Lecture 4

Game Grammars
Sources for Today’s Talk

- **Raph Koster** (one of the original proponents)
  - *Theory of Fun, 10 Years Later* (GDCOnline 2012)
  - [http://raphkoster.com](http://raphkoster.com)

- **Ernest Adams** and **Joris Dormans**
  - *Game Mechanics: Advanced Game Design*
  - One of the first serious attempts at a game grammar

- **Gabriel Eirea**, UC Berkeley
  - *Petri Nets: Properties, Analysis and Applications*
  - Source of many non-gamey examples
What Are We Talking About?

Game Grammars
Why Do We Want Game Grammars?

• Aid in **gameplay design**
  • Close disconnect between digital, non-digital
  • Understand gameplay before implementation

• Better **document gameplay**
  • Make designers wishes clear to implementer
  • Coding/software implementation is “automatic”

• Verify **gameplay balance**
  • Eliminate unwanted dominant strategies
  • Tune feedback loops for better pacing
This Sounds Vaguely Familiar…

- Aid in gameplay design
- Close disconnect between digital, non-digital
- Understand gameplay before implementation
- Better document gameplay
- Make developers' wishes clear to implementer
- Coding/software implementation is "automatic"
- Verify gameplay balance
- Eliminate unwanted dominant strategies
- Tune feedback loops for better pacing

Game Grammars
Graphical Stepwise Models

• Variations of *finite state machines*
  • Nodes represent individual steps
  • Edges are choices/transitions between steps
  • But often allow “parallel” computation

• **Notable Examples:**
  • Flow Charts
  • UML Activity Diagrams
  • Business Workflow Models
Graphical Stepwise Models

- Variations of finite state machines
  - Nodes represent individual steps
  - Edges represent transitions between steps
  - But often allow “parallel” computation

- Notable Examples:
  - Flow Charts
  - UML Activity Diagrams
  - Business Workflow Models
Verification: Model Checking

- Given a graph, logical formula $\varphi$
  - $\varphi$ expresses properties of graph
  - Checker determines if is true
- Often applied to software
  - Program as control-flow graph
  - $\varphi$ indicates acceptable paths

$$\begin{array}{l}
\text{sum} = 0 \\
\text{done} = F \\
i = 0 \\
i \leq 5 \land \neg \text{done} \\
j \geq 0 \\
\text{sum} = \text{sum} + j \\
\text{sum} > 100 \\
\text{done} = T \\
i = i + 1 \\
\text{endif} \\
\text{print sum}
\end{array}$$
Problem with State Machine Models

- State space **explosion**!
  - Resources, space continuous
  - Can discretize them (last time)
  - But that is at **individual** level
  - Full state space is all possible combinations of individuals

- **Handling simultaneity**
  - Multiple players
  - Multiple game entities
  - Simultaneous actions

Game Grammars
Petri Nets

- **State machine variant** for
  - Concurrent systems
  - Asynchronous systems
  - Distributed, parallel systems
  - Nondeterministic systems
  - Stochastic systems

- Ideal for **resource modeling**
  - Popular for modeling chemical processes
  - Useful for game economy?

\[ 2H_2 + O_2 \rightarrow 2H_2O \]
Definition

- **Directed, weighted, bipartite graph** with:
  - **Places** (nodes that represent possibilities)
  - **Transitions** (nodes between two places)
  - **Edges** (places to transitions or transitions to places)
  - **Weights** associated with each edge

- Current state is called a **marking**
  - Non-negative integer to each place
  - Represent integer at a place by “tokens”
  - Marking is tokens on all places, not just one
Definition

Marking

Place

Edge w/ Weight

H₂O

Transition

H₂

O₂

Marking

2

2

2
Transition (Firing) Rule

• Given a transition t
  • is **enabled** if each input p has ≥ w(p,t) tokens
  • may or may not **fire** if it is enabled

• If a transition fires
  • remove w(p,t) from each input place p
  • add w(t,p') to each output place p'

• This process is highly **nondeterministic**
Firing Example

\[ 2\text{H}_2 + \text{O}_2 \rightarrow 2\text{H}_2\text{O} \]
Modeling Concurrency
Modeling Concurrency

![Concurrency Diagram](image-url)
Modeling Communication Protocols

- **proc.1**
  - send msg.
  - ready to send
  - wait for ack.
  - receive ack.
  - ack. received

- **proc.2**
  - receive msg.
  - ready to receive
  - msg. received
  - send ack.
  - ack. sent
  - buffer full
  - receive ack.
  - buffer full

Game Grammars
Components of a Game Economy

- **Sources**: How a resource can increase
  - **Examples**: ammunition clips, health packs

- **Drains**: How a resource can decrease
  - **Examples**: firing weapon, player damage

- **Converters**: Changes one resource to another
  - **Example**: vendors, *Starcraft* barracks

- **Traders**: Exchange resources between entities
  - Mainly (but not always) in multiplayer games
Sources and Drains

Source

No Inputs

Drain

No Outputs
Converters and Traders

- Tokens are just *numbers*
  - No difference in token types

- Everything coded by *places*
  - Places code resources
  - Places code entities
  - Places code players
  - And combos of above

- **Color code** related groups
  - all places of same player
  - all places of same resource
What Properties Can We Analyze?

- **Reachability**
  - Is there a firing sequence from $M_0$ to $M_n$?

- **Boundedness**
  - Is there a limit to the of tokens on some place?

- **Liveness**
  - From current marking, can a transition ever fire?

- **Reversibility**
  - If $M_n$ reachable from $M_0$, is $M_0$ reachable from $M_n$?
Problems with Petri Nets

• **Synchronization and Timing**
  - Petri Nets allow arbitrary firing
  - Rules may constrain timing, coupling

• **Randomness**
  - Can effect resources coming in/out
  - Can effect synchronization and timing

• **Gameplay Effects**
  - Feedback can adjust “edge weights”
  - What about spatial properties?
Problems with Petri Nets

- **Synchronization and Timing**
  - Petri Nets allow arbitrary firing
  - Rules can constrain timing, coupling
  - Can Fix with **Timed Petri Nets**

- **Randomness**
  - Can effect resources coming in/out
  - Can effect synchronization and timing
  - Can Fix with **Stochastic Petri Nets**

- **Gameplay Effects**
  - Feedback can adjust “edge weights”
  - What about spatial properties?
  - **State Space Explosion!**

---

23
Machinations: A Proposed Grammar

- By Adams and Dormans
  - Only models resources
  - Made for strategic games

- Specialized petri net
  - Variety of marked edges
  - Specialized nodes for various economic entities

Edges
- Flow Rate 1
- Flow Rate 3
- Random Flow
- Skill-Based Flow Value
- Multiplayer Based Flow
- Strategy-Based Flow Value

Places
- Pool
- Drain
- Gate
- Source

Resources as Tokens
Specialized Node Structures

### Economic Nodes
- Converter
- Trader
- Delay
- Queue

### Gameplay Modifiers
- Modifies Edge
- Modifies Contents

### Activators
- Turns Nodes On or Off

### Triggers
- Fires When Over Theshold
- Fires When Below Theshold
Gates vs. Nondeterminism

- Gates ensure a distribution across many outputs
  - Cannot control this in a pure Petri Net
  - Non-determinism allows any combination

- Branching structures natural in rule design
Example: Pac-Man and Dot Eating

Game Grammars
Everything is a Resource

Pac-Man Ghosts

- Do not encode location
- Add resource: threat
- Class board game trick

Game Over
Lives
Reset
Threat

Evade
50%
>100

Ghost House
Maze
1/5

Add resource: threat

Game Grammars
**Everything is a Resource**

**Pac-Man Ghosts**

- Do not encode location
- Add resource: **threat**
- Class board game trick

---

**Game Over**

**Lives**

**Reset**

**Evasion Drains Threat**

**Evade**

**50%**

**>100**

**Ghosts Add Threat**
# Design vs. Verification

<table>
<thead>
<tr>
<th>Design</th>
<th>Verification</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Want ease of creation</td>
<td>• Want ease of analysis</td>
</tr>
<tr>
<td>• Quick visualization</td>
<td>• Feasible algorithms</td>
</tr>
<tr>
<td>• Can be less rigorous</td>
<td>• Definition must be rigorous</td>
</tr>
<tr>
<td>• Favors many structures</td>
<td>• Favor few structures</td>
</tr>
<tr>
<td>• Design patterns provided</td>
<td>• Encode design patterns</td>
</tr>
<tr>
<td>• Graphs are very small</td>
<td>• Graphs are very large</td>
</tr>
<tr>
<td>• Differs from implementation</td>
<td>• Many not match final design</td>
</tr>
</tbody>
</table>
Complete Pac-Man Model

Game Grammars
Advantages of Machinations

- Optimized for **verification**
  - Have software package
  - Plug in and simulate

- Software is ideal for
  - Finding dominant strategies
  - Evaluating pacing (e.g. feedback mechanisms)
  - Detecting politics

- But how good is it if model is just an **approximation**?

- **Well-designed patterns**
  - Adams *knows* economies
  - Useful economic tools

- Dynamic Patterns
  - Invest to increase flows
  - Act to resist drains
  - Build to activate nodes

- Escalation Patterns
  - Challenges more difficult
  - Actions less effective
# Spatial is Still Very Much a Problem

<table>
<thead>
<tr>
<th>Resource Affects Spatial</th>
<th>Spatial Affects Resources</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Resources can unlock areas</td>
<td>• Resources made by entities</td>
</tr>
<tr>
<td>• Keys are a trivial resource</td>
<td>• Have a spatial location</td>
</tr>
<tr>
<td>• Also use resource thresholds</td>
<td>• <strong>Ex</strong>: Time to transfer resources</td>
</tr>
<tr>
<td>• <strong>Ex</strong>: Collect all tokens to pass</td>
<td>• <strong>Ex</strong>: Sources be captured</td>
</tr>
<tr>
<td>• Resources affect difficulty</td>
<td>• Resource values are entities</td>
</tr>
<tr>
<td>• Adjust input device sensitivity</td>
<td>• Take up physical volume</td>
</tr>
<tr>
<td>• <strong>Ex</strong>: Deadeye meter in <em>RDR</em></td>
<td>• Need space to acquire</td>
</tr>
<tr>
<td>• <strong>Ex</strong>: Jet packs to increase jump</td>
<td>• <strong>Ex</strong>: Inventory in <em>Deux Ex</em></td>
</tr>
</tbody>
</table>
Spatial is Still Very Much a Problem

Spatial Affects Resources

- Resources made by entities
  - Have a spatial location
  - **Ex:** Time to transfer resources
  - **Ex:** Sources be captured

- Resource values are entities
  - Take up physical volume
  - Need space to acquire
  - **Ex:** Inventory in *Deux Ex*
What Does Raph Koster Think?

Island Life diagram

Legend

- Statistical Resource
- Friend Verb
- In-world Object
- Verb

Diagram:
- Fertilize
- Harvest
- XP/level
- Coins
- Place
- Decor
- Land
- Plots
- Space
- Delete

Game Grammars
Final Words

- Game grammars are a laudable goal
  - Specialized version of workflow problem
  - Some aspects of games lend themselves well

- Must choose between **design** or **verification**
  - Clear engineering trade-offs between these two
  - The “rigour” question is perhaps the biggest issue

- **Open problem** with lots of opportunities