

Artificial Intelligence

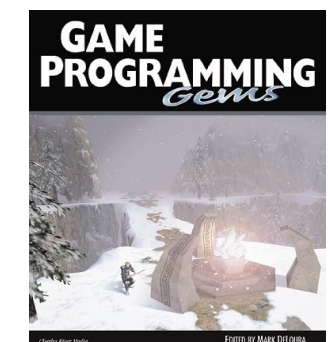
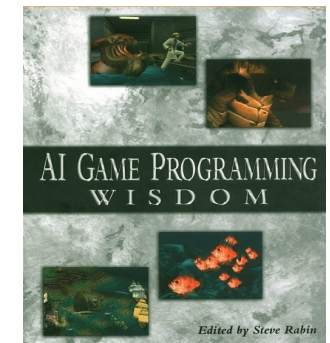
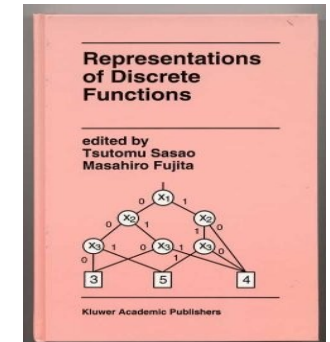
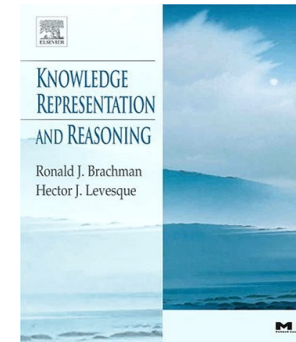
Decision Making

Frederic Koriche

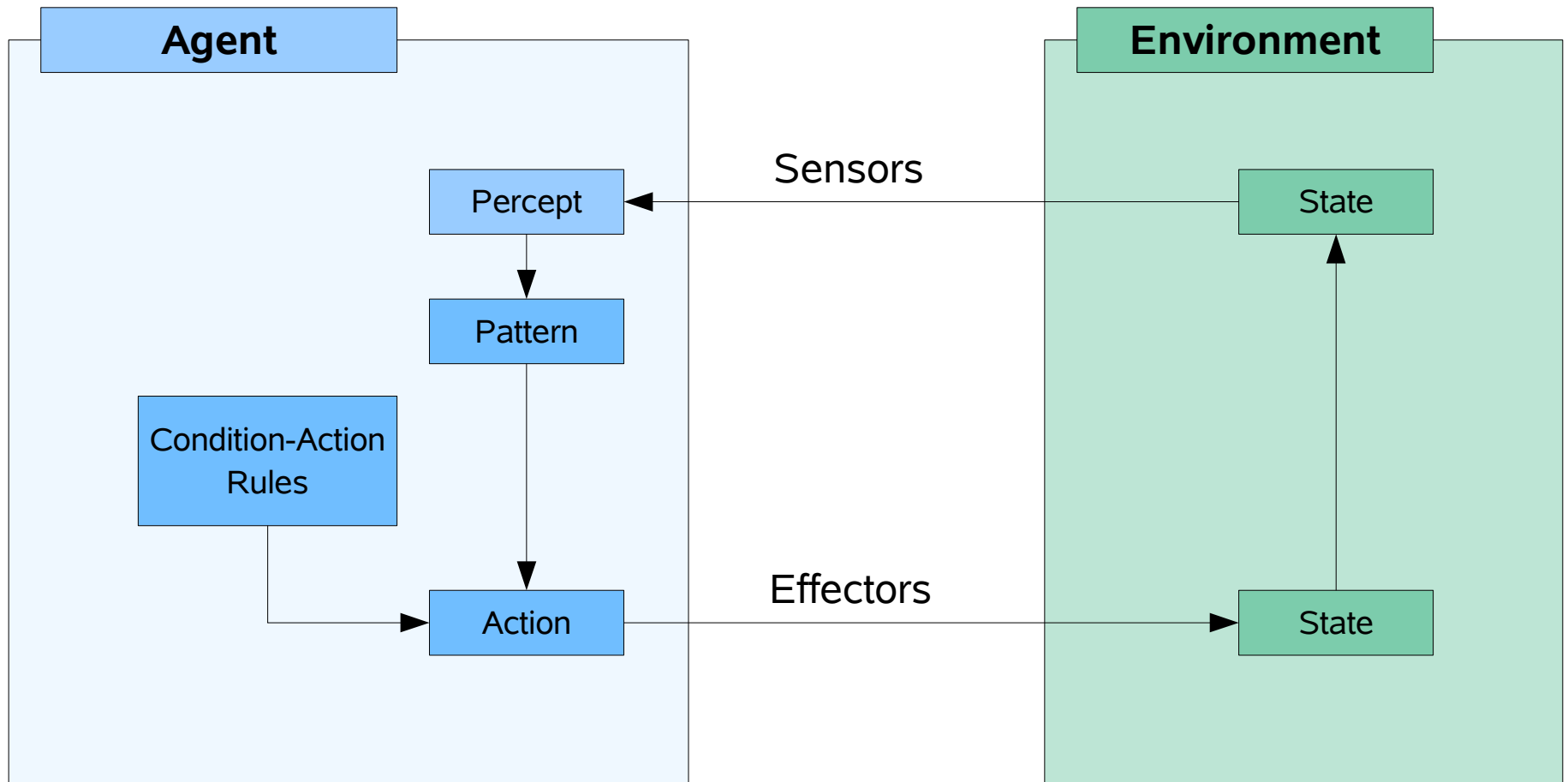
frederic.koriche@lirmm.fr

Overview

- 1.Reactive Agents
- 2.Finite State Machines
- 3.Hierarchical State Machines
- 4.Decision Trees
- 5.Decision Diagrams
- 6.Production Rule Systems

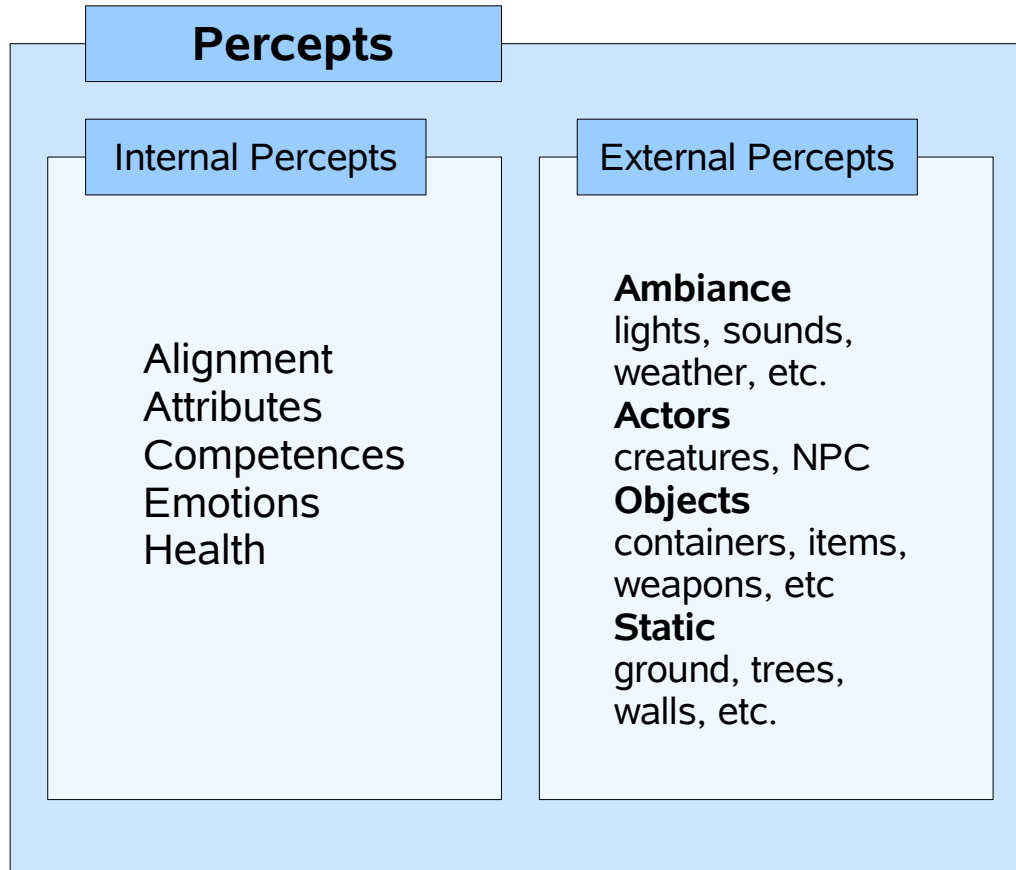


Reflexive Agents



Reflexive Agent: Makes a decision according to a prefixed set of condition-action rules

Reflexive Agents



Percepts: The agent perceives the environment through her internal and external sensors.

Reflexive Agents

Percept

- Set $\{X_1, \dots, X_n\}$ of attributes
- Domains $\{D_1, \dots, D_n\}$ of values
- Values can be nominal, ordinal or numerical
- Perception space: $\mathbf{X} = \prod_i (D_i \cup \{*\})$
- * is a hidden value
- A percept is $x \in \mathbf{X}$

Pattern

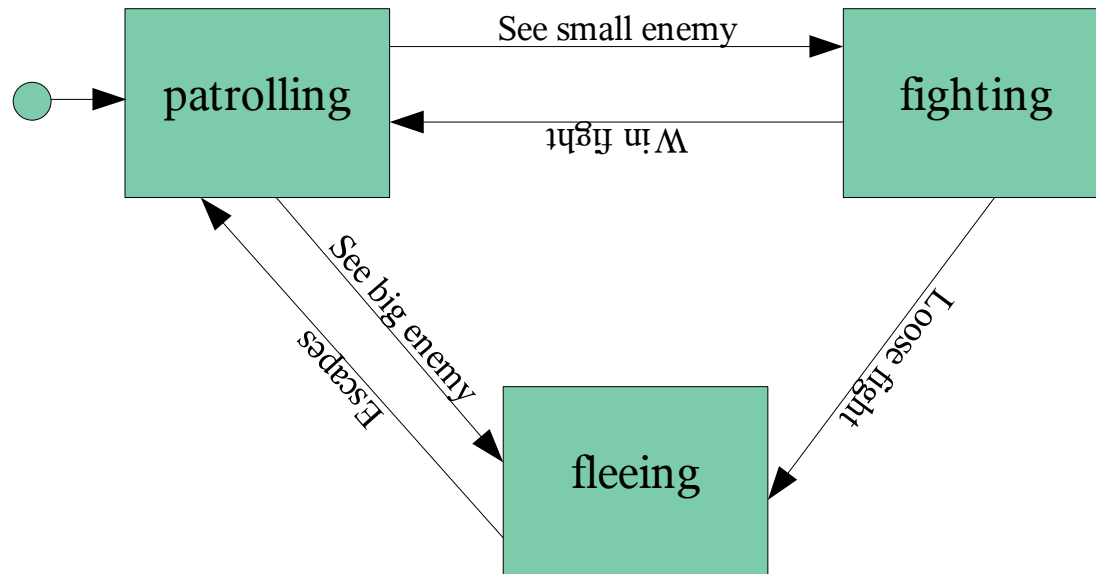
- Atoms: A
 $(X_i = v), (X_i \leq v), (X_i \geq v)$
- Literals: L is A or $\neg A$
- Conjunctive patterns:
 $L_1 \wedge L_2 \wedge \dots \wedge L_k$
- Linear patterns:
 $w_1 L_1 + w_2 L_2 + \dots + w_k L_k > \theta$

Example

A « hunger » pattern:

$$\frac{1}{3}(\text{health} < 1) + \frac{1}{3}(\text{food} = 1) + \frac{1}{3}(\text{happiness} < 1) > \frac{1}{2}$$

Finite State Machines



FSM

Directed graph (V,E) where:

- each vertex $v \in V$ is an agent's state
- each edge $(u,v) \in E$ is a transition from u to v

Finite State Machines

FSM Language

- **BeginStateMachine**: start the agent's state machine
- **EndStateMachine**: terminates the agent's state machine
- **State**(NameOfState): designates the beginning of a particular state
- **OnEnter**: responds to a state being entered by performing a series of actions
- **OnExit**: responds to a state being exited by cleaning up memory
- **OnUpdate**: responds to the update game tick
- **OnMsg**(NameOfMessage): responds to any defined message
- **SetState**(NameOfState): apply transition by sending OnExit to old state and OnEnter to new state
- **SendMessage**(Actor): send message to a particular agent

Finite State Machines

FSM Code

```
BeginStateMachine
```

```
OnEnter{SetState(Patrolling)}
```

```
State(Patrolling)
```

```
OnEnter
```

```
    { //Set Actions for Guarding }
```

```
OnUpdate
```

```
    if(FirePattern(See_Small_Enemy)
```

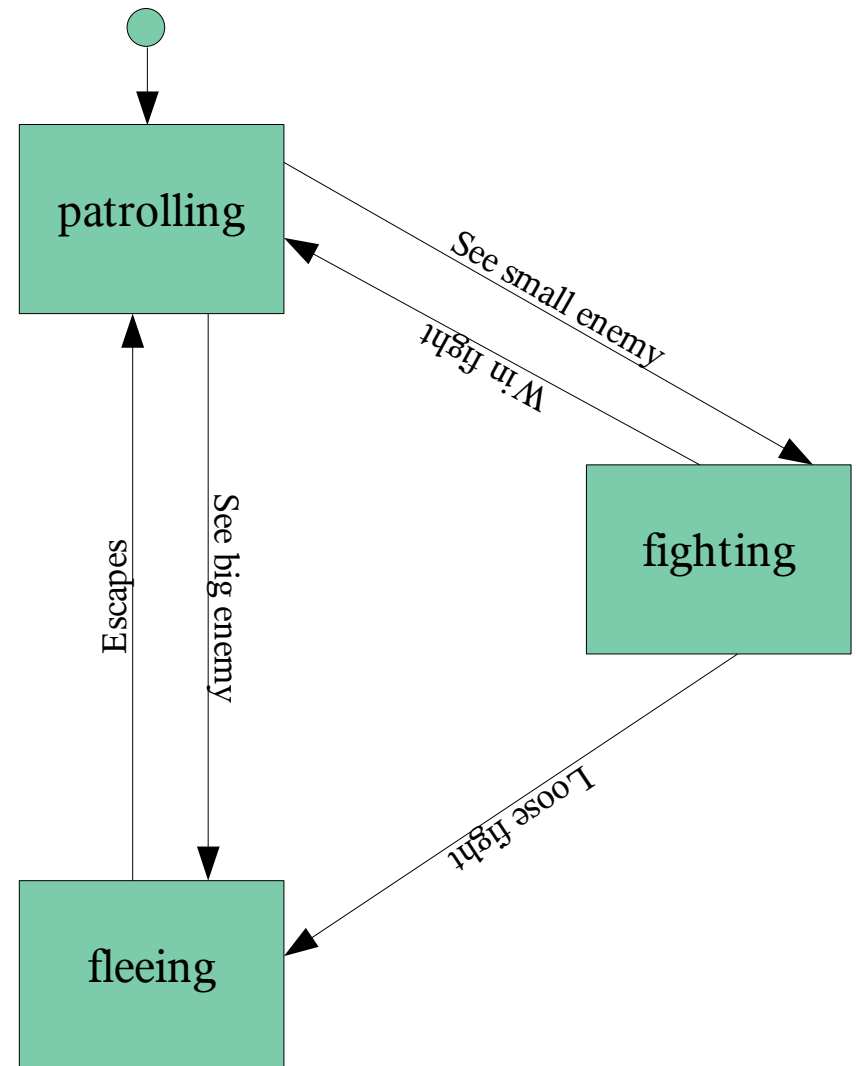
```
        SetState(Fighting)
```

```
    else if(FirePattern(See_Large_Enemy)
```

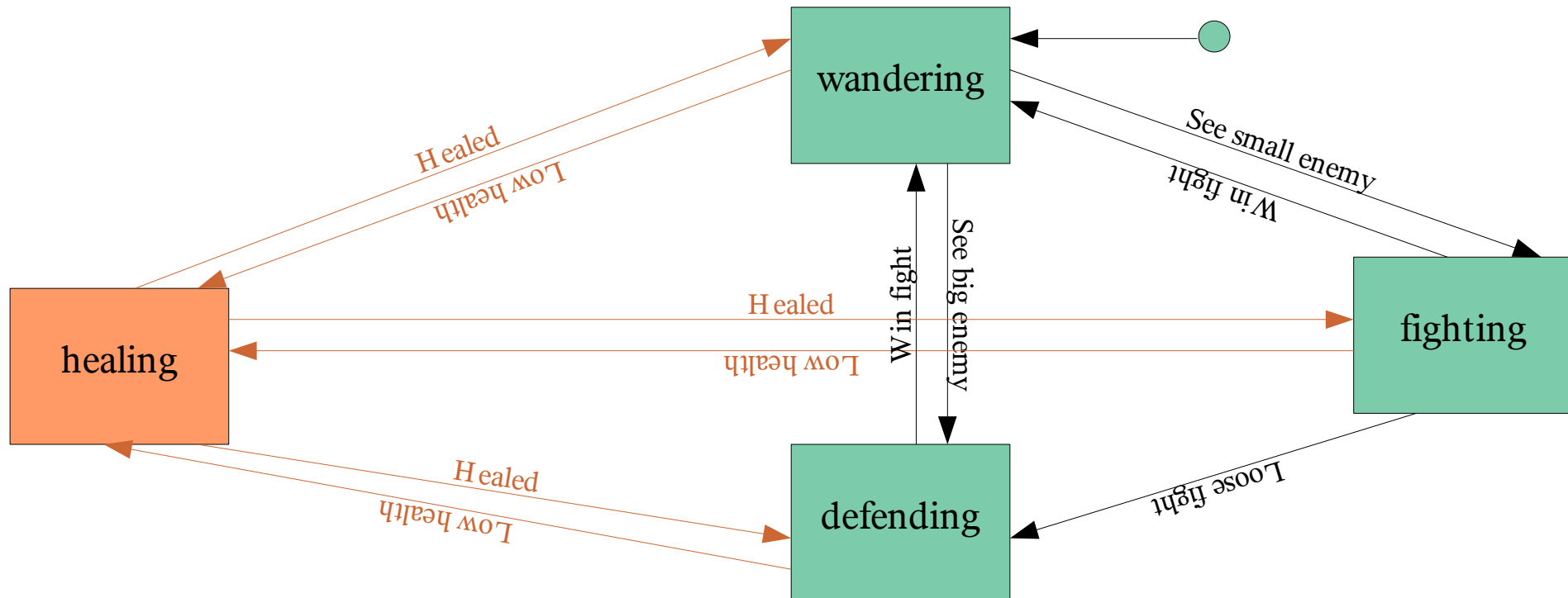
```
        SetState(Fleeing)
```

```
...
```

```
EndStateMachine
```



Hiearchical State Machines



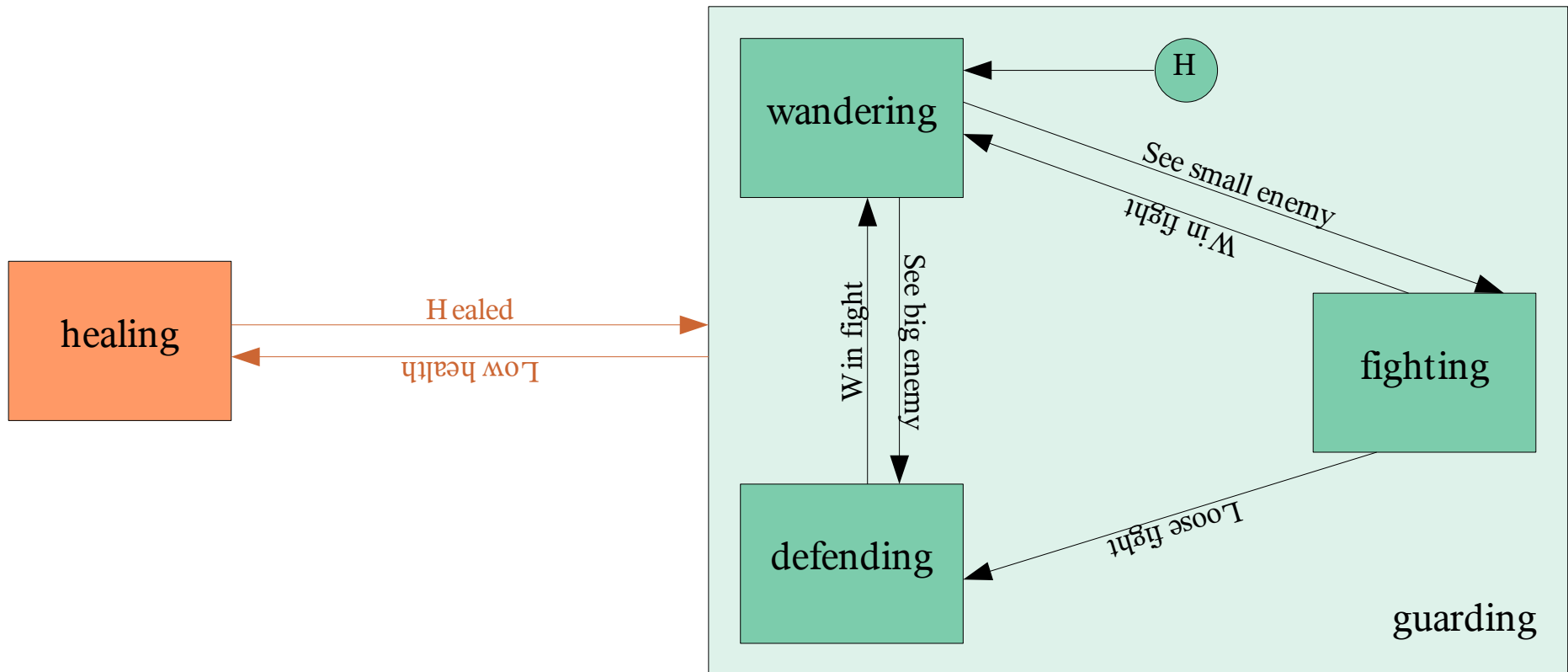
Compacting FSMs

Let $G = (V, E)$ be a FSM. Let G_1 and G_2 be subgraphs of G such that

- each v_1 in V_1 is connected to any v_2 in V_2
- each v_2 in V_2 is connected to any v_1 in V_1

Then G can be splitted into two FSMs G_1 and G_2 that keep in cache their exit state

Hierarchical State Machines



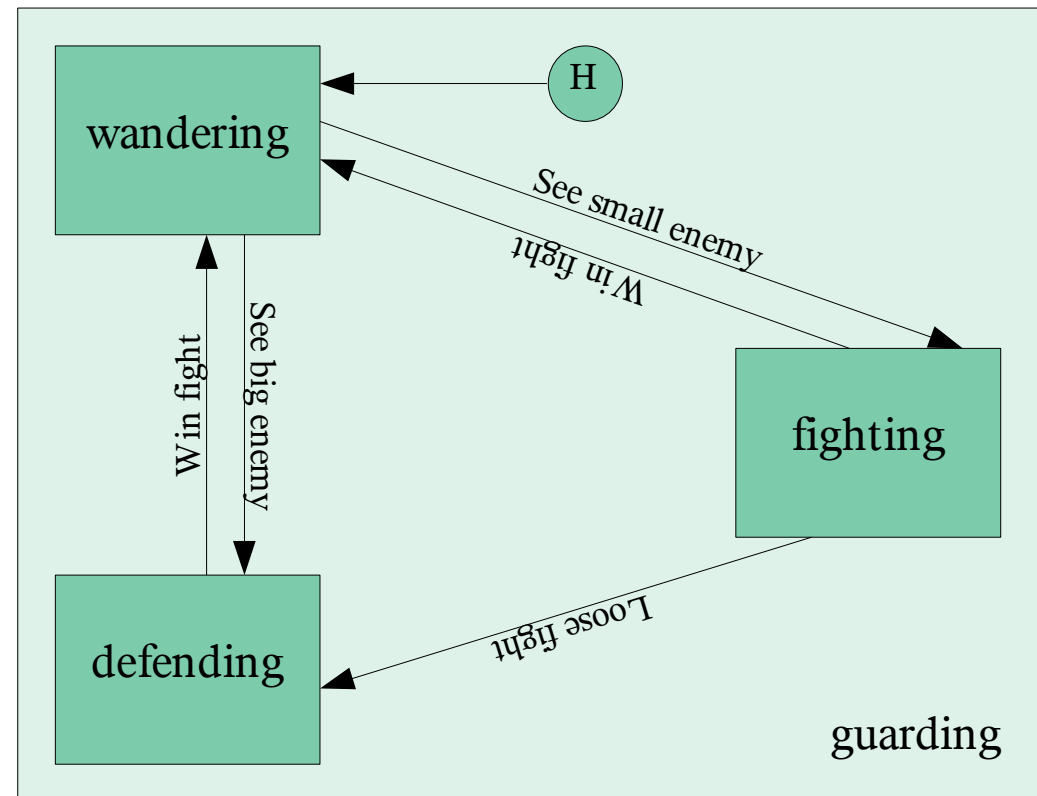
Hierarchical FSM

A hierarchical FSM of depth k is a directed graph $G = (V, E)$ where each node v in V is either a state or a hierarchical FSM of depth $k - 1$. A hierarchical FSM of depth 0 is a FSM.

Hierarchical State Machines

FSM Code

```
State(guarding)
OnEnter{SetState(cache)}
OnMsg(Low_Health)
    {SetState(healing)}
BeginStateMachine
State(wandering)
    cache = wandering
    OnEnter
        {//Set Actions}
    OnUpdate
        if(FirePattern(See_Small_Enemy))
            SetState(Fighting)
        elseif(FirePattern(See_Large_Enemy))
            SetState(Defending)
    ...
EndStateMachine
```



State Machines

Strengths

- **Memory:** low requirements
- **Time:** low requirements
- Can be **Hard-Coded**

Weaknesses

- **Modeling:** requires to know all states in advance
- **Modularity:** difficult to maintain
- **Complexity:** the number of states can grow exponentially with the number of attributes
- **Nonstochastic:** too predictable for small FSM

FSM: Memory $O(1)$, Time $O(t)$

Hierarchical FSM: Memory $O(k)$, Time $O(t)$

Parallel FSM: Memory $O(mk)$, Time $O(mt)$

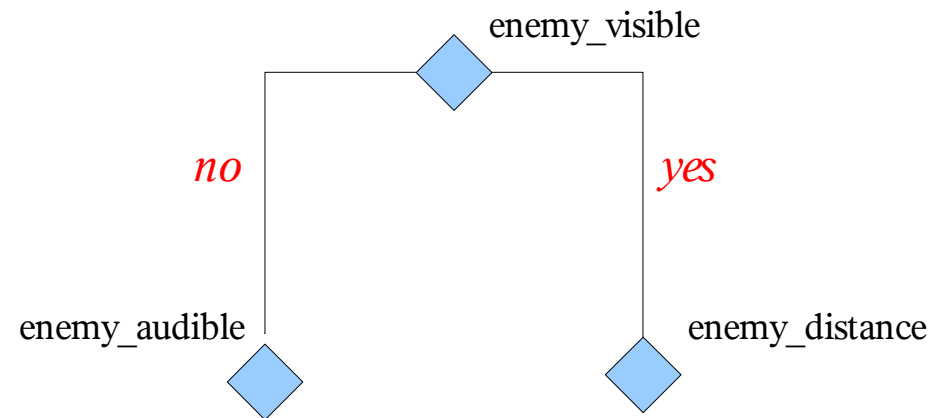
- t : number of transitions per state
- k : depth of the hierarchy
- m : number of parallel machines

Decision Trees

Decision Trees

A decision tree is a rooted tree $T = (V, E)$, where

- each internal node v in V is labeled with an attribute
- each edge e in E is labeled with a value (or interval) of the parent attribute
- each leaf v of V is labeled with an action

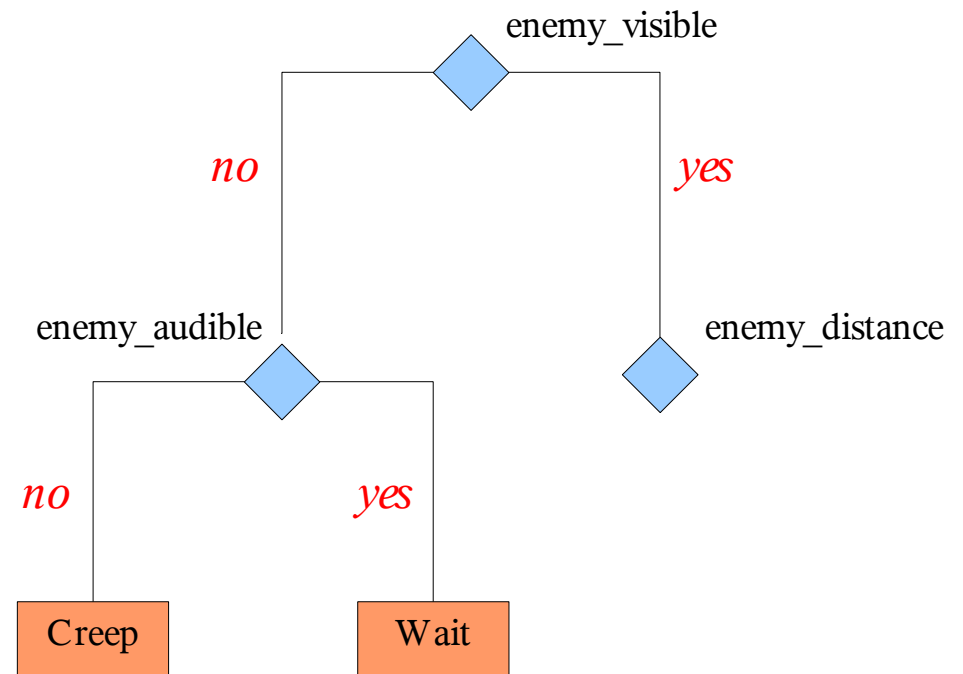


Decision Trees

Decision Trees

A decision tree is a rooted tree $T = (V, E)$, where

- each internal node v in V is labeled with an attribute
- each edge e in E is labeled with a value (or interval) of the parent attribute
- each leaf v of V is labeled with an action

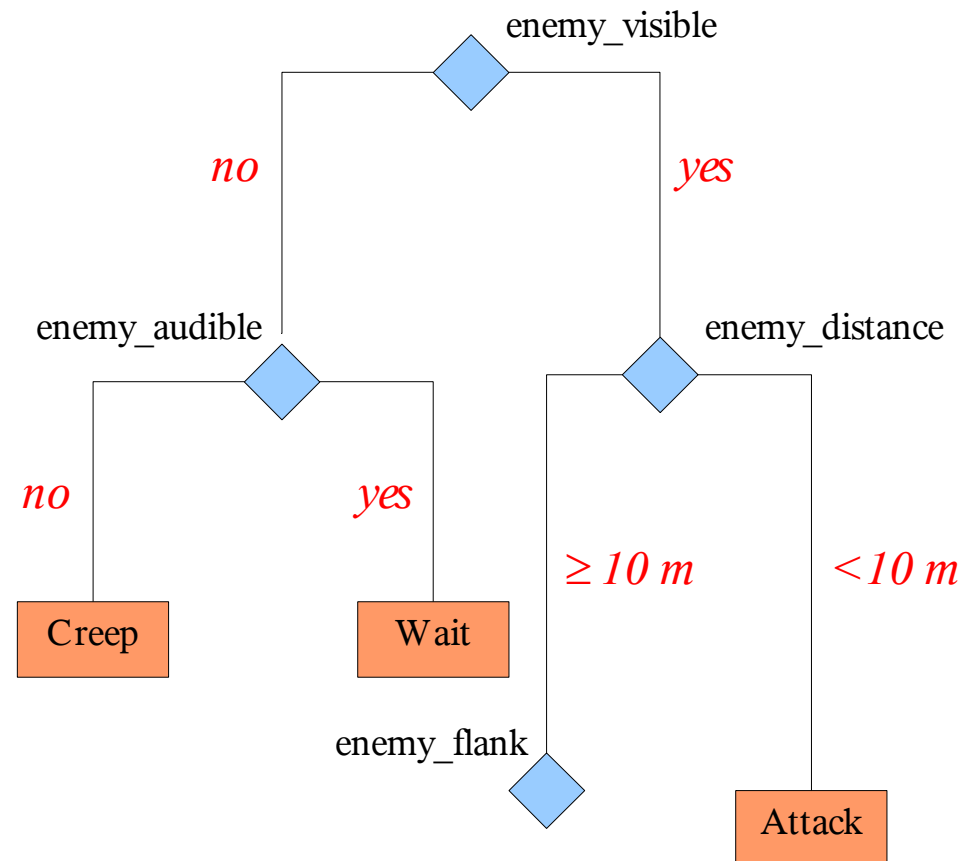


Decision Trees

Decision Trees

A decision tree is a rooted tree $T = (V, E)$, where

- each internal node v in V is labeled with an attribute
- each edge e in E is labeled with a value (or interval) of the parent attribute
- each leaf v of V is labeled with an action

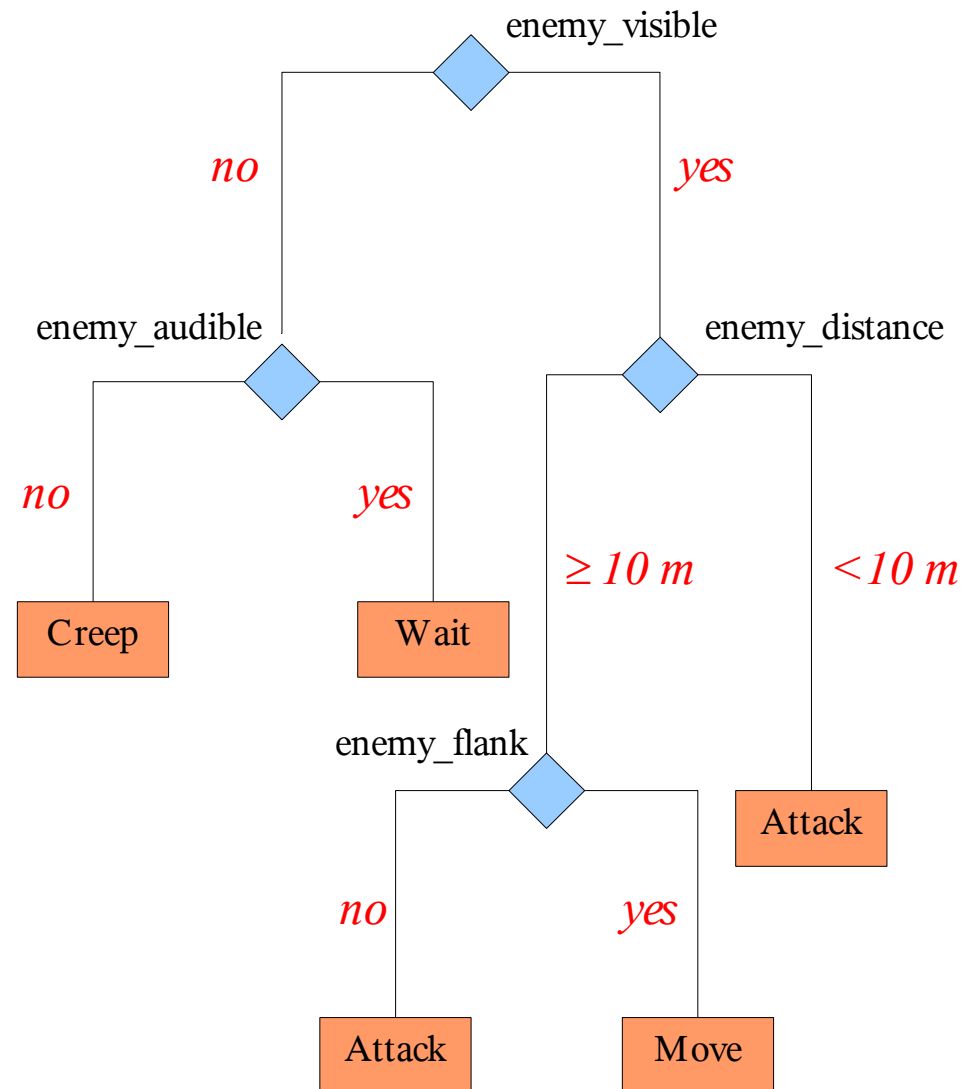


Decision Trees

Decision Trees

A decision tree is a rooted tree $T = (V, E)$, where

- each internal node v in V is labeled with an attribute
- each edge e in E is labeled with a value (or interval) of the parent attribute
- each leaf v of V is labeled with an action



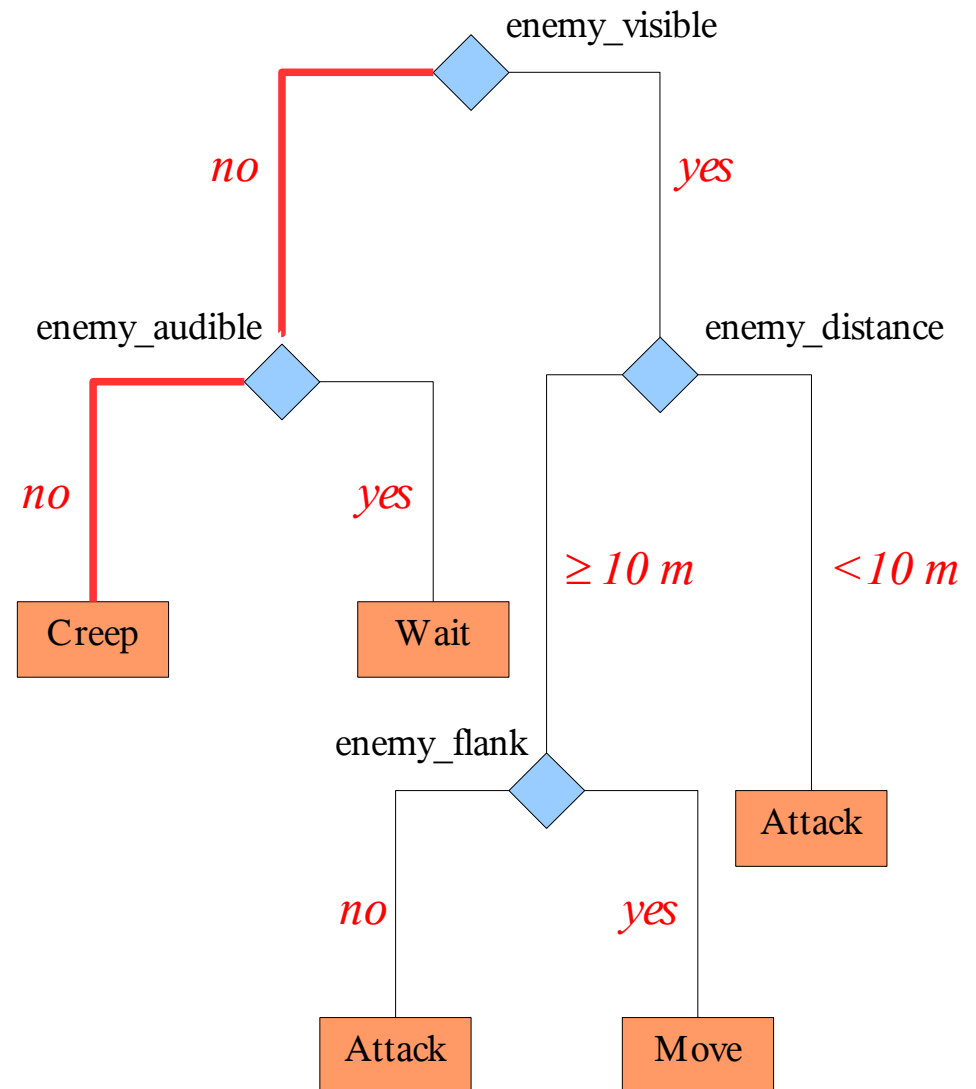
Decision Trees

Decision Making

Given a percept x , a decision is made by starting from the root and iteratively taking a branch according to x

In case of uncertainty (*), a branch is chosen at random

<i>ev</i>	<i>ea</i>	<i>ed</i>	<i>ef</i>	...
0	0	5	0	...



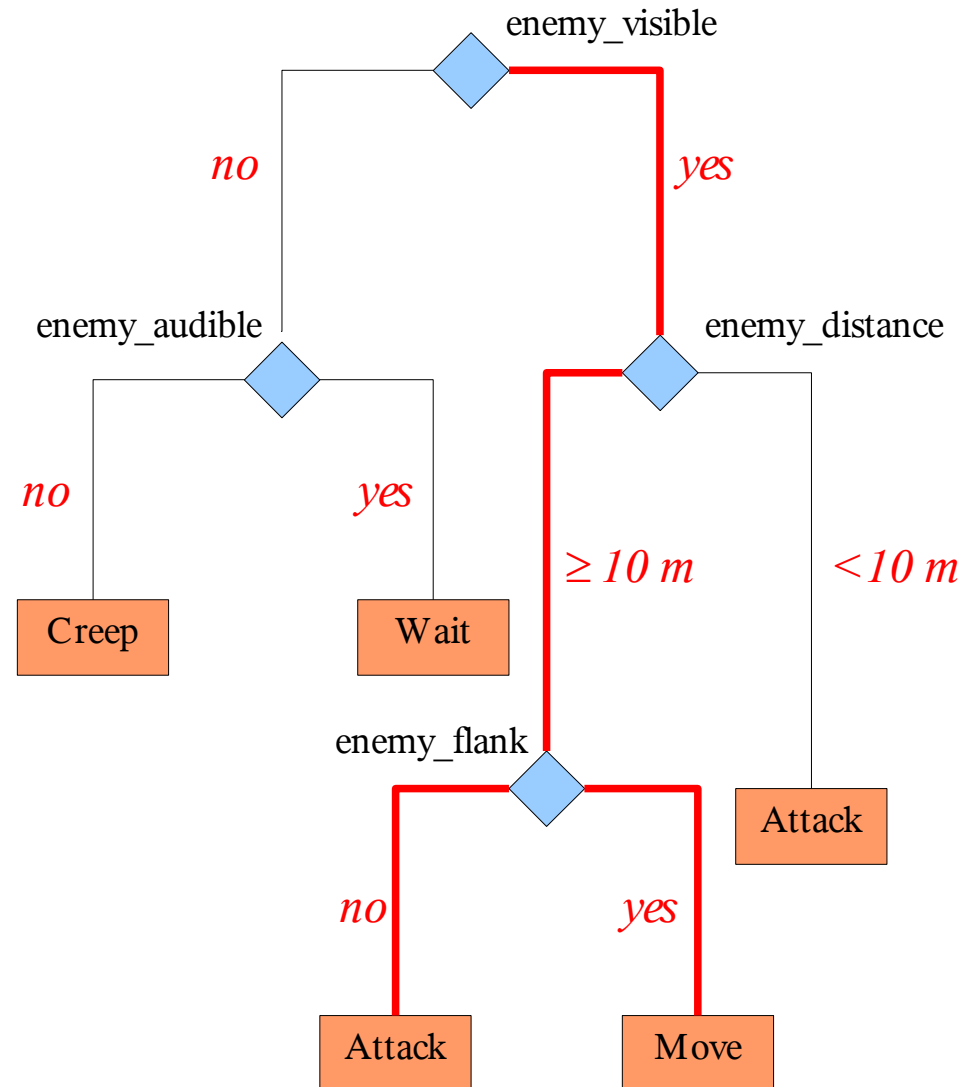
Decision Trees

Decision Making

Given a percept x , a decision is made by starting from the root and iteratively taking a branch according to x

In case of uncertainty (*), a branch is chosen at random

<i>ev</i>	<i>ea</i>	<i>ed</i>	<i>ef</i>	...
1	0	15	*	...



Decision Trees

Expressiveness

Decision trees have the expressiveness of deterministic decision rules

$\neg \text{enemy_visible} \wedge \neg \text{enemy_audible} \rightarrow \text{Creep}$

$\neg \text{enemy_visible} \wedge \text{enemy_audible} \rightarrow \text{Wait}$

$\text{enemy_visible} \wedge (\text{enemy_distance} < 10\text{m}) \rightarrow \text{Attack}$

$\text{enemy_visible} \wedge (\text{enemy_distance} \geq 10\text{m}) \wedge \neg \text{enemy_flanck} \rightarrow \text{Attack}$

$\text{enemy_visible} \wedge (\text{enemy_distance} \geq 10\text{m}) \wedge \text{enemy_flanck} \rightarrow \text{Move}$

Decision Trees

Expressiveness

Decision trees have the expressiveness of deterministic Decision rules

$\neg \text{enemy_visible} \wedge \neg \text{enemy_audible} \rightarrow \text{Creep}$

$\neg \text{enemy_visible} \wedge \text{enemy_audible} \rightarrow \text{Wait}$

$\text{enemy_visible} \wedge (\text{enemy_distance} < 10\text{m}) \rightarrow \text{Attack}$

$\text{enemy_visible} \wedge (\text{enemy_distance} \geq 10\text{m}) \wedge \neg \text{enemy_flanck} \rightarrow \text{Attack}$

$\text{enemy_visible} \wedge (\text{enemy_distance} \geq 10\text{m}) \wedge \text{enemy_flanck} \rightarrow \text{Move}$

ev ea ed ef ...

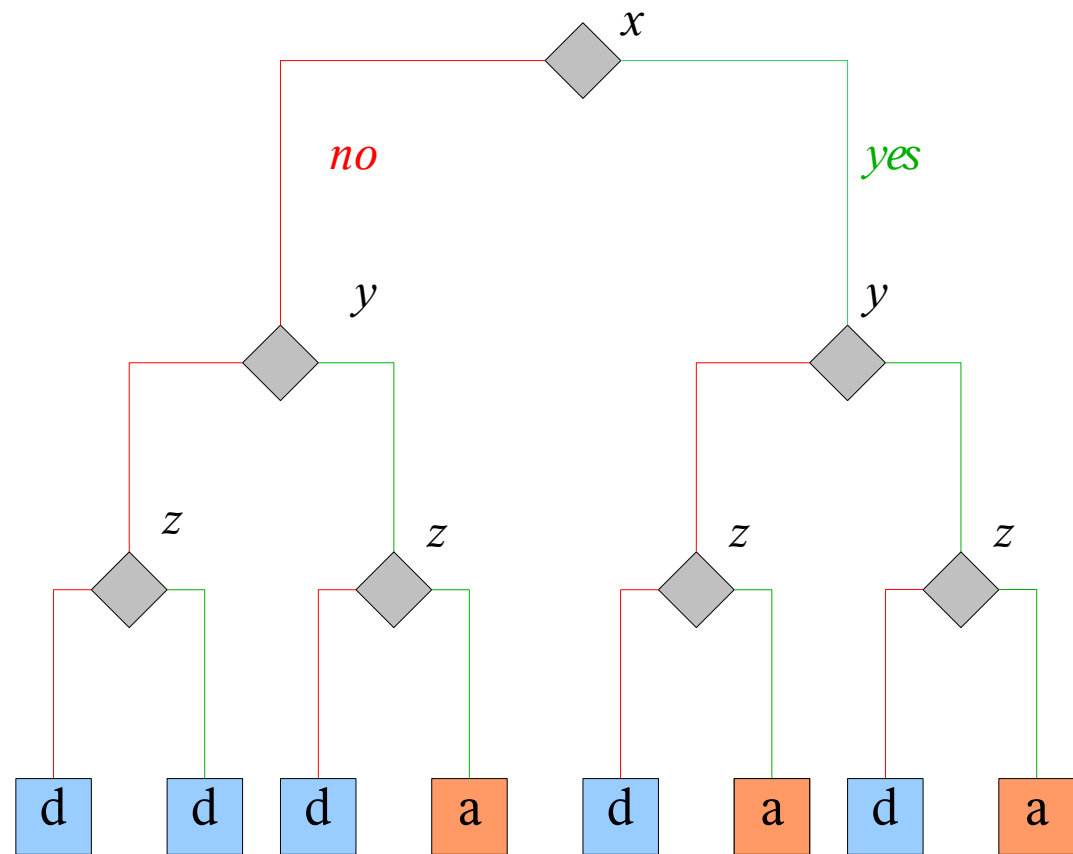
0	0	5	0	...
---	---	---	---	-----

Decision Diagrams

Decision Diagrams

A decision diagram is directed graph that represents a decision tree into a compiled form.

- BDD: the choices are binary
- OBDD: the attributes are ordered according to a predefined ordering

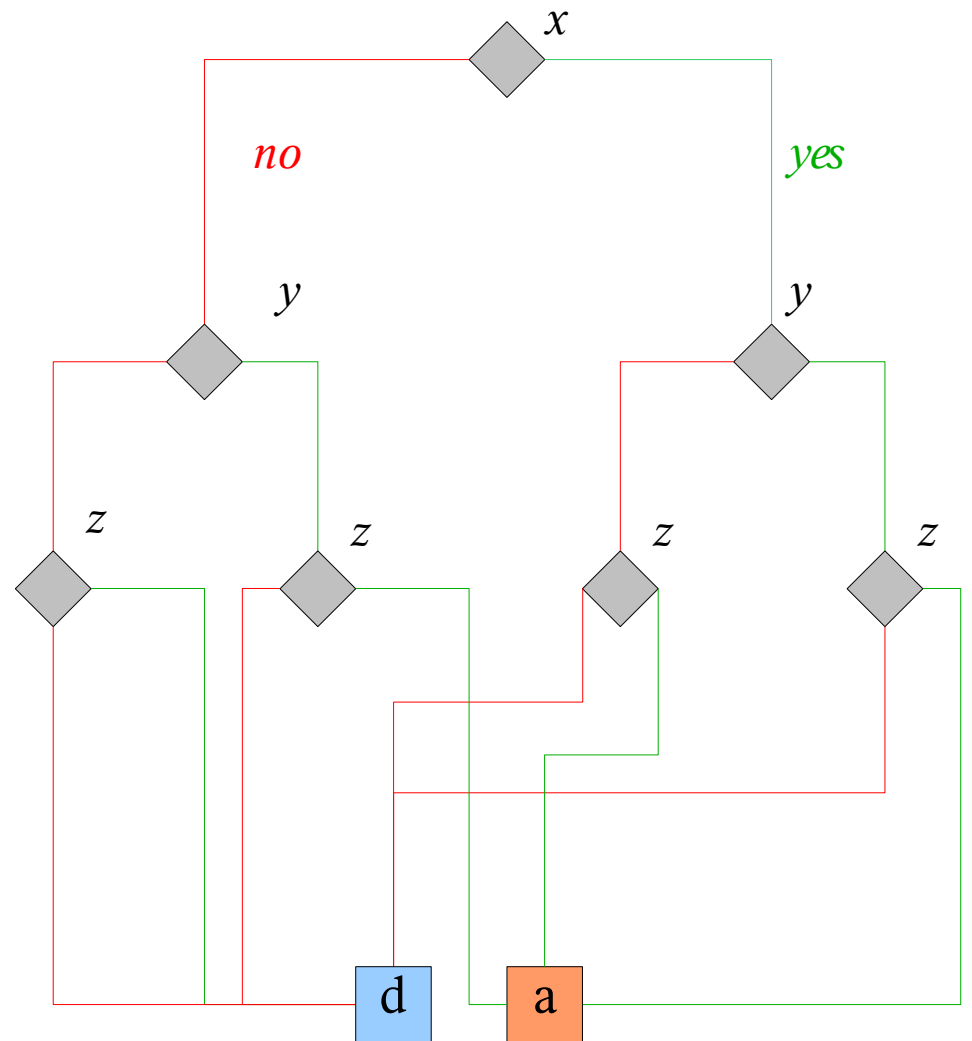


15 nodes

Decision Diagrams

Compiling a DT

Remove Duplicate Terminals:
eliminate all but one leaf with a
given label and redirect all arcs.

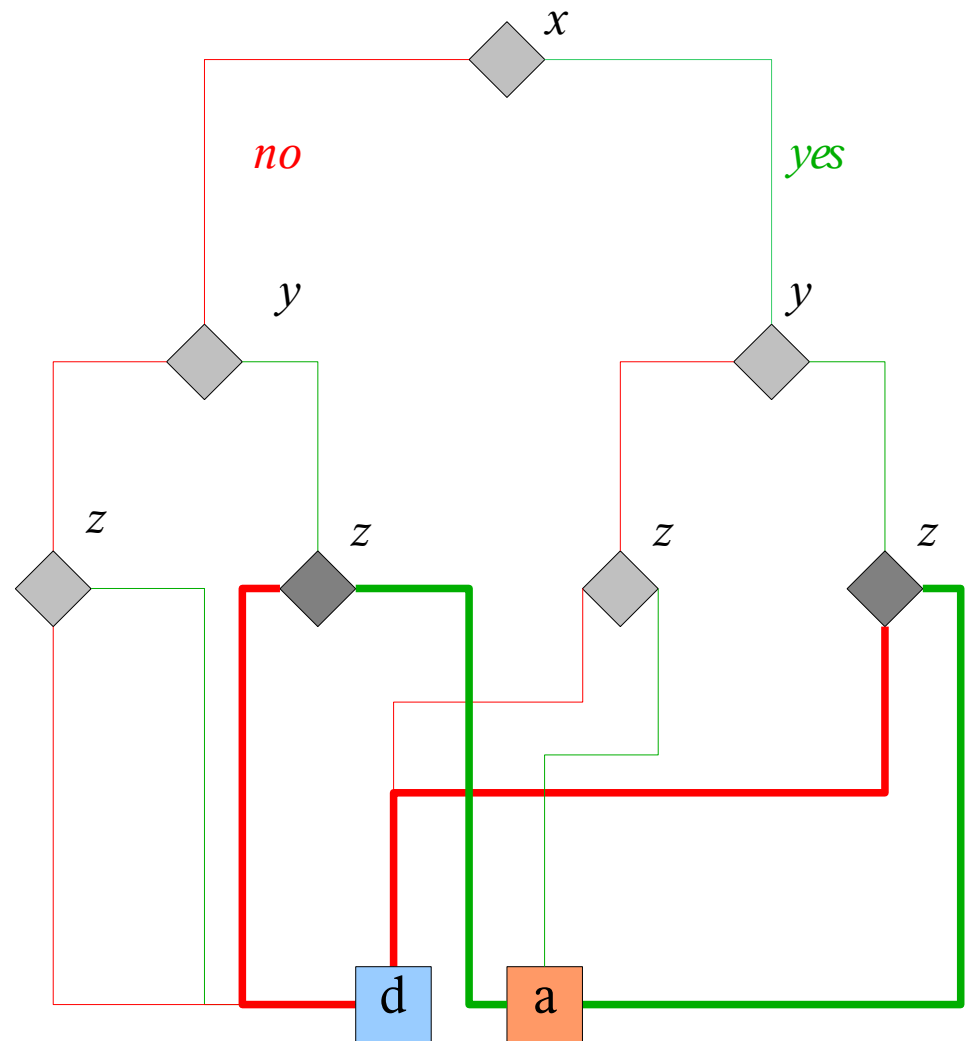


Decision Diagrams

Compiling a DT

Remove Duplicate Terminals:
eliminate all but one leaf with a given label and redirect all arcs.

Remove Duplicate Nonterminals:
if nonterminal vertices have u and v are labelled by the same attributes and have the same subtrees, then eliminate one of u or v , and redirect incoming arcs.

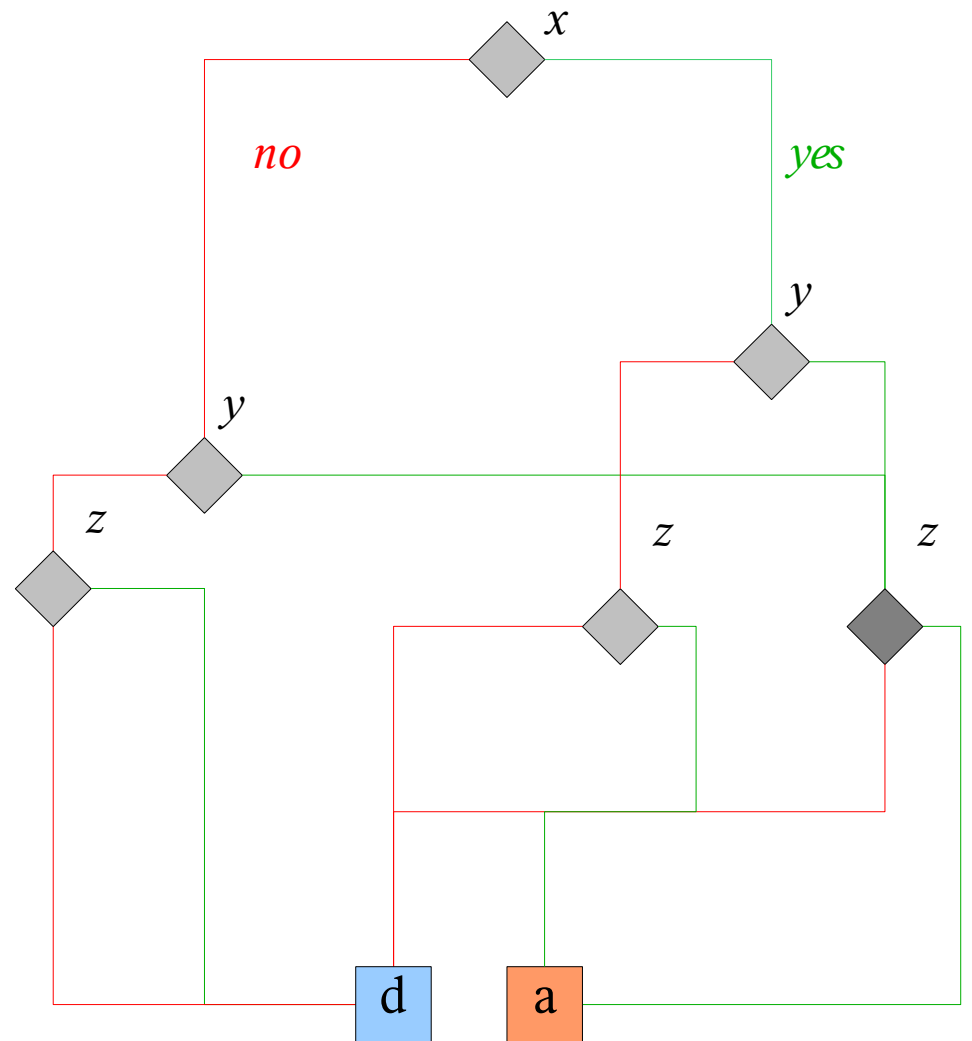


Decision Diagrams

Compiling a DT

Remove Duplicate Terminals:
eliminate all but one leaf with a given label and redirect all arcs.

Remove Duplicate Nonterminals:
if nonterminal vertices have u and v are labelled by the same attributes and have the same subtrees, then eliminate one of u or v , and redirect incoming arcs.

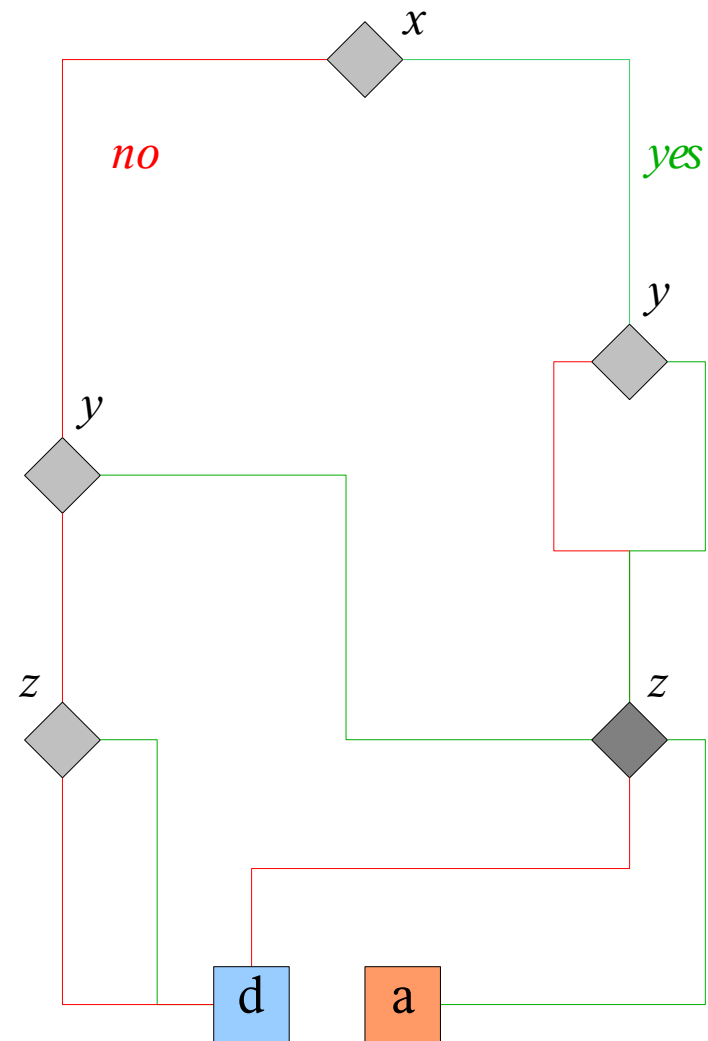


Decision Diagrams

Compiling a DT

Remove Duplicate Terminals:
eliminate all but one leaf with a given label and redirect all arcs.

Remove Duplicate Nonterminals:
if nonterminal vertices have u and v are labelled by the same attributes and have the same subtrees, then eliminate one of u or v , and redirect incoming arcs.



7 nodes

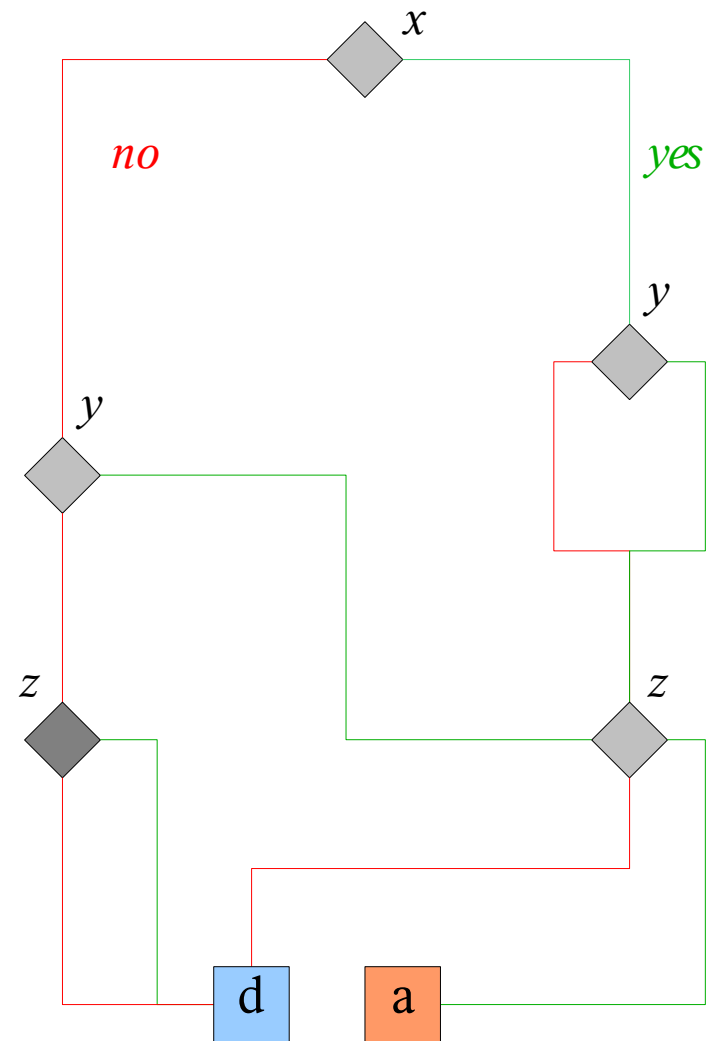
Decision Diagrams

Compiling a DT

Remove Duplicate Terminals:
eliminate all but one leaf with a given label and redirect all arcs.

Remove Duplicate Nonterminals:
if nonterminal vertices have u and v are labelled by the same attributes and have the same subtrees, then eliminate one of u or v , and redirect incoming arcs.

Remove Redundant Tests:
if nonterminal vertex u has the same subtrees then eliminate u and redirect incoming arcs.



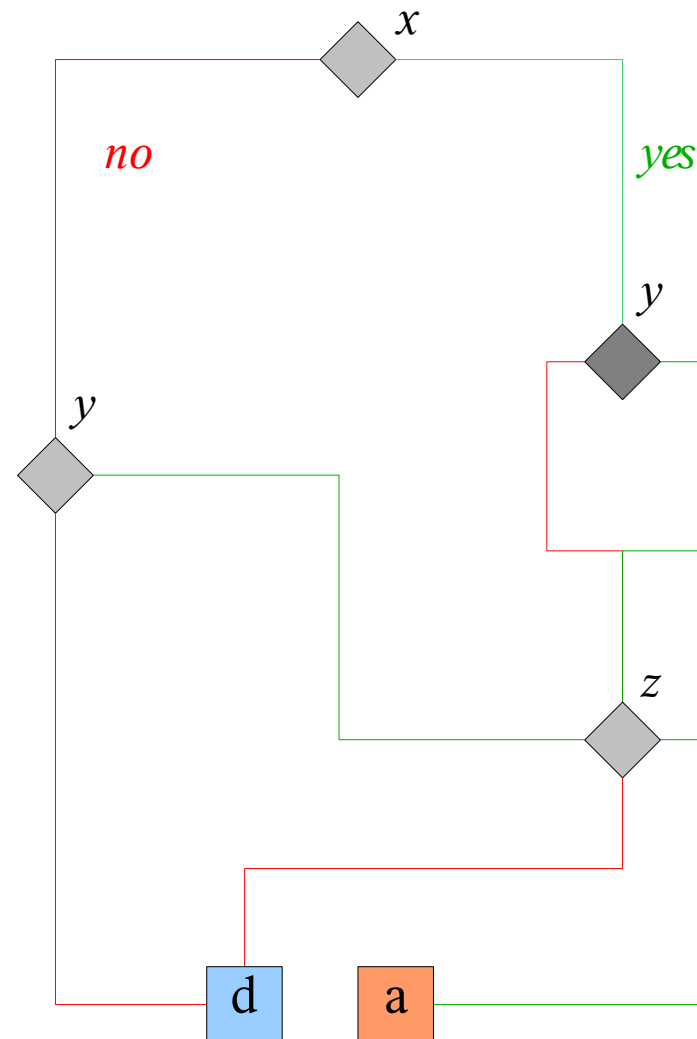
Decision Diagrams

Compiling a DT

Remove Duplicate Terminals:
eliminate all but one leaf with a given label and redirect all arcs.

Remove Duplicate Nonterminals:
if nonterminal vertices have u and v are labelled by the same attributes and have the same subtrees, then eliminate one of u or v , and redirect incoming arcs.

Remove Redundant Tests:
if nonterminal vertex u has the same subtrees then eliminate u and redirect incoming arcs.



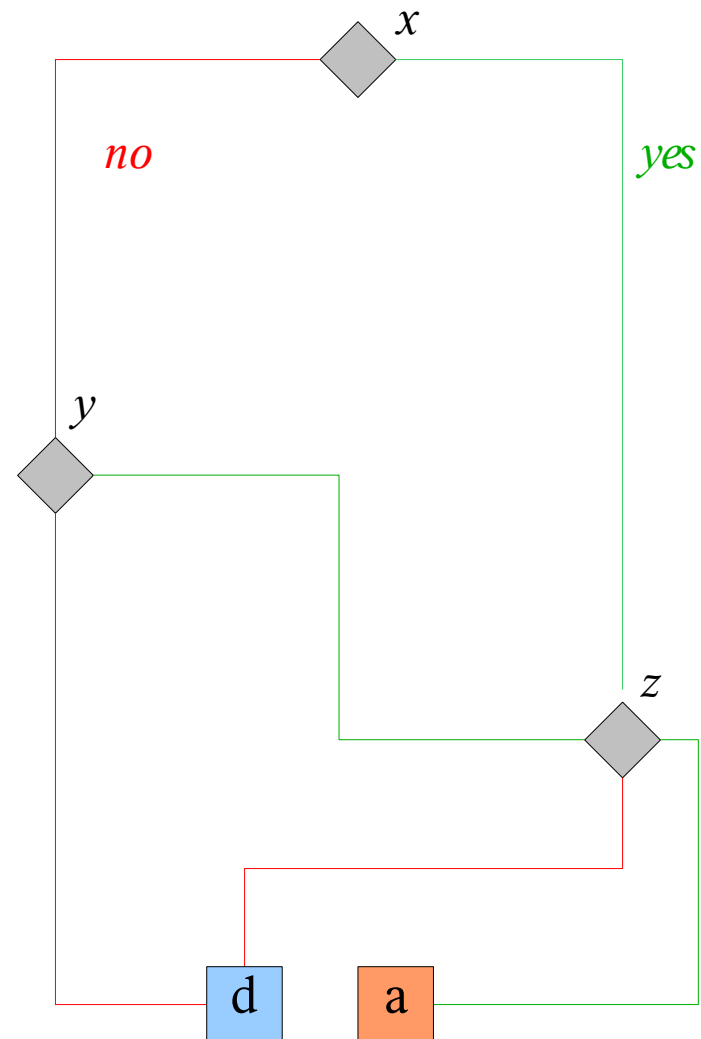
Decision Diagrams

Compiling a DT

Remove Duplicate Terminals:
eliminate all but one leaf with a given label and redirect all arcs.

Remove Duplicate Nonterminals:
if nonterminal vertices have u and v are labelled by the same attributes and have the same subtrees, then eliminate one of u or v , and redirect incoming arcs.

Remove Redundant Tests:
if nonterminal vertex u has the same subtrees then eliminate u and redirect incoming arcs.



5 nodes

Decision Trees

Strengths

- **Modeling:** reduced to local decisions
- **Time:** low requirements
- Large DT can be **compiled** into small Decision Diagrams

Weaknesses

- **Lack of Modularity:** large or compiled DT are difficult to maintain
- **Reduced expressiveness:** cannot include linear patterns easily
- **Nonstochastic:** difficult to include random behavior

DT: Memory $O(n)$, Time $O(\log n)$

DD: Memory $< O(n)$, Time $< O(\log n)$

Production Rule Systems

PRS

A PRS is a set of **decision rules**, together with a **conflict resolution** strategy.

Rule Base

{
enemy_visible → Attack
enemy_audible → Wait
enemy_flanck → Move
¬ enemy_audible → Creep

Conflict Resolution

Order strategy

Production Rule Systems

Decision Rules

A **decision rule** is an expression $P \rightarrow A$, where P is a pattern and A an action

$(\text{hapiness} = 1) \rightarrow \text{Wander}$

$(\text{health} < \frac{1}{2}) \wedge (\text{food} = 0) \rightarrow \text{Sleep}$

$[\frac{1}{3}(\text{health} < 1) + \frac{1}{3}(\text{food} = 1) + \frac{1}{3}(\text{hapiness} < 1) > \frac{1}{2}] \rightarrow \text{Hunt}$

Production Rule Systems

Conflict Resolution

Order: pick the first applicable rule in order of presentation

Specificity: select the applicable rule whose conditions are most specific

Weighting: select the applicable rule whose weight, or vote, is majoritary

enemy_visible → Attack

enemy_visible \wedge enemy_audible → Wait

enemy_flanck → Move

\neg enemy_flanck → Attack

\neg enemy_audible → Creep

ev ea ed ef ...

1	1	5	0	...
---	---	---	---	-----

Production Rule Systems

Conflict Resolution

Order: pick the first applicable rule in order of presentation

Specificity: select the applicable rule whose conditions are most specific

Weighting: select the applicable rule whose weight, or vote, is majoritary

enemy_visible \rightarrow Attack

enemy_visible \wedge enemy_audible \rightarrow Wait

enemy_flanck \rightarrow Move

\neg enemy_flanck \rightarrow Attack

\neg enemy_audible \rightarrow Creep

ev ea ed ef ...

1	1	5	0	...
---	---	---	---	-----

Production Rule Systems

Conflict Resolution

Order: pick the first applicable rule in order of presentation

Specificity: select the applicable rule whose conditions are most specific

Weighting: select the applicable rule whose weight, or vote, is majoritary

$\text{enemy_visible} \rightarrow \text{Attack}$

$\text{enemy_visible} \wedge \text{enemy_audible} \rightarrow \text{Wait}$

$\text{enemy_flanck} \rightarrow \text{Move}$

$\neg \text{enemy_flanck} \rightarrow \text{Attack}$

$\neg \text{enemy_audible} \rightarrow \text{Creep}$

ev ea ed ef ...

1	1	5	0	...
---	---	---	---	-----

Production Rule Systems

Strengths

- **Expressiveness:** any pattern
- **Modeling:** reduced to local decisions
- **Modularity:** easy to maintain
- **Memory:** low requirements
- **Time:** low requirements

Weaknesses

- **Nonstochastic:** difficult to include random behavior (random resolution strategies)
- **Nonpredictability:** determining the behavior of multiple agents can be undecidable

PRS: Memory $O(m)$, Time $O(m n)$