Game Reformulation
**Metagaming**

*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-Graph Search
  Endgame book

Structural:
  Change of Framework (e.g. state machines to propnets)
  Game Reformulation (e.g. game decomposition)

Engineering:
  Compilation (machine language, fpga’s)
Game Reformulation

Conceptual Reformulation - Changing propositions and/or connectivity within game

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

Game Factoring - dividing game into sub-games
Factoring
Hodgepodge

Hodgepodge = Chess + Othello

Analysis of joint game:
Branching factor as given to players: $a*b$
Fringe of tree at depth n as given: $(a*b)^n$
Fringe of tree at depth n factored: $a^n+b^n$
Double Tic-Tac-Toe

Double Tic Tac Toe = TTT + TTT

Analysis of joint game:
   Branching factor: 81, 64, 49, 36, 25, 16, 9, 4, 1
   Branching factor: 9, 8, 7, 6, 5, 4, 3, 2, 1
Game Factoring

Method:

- Compute factors
- Use factors to generate submoves
- Assemble overall move from submoves

Cases:

- Initial factors
- Conditional factors
Propositional Net

![Propositional Net Diagram]
Factorable Example
Propnet Factors

A propositional net $M$ is a factor of $N$ if and only if $M$ is contained in $N$ and there are no connections between the components in $M$ and the components in $N-M$.

Factors of propositional nets can be found in polynomial time (in terms of the size of the nets).
Open Propositional Automata

An open propositional automaton is a structure of the form $<N,m,t>$, where $N$ is a propnet with one or more input propositions, where $m$ is a base marking (the initial marking), where $t$ is a proposition in $N$ (the terminal node).
GGP Version

For GGP purposes, we augment to $<N,m,t,l,g>$, where $l$ is a proposition in $N$ (the *legality* node for the input propositions), and where $g$ is a proposition in $N$ (the *goal* node).
Determination

The goal and legality and terminal nodes must be determined by the base propositions. A proposition $p$ is determined by a set of propositions $P$ iff $p$ is a member of $P$ or is the output of a gate with all inputs determined by $P$. 
2-Port Open Propositional Automata

An *n*-port open propositional automaton is a tuple of the form \(<N,m,t,I_1,l_1,g_1,I_2,l_2,g_2>\), where \(N\) is an arbitrary propositional net, where \(m\) is a base marking (the *initial marking*), where \(t\) is a proposition in \(N\) (the *terminal* node), where \(I_1, I_2\) is a partition of the input propositions in \(N\), where \(L_1\) and \(L_2\) are sets of propositions in \(N\) (the *legality* nodes for each role), and where \(g_1\) and \(g_2\) are proposition in \(N\) (the *goal* nodes for each role).

The goal and legality and terminal nodes must be determined by the base propositions.
Example
n-Port Open Propositional Automata

An *n*-port open propositional automaton is a tuple of the form $<N, m, t, I_1, l_1, g_1, \ldots, I_n, l_n, g_n>$, where $N$ is an arbitrary propositional net, where $m$ is a base marking (the *initial marking*), where $t$ is a proposition in $N$ (the *terminal* node), where $I_1, \ldots, I_n$ is a partition of the input propositions in $N$, where $L_i$ is a set of propositions in $N$ (the *legality* nodes for role $I$ for role $i$), and where $g_i$ is a proposition in $N$ (the *goal* node for role $i$).

The goal and legality and terminal nodes must be determined by the base propositions.
Game Factoring

Method:
- Compute factors
- Use factors to generate “subplans”
- Reassemble overall plan from subplans

Patterns:
* Disjunctive Factors
+ Interleaved Conjunctive Factors
+ Simultaneous Conjunctive Factors
X Sequential Factors

* Initial factors versus X conditional factors
Disjunctive Factoring

Disjunctive Goals
  Delete disjunctive goal
  Make disjuncts goals for each factor

Solve one of two problems and use that solution.

In multiple player games, your player must be sure it does not lose other factors before it wins the factor it has selected.
Example
Example
Performance

? (time (genplan propcompbobuttons))
  407 states
  1,118 milliseconds.
  605,728 bytes of memory allocated.
(PROG A B A D E D)

? (time (multiplan propcommbobuttons))
  14 states
  53 milliseconds
  22,320 bytes of memory allocated.
(PROG A B A D E D)
Partition time: 1 millisecond.
Interleaved Conjunctive Factoring

Conjunctive Goals
   Delete conjunctive goal
   Make conjuncts goals for each factor

In *interleaved* conjunctive factoring, the actions of each factor are paired with noops for other factors until goal is reached, after which actions in other factors are executed.

In *simultaneous* conjunctive factoring, actions must be decomposed into actions for each factor and recomposed into joint actions.
Relative Inertia

In interleaved conjunctive factoring, a player must show that a non-action leaves everything the same, i.e. that the propositions in a factor do not change unless one of the actions in the factor is executed.
Relatively Inert Prop Net
$q_1, q_2$ not relatively inert. If $q_2$ is true and $p_2$ is false then $q_2$ becomes false.
Simultaneous Conjunctive Factoring

Conjunctive Goals
  Delete conjunctive goal
  Make conjuncts goals for each factor

Solve each problem, conjoin solutions.

In simultaneous conjunctive factoring, actions must be decomposed into actions for each factor and recomposed into joint actions.
Restructuring and Reformulation

In some cases, it may be necessary and it may be possible to restructure a propositional net so that it becomes factorable. *Restructuring* here means same propositions but different gates and transitions.

In some case, it may be necessary to reformulate goals and/or actions so that they become factorable and still satisfy the properties described earlier. *Reformulation* here means different propositions as well as different gates and transitions.
Example of Rewriting

Original Version

\[ p' := a \land \neg p \]
\[ q' := a \land \neg q \]
\[ s' := a \land s \]
\[ t' := a \land t \]

Rewriting

\[ p' := a \land \neg p \]
\[ q' := a \land \neg q \]
\[ s' := \neg b \land s \]
\[ t' := \neg b \land t \]
Buttons and Lights

Pressing button $a$ toggles $p$.
Pressing button $b$ interchanges $p$ and $q$. 
Double Buttons and Lights

Pressing button \( a \) toggles \( p \), toggles \( s \).
Pressing button \( b \) toggles \( p \), interchanges \( s \) and \( t \).
Pressing button \( c \) interchanges \( p \) and \( q \), toggles \( s \).
Pressing button \( d \) interchanges \( p \) and \( q \), \( s \) and \( t \).
Original Version

\[ p' \leftarrow a \land \neg p \]
\[ q' \leftarrow a \land q \]
\[ s' \leftarrow a \land \neg s \]
\[ t' \leftarrow a \land t \]

\[ p' \leftarrow c \land \neg p \]
\[ q' \leftarrow c \land q \]
\[ s' \leftarrow c \land t \]
\[ t' \leftarrow c \land s \]
Action Grouping

Actions grouped according to behavior.

\[
\begin{align*}
  p' & : {a, c} & & \sim p & & p' & : {b, d} & & q \\
  q' & : {a, c} & & q & & q' & : {b, d} & & \sim q \\
  s' & : {a, b} & & \sim s & & s' & : {c, d} & & \sim s \\
  t' & : {a, b} & & t & & t' & : {c, d} & & t
\end{align*}
\]

Note the partition on propositions.
Import and Export

Import:

\[
\begin{align*}
e & : - a \\
e & : - b \\
f & : - c \\
f & : - d \\
g & : - a \\
g & : - c \\
h & : - b \\
h & : - d
\end{align*}
\]

Export:

\[
\begin{align*}
a & : - e \ & g \\
b & : - e \ & h \\
c & : - f \ & g \\
d & : - f \ & h
\end{align*}
\]
Reformulated Version

\[ p' : = e \land \neg p \]
\[ q' : = e \land q \]
\[ p' : = f \land q \]
\[ q' : = f \land p \]
\[ s' : = g \land \neg s \]
\[ t' : = g \land t \]
\[ s' : = h \land t \]
\[ t' : = h \land s \]
Celebrating AAAI’s 25th Anniversary