
The result of applying a rule to a dataset is defined to be the set of all $\psi$ such that

(1) $\psi$ is the head of an instance of the rule,

(2) every positive subgoal in the instance is in the dataset,

(3) no negative subgoal is in the dataset.
Example

Dataset
- p(a, b)
- p(b, c)
- p(c, d)
- p(d, c)

Result
- r(c, d)
- r(d, c)

Rule
- \( r(X, Y) :- p(X, Y) \land p(Y, X) \)

Positive instances (2)
- \( r(c, d) :- p(c, d) \land p(d, c) \)
- \( r(d, c) :- p(d, c) \land p(c, d) \)

Negative instances (14)
- \( r(a, b) :- p(a, b) \land p(b, a) \)
- \( r(b, c) :- p(b, c) \land p(b, a) \)

...
Example

Dataset

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>p(a,b)</td>
<td>r(a,b)</td>
</tr>
<tr>
<td>p(b,c)</td>
<td>r(b,c)</td>
</tr>
<tr>
<td>p(c,d)</td>
<td></td>
</tr>
<tr>
<td>p(d,c)</td>
<td></td>
</tr>
</tbody>
</table>

Result

Rule

\[ r(X, Y) \leftarrow p(X, Y) \land \neg p(Y, X) \]

Positive instances (2)

\[ r(a, b) \leftarrow p(a, b) \land \neg p(b, a) \]
\[ r(b, c) \leftarrow p(b, c) \land \neg p(c, b) \]

Negative instances (14)

\[ r(a, c) \leftarrow p(a, c) \land \neg p(c, a) \]
\[ r(c, d) \leftarrow p(c, d) \land \neg p(d, c) \]
...
A **ruleset** is a finite set of rules.

\[
\begin{align*}
  r(X, Y) & : \ p(X, Y) \ & q(X) \\
  r(X, Y) & : \ p(X, Y) \ & \neg q(Y)
\end{align*}
\]

Note that there can be more than one rule defining a relation.
Using this notion, we define the closure of a ruleset on a dataset to be the result of

(1) **applying** the rules in the ruleset to facts in the dataset,

(2) **adding** the results to the dataset, and

(3) **repeating** until nothing new is added.
Identity

\text{same}(t_1,t_2) \text{ is true iff } t_1 \text{ and } t_2 \text{ are identical}
\text{distinct}(t_1,t_2) \text{ is true iff } t_1 \text{ and } t_2 \text{ are different}

Examples

\text{same}(a,a) \text{ is true}
\text{same}(a,b) \text{ is false}
\text{distinct}(a,a) \text{ is false}
\text{distinct}(a,b) \text{ is true}

NB: This is \textbf{not} ordinary equality (e.g. 2+2 = 4)

\text{same}(\text{plus}(2,2),4) \text{ is false}
\text{distinct}(\text{plus}(2,2),4) \text{ is true}
Dataset
   p(a,b)
   p(a,c)
   p(b,d)
   p(d,c)

Example
   divergence(X,L) :-
      p(X,Y) &
      countofall(Z,p(X,Z),L)

Result
   divergence(a,2)
   divergence(b,1)
   divergence(d,1)
Grandparents:

\[
\text{grandparent}(X,Z) :\text{ parent}(X,Y) \& \text{ parent}(Y,Z)
\]

Data:

\[
\begin{align*}
\text{parent}(\text{art},\text{bob}) \\
\text{parent}(\text{art},\text{bea}) \\
\text{parent}(\text{bob},\text{cal}) \\
\text{parent}(\text{bob},\text{cam}) \\
\text{parent}(\text{bea},\text{cat}) \\
\text{parent}(\text{bea},\text{coe})
\end{align*}
\]

View:

\[
\begin{align*}
\text{grandparent}(\text{art},\text{cal}) \\
\text{grandparent}(\text{art},\text{cam}) \\
\text{grandparent}(\text{art},\text{cat}) \\
\text{grandparent}(\text{art},\text{coe})
\end{align*}
\]
Ancestors:

\[
\text{ancestor}(X,Z) :- \text{parent}(X,Z) \\
\text{ancestor}(X,Z) :- \text{parent}(X,Y) \land \text{ancestor}(Y,Z)
\]

Data:  
\[
\begin{align*}
\text{parent}(\text{art}, \text{bob}) \\
\text{parent}(\text{art}, \text{bea}) \\
\text{parent}(\text{bob}, \text{cal}) \\
\text{parent}(\text{bob}, \text{cam}) \\
\text{parent}(\text{bea}, \text{cat}) \\
\text{parent}(\text{bea}, \text{coe})
\end{align*}
\]

View:  
\[
\begin{align*}
\text{ancestor}(\text{art}, \text{bob}) \\
\text{ancestor}(\text{art}, \text{bea}) \\
\text{ancestor}(\text{bob}, \text{cal}) \\
\text{ancestor}(\text{bob}, \text{cam}) \\
\text{ancestor}(\text{bea}, \text{cat}) \\
\text{ancestor}(\text{bea}, \text{coe}) \\
\text{ancestor}(\text{art}, \text{cal}) \\
\text{ancestor}(\text{art}, \text{cam}) \\
\text{ancestor}(\text{art}, \text{cat}) \\
\text{ancestor}(\text{art}, \text{coe})
\end{align*}
\]
**Example**

**Personhood:**

\[
\begin{align*}
\text{person}(X) &: \text{ parent}(X,Y) \\
\text{person}(Y) &: \text{ parent}(X,Y)
\end{align*}
\]

**Data:**

- parent(art,bob)
- parent(art,bea)
- parent(bob,cal)
- parent(bob,cam)
- parent(bea,cat)
- parent(bea,coe)

**View:**

- person(art)
- person(bob)
- person(cal)
- person(cam)
- person(bea)
- person(coe)
Childlessness:

\[
\text{childless}(X) : - \\
\quad \text{person}(X) \ & \\
\quad \text{countofall}(Y, \text{parent}(X, Y), 0)
\]

Data:

- parent(art, bob)
- parent(art, bea)
- parent(bob, cal)
- parent(bob, cam)
- parent(beam, cat)
- parent(beam, coe)

View:

- childless(cal)
- childless(cam)
- childless(cat)
- childless(coe)
At Least Two Children:

\[
\text{atleasttwo}(X) :- \\
\quad \text{parent}(X,Y) \land \text{parent}(X,Z) \land \text{distinct}(Y,Z)
\]

Data:

- parent(art,bob)
- parent(art,bea)
- parent(art,ben)
- parent(bob,cam)
- parent(bea,cat)
- parent(bea,coe)

View:

- atleasttwo(art)
- atleasttwo(bea)
Exactly Two Children:

```
exactlytwo(X) :-
    parent(X,Y) &
    evaluate(countofall(Z,parent(X,Z)),2)
```

Data:
- parent(art,bob)
- parent(art,bea)
- parent(art,ben)
- parent(bob,cam)
- parent(bea,cat)
- parent(bea,coe)

View:
- exactlytwo(bea)
A rule is *safe* if and only if every variable in the head and every variable in any negative subgoal appears in some positive subgoal in the body.

Safe Rule:
\[
x(X, Z) :- p(X, Y) \land q(Y, Z) \land \neg r(X, Y)
\]

Unsafe Rule:
\[
x(X, Z) :- p(X, Y) \land q(Y, X)
\]

Unsafe Rule:
\[
x(X, Y) :- p(X, Y) \land \neg q(Y, Z)
\]
A **semipositive** ruleset is one in which negations apply only to base relations, i.e. there are no subgoals with negated views.

Base Relations: \{m, n, p\}
View Relations: \{q, r\}

Example:

\[
\begin{align*}
  r(X) & : - p(X,Y) \& q(Y) \\
  q(Y) & : - m(Y,Z) \& \neg n(Z)
\end{align*}
\]

Non-Example:

\[
\begin{align*}
  r(X) & : - p(X,Y) \& \neg q(Y) \\
  q(Y) & : - m(Y,Z) \& \neg n(Z)
\end{align*}
\]

See article and book on Logic Programming for semantics of non-semipositive rulesets.
Dynamic Logic Programs
Views

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

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

View

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

Operation

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

\[
t=1 \\
t=2 \\
\]
Operations

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

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

View

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

View

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

Operation

\[ t=1 \quad \rightarrow \quad t=2 \]
Operation constants represent operations.

- **stack** - place one block on another
- **mark** - place a specific mark in a row and a column

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

- 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:**

- 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

\[ c(X) :: p(X,Y) \land q(X) \implies \neg q(X) \land c(Y) \]
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**

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

A dynamic logic program is a collection of view definitions and operation definitions.
A operation rule is **safe** if and only if every variable in every literal on the right hand side appears 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**

\[
c(X) ::
\]

\[
p(X, Y) & \neg q(X) \implies
\]

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

**Unsafe Operation Rule**

\[
c(X) ::
\]

\[
p(X, Y) & \neg q(Z) \implies
\]

\[
\neg p(X, Y) & q(W) & c(Y)
\]
Active and Inactive Rule Instances

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 $\Gamma$ and
2. the conditions of the rule are all true in $\Delta$.

Otherwise, the instance is inactive.
Dataset: \{p(a), p(b), p(c), q(a), q(b), q(c), r(b)\}

**Action Definition:**

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

**Actions:** \{u(a), u(b)\}

**Active Instance:**

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

**Inactive Instances:**

\[ u(b) ::= p(b) \land q(b) \land \neg r(b) \implies \neg p(b) \land r(b) \]
\[ u(c) ::= p(c) \land q(c) \land \neg r(c) \implies \neg p(c) \land r(c) \]
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 set with respect to a rule set are the positive effects in the expansion.

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

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

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

**Actions:** \{u(a), u(b)\}

**Active Instance:**
\[
u(a) :: p(a) & q(a) & \neg r(a) \implies \neg p(a) & r(a)
\]

**Expansion:** \{\neg p(a), r(a)\}

**Negative Updates:** \{p(a)\}

**Positive Updates:** \{r(a)\}
Given a rule set, the result of applying an action to a dataset is the dataset that results from *deleting all of the negative effects* of the action from the dataset and *adding in all of the positive effects*. 
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.

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

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

Actions: \{u(a), u(b)\}

Expansion: \{\neg p(a), r(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)\}

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

**Action:** \(u(a)\)

**Expansion:** \{\neg r(a), r(a)\}

**Negative Updates:** \{r(a)\}

**Positive Updates:** \{r(a)\}

**Result:** \{p(a), p(b), p(c), q(a), q(b), q(c), r(a)\}
Game Description
Structured State Machine

The diagram illustrates a state transition system with the following states and transitions:

- States: 0, 25, 50, 100
- Transitions:
  - From 0 to 50 on input $f(a)$
  - From 50 to 100 on input $f(a)$
  - From 0 to 25 on input $f(b)$
  - From 25 to 50 on input $f(b)$
  - From 50 to 100 on input $f(b)$

States are labeled as $q(a,b)$, $p(a)$, $q(a,b)$, and $q(b,a)$.
Game-Independent Vocabulary

Relation Constants:

- `role(role)`
- `base(proposition)`
- `action(action)`
- `init(proposition)`
- `legal(action)`
- `goal(role,number)`
- `terminal`

Object Constants:

- `0, 1, 2, 3, ... , 100 - numbers (i.e. entities)`
Object constants:
  Roles: \(x, o\)
  Entities in propositions and actions: \(a, b, c\)

Relation Constants:
  Helper relations: \(r, s\)
Propositions and actions are terms in GDL.

Object constants: \( a, b, c \)
Relations: \( p, q \)
Operations: \( f, g \)

Actions are treated as terms.

Logic program rules say which propositions are true and which actions are performed (as shown in what follows) and specify the results of performing actions in states.
A game description in GDL is a dynamic logic program using GDL’s game-independent and the game’s game-specific vocabulary subject to the following constraints.

Inputs: state description

Outputs: role, base, action, init, legal, goal, terminal

(1) role, base, action, init - independent of state
(2) legal, goal, terminal - depend on state

Actions
States

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)
Game-Specific Vocabulary

Objects:
  x, o - roles
  1, 2, 3 - indices of rows and columns
  b - blank

Relations:
  cell(index,index,mark) --> proposition
  control(role) --> proposition
  row(index,mark)
  column(index,mark)
  diagonal(mark)
  line(mark)
  open

Operation:
  mark(index,index)
\text{role}(x) \\
\text{role}(o)
Propositions

\texttt{base(cell(X,Y,W)) :-}
\texttt{\hspace{1em} index(X) \&}
\texttt{\hspace{1em} index(Y) \&}
\texttt{\hspace{1em} filler(W)}

\texttt{base(control(W)) :-}
\texttt{\hspace{1em} role(W)}

\texttt{index(1)}
\texttt{index(2)}
\texttt{index(3)}

\texttt{filler(x)}
\texttt{filler(o)}
\texttt{filler(b)}
\textbf{Actions}

\begin{verbatim}
action(mark(X,Y)) :- index(X) & index(Y)
\end{verbatim}
\textbf{Initial State}

\begin{verbatim}
init(cell(1,1,b))
init(cell(1,2,b))
init(cell(1,3,b))
init(cell(2,1,b))
init(cell(2,2,b))
init(cell(2,3,b))
init(cell(3,1,b))
init(cell(3,2,b))
init(cell(3,3,b))
init(control(x))
\end{verbatim}
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)
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)
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)
Goals and Termination

\[
\text{goal}(x, 100) :\text{=} \text{line}(x) \\
\text{goal}(x, 50) :\text{=} \sim\text{line}(x) \& \sim\text{line}(o) \\
\text{goal}(x, 0) :\text{=} \text{line}(o) \\
\text{goal}(o, 100) :\text{=} \text{line}(o) \\
\text{goal}(o, 50) :\text{=} \sim\text{line}(x) \& \sim\text{line}(o) \\
\text{goal}(o, 0) :\text{=} \text{line}(x)
\]

\[
\text{terminal} :\text{=} \text{line}(W) \\
\text{terminal} :\text{=} \sim\text{open}
\]
Termination

A game description in GDL terminates if and only if every sequence of legal moves from the initial state reaches a terminal state after a finite number of steps.
A game is playable if and only some player has control in every non-terminal state and the player in control has at least one legal move in every non-terminal state.

Note that in chess, if a player cannot move, it is a stalemate. Fortunately, this is a terminal state.

*In GGP, we guarantee that every game is playable.*
A game is strongly winnable if and only if, for some player, there is a sequence of individual moves of that player that leads to a goal state in which that player gets 100 points.

A game is weakly winnable if and only if, for every player, there is some sequence of moves of the players that leads to a goal state in which that player gets more than 0 points.

In GGP, every game is weakly winnable, and all single player games are strongly winnable.
What we see:

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

What the player sees:

\[
\text{welcoul}(M,N) ::
\quad \text{control}(Z) \implies \neg \text{dupse}(M,N,erol) \land \text{dupse}(M,N,Z)
\]
Game Resources
Sierra is browser-based IDE (interactive development environment) for Epilog.

- Saving and loading files
- Visualization of datasets
- Querying datasets
- Transforming datasets
- Interpreter (for view definitions, action definitions)
- Trace capability (useful for debugging rules)
- Analysis tools (error checking and optimizing rules)

http://epilog.stanford.edu/homepage/sierra.php
<table>
<thead>
<tr>
<th>File</th>
<th>Dataset</th>
<th>Channel</th>
<th>Ruleset</th>
<th>Operation</th>
<th>Settings</th>
</tr>
</thead>
</table>


\[ p(c,d) \]
\[ p(a,b) \]
\[ p(b,c) \]
\[
p(c, d) \\
p(a, b) \\
p(b, c) \\
\]
\[ p(a, b) \\
p(b, c) \\
p(c, d) \]
\[ p(a, b) \]
\[ p(b, c) \]
\[ p(c, d) \]
Syntax error.
\( p(a, b) \)
\( p(b, c) \)
\( p(c, d) \)
\[ p(a,b) \]
\[ p(b,c) \]
\[ p(c,d) \]
New Dataset
<table>
<thead>
<tr>
<th>File</th>
<th>Dataset</th>
<th>Channel</th>
<th>Ruleset</th>
<th>Operation</th>
<th>Settings</th>
</tr>
</thead>
</table>

Load Configuration
http://gamemaster.stanford.edu/homepage/rulechecker.php
http://gamemaster.stanford.edu/homepage/gamechecker.php
Gamemaster

Protocol: standalone

Game: tictactoe

X O X

Control: o

Move:  

http://gamemaster.stanford.edu/extras/standalone.html