Recipes: Non-Deterministic and Hierarchical Behavior Selection

Gal A. Kaminka

Non-Deterministic Markovian Behavior Selection

Reminder: Deterministic FSM Selection

```
1  W knowledge base, g goal, B behavior FSM
2
3  Let b = starting behavior in B  // starting state
4  START(b) // start execution of b
5  Let W' be PERCEIVE(W) // W' is updated
6  E = new beliefs in W' // (W'-W)
7  if E matches event on outgoing transition to b':
8      STOP(b) // stop execution of b
9      Let b = b'  // update b
10     goto 4
11 goto 5
```

But: Non-Determinism can be Useful

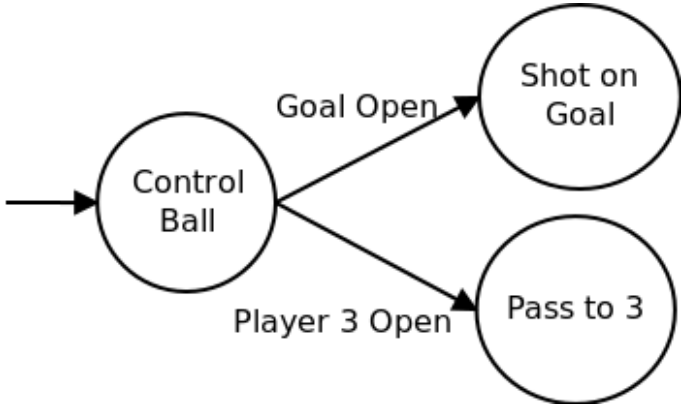
- Allow partial-order (lazy commitment to order)
  - Outgoing transitions marked by complementary events
  - More than one can be taken in principle

But: Non-Determinism can be Useful

- Allow partial-order (lazy commitment to order)
  - Outgoing transitions marked by complementary events
  - More than one can be taken in principle
- Delay grounding
  - Example: pass to open player? There can be several.
  - i.e., several grounded instantiations with same incoming event

In depth: Partially-ordered Markovian Selection

Easier to specify PARTIAL conditions for selection



. . . than FULLY-SPECIFIED conditions

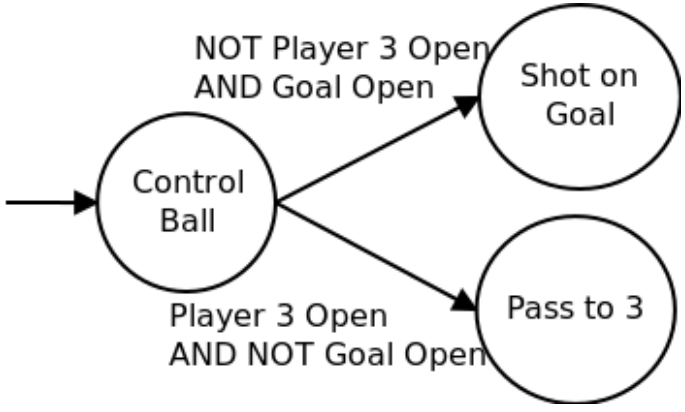
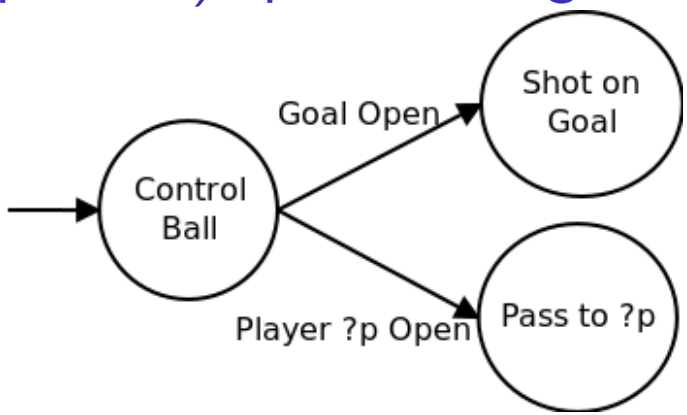


Figure 1: Total Order

In depth: Delayed grounding

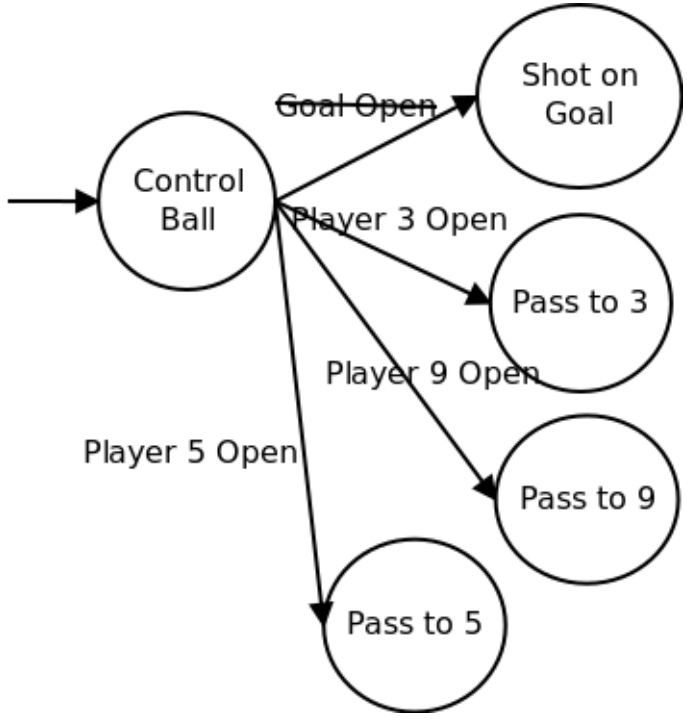
The programmer (planner) specifies ungrounded behavior



In depth: Delayed grounding

System grounds behaviors at run-time

Think of templates and instantiations, classes and instances



Must non-deterministically choose between multiple behaviors

Recipes, and BDI Agents

Non-determistic FSMs lead to thinking about recipes:

- instantiated at runtime (ungrounded specification)
- ad-hoc choice procedures (i.e., CHOOSE() procedures)
- complex events testing beliefs, allowing loops

CHOOSE() procedures for execution-time decision making

- Yes, it does look like the familiar agent design again. . .
- But this time, CHOOSE() is between behaviors, not actions
  - No action models; don't know what the effects are
  - CHOOSE can be arbitrary, random, or much much smarter

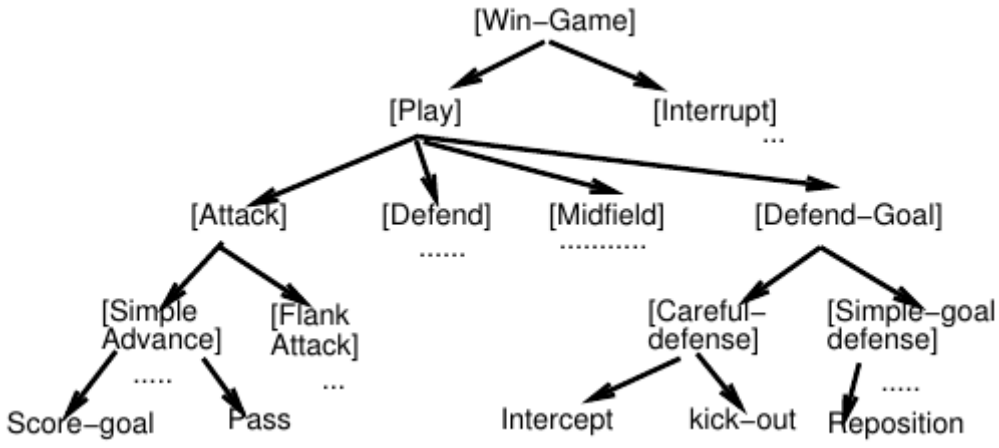
And expanding beyond FSMs (automata theory)

- Context-free, context-sensitive CHOOSE()
  - Utilize memory, history
- Hierarchies
  - May even change CHOOSE() mechanism depending on context

Hierarchical Recipes

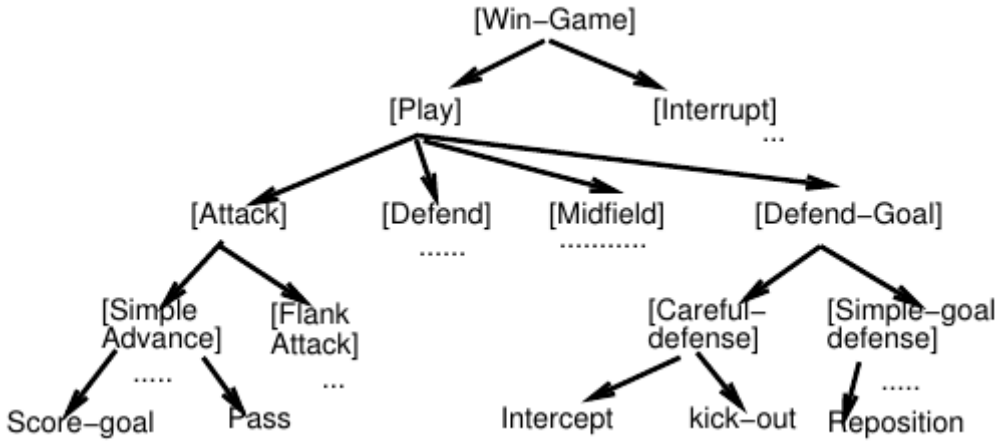
Hierarchies for Modularity and Reuse

- Organize plan as a sequence of hierarchies
- Each hierarchy composed of *multiple* behaviors



Hierarchies for Modularity and Reuse

- Organize plan as a sequence of hierarchies
- Each hierarchy composed of *multiple* behaviors



- Behaviors at lower levels can be re-used (multiple parents)
- Formally: a DAG (directed acyclic graph)
  - In most models: along hierarchical transitions

And/Or vs Or Graphs

- FSMs are **or graphs**: Take this action, or that action.
- Hierarchies open possibility for **and/or graphs**
  - **and node**: All children transitions must be taken

Execution Models

Two models of execution:

- **Layered**: Stack of threads (parallel actions)
  - RCS, ATLANTIS, BITE
- **Hierarchical**: Abstract actions, decomposed
  - Complex actions decomposed into more basic actions
  - HTN Planners, RAE executor (covered by T.A.)

Layered Execution Algorithm

(distributed in class)