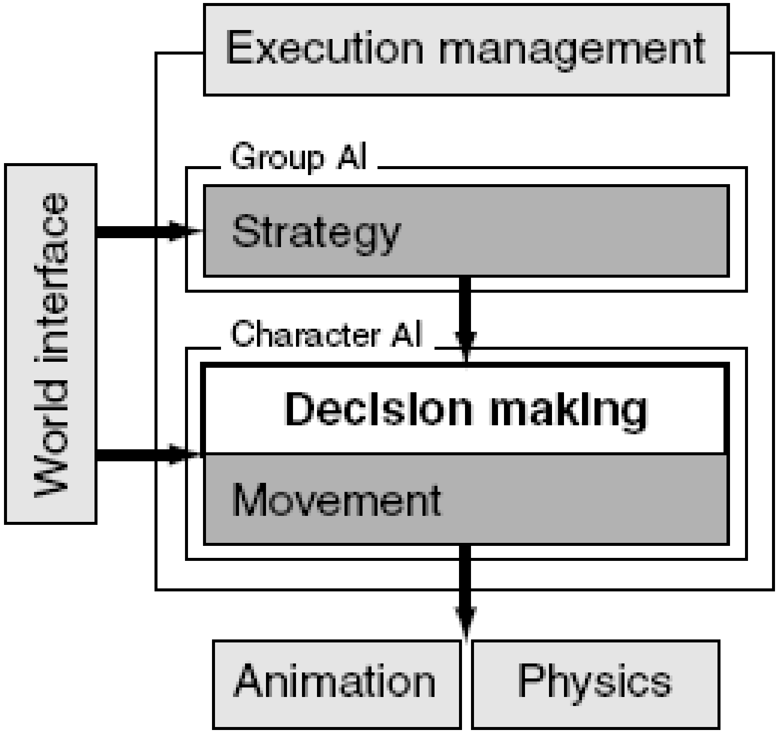
Intelligence I:
Basic Decision-Making Mechanisms

AI Architecture

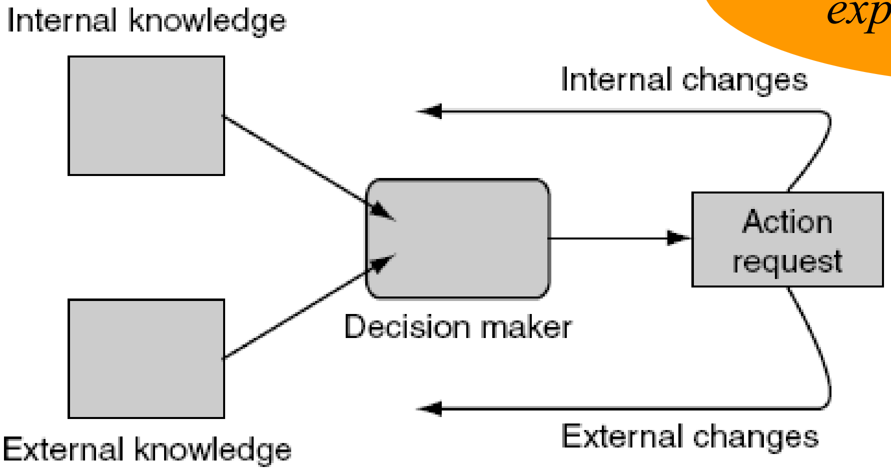


from “Artificial Intelligence for Games” by I. Millington & J. y ge

Decision-making Framework

e.g., health level, availability of ammunition, …

e.g., attack, flee, explore the sound, …



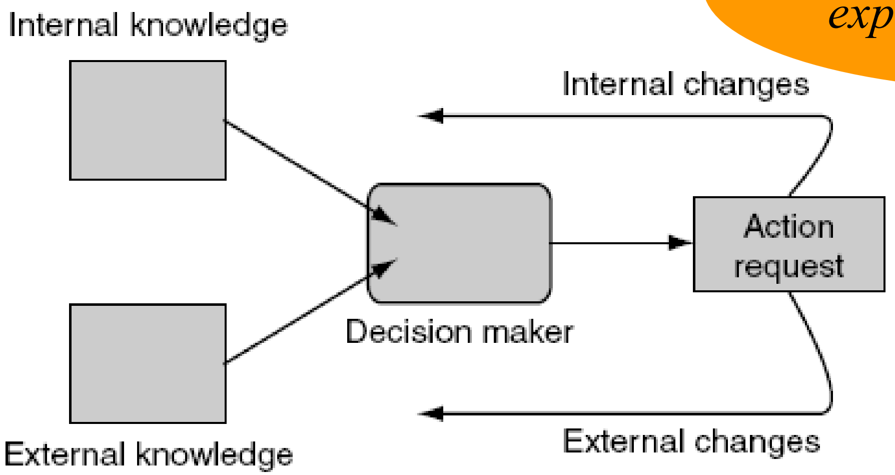
e.g., map, see enemy, hear sound, …

from “Artificial Intelligence for Games” by I. Millington & J. y ge

Decision-making Framework

e.g., health level, availability of ammunition, …

e.g., attack, flee, explore the sound, …



e.g., map, see enemy, hear sound, …

Any ideas for how to implement decision-making?

from "Artificial Intelligence for Games" by I. Millington & J. y ge

Basic Decision-making Mechanisms for this Class

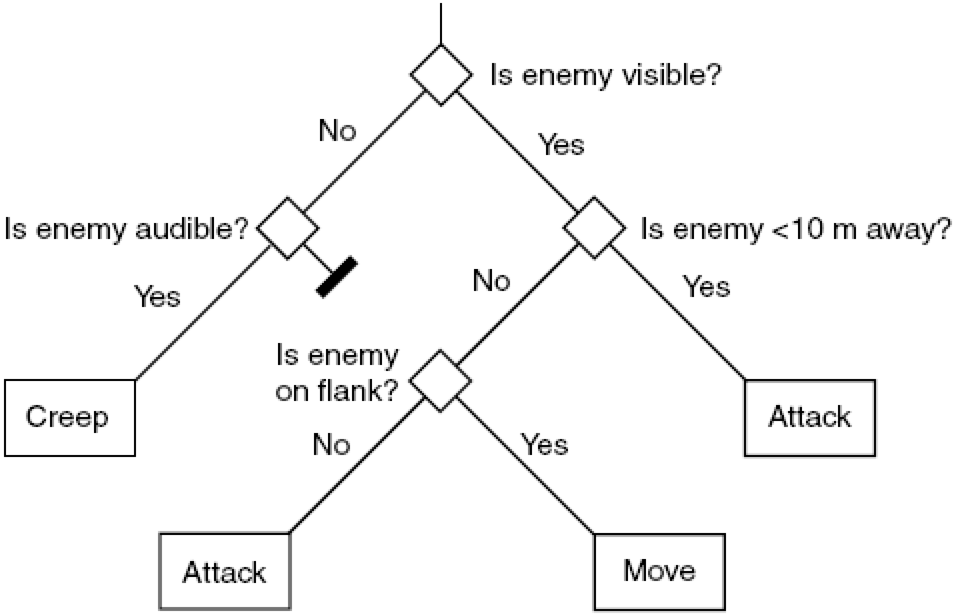
- Decision Trees
- Finite-state Machines
- Basic Behavior Trees

Basic Decision-making Mechanisms for this Class

- Decision Trees
- Finite-state Machines
- Basic Behavior Trees

Decision Trees

- Formalization of a set of nested if-then rules
- Very popular: easy-to-implement, intuitive (=easy-to-debug) and fast
- Require careful manual design (theoretically, learning trees is also possible)

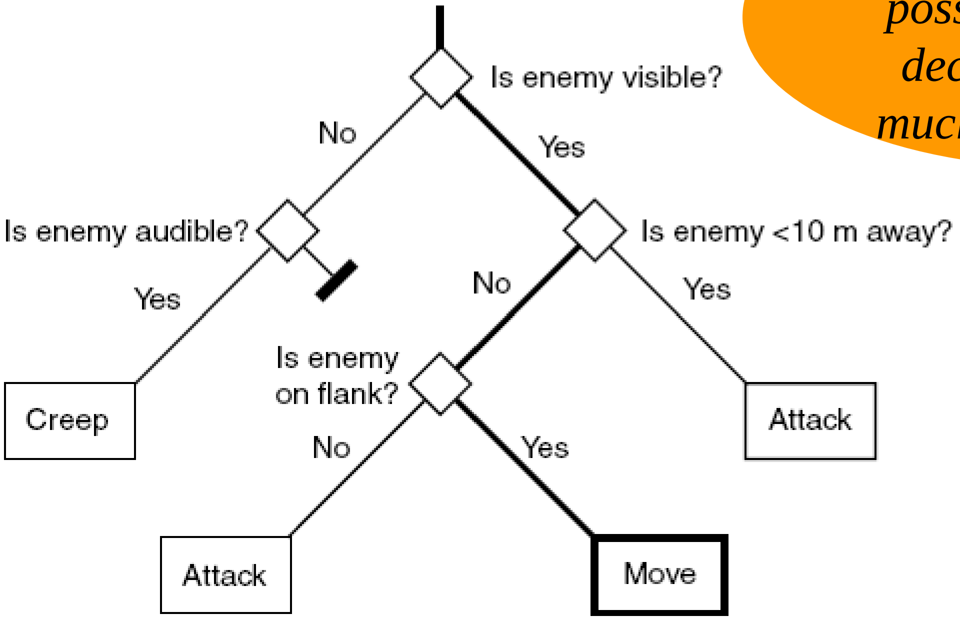


from “Artificial Intelligence for Games” by I. Millington & J. y ge

Decision Trees

- Formalization of a set of nested if-then rules
- Very popular: easy-to-implement, intuitive (=easy-to-debug) and fast
- Require careful manual design (theoretically, learning trees is also possible)

K-ary trees are possible but binary decision trees are much more common



from "Artificial Intelligence for Games" by I. Millington & J. y ge

Decision Trees

- Support for multi-valued input variables in binary decision-trees

Example:

Depending on the size of the enemy troops,
attack, stand ground or retreat

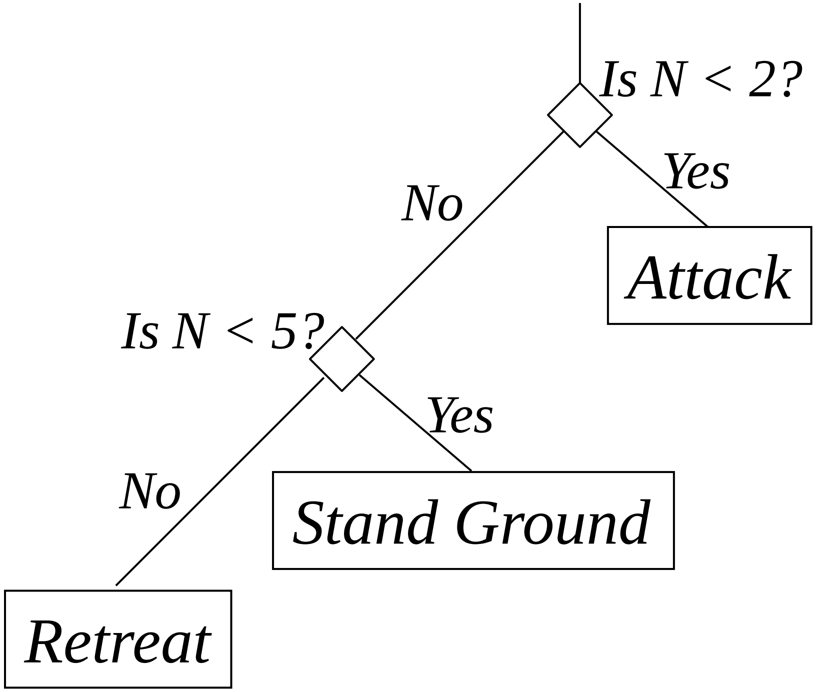
How to implement in a binary decision trees?

Decision Trees

- Support for multi-valued input variables in binary decision-trees

Example:

Depending on the size of the enemy troops, attack, stand ground or retreat



N=size of the enemy troops

Decision Trees

- Support for continuous input variables in binary decision-trees

Example:

Depending on the distance to the enemy,

hand-to-hand combat, shoot, hide

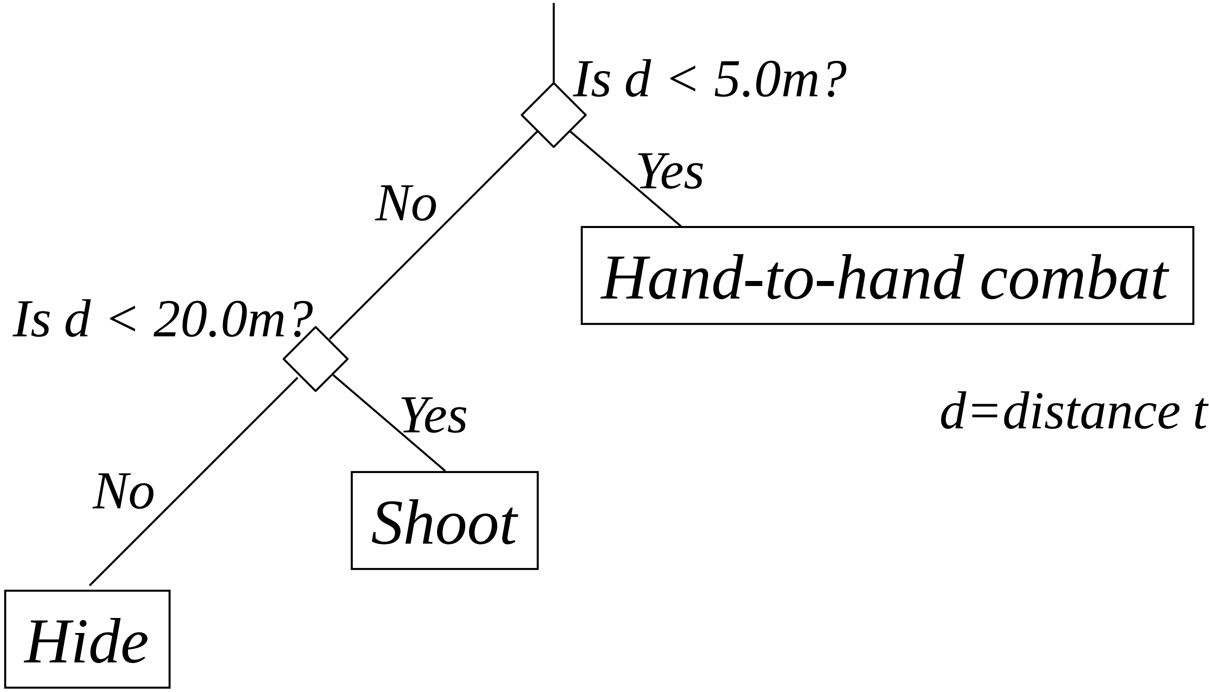
How to implement in a binary decision trees?

Decision Trees

- Support for continuous input variables in binary decision-trees

Example:

Depending on the distance to the enemy,
hand-to-hand combat, shoot, hide



d=distance to enemy

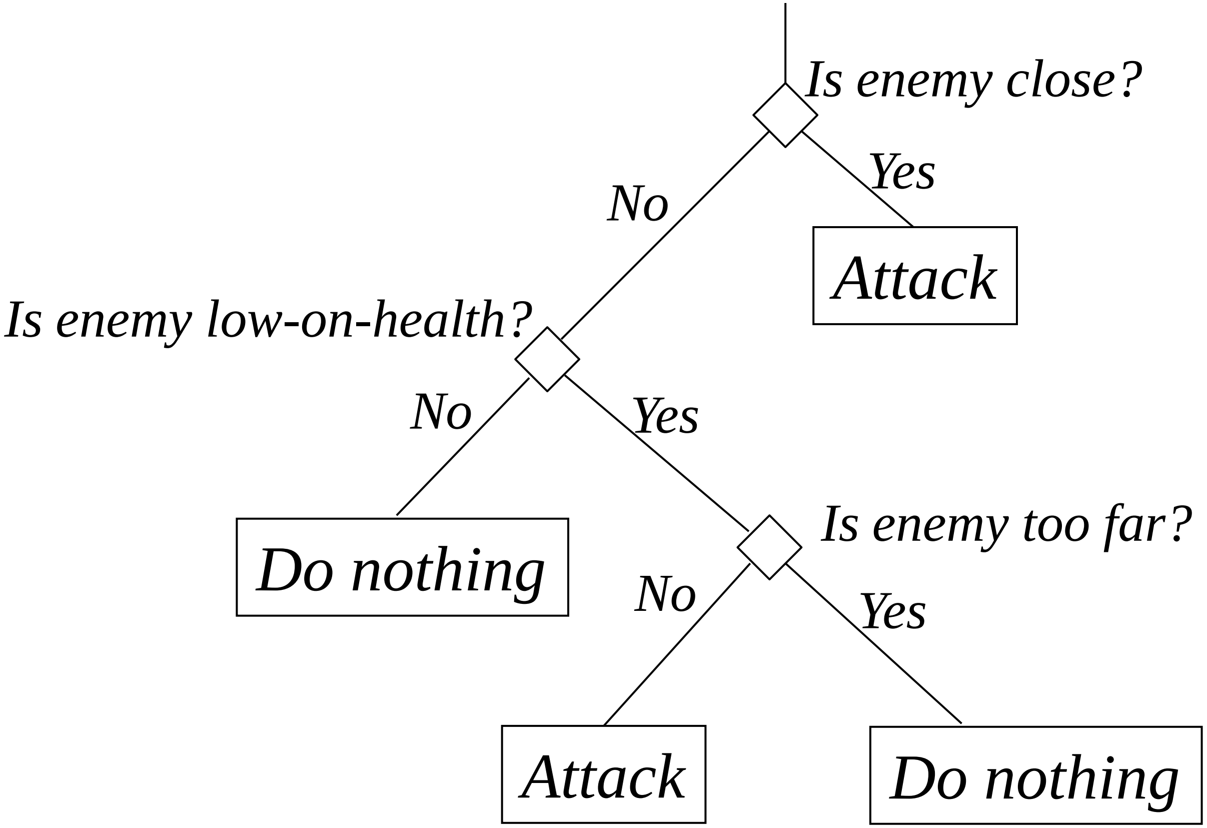
Decision Trees

- Support for complex decision formulae

Example:

Attack whenever

enemy is close OR (low-on-health AND not too far)



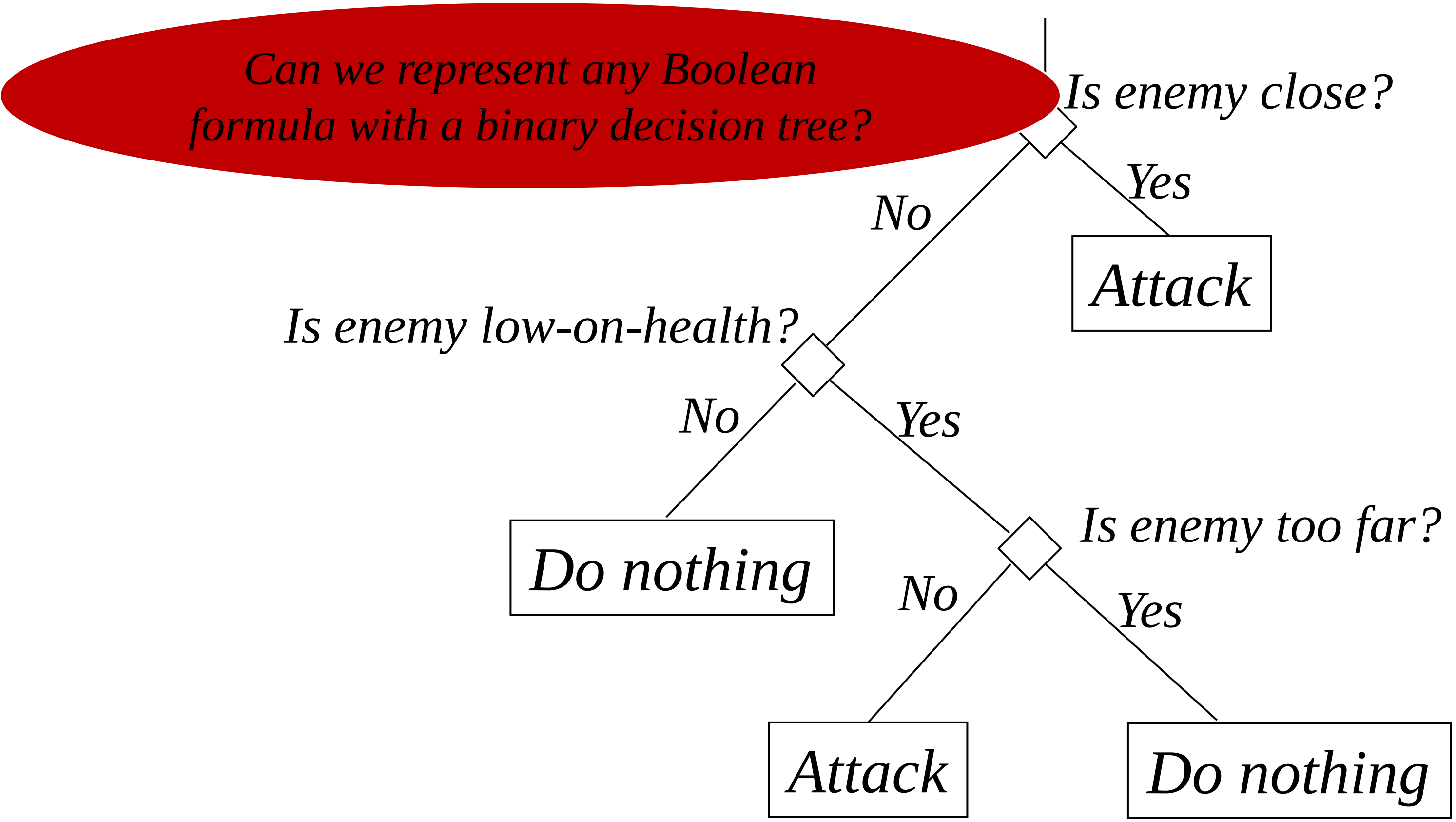
Decision Trees

- Support for complex decision formulae

Example:

Attack whenever

enemy is close OR (low-on-health AND not too far)



Can we represent any Boolean formula with a binary decision tree?

Decision Trees

- Support for complex decision formulae

Each row is a branch in the tree

True?

Example:

Attack whenever

enemy is close OR (low-on-health AND not too far)

‖

A OR (B AND C)

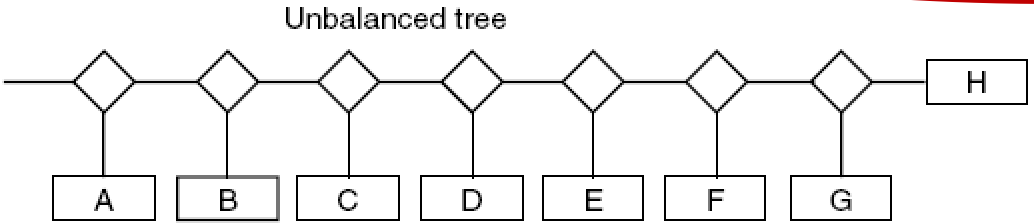
‖

A	B	C	Outcome
0	0	0	No
1	0	0	Yes
0	1	0	No
1	1	0	Yes
…	…	…	…

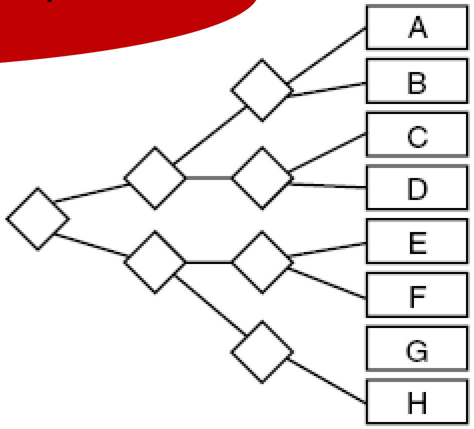
Decision Trees

- Making the decision tree traversal fast is important

Which decision tree is better for decision-making?



vs.



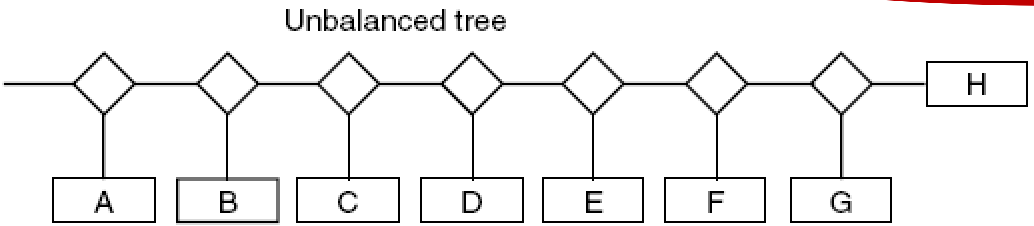
from "Artificial Intelligence for Games" by I. Millington & J. y ge

Decision Trees

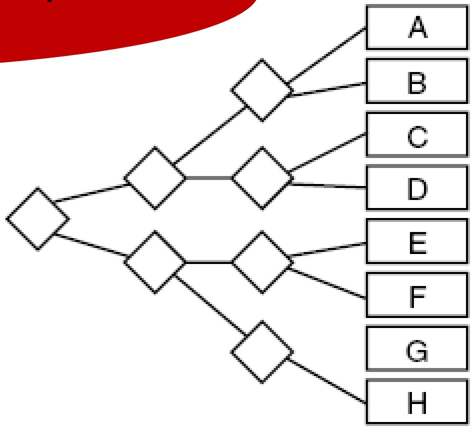
- Making the decision tree traversal fast is im

What does it depend on?

Which decision tree is better for decision-making?



vs.



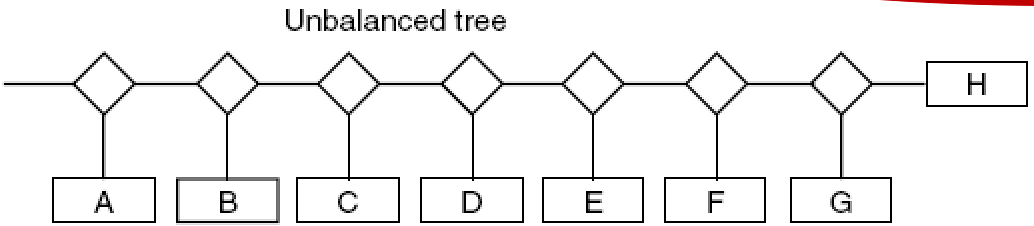
from "Artificial Intelligence for Games" by I. Millington & J. y ge

Decision Trees

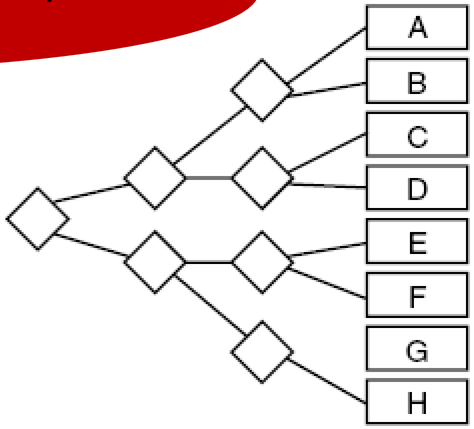
- Making the decision tree traversal fast is imp

What does it depend on?

Which decision tree is better for decision-making?



vs.



- Frequency (probability) of outcome (e.g., what if A happens 99% of the time)

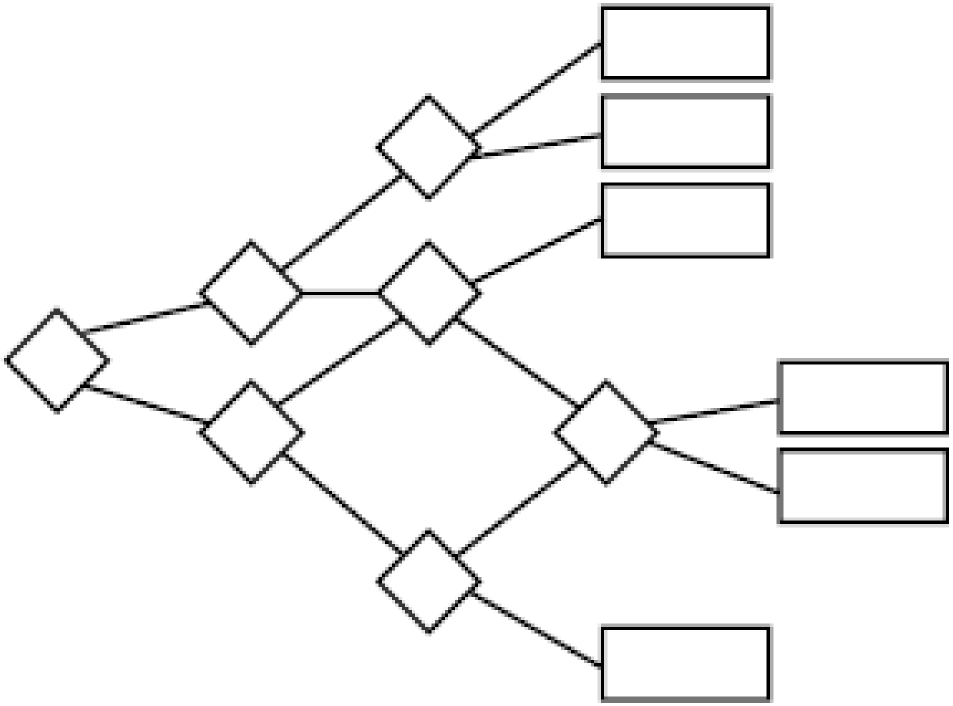
- The computational complexity of the test (e.g., what if testing for G is very expensive)

from “Artificial Intelligence for Games” by I. Millington & J. y ge

Decision Trees

- Making the decision tree traversal fast is important

Merging the branches:



from “Artificial Intelligence for Games” by I. Millington & J. y ge

Decision Trees

- How to deal with the predictability of AI?

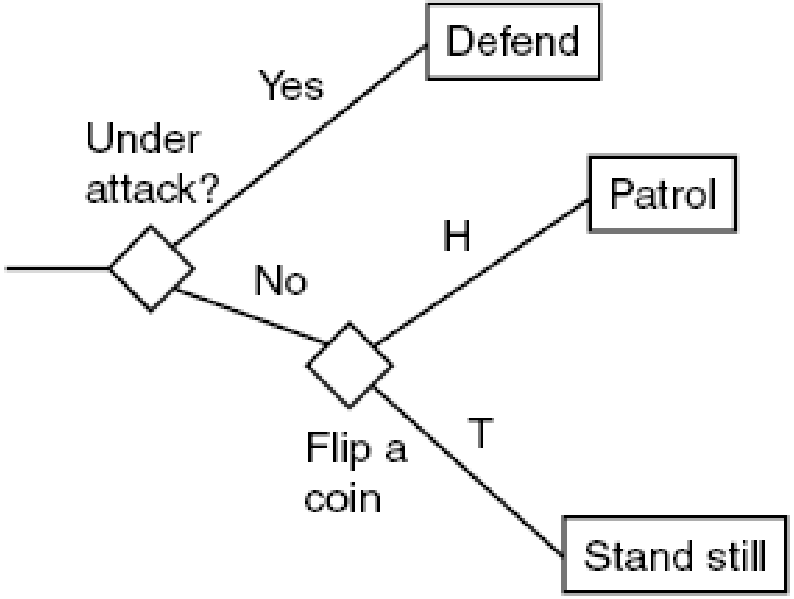
Any ideas?

Decision Trees

- How to deal with the predictability of AI?

  Random Decision Trees:

To avoid switching every frame: use hysteresis (memory)



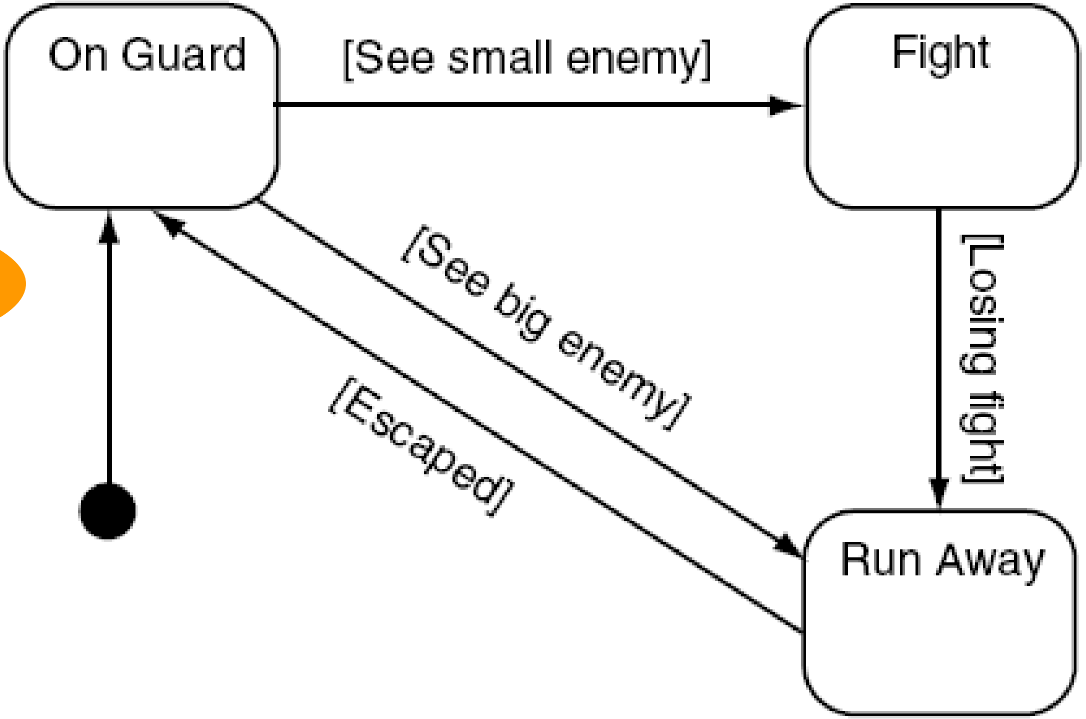
from “Artificial Intelligence for Games” by I. Millington & J. y ge

Basic Decision-making Mechanisms for this Class

- Decision Trees
- Finite-state Machines
- Basic Behavior Trees

Finite State Machines

- Basic FSM



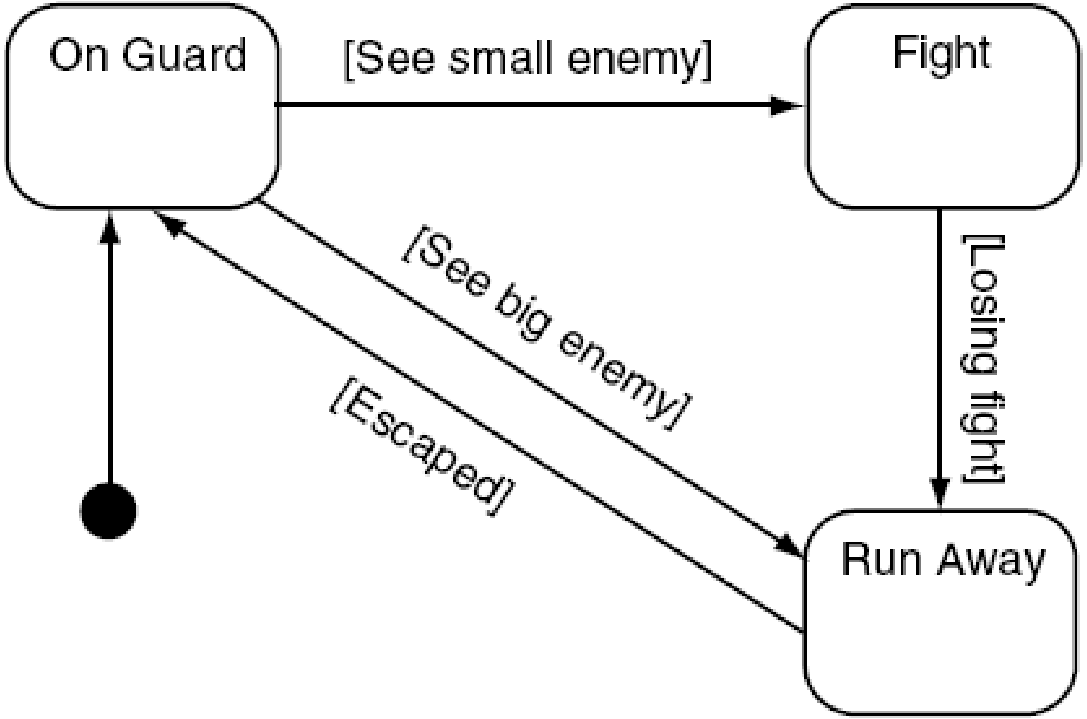
Initial State

from “Artificial Intelligence for Games” by I. Millington & J. y ge

Finite State Machines

- Basic FSM

Advantages over Decision Trees?



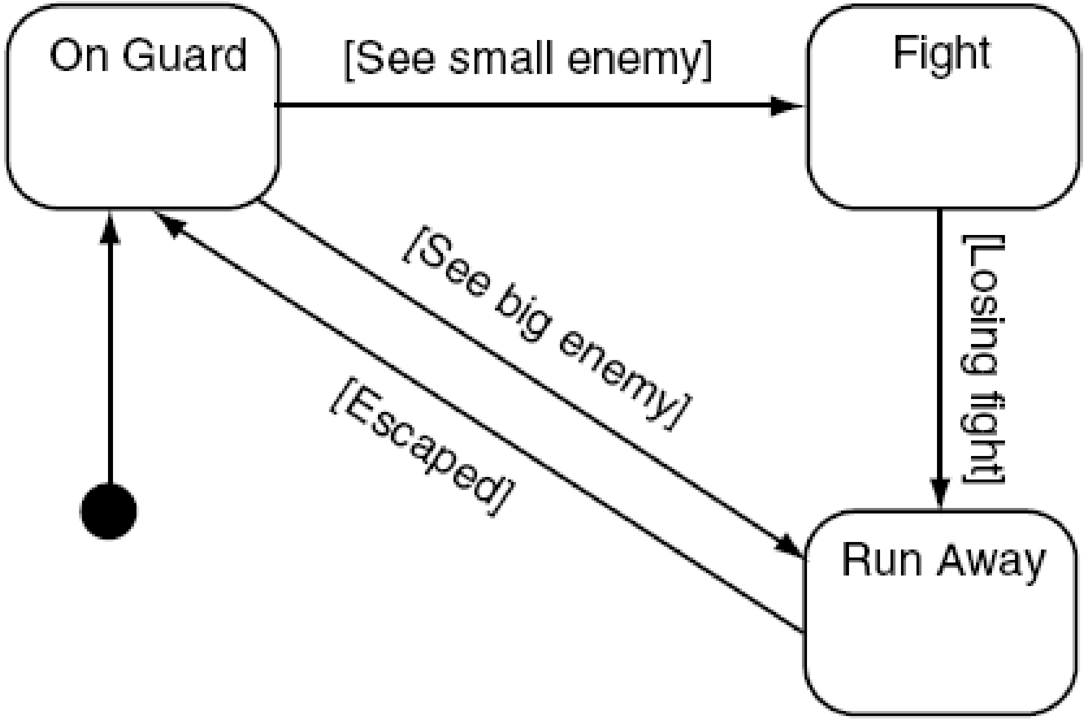
Disadvantages over Decision Trees?

from “Artificial Intelligence for Games” by I. Millington & J. y ge

Finite State Machines

- Basic FSM

How to introduce unpredictability?

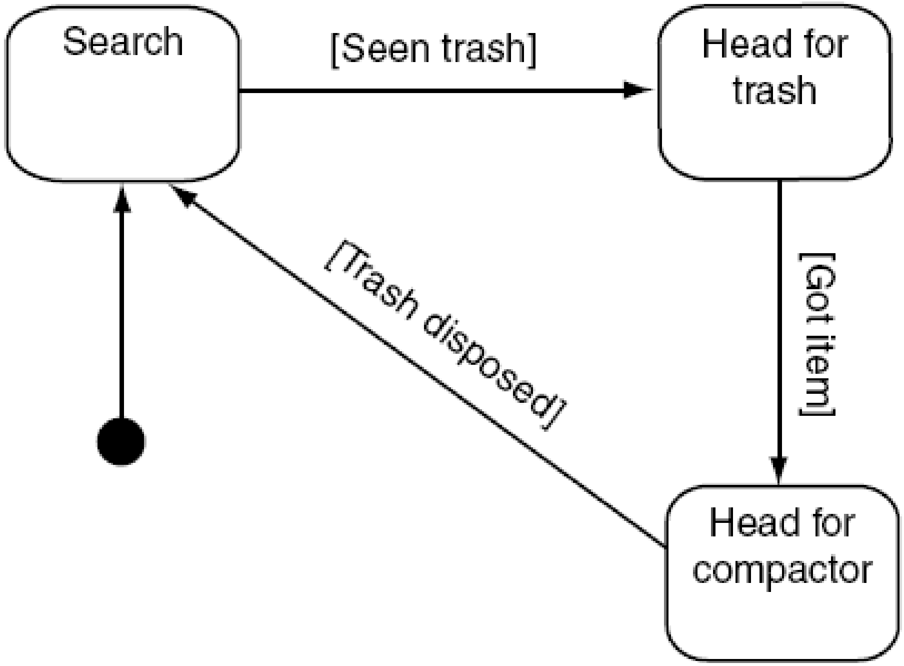


from “Artificial Intelligence for Games” by I. Millington & J. y ge

Finite State Machines

- Hierarchical FSM

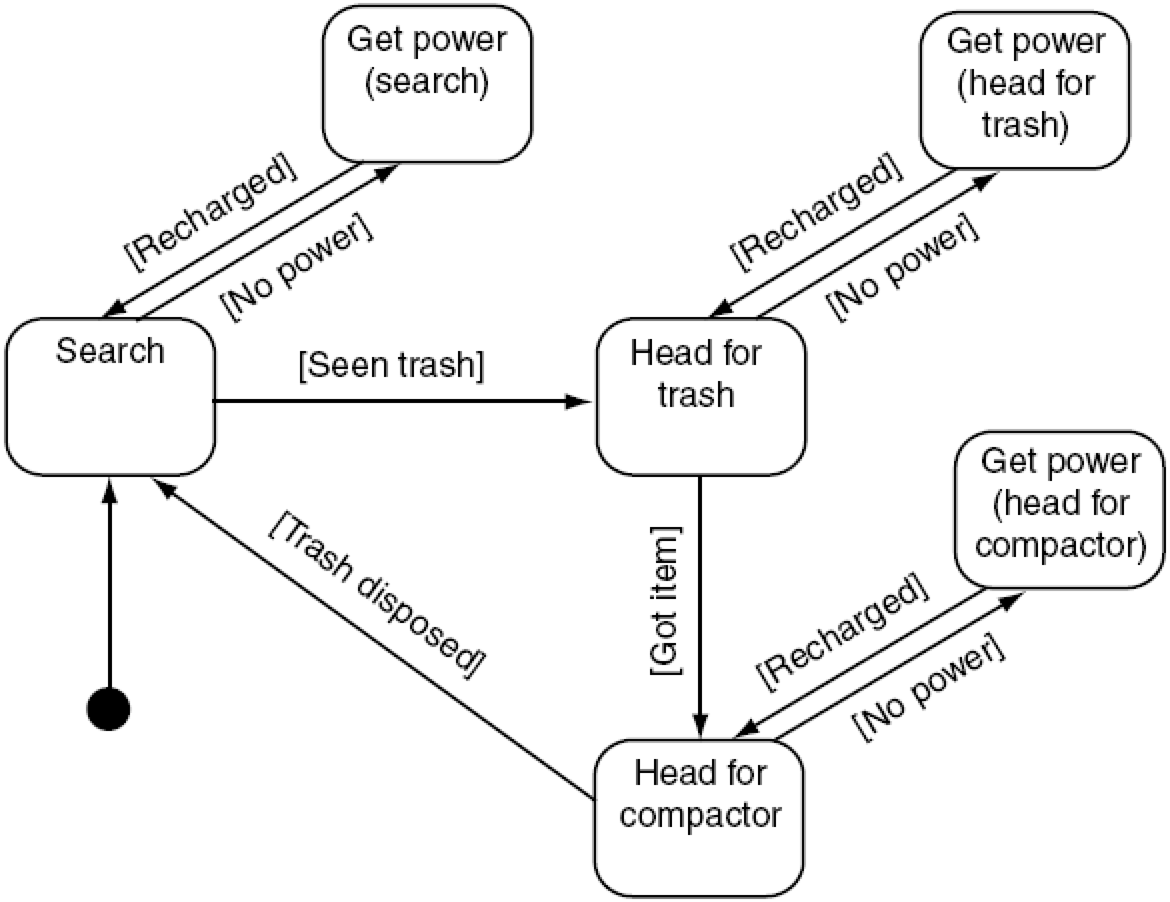
How it changes to react to “re-charge now” alarm?



from “Artificial Intelligence for Games” by I. Millington & J. y ge

Finite State Machines

- Hierarchical FSM



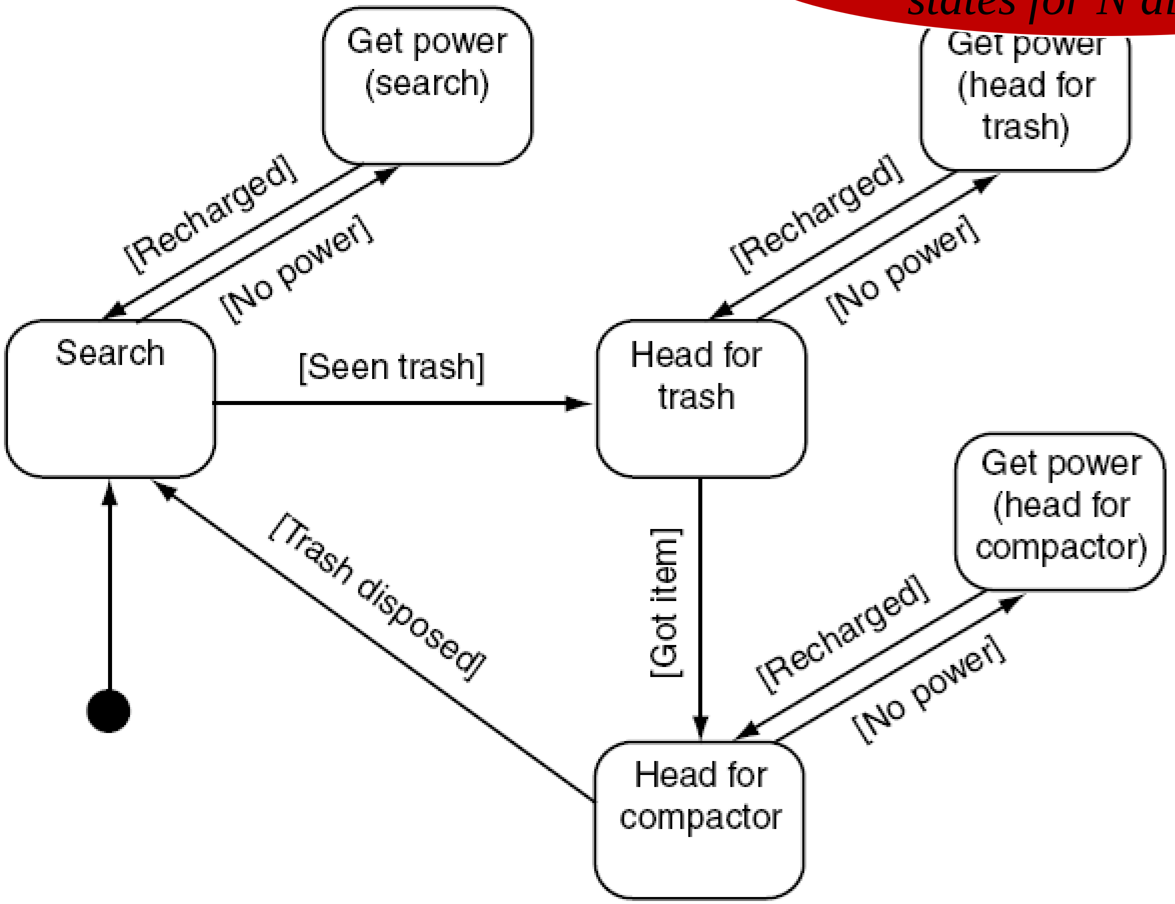
from “Artificial Intelligence for Games” by I. Millington & J. y ge

Finite State Machines

- Hierarchical FSM

What if an additional alarm?

Upper bound on # of states for N alarms?

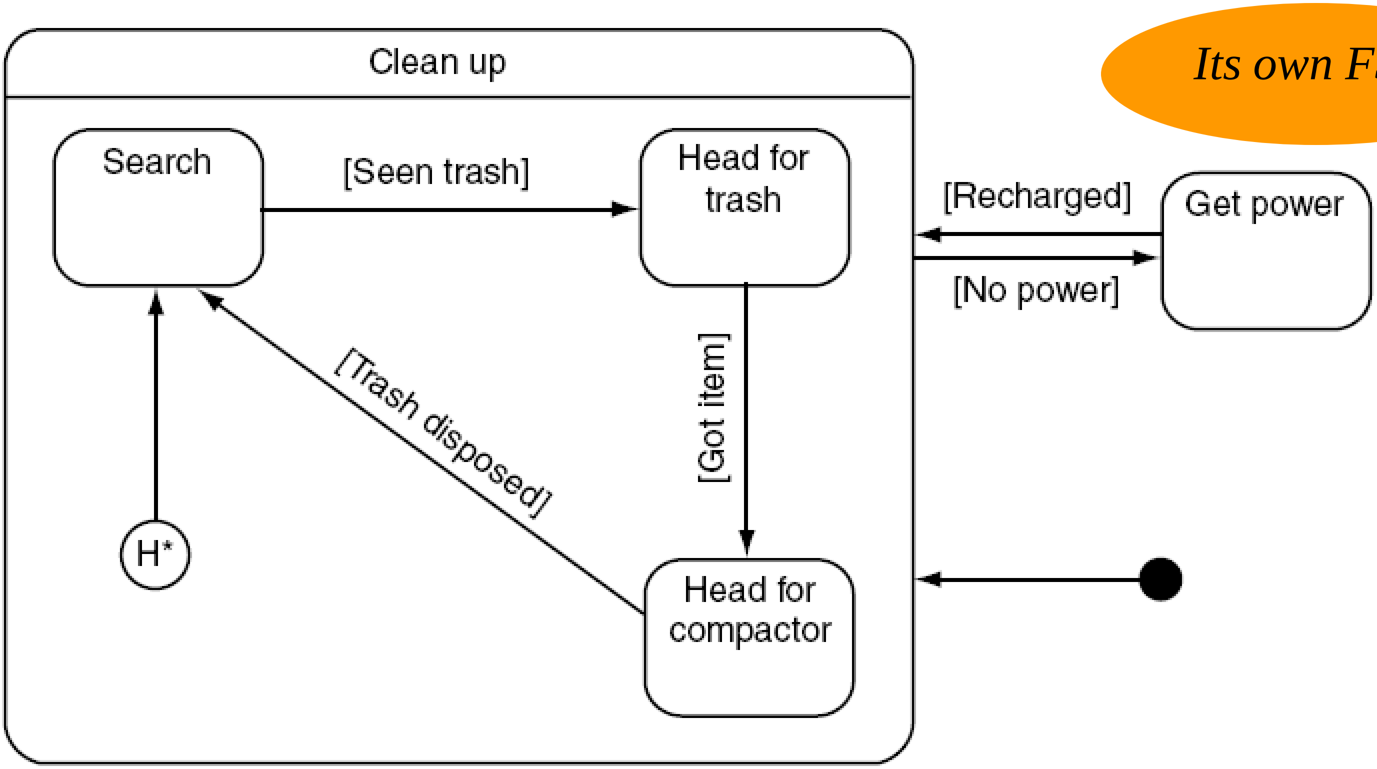


from “Artificial Intelligence for Games” by I. Millington & J. y ge

Finite State Machines

- Hierarchical FSM: strict hierarchy with only global alarms

State: [State at Level i, … State at Level 1] (for i ***active*** *FSMs)*
All triggers get acted upon by FSM at level i
Whenever FSM at Level i exits, FSM at level i-1 becomes dominant



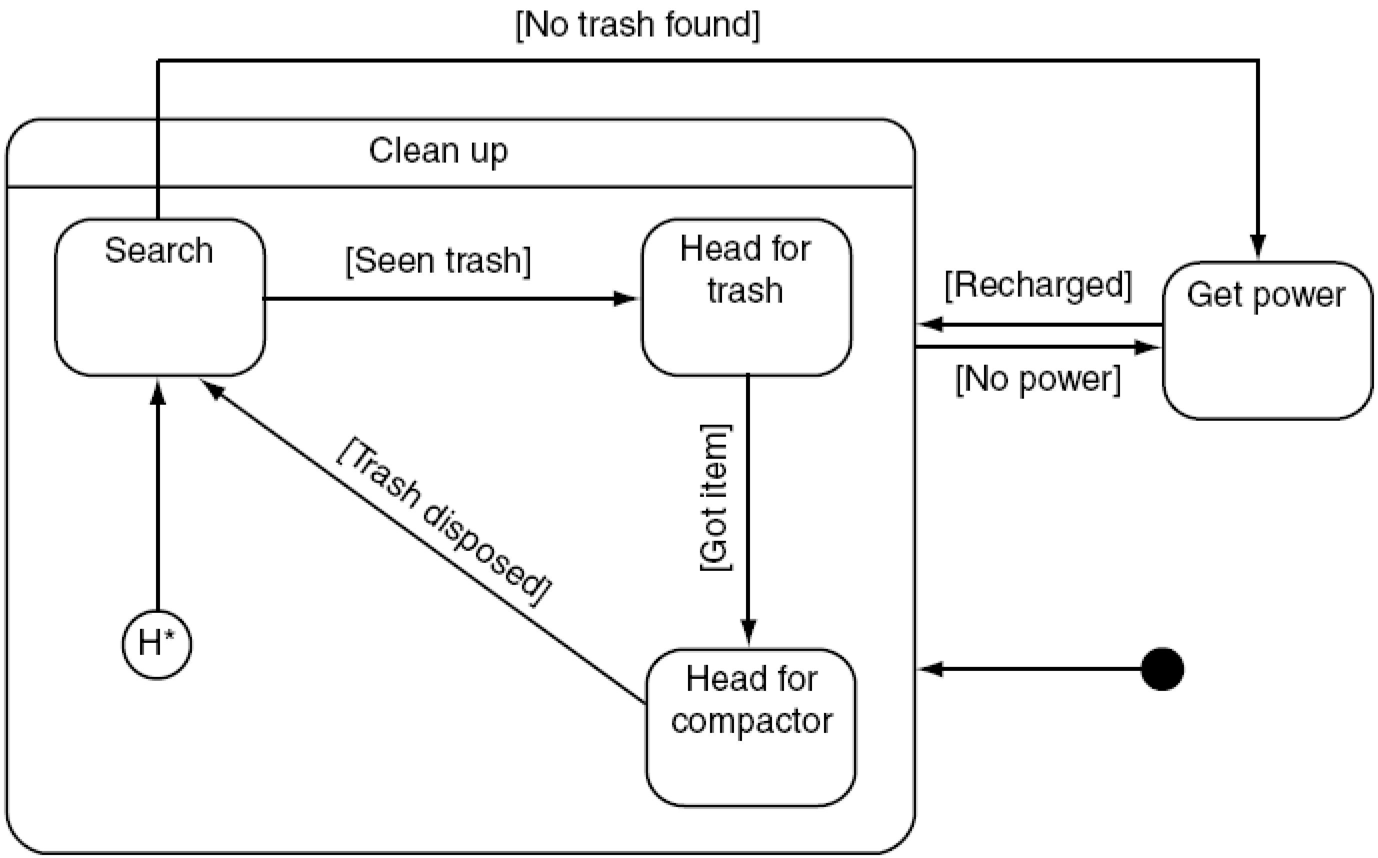
Its own FSM

from "Artificial Intelligence for Games" by I. Millington & J. y ge

Finite State Machines

- Hierarchical FSM: strict hierarchy with additional direct transitions

Direct transitions between levels allow to ***leave*** *the source state*



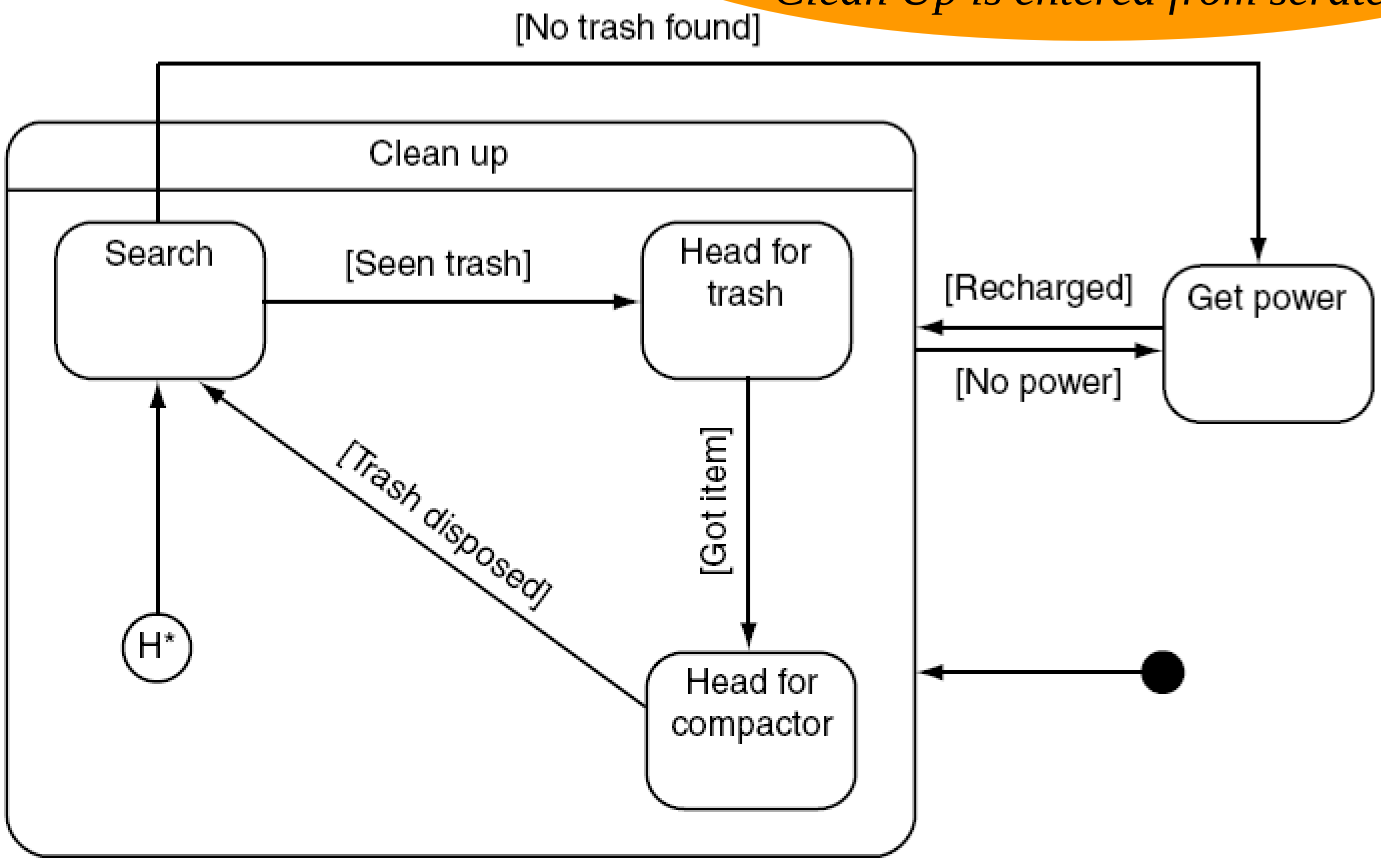
from “Artificial Intelligence for Games” by I. Millington & J. y ge

Finite State Machines

- Hierarchical FSM: strict hierarchy with additional direct transitions

Direct transitions between lev

Search state is left
When GetPower is done,
Clean Up is entered from scratch

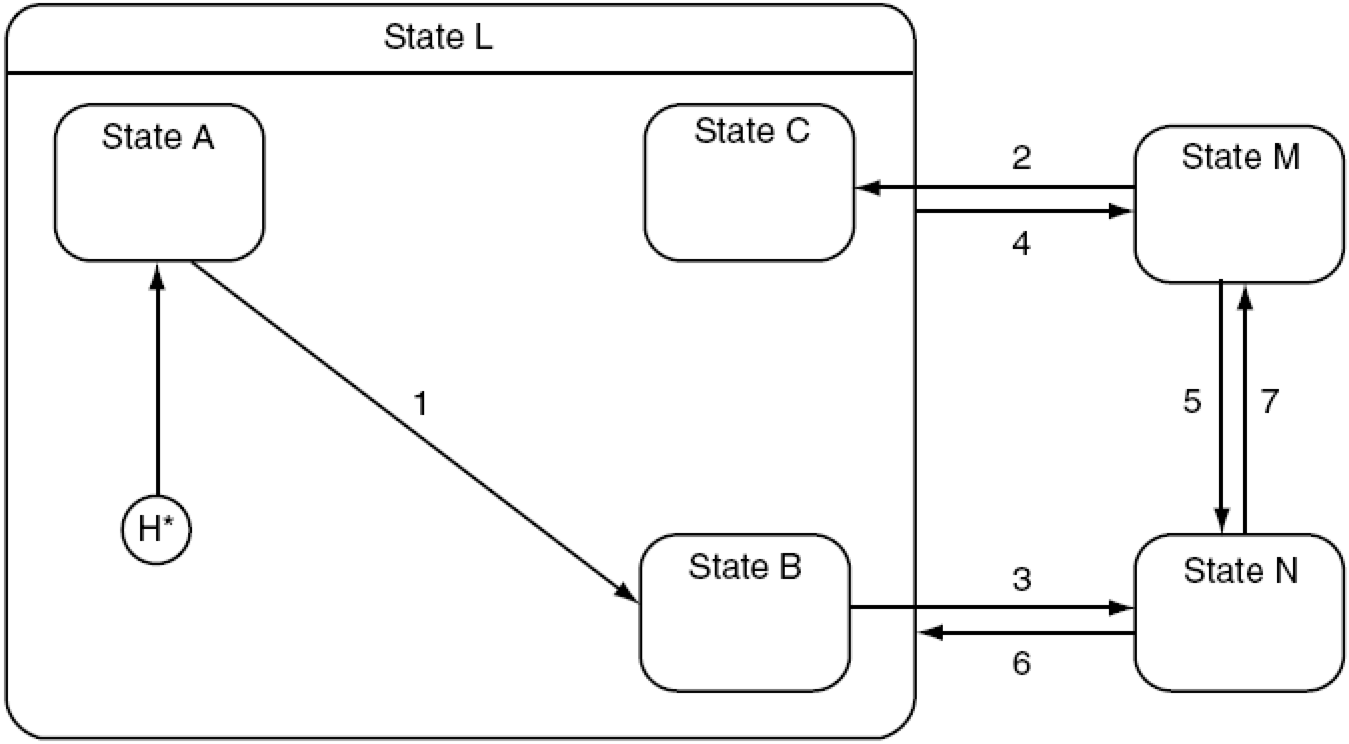


from "Artificial Intelligence for Games" by I. Millington & J. y ge

Finite State Machines

- Hierarchical FSM: strict hierarchy with additional direct transitions

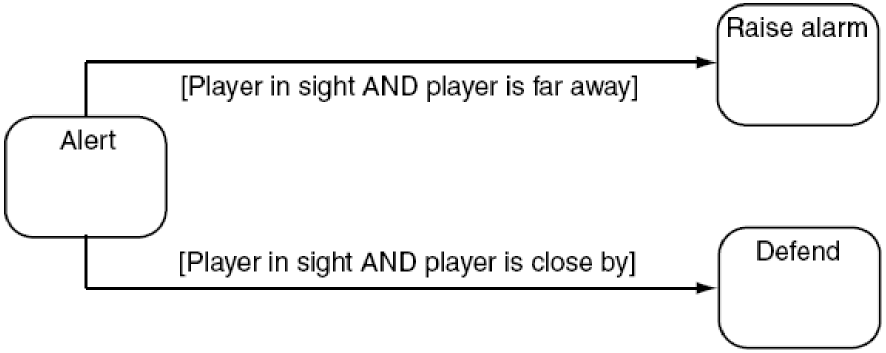
More complex example:



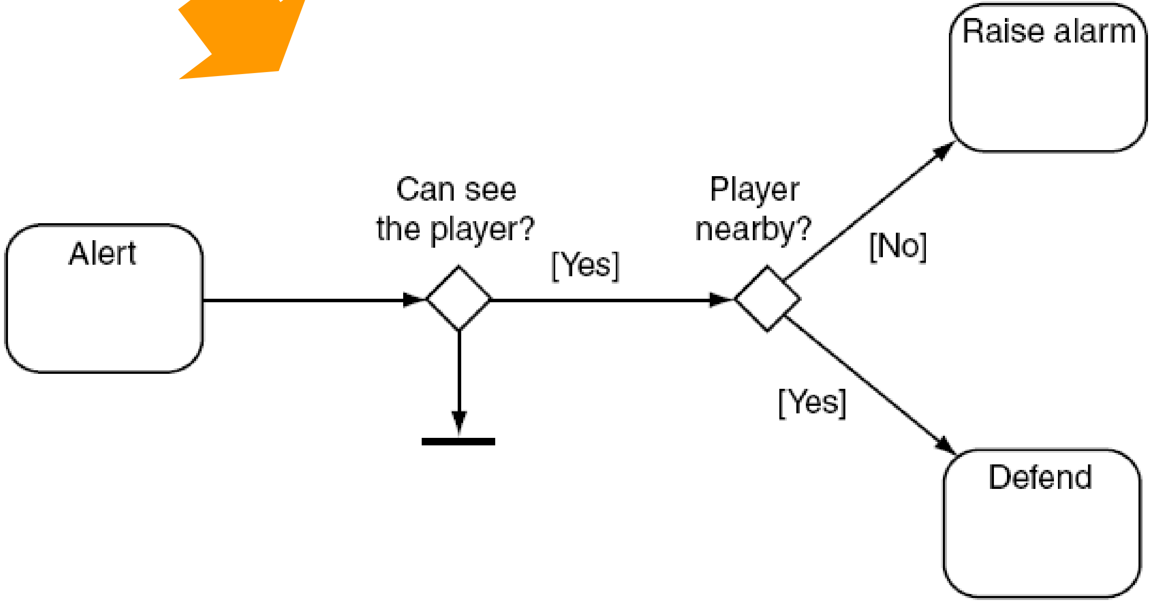
from "Artificial Intelligence for Games" by I. Millington & J. y ge

Finite State Machines

- Combining FSM and decision trees


Raise alarm
[Player in sight AND player is far away]
Alert
[Player in sight AND player is close by]
Defend

Why bother?



from "Artificial Intelligence for Games" by I. Millington & J. y ge

Basic Decision-making Mechanisms for this Class

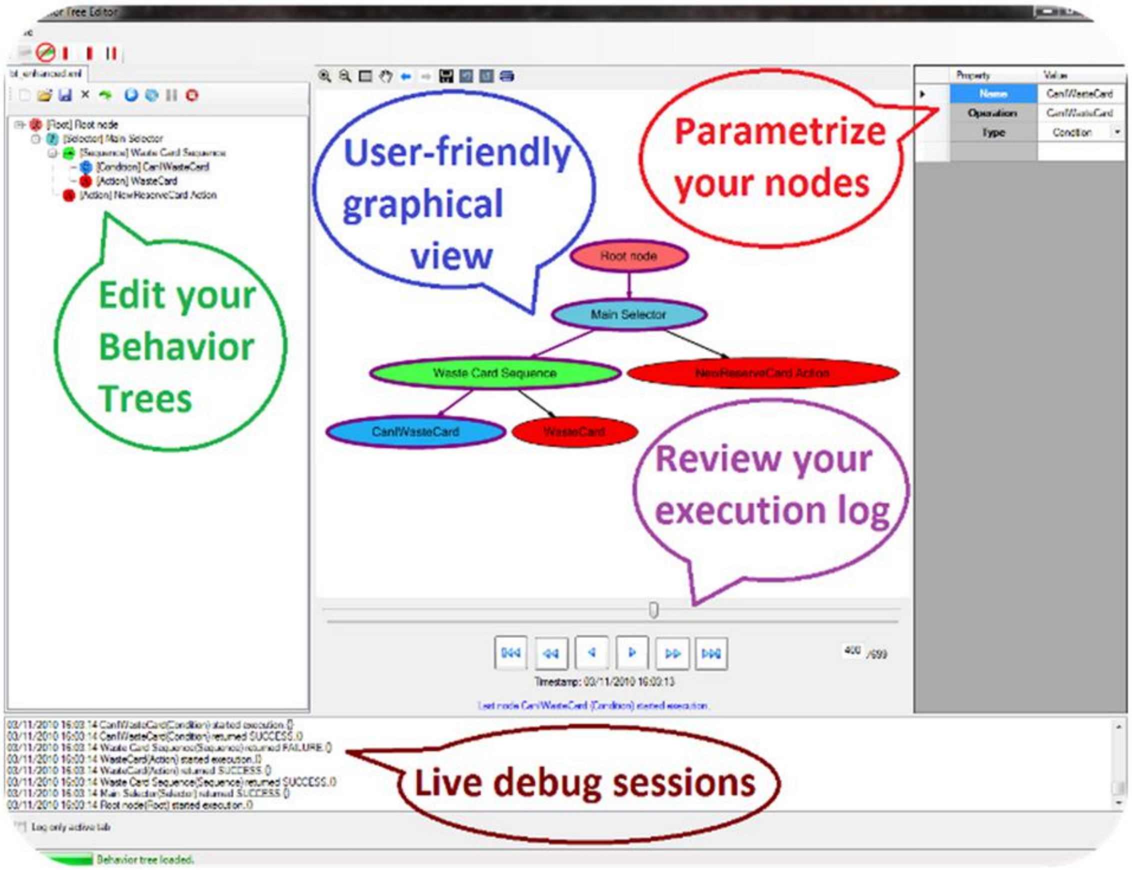
- Decision Trees
- Finite-state Machines
- Basic Behavior Trees

Basic Behavior Trees

- Became very popular after Halo 2 game [2004]

Basic Behavior Trees

- Became very popular after Halo 2 game [2004]
- Especially, when coupled with graphical interfaces to edit them



Property	Value
Name	CanIWasteCard
Operation	CanIWasteCard
Type	Condition

Timestamp: 03/11/2010 16:03:13

Last node CanIWasteCard (Condition) started execution.

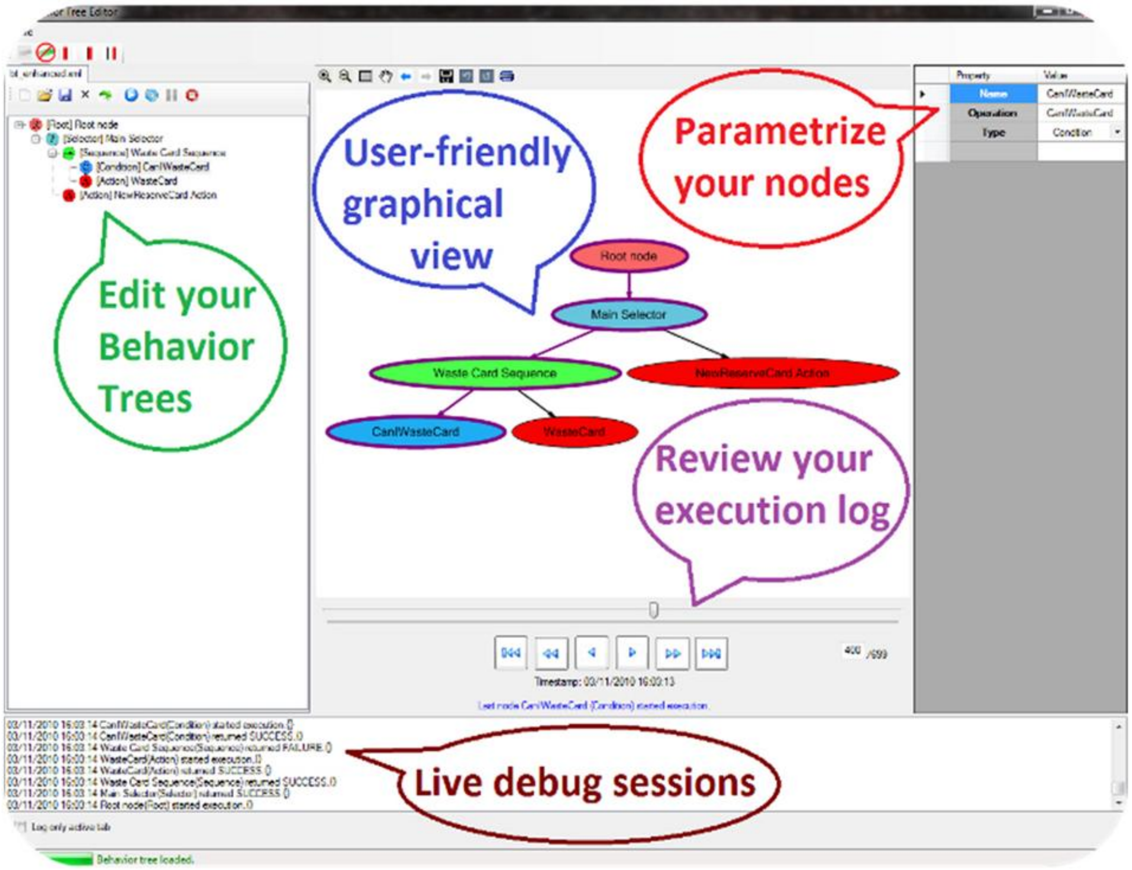
```
03/11/2010 16:03:14 CanIWasteCard(Condition) started execution.()
03/11/2010 16:03:14 CanIWasteCard(Condition) returned SUCCESS.()
03/11/2010 16:03:14 Waste Card Sequence(Sequence) returned FAILURE.()
03/11/2010 16:03:14 WasteCard(Action) started execution.()
03/11/2010 16:03:14 WasteCard(Action) returned SUCCESS.()
03/11/2010 16:03:14 Waste Card Sequence(Sequence) returned SUCCESS.()
03/11/2010 16:03:14 Main Selector(Selector) returned SUCCESS.()
03/11/2010 16:03:14 Root node(Root) started execution.()
```

Behavior tree loaded.

Screenshot of GameBrains GUI

Basic Behavior Trees

- Collection of simple tasks arranged in a tree structure
- One of the main advantages: Tasks and sub-trees can be reused!!!



[Root] Root node
[Selector] Main Selector
[Sequence] Waste Card Sequence
[Condition] CanIWasteCard
[Action] WasteCard
[Action] NewReserveCard Action

Property	Value
Name	CanIWasteCard
Operation	CanIWasteCard
Type	Condition

Timestamp: 03/11/2010 16:03:13

Last node CanIWasteCard (Condition) started execution.

```
03/11/2010 16:03:14 CanIWasteCard(Condition) started execution.()
03/11/2010 16:03:14 CanIWasteCard(Condition) returned SUCCESS.()
03/11/2010 16:03:14 Waste Card Sequence(Sequence) returned FAILURE.()
03/11/2010 16:03:14 WasteCard(Action) started execution.()
03/11/2010 16:03:14 WasteCard(Action) returned SUCCESS.()
03/11/2010 16:03:14 Waste Card Sequence(Sequence) returned SUCCESS.()
03/11/2010 16:03:14 Main Selector(Selector) returned SUCCESS.()
03/11/2010 16:03:14 Root node(Root) started execution.()
```

☐ Log only active tab

Behavior tree loaded.

Screenshot of GameBrains GUI

Basic Behavior Trees

- Type of tasks: *Conditions, Actions and Composites*
- Each returning either success or failure

*“**Condition**” task tests for a condition*

Examples of behavior trees that consist of Conditions tasks only:

Door open?

Health level OK?

Enemy close-by?

…

Basic Behavior Trees

- Type of tasks: *Conditions*, *Actions and Composites*
- Each returning either success or failure

*“**Action**” task alters the state of the game*

Examples of behavior trees that consist of Actions tasks only:

Move to room *Find a path* *Play audio sound* *Talk to the player* …

Basic Behavior Trees

- Type of tasks: *Conditions, Actions and Composites*
- Each returning either success or failure

Condition and Action tasks are always at the leafs of the tree
*"**Composite**" task sequences through them*

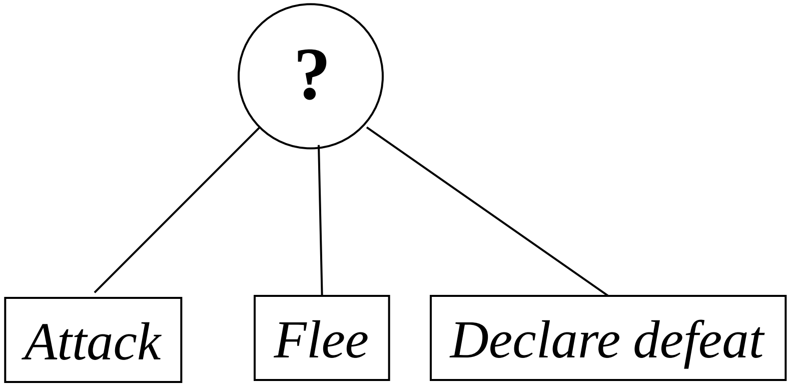
Basic Behavior Trees

- Type of tasks: *Conditions, Actions and Composites*
- Each returning either success or failure

Condition and Action tasks are always at the leafs of the tree
*"**Composite**" task sequences through them*

Two types of Composite tasks:

***Selector** returns as soon as the first leaf task is successful*



Basic Behavior Trees

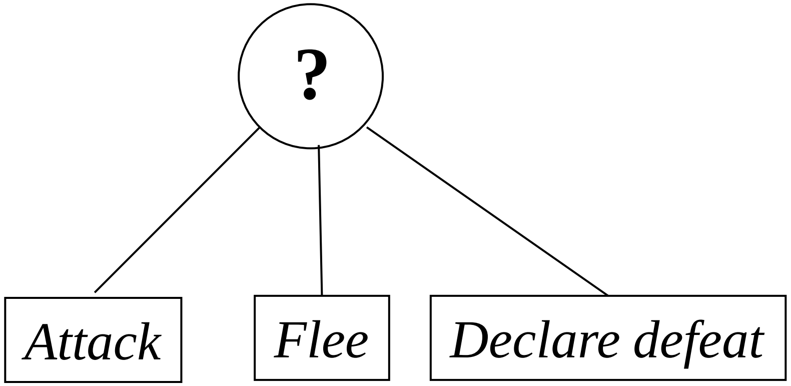
- Type of tasks: *Conditions, Actions and Composites*
- Each returning either success or failure

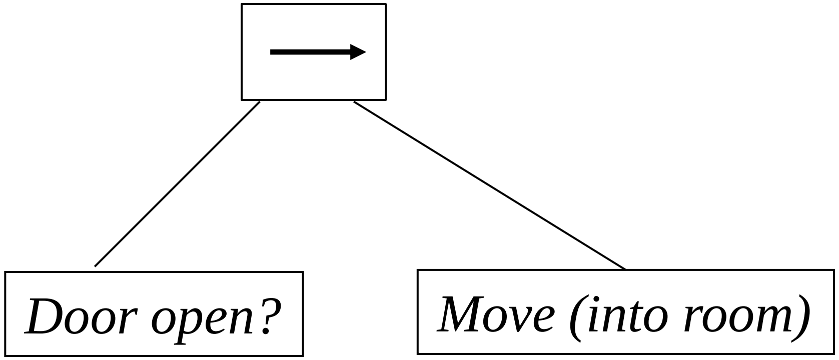
Condition and Action tasks are always at the leafs of the tree
*“**Composite**” task sequences through them*

Two types of Composite tasks:

***Selector** returns as soon as the first leaf task is successful*

***Sequencer** returns as soon as the first leaf task fails*





Basic Behavior Trees

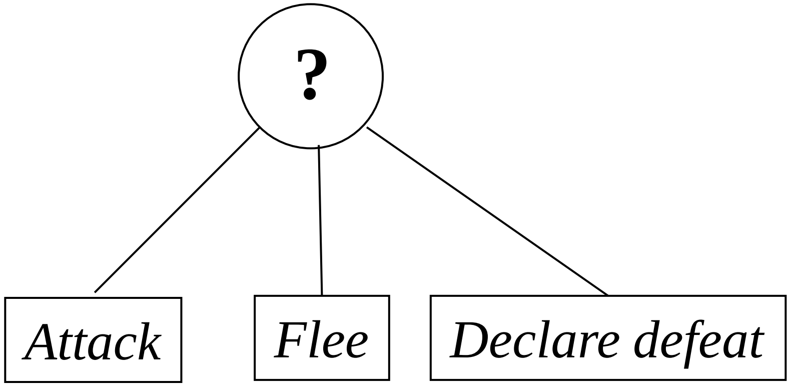
- Type of tasks: *Conditions, Actions and Composites*
- Each returning either success or failure

Condition and Action tasks are always at the leafs of the tree
*"**Composite**" task sequences through them*

Two types of Composite tasks:

***Selector** returns as soon as the first leaf task is successful*

***Sequencer** returns as soon as the first leaf task fails*

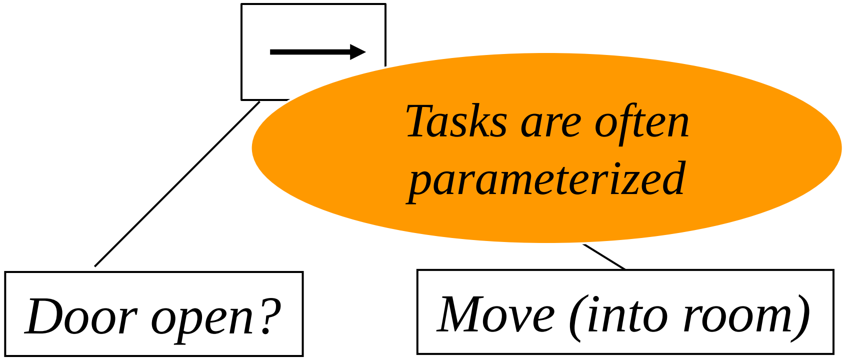


?

Attack

Flee

Declare defeat



⟶

Door open?

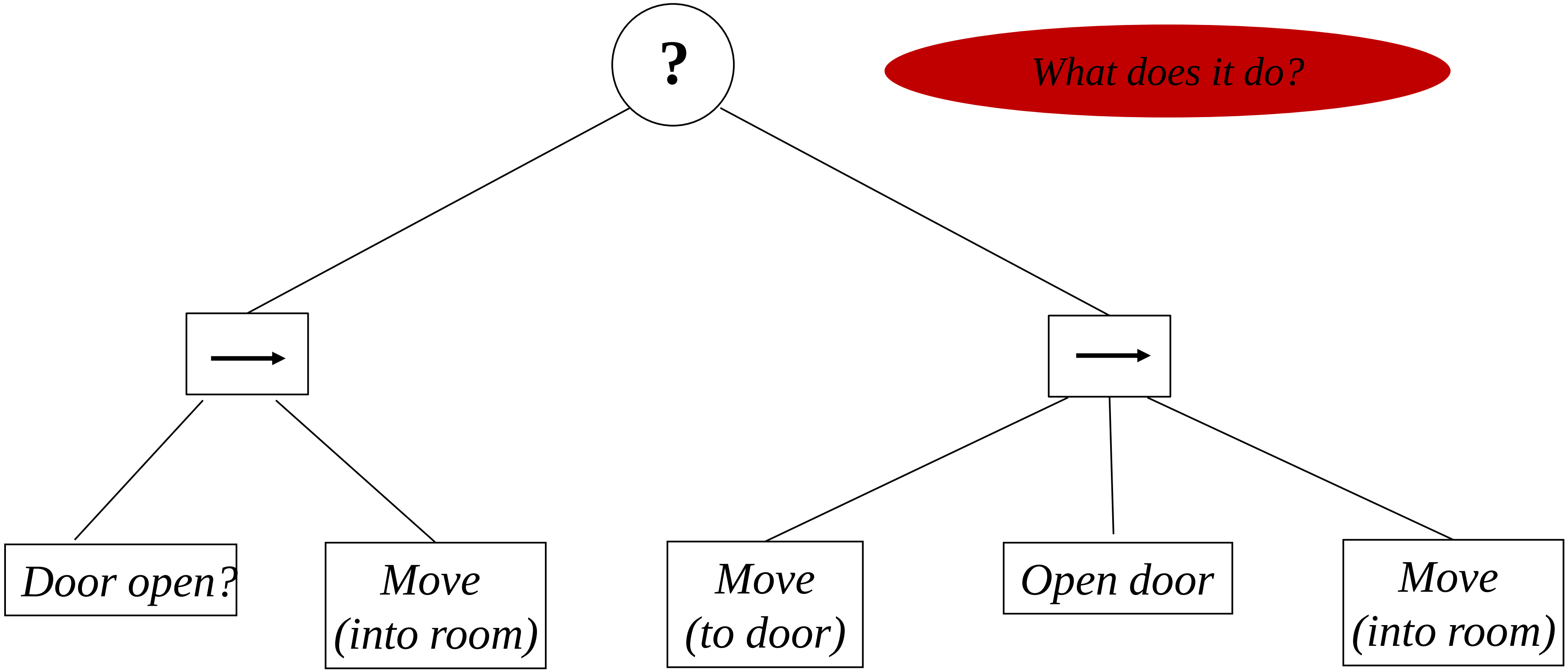
Move (into room)

Tasks are often parameterized

Basic Behavior Trees

- Type of tasks: *Conditions, Actions and Composites*
- Each returning either success or failure

Example:

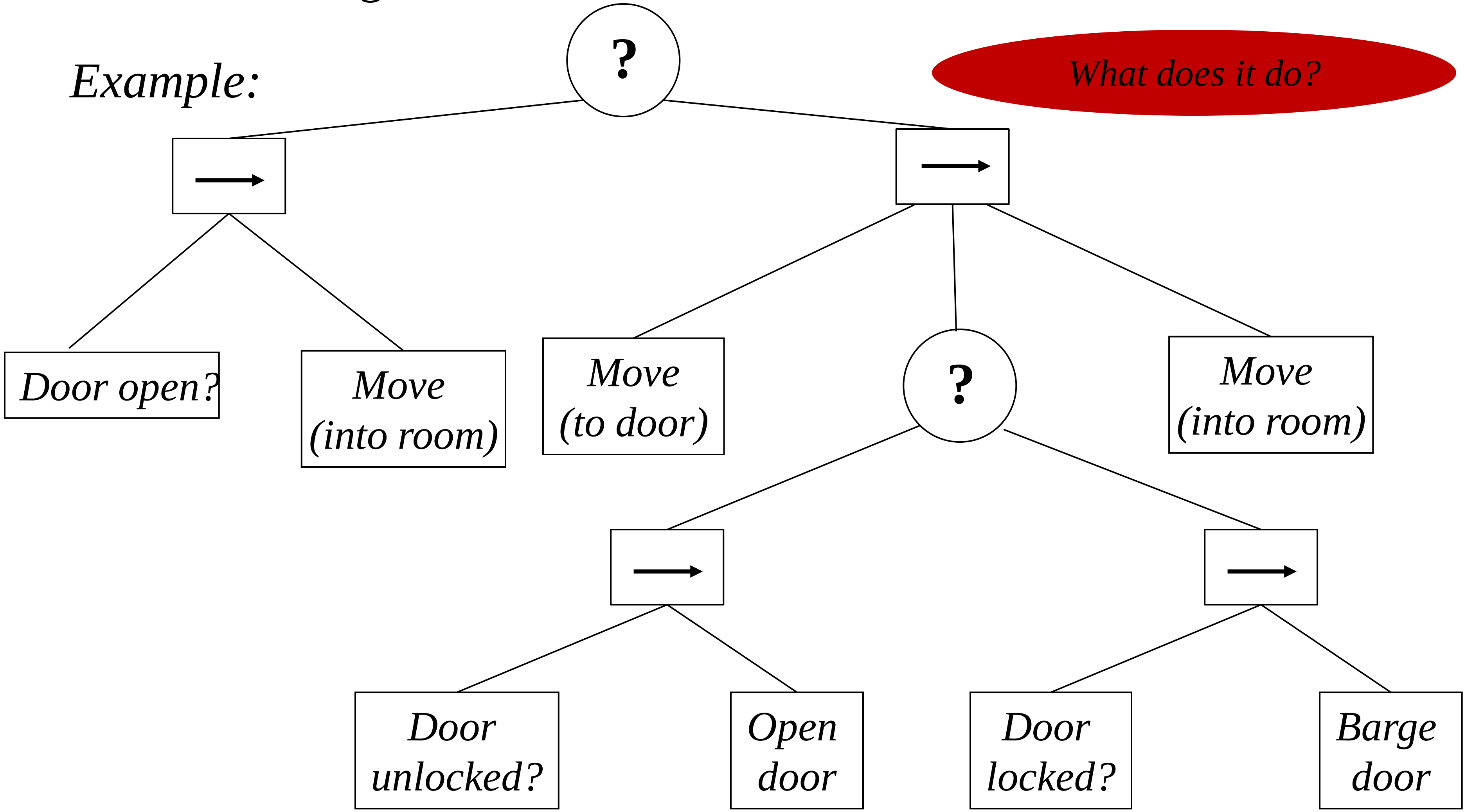


What does it do?

Basic Behavior Trees

- Type of tasks: *Conditions, Actions and Composites*
- Each returning either success or failure

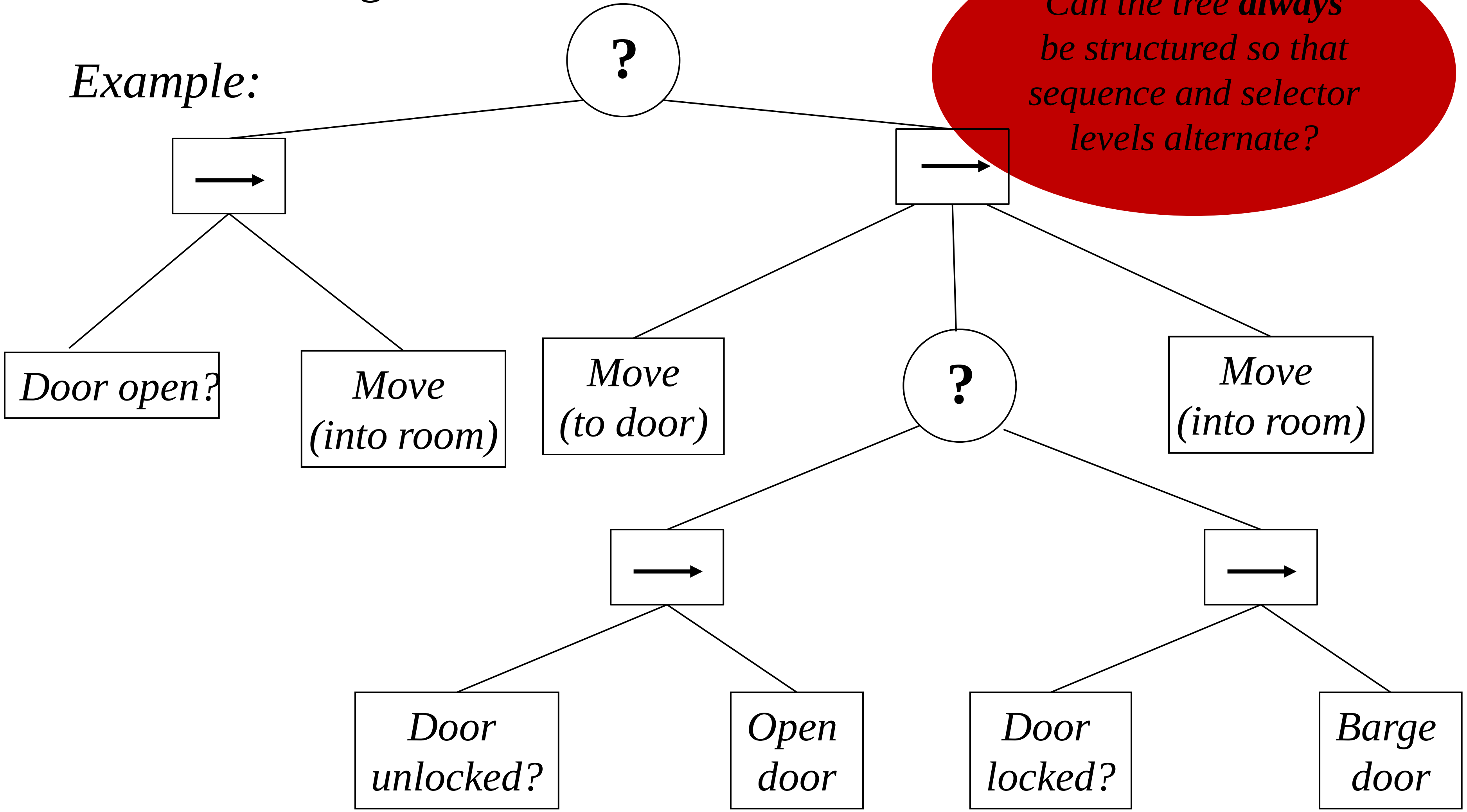
Example:



Basic Behavior Trees

- Type of tasks: *Conditions, Actions and Composites*
- Each returning either success or failure

Example:

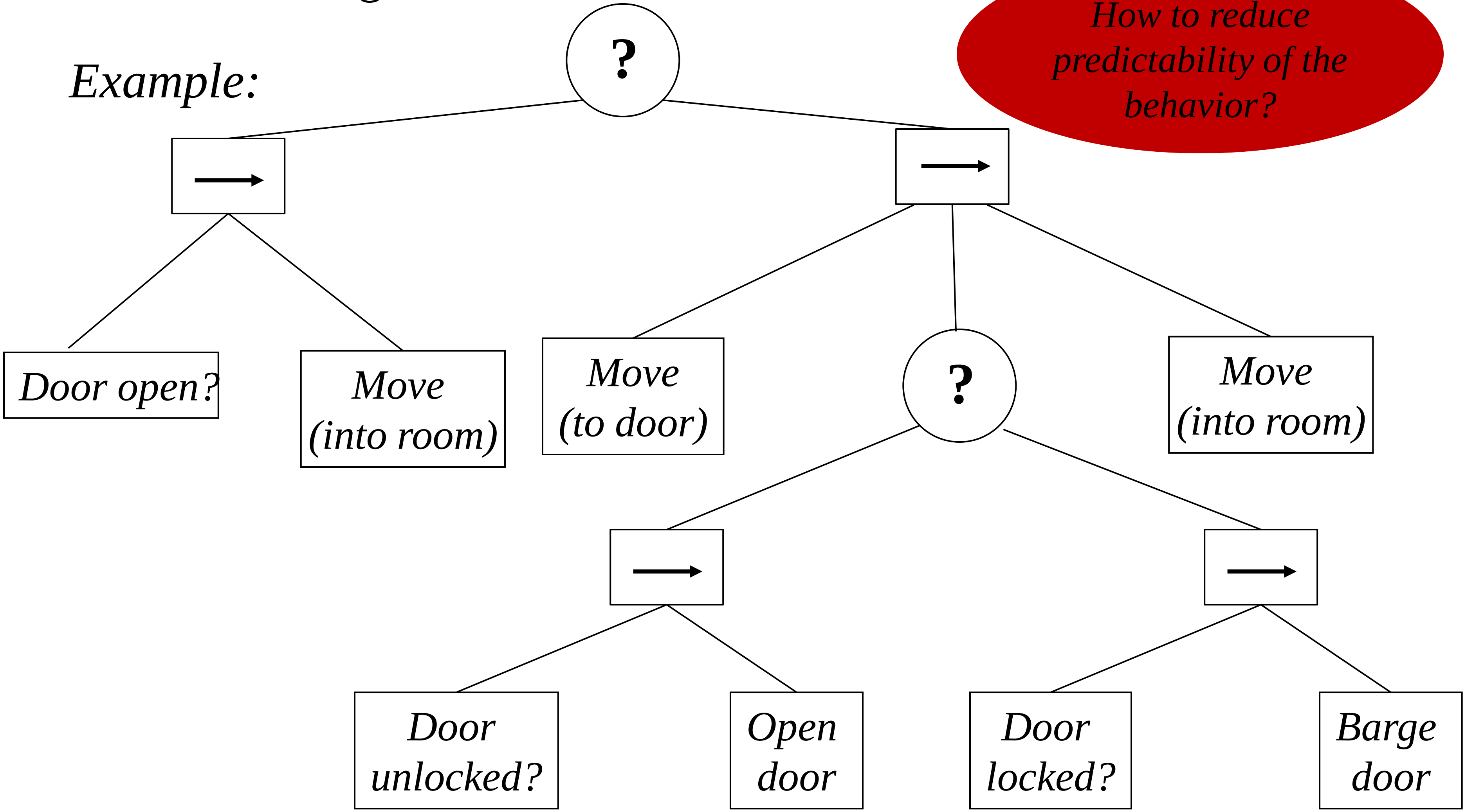


Can the tree ***always*** *be structured so that sequence and selector levels alternate?*

Basic Behavior Trees

- Type of tasks: *Conditions, Actions and Composites*
- Each returning either success or failure

Example:



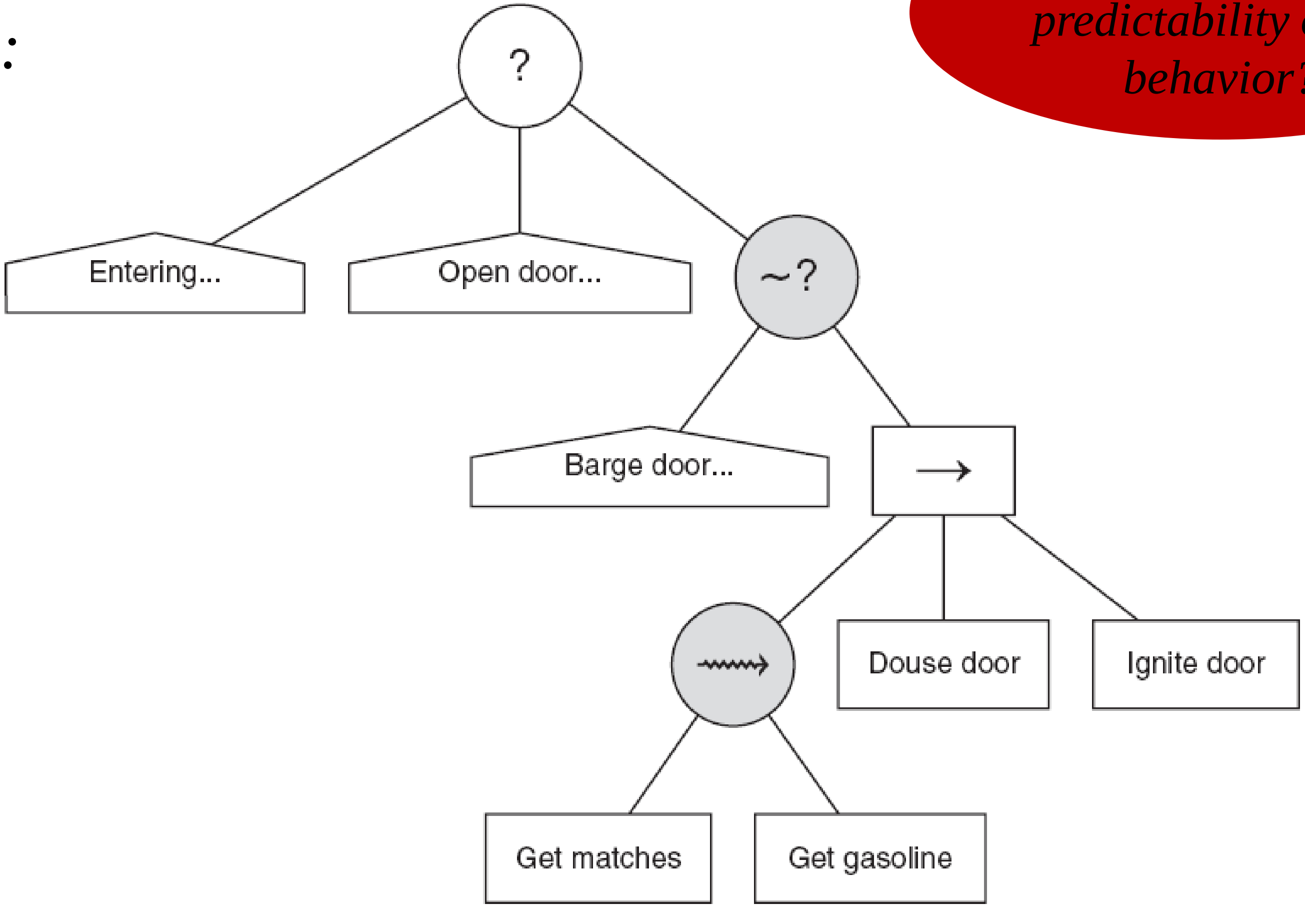
How to reduce predictability of the behavior?

Basic Behavior Trees

- Behavior trees with **Order Randomization for some Sequencers and Selectors**

How to reduce predictability of the behavior?

Example:



from “Artificial Intelligence for Games” by I. Millington & J. y ge