General Game Playing

Game Description

Michael Genesereth
Computer Science Department
Stanford University
State Machine
Structured Actions and States

0 / 0

\( q(a,b) \) \hspace{2cm} \( p(a) q(a,b) \) \hspace{2cm} \( p(b) q(a,b) \) \hspace{2cm} \( p(a) p(b) q(a,b) \)

\( f(a) \) \hspace{2cm} \( f(b) \) \hspace{2cm} \( f(b) \) \hspace{2cm} \( f(a) \)

0 / 0 50 / 50 50 / 50 100 / 0

\( q(b,a) \) \hspace{2cm} \( p(a) q(b,a) \) \hspace{2cm} \( p(b) q(b,a) \) \hspace{2cm} \( p(b) q(b,a) \)

\( f(a) \) \hspace{2cm} \( f(b) \) \hspace{2cm} \( f(b) \) \hspace{2cm} \( f(b) \)

0 / 0 25 / 25 25 / 25 0 / 100
Logical Encoding

\[
\text{init} \text{(cell}(1,1,b))
\]
\[
\text{init} \text{(cell}(1,2,b))
\]
\[
\text{init} \text{(cell}(1,3,b))
\]
\[
\text{init} \text{(cell}(2,1,b))
\]
\[
\text{init} \text{(cell}(2,2,b))
\]
\[
\text{init} \text{(cell}(2,3,b))
\]
\[
\text{init} \text{(cell}(3,1,b))
\]
\[
\text{init} \text{(cell}(3,2,b))
\]
\[
\text{init} \text{(cell}(3,3,b))
\]
\[
\text{init} \text{(control}(x))
\]

\[
\text{legal} \text{(P,mark}(X,Y)) :-
\]
\[
\text{true} \text{(cell}(X,Y,b)) \&
\]
\[
\text{true} \text{(control}(P))
\]

\[
\text{legal} \text{(x,noop)} :-
\]
\[
\text{true} \text{(control}(o))
\]

\[
\text{legal} \text{(o,noop)} :-
\]
\[
\text{true} \text{(control}(x))
\]

\[
\text{next} \text{(cell}(M,N,P)) :-
\]
\[
\text{does} \text{(P,mark}(M,N))
\]

\[
\text{next} \text{(cell}(M,N,Z)) :-
\]
\[
\text{does} \text{(P,mark}(M,N)) \&
\]
\[
\text{true} \text{(cell}(M,N,Z)) \& Z#b
\]

\[
\text{next} \text{(cell}(M,N,b)) :-
\]
\[
\text{does} \text{(P,mark}(M,N)) \&
\]
\[
\text{true} \text{(cell}(M,N,b)) \& (M#J \mid N#K)
\]

\[
\text{legal} \text{(control}(x)) :-
\]
\[
\text{true} \text{(control}(o))
\]

\[
\text{next} \text{(control}(x)) :-
\]
\[
\text{true} \text{(control}(o))
\]

\[
\text{next} \text{(control}(o)) :-
\]
\[
\text{true} \text{(control}(x))
\]

\[
\text{line}(P) :-
\]
\[
\text{true} \text{(cell}(M,1,P)) \&
\]
\[
\text{true} \text{(cell}(M,2,P)) \&
\]
\[
\text{true} \text{(cell}(M,3,P))
\]

\[
\text{row}(M,P) :-
\]
\[
\text{true} \text{(cell}(M,1,P)) \&
\]
\[
\text{true} \text{(cell}(M,2,P)) \&
\]
\[
\text{true} \text{(cell}(M,3,P))
\]

\[
\text{column}(N,P) :-
\]
\[
\text{true} \text{(cell}(1,N,P)) \&
\]
\[
\text{true} \text{(cell}(2,N,P)) \&
\]
\[
\text{true} \text{(cell}(3,N,P))
\]

\[
\text{diagonal}(P) :-
\]
\[
\text{true} \text{(cell}(1,1,P)) \&
\]
\[
\text{true} \text{(cell}(1,2,P)) \&
\]
\[
\text{true} \text{(cell}(1,3,P))
\]

\[
\text{diagonal}(P) :-
\]
\[
\text{true} \text{(cell}(1,1,P)) \&
\]
\[
\text{true} \text{(cell}(2,2,P)) \&
\]
\[
\text{true} \text{(cell}(3,3,P))
\]

\[
\text{terminal} :- \text{line}(P)
\]
\[
\text{terminal} :- \sim \text{open}
\]

\[
\text{goal}(x,100) :- \text{line}(x)
\]
\[
\text{goal}(x,50) :- \text{draw}
\]
\[
\text{goal}(x,0) :- \text{line}(o)
\]

\[
\text{goal}(o,100) :- \text{line}(o)
\]
\[
\text{goal}(o,50) :- \text{draw}
\]
\[
\text{goal}(o,0) :- \text{line}(x)
\]

\[
\text{line}(P) :- \text{row}(M,P)
\]
\[
\text{line}(P) :- \text{column}(N,P)
\]
\[
\text{line}(P) :- \text{diagonal}(P)
\]

\[
\text{open} :- \text{true}(\text{cell}(M,N,b))
\]
\[
\text{draw} :- \sim\text{line}(x) \&
\]
\[
\sim\text{line}(o)
\]
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 is a specialization of a logic programming language called Epilog.
Datasets

Virtual Dataset

Base Dataset

Virtual Dataset

Operation

View

g(a,c)  
g(b,d)

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

t=1

View

g(c,a)  
g(d,b)

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

t=2
View Definitions

Virtual Dataset

Base Dataset

Virtual Dataset

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

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

G(a, c)
G(b, d)

G(c, a)
G(d, b)

Operation

t=1

Base Dataset

Virtual Dataset

t=2
Operation Definitions

Virtual Dataset

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

View

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

Base Dataset

Action

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

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)
Constants are strings of lower case letters, digits, underscores, and periods.

Examples:
   a, b, c, 1, 2, 3, king, white_knight
   p, q, r, cell, adjacent

Non-examples:
   Cell, white-knight
Types of Constants

Object constants represent objects.
    a, b, c, 1, 2, 3, king, white_knight

Function constants represent functions.
    f, g, piece, list, set

Relation constants represent relations.
    p, q, r, cell, adjacent
A **fact / factoid / datum** is an expression formed from a predicate and $n$ symbols enclosed in parentheses and separated by commas.

Symbols: $a, b$
Predicate: $p, q$

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.

Symbols:  $a, b$
Predicate: $p, q$

**Herbrand Base:**

$$\{p(a,a), p(a,b), p(b,a), p(b,b), q(a), q(b)\}$$
A **dataset** is any set of factoids that can be formed from a vocabulary, i.e. a *subset of the Herbrand base*.

**Herbrand Base:**

\[
\{ p(a,a), p(a,b), p(b,a), p(b,b), q(a), q(b) \}
\]

**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.
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.
We sometimes want to talk about complex objects made up of simpler objects.

Examples:
- the list of a, b, and c
- the cell in row 2 and column 3

Solution: *Compound names:*
- list(a, b, c)
- piece(white, knight)
Types of Constants

Object constants represent objects.
   a, b, c, 1, 2, 3, king, white_knight

Function constants
   f, g, piece, list, set

Relation constants represent relations.
   p, q, r, cell, adjacent
A **compound name** is an expression formed from a *constructor* and *n symbols* enclosed in parentheses and separated by commas.

Symbols:  a, b  
Constructor:  f, g  

Compound Names:  f(a,b), f(b,a), g(a), g(b)

*This allows us to refer to complex objects made up of simple objects. How do we refer to complex objects made up of other complex objects?*
A **compound name** is an expression formed from a *constructor* and *n symbols or compound names* enclosed in parentheses and separated by commas.

Symbols:  \( a, b \)
Constructor:  \( f, g \)

Compound Names:  \( f(a,b), f(b,a), g(a), g(b) \)
Compound Names:  \( f(g(a),b), g(f(a,b)) \)
Compound Names:  \( g(g(a)), g(f(g(a),g(b))) \)
Compound Names:  \( g(g(g(a))) \)
Compound Names:  \( g(g(g(g(a)))) \)
A **datum / factoid** is an expression formed from a predicate and *n symbols or compound names* enclosed in parentheses and separated by commas.

Symbols:  $a, b$
Constructors:  $f, g$
Predicate:  $p$

Sample Datum:  $p(a, g(a))$
Sample Datum:  $p(f(a,b), g(b))$

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: $a, b$

Constructors: $f, g$

Predicates: $p, q$

Dataset: \{p(a, g(a)), p(g(b), a), q(f(a, 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*. 
View Definitions
View Definitions

Virtual Dataset

View

Base Dataset

Operation

Virtual Dataset

View

Base Dataset
A **constant** is a string of lower case letters, digits, underscores, and periods.

```
a, b, c, 1, 2, 3, king, white_knight
f, g, piece, list, set
p, q, r, cell, adjacent
```

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

```
X   Y23   Somebody  _
```
The head of an instance of a rule is true if every positive subgoal is true and every negated subgoal is false.
An instance of a rule is a rule in which all variables have been consistently replaced by symbols or compound names.

Rule

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

Symbols

\{a, b\}

Instances

\[ r(a,a) :\neg p(a,a) & \neg q(a) \]
\[ r(a,b) :\neg p(a,b) & \neg q(b) \]
\[ r(b,a) :\neg p(b,a) & \neg q(a) \]
\[ r(b,b) :\neg p(b,b) & \neg q(b) \]
The **result** of applying $r$ to dataset $\Delta$ is the set of all $\psi$ such that

1. $\psi$ is the head of an arbitrary instance of $r$,
2. every positive subgoal in the instance is in $\Delta$, and
3. no negated subgoal in the instance is in $\Delta$.

The **closure** of a ruleset on a dataset is 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.

*The full definition is slightly more complicated. See below.*

http://logicprogramming.stanford.edu/miscellaneous/dlp.html
Grandparents:

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

Data:

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

View:

- grandparent(art,cal)
- grandparent(art,cam)
- grandparent(art,cat)
- grandparent(art,coe)
Personhood:

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

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)
Example (step 1)

Ancestors:

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

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:

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

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

Initial 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:

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

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

Difference

\textit{distinct}(t_1,t_2) \text{ is true iff } t_1 \text{ and } t_2 \text{ are different}
\textit{mutex}(t_1,\ldots,t_n) \text{ is true iff } t_1,\ldots,t_2 \text{ are all different}

Examples

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

Safety: Variables okay but no \textit{unbound} variables allowed!!!
Dataset \{p(a,b), p(a,c), p(b,d)\}

Example

\[
goal(X,L) :- \\
p(X,Y) \ & \\
\text{countofall}(Z,p(X,Z),L)
\]

Result \{goal(a,2), goal(b,1)\}

Example

\[
goal(X,L) :- \\
p(X,Y) \ & \\
\text{setofall}(Z,p(X,Z),L)
\]

Result \{goal(a,[b,c]), goal(b,[d])\}
Operation Definitions
Dynamics

Virtual Dataset

Base Dataset

Virtual Dataset

Base Dataset

View

Action

t=1

t=2

\( p(a,b) \)

\( p(b,c) \)

\( p(c,d) \)

\( p(b,a) \)

\( p(c,b) \)

\( p(d,c) \)

\( g(a,c) \)

\( g(b,d) \)

\( g(c,a) \)

\( g(d,b) \)
**Operation Constants**

*Operation constants* represent operations.

**move** - move a piece from one square to another  
**mark** - place a specific mark in a row and a column

Same spelling conventions as other constants.
An **action** is an application of an operation to objects.

In what follows, we denote actions using a syntax analogous to that of *compound names* and *factoids*, viz. an operation constant followed by $n$ terms enclosed in parentheses and separated by commas.

**Examples:**

- $\text{mark}(1,2)$
- $\text{move}(e2,e4)$
If the head of an instance is performed and the conditions are true, then the negated consequences are removed and the positive consequences are added.
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.
Given a rule set, the result of applying an action to a dataset $\Delta$ is the dataset that results from (1) *deleting all of the negative effects* of the action from the dataset and (2) *adding in all of the positive effects*.

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

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

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), r(b)\}

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

Action: \( u(b) \)

Negative effects: none
Positive effects: none

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

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

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

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)\}
Game Description
States

X
| O
| X
Game-Specific Vocabulary

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

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

Operation:
  mark(index,index)
Relation Constants:

role\( (role) \) - means that role is a role
base\( (factoid) \)
action\( (action) \)
init\( (factoid) \)
control\( (role) \)
legal\( (action) \)
goal\( (role, number) \)
terminal

Object Constants:

0, 1, 2, 3, … , 100 - numbers
A game description in GDL is a dynamic logic program (1) that defines the game-independent relations in terms of the game’s game-specific vocabulary and (2) that defines the operations of the game.

Independent of state:
  role, base, action, init

State-dependent:
  control, legal, goal, terminal

Operations
\texttt{role}(x) \\
\texttt{role}(o)
Base Relations and Input Operations

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

Relations:
- \( \rightarrow \text{cell}(\text{index}, \text{index}, \text{mark}) \)
- \( \rightarrow \text{control}(\text{role}) \)
  - \( \text{row}(\text{index}, \text{mark}) \)
  - \( \text{column}(\text{index}, \text{mark}) \)
  - \( \text{diagonal}(\text{mark}) \)
  - \( \text{line}(\text{mark}) \)
  - open

Operation:
- \( \rightarrow \text{mark}(\text{index}, \text{index}) \)
base(cell(X,Y,W)) :-
    index(X) &
    index(Y) &
    filler(W)

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

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

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

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

\[
\text{init}(\text{cell}(1,1,b)) \\
\text{init}(\text{cell}(1,2,b)) \\
\text{init}(\text{cell}(1,3,b)) \\
\text{init}(\text{cell}(2,1,b)) \\
\text{init}(\text{cell}(2,2,b)) \\
\text{init}(\text{cell}(2,3,b)) \\
\text{init}(\text{cell}(3,1,b)) \\
\text{init}(\text{cell}(3,2,b)) \\
\text{init}(\text{cell}(3,3,b)) \\
\text{init}(\text{control}(x))
\]
\textbf{Legal Moves}:

\texttt{legal(mark(M,N)) :- cell(M,N,b)}

\textbf{State:}

\begin{align*}
\text{cell(1,1,x)} \\
\text{cell(1,2,b)} \\
\text{cell(1,3,b)} \\
\text{cell(2,1,b)} \\
\text{cell(2,2,o)} \\
\text{cell(2,3,b)} \\
\text{cell(3,1,b)} \\
\text{cell(3,2,b)} \\
\text{cell(3,3,x)} \\
\text{control(o)}
\end{align*}

\textbf{Legal Moves:}

\begin{align*}
\text{mark(1,2)} \\
\text{mark(1,3)} \\
\text{mark(2,1)} \\
\text{mark(2,3)} \\
\text{mark(3,1)} \\
\text{mark(3,2)}
\end{align*}
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)

open :- cell(X,Y,b)
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
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 \textit{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.

\textit{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:

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

What the player sees:

\[ \text{wlcoul}(M,N) :: \]
\[ \text{control}(Z) ==> \neg \text{dupse}(M,N,erol) \land \text{dupse}(M,N,Z) \]
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/sierra/sierra.html
http://gamemaster.stanford.edu/homepage/rulecheckerintro.php
http://gamemaster.stanford.edu/homepage/standaloneopen.php
http://gamemaster.stanford.edu/homepage/standaloneopen.php
GENERAL
GAME PLAYING