Metagaming is match-independent game processing, i.e. game processing that is done independent of any particular opponent or any particular state.

Objective of metagaming - to optimize performance in playing specific matches of the game.

Usually done offline, i.e. during the startclock or between moves or in parallel with regular game play.
Examples

Boring:
Headstart on Game Tree Search
Endgame book

Interesting:
Game Grounding (formerly known as "propnets")
Game Reformulation (e.g. symmetries, factors)
Symbolic Reasoning (e.g. subgoal ordering)
**Grounding** is the process of *instantiating* game rules, i.e. replacing all variables by ground terms.

**Symbolizing** is the process of replacing ground propositions by atomic symbols.
Grounding

\[
\text{row}(M,R) \ :- \ \text{cell}(M,1,R) \ & \ \text{cell}(M,2,R) \ & \ \text{cell}(M,3,R) \\
\text{col}(N,R) \ :- \ \text{cell}(1,N,R) \ & \ \text{cell}(2,N,R) \ & \ \text{cell}(3,N,R) \\
\text{row}(1,x) \ :- \ \text{cell}(1,1,x) \ & \ \text{cell}(1,2,x) \ & \ \text{cell}(1,3,x) \\
\text{row}(2,x) \ :- \ \text{cell}(2,1,x) \ & \ \text{cell}(2,2,x) \ & \ \text{cell}(2,3,x) \\
\text{row}(3,x) \ :- \ \text{cell}(3,1,x) \ & \ \text{cell}(3,2,x) \ & \ \text{cell}(3,3,x) \\
\text{row}(1,o) \ :- \ \text{cell}(1,1,o) \ & \ \text{cell}(1,2,o) \ & \ \text{cell}(1,3,o) \\
\text{row}(2,o) \ :- \ \text{cell}(2,1,o) \ & \ \text{cell}(2,2,o) \ & \ \text{cell}(2,3,o) \\
\text{row}(3,o) \ :- \ \text{cell}(3,1,o) \ & \ \text{cell}(3,2,o) \ & \ \text{cell}(3,3,o) \\
\text{col}(1,x) \ :- \ \text{cell}(1,1,x) \ & \ \text{cell}(2,1,x) \ & \ \text{cell}(3,1,x) \\
\text{col}(2,x) \ :- \ \text{cell}(1,2,x) \ & \ \text{cell}(2,2,x) \ & \ \text{cell}(3,2,x) \\
\text{col}(3,x) \ :- \ \text{cell}(1,3,x) \ & \ \text{cell}(2,3,x) \ & \ \text{cell}(3,3,x) \\
\text{col}(1,o) \ :- \ \text{cell}(1,1,o) \ & \ \text{cell}(2,1,o) \ & \ \text{cell}(3,1,o) \\
\text{col}(2,o) \ :- \ \text{cell}(1,2,o) \ & \ \text{cell}(2,2,o) \ & \ \text{cell}(3,2,o) \\
\text{col}(3,o) \ :- \ \text{cell}(1,3,o) \ & \ \text{cell}(2,3,o) \ & \ \text{cell}(3,3,o) 
\]
Symbolizing

\[
\begin{align*}
\text{row}(1,x) & : - \text{cell}(1,1,x) \land \text{cell}(1,2,x) \land \text{cell}(1,3,x) \\
\text{row}(2,x) & : - \text{cell}(2,1,x) \land \text{cell}(2,2,x) \land \text{cell}(2,3,x) \\
\text{row}(3,x) & : - \text{cell}(3,1,x) \land \text{cell}(3,2,x) \land \text{cell}(3,3,x) \\
\text{row}(1,o) & : - \text{cell}(1,1,o) \land \text{cell}(1,2,o) \land \text{cell}(1,3,o) \\
\text{row}(2,o) & : - \text{cell}(2,1,o) \land \text{cell}(2,2,o) \land \text{cell}(2,3,o) \\
\text{row}(3,o) & : - \text{cell}(3,1,o) \land \text{cell}(3,2,o) \land \text{cell}(3,3,o)
\end{align*}
\]
Benefits
   Ground descriptions are simpler
   Interpreting ground descriptions more efficient

Disadvantages:
   Ground descriptions are more verbose / larger
   Extra rules can lead to greater cost
   (but extra cost offset by greater efficiency)

Grounding takes time
Not all descriptions can be grounded
Benefits
Ground descriptions are simpler
Interpreting ground descriptions more efficient

Disadvantages:
Ground descriptions are more verbose / larger
Extra rules can lead to greater cost
(but extra cost offset by greater efficiency)

Grounding takes time
Not all descriptions can be grounded
Grounding
(1) Compute base propositions.

(2) Compute actions.

(3) Iterate over view rules, matching subgoals to propositions, adding corresponding rule heads to propositions. Repeat to fixpoint.

(4) Iterate over view rules matching subgoals to propositions, adding corresponding rules to output.

(5) Iterate over operation rules matching heads to actions and conditions to propositions, adding corresponding rules to output.
Computing Base Propositions

role(x)  index(1)
role(o)  index(2)
index(3)

\[
\begin{align*}
\text{base}(\text{cell}(M,N,x)) & : \text{index}(M) \land \text{index}(N) \\
\text{base}(\text{cell}(M,N,o)) & : \text{index}(M) \land \text{index}(N) \\
\text{base}(\text{cell}(M,N,b)) & : \text{index}(M) \land \text{index}(N) \\
\text{base}(\text{control}(R)) & : \text{role}(R)
\end{align*}
\]
index(1)
index(2)
index(3)

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

mark(1,1)
mark(1,2)
...
mark(3,1)
mark(3,3)
Adding View Propositions

\[
\text{row}(M,R) :- \text{cell}(M,1,R) \land \text{cell}(M,2,R) \land \text{cell}(M,3,R)
\]
Adding View Propositions

\[
\begin{align*}
\text{cell}(1,1,x) & \quad \text{row}(1,x) \\
\text{cell}(1,2,x) & \quad \text{row}(2,x) \\
\ldots & \quad \text{row}(3,x) \\
\text{control}(x) & \quad \text{row}(1,o) \\
\text{control}(o) & \quad \text{row}(2,o) \\
& \quad \text{row}(3,o)
\end{align*}
\]

\[
\text{line}(R) ::= \text{row}(M,R)
\]

line(x) 
line(o)
Instantiating View Rules

\[
\begin{align*}
\text{cell}(1,1,x) & \quad \text{row}(1,x) \quad \ldots \\
\text{cell}(1,2,x) & \quad \text{row}(2,x) \quad \ldots \\
\vdots & \quad \text{row}(3,x) \quad \ldots \\
\text{control}(x) & \quad \text{row}(1,o) \quad \ldots \\
\text{control}(o) & \quad \text{row}(2,o) \quad \ldots \\
& \quad \text{row}(3,o) \quad \ldots \\
\end{align*}
\]

\[
\begin{align*}
\text{line}(R) & :- \text{row}(M,R) \\
\text{line}(x) & :- \text{row}(1,x) \\
\text{line}(x) & :- \text{row}(2,x) \\
\text{line}(x) & :- \text{row}(3,x) \\
\text{line}(o) & :- \text{row}(1,o) \\
\text{line}(o) & :- \text{row}(2,o) \\
\text{line}(o) & :- \text{row}(3,o) \\
\end{align*}
\]
Instantiating Operation Rules

mark(1,1) :: control(x) ==> cell(1,1,x) & ~cell(1,1,b)
mark(1,2) :: control(x) ==> cell(1,2,x) & ~cell(1,2,b)
mark(1,3) :: ...
mark(3,1) :: control(o) ==> cell(3,1,o) & ~cell(3,1,b)
mark(3,2) :: control(o) ==> cell(3,2,o) & ~cell(3,2,b)
mark(3,3) :: control(o) ==> cell(3,3,o) & ~cell(3,3,b)

mark(M,N) :: control(R) ==> cell(M,N,R) & ~cell(M,N,b)
General Game Playing
Game Grounder

Game Description:

```prolog
;; Buttons and Lights
;; Components

{role robot}  
{base p} 
{base q} 
{base r} 
{base 1} 
{base 2} 
{base 3} 
{base 4} 
{base 5} 
{base 6}
```

Output: 3 milliseconds

```prolog
role(robot)  
base(p)  
base(q)  
base(r)  
base(1)  
base(2)  
base(3)  
base(4)  
base(5)  
base(6)  
base(7)  
input(robot,a)  
input(robot,b)  
input(robot,c)  
init(1)  
legal(robot,a)  
legal(robot,b)  
legal(robot,c)
```
init(cell(M,N,b)) :- index(M) & index(N)

init(cell(1,1,b)) :- index(1) & index(1)
init(cell(1,2,b)) :- index(1) & index(2)
...  
init(cell(3,2,b)) :- index(3) & index(2)
init(cell(3,3,b)) :- index(3) & index(3)

init(cell(1,1,b))
init(cell(1,1,b))
...  
init(cell(1,1,b))
init(cell(1,1,b))
Execution
Playing Ground / Symbolized Games

Good News
   Our interpreter works with ground / symbolized games

Better News
   We can do better
   Specialized interpreters for ground / symbolized games
Our *general interpreter* must allow for *unification* of expressions involving variables.

Upshot: Unifications are expensive.

In a *ground interpreter*, unifications are replaced by *string equality*.

Upshot: String equality testing is *relatively* inexpensive.
function findcontrol (facts, rules)
{return grounditem('control', facts, rules)}

function findlegalp (move, facts, rules)
{return groundfindp(seq('legal', move), facts, rules)}

function findlegalx (facts, rules)
{return grounditem('legal', facts, rules)}

function findlegals (facts, rules)
{return grounditems('legal', facts, rules)}

function findreward (role, facts, rules)
{var value = groundvalue('goal', role, facts, rules);
 if (value) {return value};
 return 0}

function findterminalp (facts, rules)
{return groundfindp('terminal', facts, rules)}
function basefindbackground (n,x,p,pl,al,cont,results,facts,rules)
  {var data = envlookupfacts(p,al,facts);
   for (var i=0; i<data.length; i++)
     {var bl = {};
      var ol = seq();
      if (vnifyp(data[i],bl,p,al,ol))
        {var ans = basesomeexit(n,x,pl,al,cont,results,facts,rules);
         backup(ol);
         if (answer) {return answer}};
     return false}

function groundfindbackground (p,facts,rules)
  {for (var i=0; i<facts.length; i++)
   {if (equalp(facts[i],p)) {return true}};
  return false}
function basefindbackground (n,x,p,pl,al,cont,results,facts(rules)
   {var data = envlookupfacts(p,al,facts);
    for (var i=0; i<data.length; i++)
       {var bl = {};
        var ol = seq();
        if (vnifyp(data[i],bl,p,al,ol))
           {var ans = basesomeexit(n,x,pl,al,cont,results,facts,.rules);
            backup(ol);
            if (answer) {return answer};
        }
    return false
}

function groundfindbackground (p,facts, rules)
   {for (var i=0; i<facts.length; i++)
    {if (equalp(facts[i],p)) {return true};
    return false
}
Specialized Player for Symbolized Games

All the benefits of ground games and more.

Once symbolized, propositions are represented as strings, and we can represent states using *associative arrays* (logarithmic access and update) rather than *lists* (linear access and update).

Benefit: Especially valuable for games where states characterized by many propositions.

Benefit: In Monte Carlo depth charges, there is no need to copy states; we can just change in place.
function findinits (rules)
    {return basefinds('P',seq('init','P'),seq(),rules)}

function findinits (rules)
    {var state = basefinds('P',seq('init','P'),[ ],rules));
    return makestate(state)}

function makestate (facts)
    {var newstate = {};
    for (var i=0; i<facts.length; i++)
        {newstate[facts[i]] = true};
    return newstate}
Ground:

function groundfindbackground (p,facts,rules)
  {for (var i=0; i<facts.length; i++)
    {if (equalp(facts[i],p)) {return true};
      return false}

Symbol:

function groundfindbackground (p,facts,rules)
  {return (facts[p]===true)}
function simulate (move,state,rules)
    {var deltas = groundexpand(move,state,rules);
        var newstate = Object.assign({},state);
        for (var i=0; i<deltas.length; i++)
            {var delta = deltas[i];
                if (!symbolp(delta) && delta[0]==='not')
                    {delete(newstate[delta[1]]);}};
        for (var i=0; i<deltas.length; i++)
            {var delta = deltas[i];
                if (symbolp(delta))
                    {newstate[delta] = true; continue};
                if (delta[0]==='not') {continue};
                newstate[delta] = true};
    return newstate}

No need to copy on depth charges!!
Specialized Ground Interpreter

Browser-Based Players

- Legal - Legal player
- Random - Random player
- OneStep - One Step player
- Minimax - Full Minimax player
- MinimaxDepth - Minimax player with fixed depth
- MinimaxID - Minimax player with iterative deepening
- Greedy - Greedy player
- MCS - Monte Carlo Search player

JavascriptCode

- EpilogJS - Logic Programming Interpreter
- Basics - Basic subroutines for general games
- Ground - Basic subroutines for ground games
- Symbol - Basic subroutines for symbolized games
- ParametricJS - Code for standard players
- PlayerJS - Player set-up for NodeJS

Older General Game Playing Websites

- Tiltyard (web site) - allows users to register players for automatic round robin competition against other general game playing programs. Warning: Games on this website do not necessarily comply with the current General Game Playing standard.

- GGP.org (web page) - General website on GGP. Contains information on how to develop software for GGP. Warning: Games on this website do not necessarily comply with the current General Game Playing standard.

Questions and Comments
Specialized Symbol Interpreter

Browser-Based Players

- Legal - Legal player
- Random - Random player
- Onestep - One Step player
- Minimax - Full Minimax player
- Minimaxdepth - Minimax player with fixed depth
- Minimaxid - Minimax player with iterative deepening
- Greedy - Greedy player
- MCS - Monte Carlo Search player

JavascriptCode

- EpilogJS - Logic Programming Interpreter
- Basics - Basic subroutines for general games
- Ground - Basic subroutines for ground games
- Symbol - Basic subroutines for symbolized games
- ParametricJS - Code for standard players
- PlayerJS - Player set-up for NodeJS

Older General Game Playing Websites

- Tillyard (web site) - allows users to register players for automatic round robin competition against other general game playing programs. Warning: Games on this website do not necessarily comply with the current General Game Playing standard.
- GGP.org (web page) - General website on GGP. Contains information on how to develop software for GGP. Warning: Games on this website do not necessarily comply with the current General Game Playing standard.

Questions and Comments
Analysis
Good News
  General interpreter works on ground rules.
  Our special interpreters are faster on ground rules.

Bad News
  They sometimes play slower!

What?

Why?
Analysis

Original Rule

\[ \text{goal}(X,Y) ::= \ p(X) \ & \ q(X) \ & \ r(X,Y) \]

\[ 1 + (n^2+2n) + n*((n^2+2n) + 1*(n^2+2n)) = 2n^3 + 5n^2 + 2n + 1 \]

Ground Rules

\[ \text{goal}(a,a) ::= \ p(a) \ & \ q(a) \ & \ r(a,a) \]

\[ \ldots \]

\[ \text{goal}(c,c) ::= \ p(c) \ & \ q(c) \ & \ r(c,c) \]

\[ n^2 * (1 + 3*(n^2 + 2n)) = 3n^4 + 6n^3 + n^2 \]
Number of Comparisons

Original Rule

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

\[
1 + (n^2 + 2n) + n^2((n^2 + 2n) + 1*(n^2 + 2n)) = 2n^3 + 5n^2 + 2n + 1
\]

106 nodes for \( n = 3 \)

Ground Rules

\[
goal(a, a) :- p(a) \land q(a) \land r(a, a)
\]

... 

\[
goal(c, c) :- p(c) \land q(c) \land r(c, c)
\]

\[
n^2 * (1 + 3*(n^2 + 2n)) = 3n^4 + 6n^3 + n^2
\]

414 nodes for \( n = 3 \)
# Experimental Results

## Table of Results

<table>
<thead>
<tr>
<th>Game</th>
<th>Depth</th>
<th>Result</th>
<th>Terms</th>
<th>Nodes</th>
<th>Normal Normal</th>
</tr>
</thead>
<tbody>
<tr>
<td>buttonsandlights</td>
<td>6</td>
<td>100</td>
<td>729</td>
<td>1093</td>
<td>20</td>
</tr>
<tr>
<td>tictactoe7</td>
<td>7</td>
<td>50</td>
<td>3468</td>
<td>7332</td>
<td>650</td>
</tr>
<tr>
<td>tictoctoe7</td>
<td>7</td>
<td>50</td>
<td>5040</td>
<td>13700</td>
<td>550</td>
</tr>
<tr>
<td>connectfour</td>
<td>4</td>
<td>0</td>
<td>4096</td>
<td>4681</td>
<td>2700</td>
</tr>
<tr>
<td>nineboard</td>
<td>4</td>
<td>50</td>
<td>6120</td>
<td>6914</td>
<td>9300</td>
</tr>
</tbody>
</table>

### Experimental Results

<table>
<thead>
<tr>
<th>Game</th>
<th>Depth</th>
<th>Result</th>
<th>Terms</th>
<th>Nodes</th>
<th>Normal Normal</th>
<th>Normal Ground</th>
</tr>
</thead>
<tbody>
<tr>
<td>buttonsandlights</td>
<td>6</td>
<td>100</td>
<td>729</td>
<td>1093</td>
<td>20</td>
<td>25</td>
</tr>
<tr>
<td>tictactoe7</td>
<td>7</td>
<td>50</td>
<td>3468</td>
<td>7332</td>
<td>650</td>
<td>800</td>
</tr>
<tr>
<td>tictactoe7</td>
<td>7</td>
<td>50</td>
<td>5040</td>
<td>13700</td>
<td>550</td>
<td>800</td>
</tr>
<tr>
<td>connectfour</td>
<td>4</td>
<td>0</td>
<td>4096</td>
<td>4681</td>
<td>2700</td>
<td>3200</td>
</tr>
<tr>
<td>nineboard</td>
<td>4</td>
<td>50</td>
<td>6120</td>
<td>6914</td>
<td>9300</td>
<td>38000</td>
</tr>
</tbody>
</table>
## Experimental Results

<table>
<thead>
<tr>
<th>Game</th>
<th>Depth</th>
<th>Result</th>
<th>Terms</th>
<th>Nodes</th>
<th>Normal Normal</th>
<th>Normal Ground</th>
<th>Ground Ground</th>
</tr>
</thead>
<tbody>
<tr>
<td>buttonsandlights</td>
<td>6</td>
<td>100</td>
<td>729</td>
<td>1093</td>
<td>20</td>
<td>25</td>
<td>9</td>
</tr>
<tr>
<td>tictactoe7</td>
<td>7</td>
<td>50</td>
<td>3468</td>
<td>7332</td>
<td>650</td>
<td>800</td>
<td>250</td>
</tr>
<tr>
<td>tictoctoe7</td>
<td>7</td>
<td>50</td>
<td>5040</td>
<td>13700</td>
<td>550</td>
<td>800</td>
<td>240</td>
</tr>
<tr>
<td>connectfour</td>
<td>4</td>
<td>0</td>
<td>4096</td>
<td>4681</td>
<td>2700</td>
<td>3200</td>
<td>870</td>
</tr>
<tr>
<td>nineboard</td>
<td>4</td>
<td>50</td>
<td>6120</td>
<td>6914</td>
<td>9300</td>
<td>38000</td>
<td>10500</td>
</tr>
</tbody>
</table>
## Experimental Results

<table>
<thead>
<tr>
<th>Game</th>
<th>Depth</th>
<th>Result</th>
<th>Terms</th>
<th>Nodes</th>
<th>Normal Normal</th>
<th>Normal Ground</th>
<th>Ground Ground</th>
<th>Symbol Symbol</th>
</tr>
</thead>
<tbody>
<tr>
<td>buttonsandlights</td>
<td>6</td>
<td>100</td>
<td>729</td>
<td>1093</td>
<td>20</td>
<td>25</td>
<td>9</td>
<td>8</td>
</tr>
<tr>
<td>tictactoe7</td>
<td>7</td>
<td>50</td>
<td>3468</td>
<td>7332</td>
<td>650</td>
<td>800</td>
<td>250</td>
<td>130</td>
</tr>
<tr>
<td>tictoctoe7</td>
<td>7</td>
<td>50</td>
<td>5040</td>
<td>13700</td>
<td>550</td>
<td>800</td>
<td>240</td>
<td>150</td>
</tr>
<tr>
<td>connectfour</td>
<td>4</td>
<td>0</td>
<td>4096</td>
<td>4681</td>
<td>2700</td>
<td>3200</td>
<td>870</td>
<td>600</td>
</tr>
<tr>
<td>nineboard</td>
<td>4</td>
<td>50</td>
<td>6120</td>
<td>6914</td>
<td>9300</td>
<td>38000</td>
<td>10500</td>
<td>960</td>
</tr>
</tbody>
</table>
Propositional Nets
Propositional Net Components

Propositions

Connectives

Transitions
Propositional Net
Propositional Net

Action

View Proposition

Base Proposition
A marking for a propositional net is a function from the propositions $P$ to boolean values.

$$m: P \rightarrow \{true, false\}$$

Think of a marking as a state of a game.
Pressing button $a$ toggles $p$.
Pressing button $b$ interchanges $p$ and $q$. 

Simple Buttons and Lights
Field Programmable Gate Arrays (FPGAs)
Field Programmable Gate Arrays (FPGAs)