Stavros Vassos, University of Athens, Greece stavrosv@di.uoa.gr May 2012

INTRODUCTION TO AI STRIPS PLANNING

.. and Applications to Video-games!

Course overview

- Lecture 1: Game-inspired competitions for AI research, **AI decision making for non-player characters in games**
- Lecture 2: STRIPS planning, state-space search
- Lecture 3: Planning Domain Definition Language (PDDL), using an award winning planner to solve Sokoban
- Lecture 4: Planning graphs, domain independent heuristics for STRIPS planning
- Lecture 5: Employing STRIPS planning in games: SimpleFPS, iThinkUnity3D, SmartWorkersRTS
- Lecture 6: Planning beyond STRIPS

Artificial Intelligence and Video Games

- Video Games:
  - Finite State Machines
  - Decision Diagrams
  - Behavior Trees
  - Goal Oriented Action Planning
- Academic AI on agents:
  - Knowledge representation, First-order logic, Classical planning, Planning with preferences, …
  - Belief-Desire-Intention architecture, Agent-based programming, …
  - Probabilistic reasoning, Bayesian networks, Utility theory, Markov Decision Processes, …

Artificial Intelligence and Video Games

- Game engine:
  - C++
  - Creates game-world objects with (x,y,z) coordinates and calculates what happens to them on every frame
  - E.g., a crate is up in the air on frame1. On frame 2 the game engine will calculate the new position, etc

Artificial Intelligence and Video Games

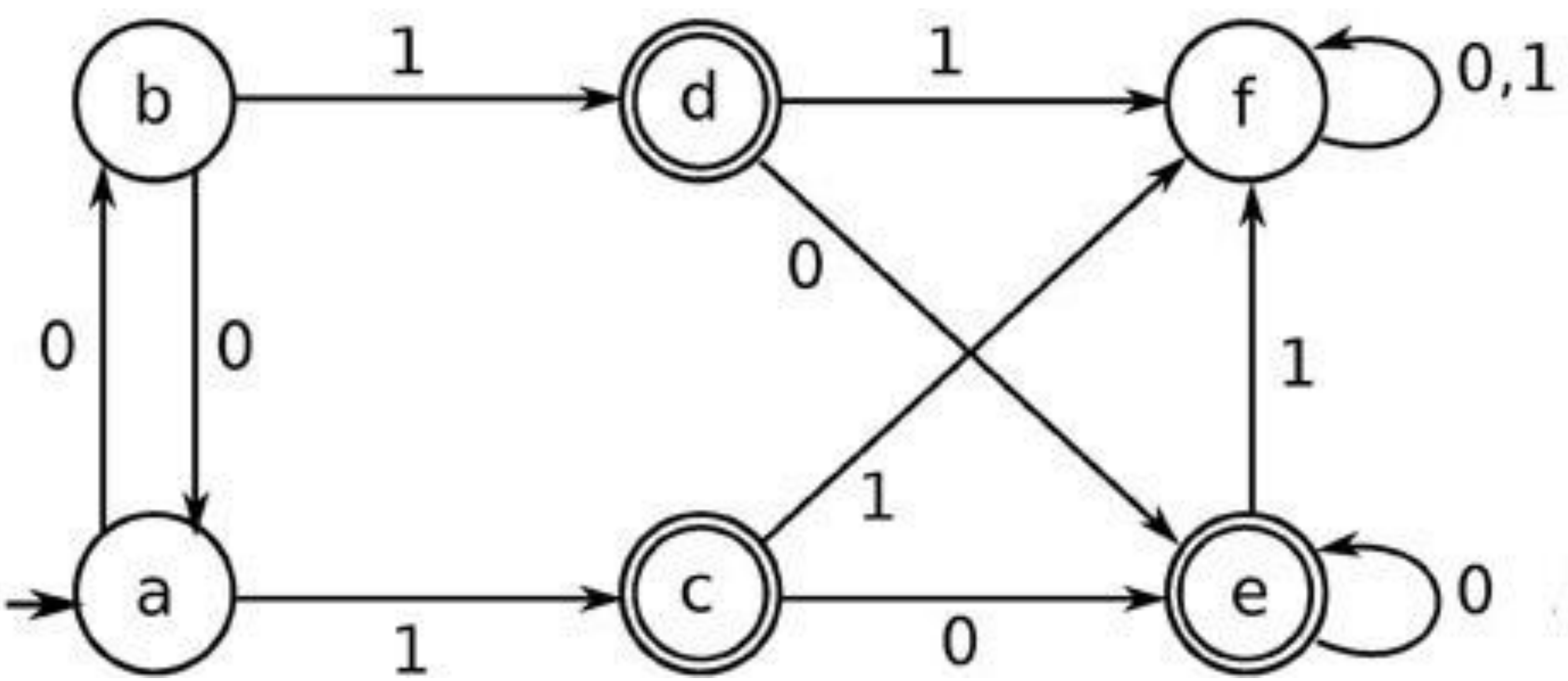
- Game engine:
  - C++
  - Creates game-world objects with (x,y,z) coordinates and calculates what happens to them on every frame
  - E.g., a crate is up in the air on frame1. On frame 2 the game engine will calculate the new position, etc
  - Same for **non-player characters** (NPCs)!

Finite State Machines (FSMs)

- Video Games:
  - Finite State Machines
  - Decision Diagrams
  - Behavior Trees
  - Goal Oriented Action Planning
- Academic AI on agents:
  - Knowledge representation, First-order logic, Classical planning, Planning with preferences, …
  - Belief-Desire-Intention architecture, Agent-based programming, …
  - Probabilistic reasoning, Bayesian networks, Utility theory, Markov Decision Processes, …

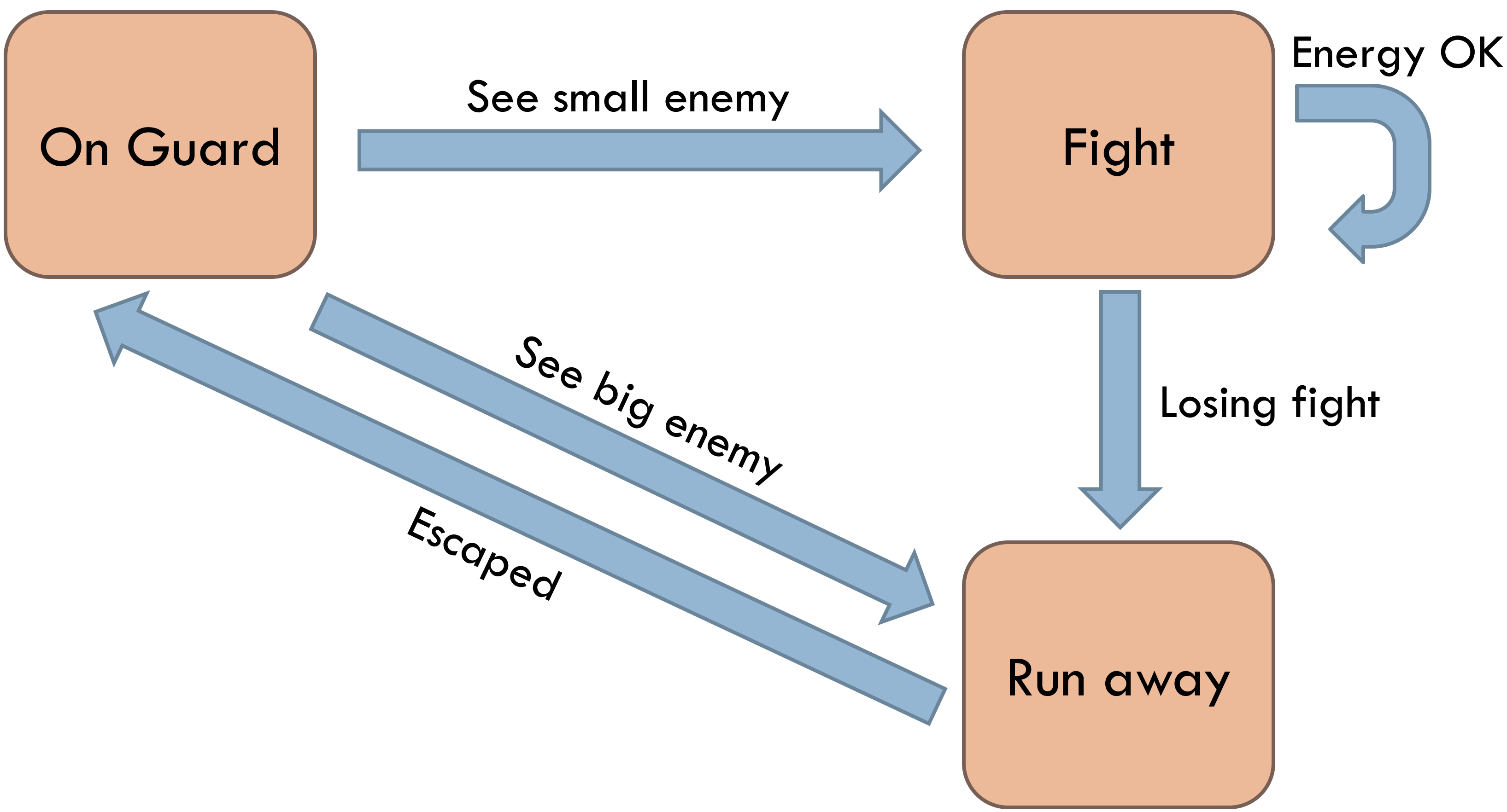
Finite State Machines (FSMs)

- Recognize a formal language



Finite State Machines (FSMs)

- NPC behavior based on high-level states



Finite State Machines (FSMs)

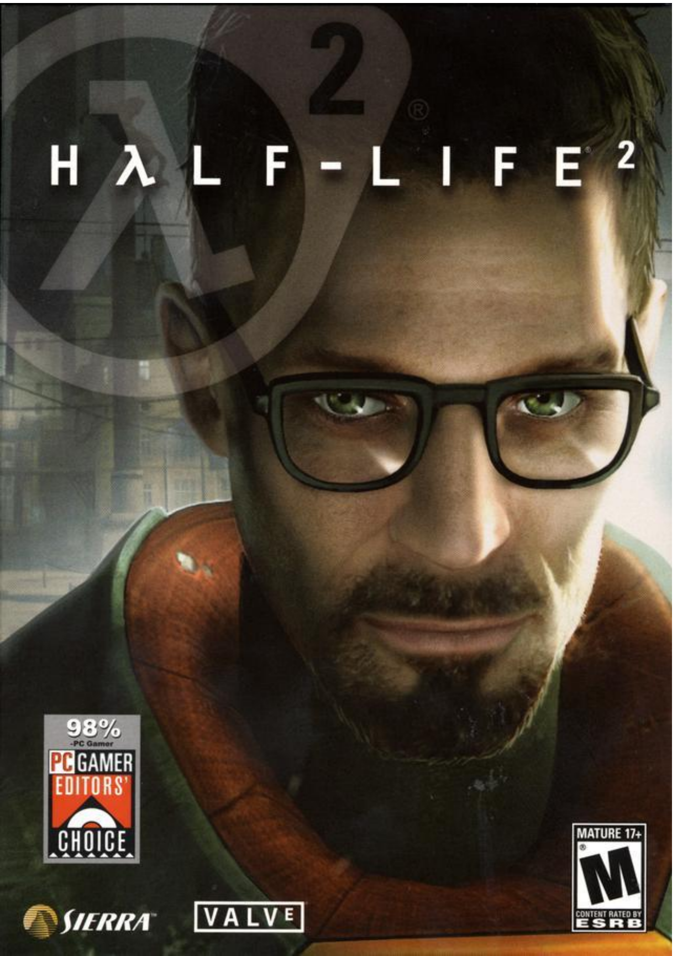
- Traditionally one of the first techniques for NPC behavior
- Very simple to understand
- Very simple to implement
  - E.g., directly using if-then-else statements

Finite State Machines (FSMs)


```
int NPC::think(){
  if (state==ONGUARD && seeSmallEnemy()){
    state=FIGHT;
    makeScarySound();
  }
  else if (state==FIGHT && energy>30){
    ...
  }
  else if ...
}
```

Finite State Machines (FSMs)

- Let's see some code from a commercial game



- HL2-SDK, npc_BaseZombie.cpp
- lines 1828-1870

```
switch ( m_NPCState )
{
case NPC_STATE_COMBAT:
...
case NPC_STATE_ALERT:
...
}
```

Finite State Machines (FSMs)

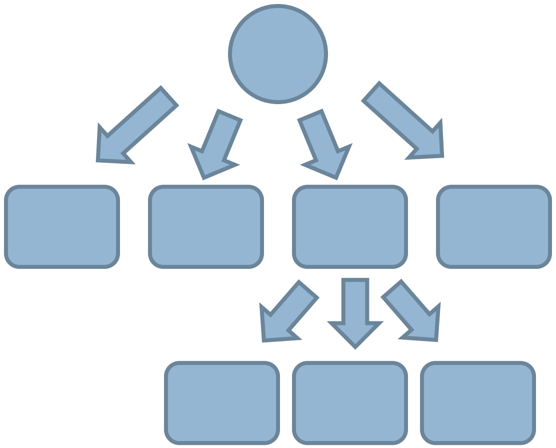
- Traditionally one of the first techniques for NPC behavior
- Very simple to understand
- Very simple to implement
  - E.g., directly using if-then-else statements
- Separation between the work of the programmer and the game designers
- But also simplistic in the behaviors that can be expressed…

Behavior Trees (BTs)

- Video Games:
  - Finite State Machines
  - Decision Diagrams
  - Behavior Trees
  - Goal Oriented Action Planning
- Academic AI on agents:
  - Knowledge representation, First-order logic, Classical planning, Planning with preferences, …
  - Belief-Desire-Intention architecture, Agent-based programming, …
  - Probabilistic reasoning, Bayesian networks, Utility theory, Markov Decision Processes, …

Behavior Trees (BTs)

- NPC behavior based on more refined conditions and strategies
  - **Tasks** have a common basic structure: they are given CPU time to **do something** and **return success or failure**
  - Leaf tasks: check a condition or execute some code
  - Composite tasks: return value depend on child tasks

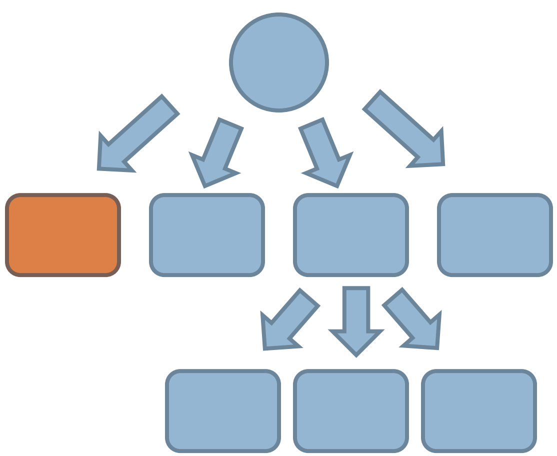


Behavior Trees (BTs)

- NPC behavior based on more refined conditions and strategies
  - Tasks have a common basic structure: they are given CPU time to do something and return success or failure
  - **Leaf tasks**: check a **condition** or **execute some code**
  - Composite tasks: return value depend on child tasks

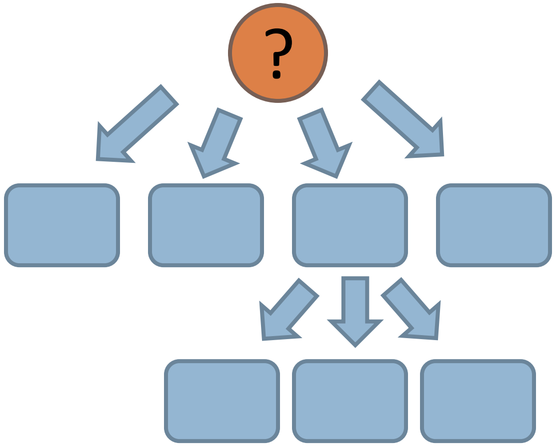
E.g., succeed if the door in front of the NPC is open

E.g., kick the door in front of the NPC



Behavior Trees (BTs)

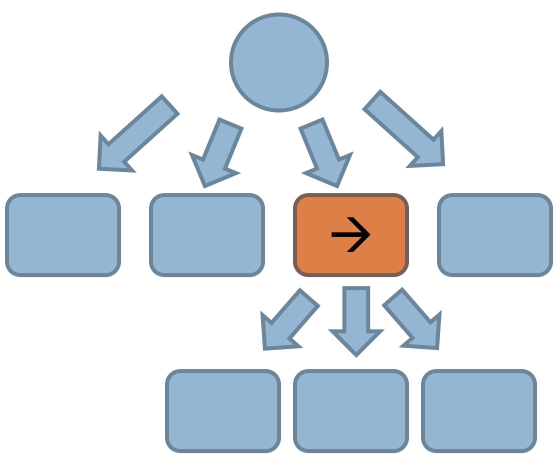
- NPC behavior based on more refined conditions and strategies
  - Tasks have a common basic structure: they are given CPU time to do something and return success or failure
  - Leaf tasks: check a condition or execute some code
  - **Composite tasks**: return value depend on **child tasks**



E.g., succeed if **any** of the child tasks succeed

Behavior Trees (BTs)

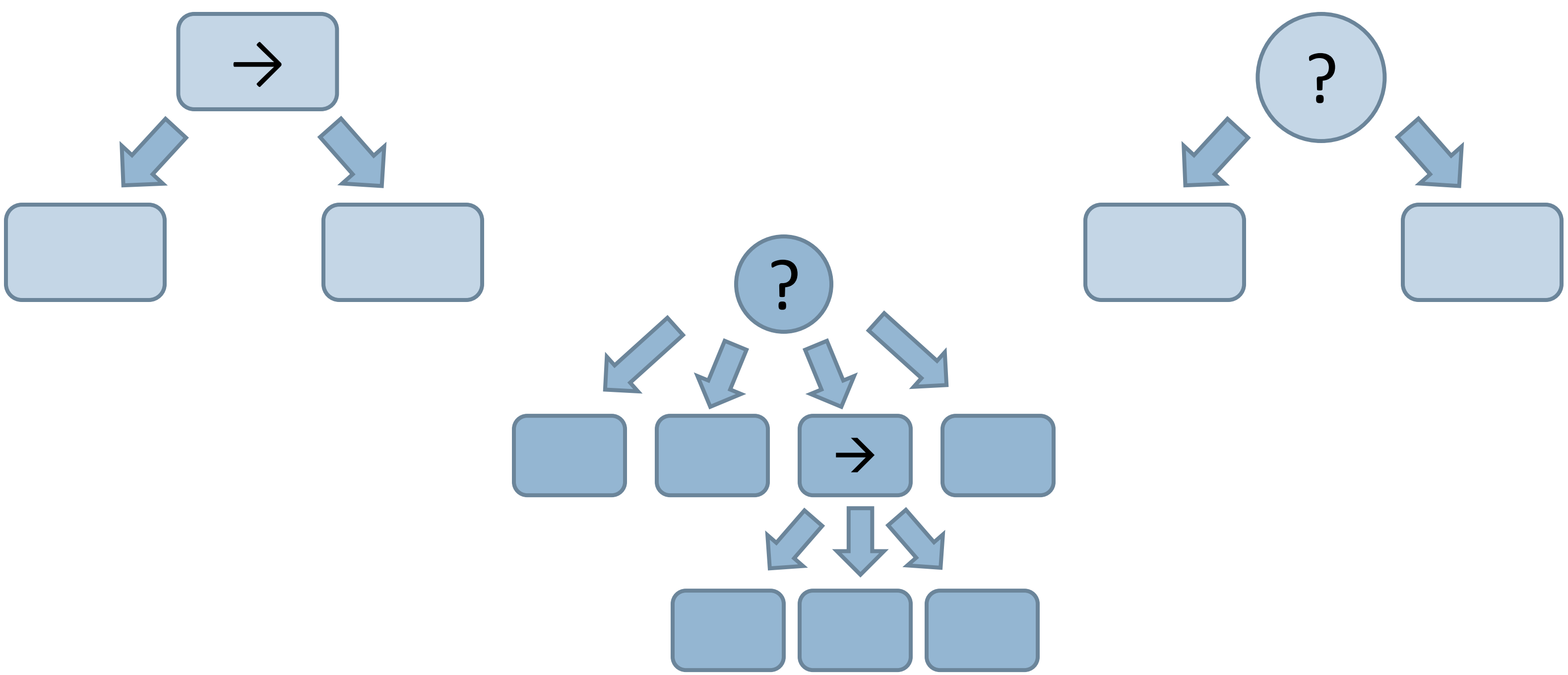
- NPC behavior based on more refined conditions and strategies
  - Tasks have a common basic structure: they are given CPU time to do something and return success or failure
  - Leaf tasks: check a condition or execute some code
  - **Composite tasks**: return value depend on child tasks



E.g., succeed if **all** of the child tasks succeed

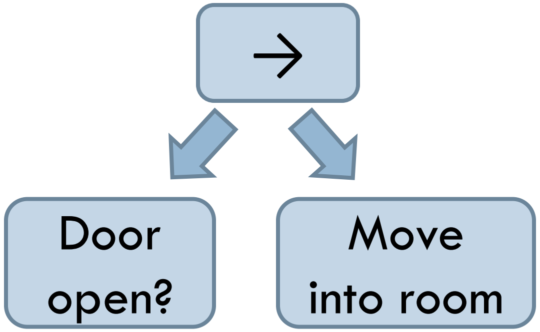
Behavior Trees (BTs)

- NPC behavior based on more refined conditions and strategies
- **Sequence** task and **Selector** task



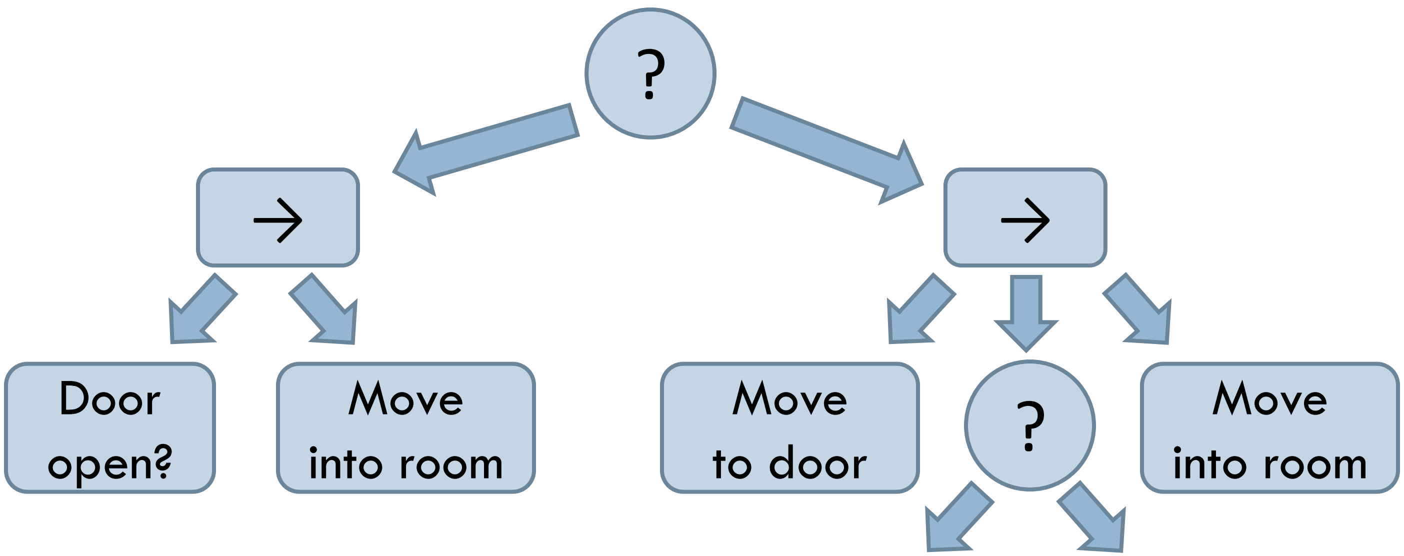
Behavior Trees (BTs)

- NPC behavior based on more refined conditions and strategies



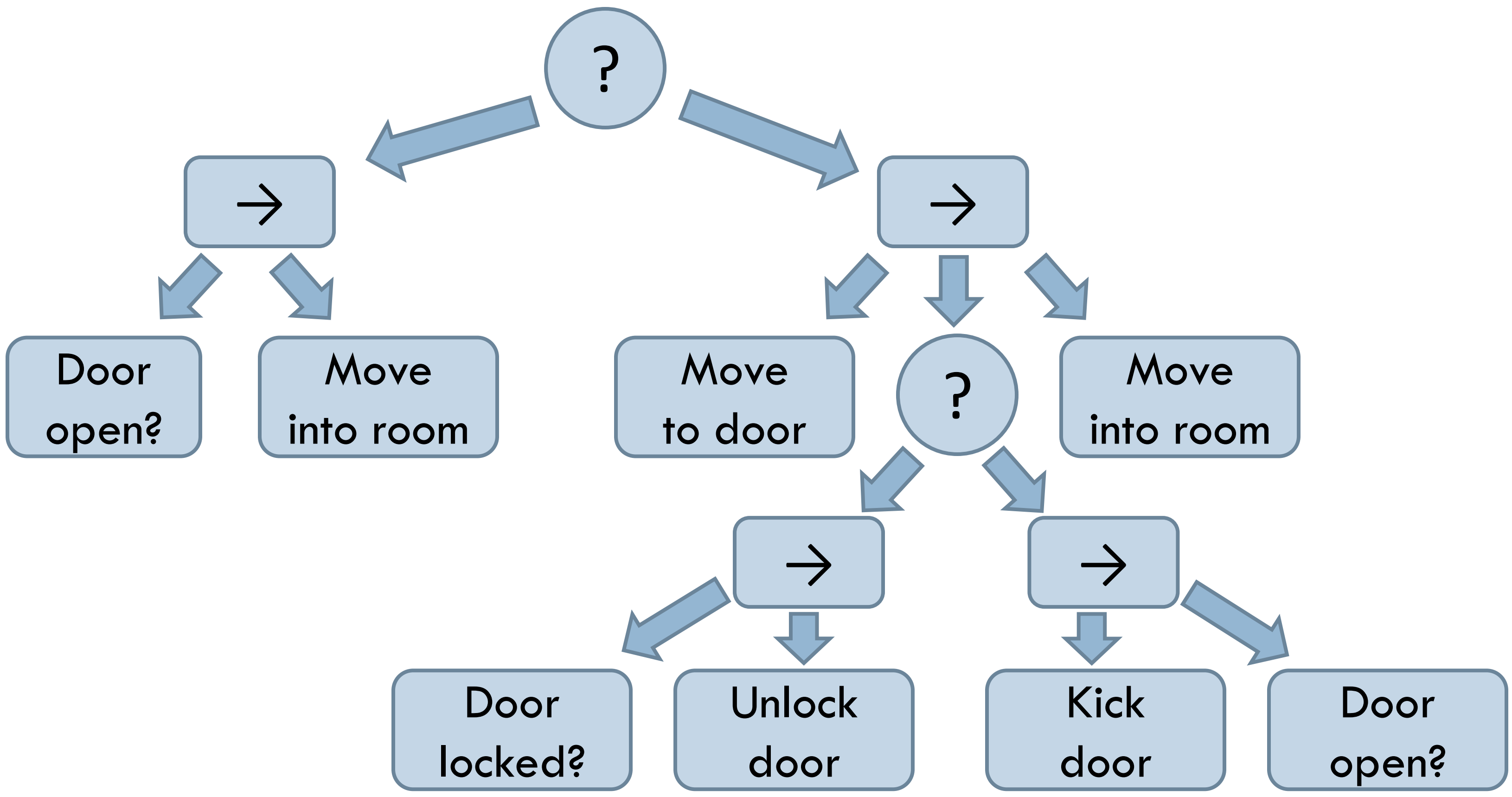
Behavior Trees (BTs)

- NPC behavior based on more refined conditions and strategies



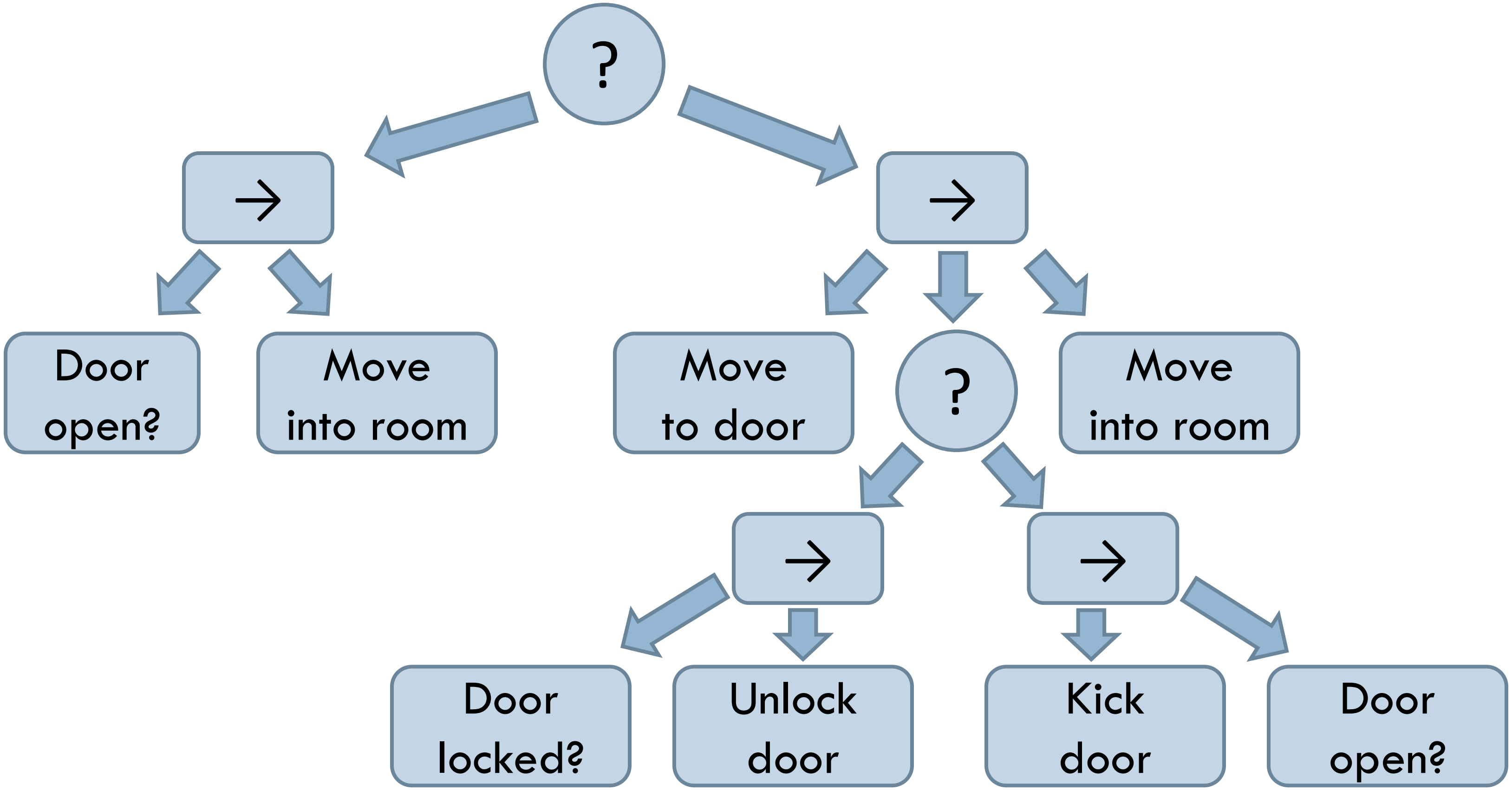
Behavior Trees (BTs)

- NPC behavior based on more refined conditions and strategies



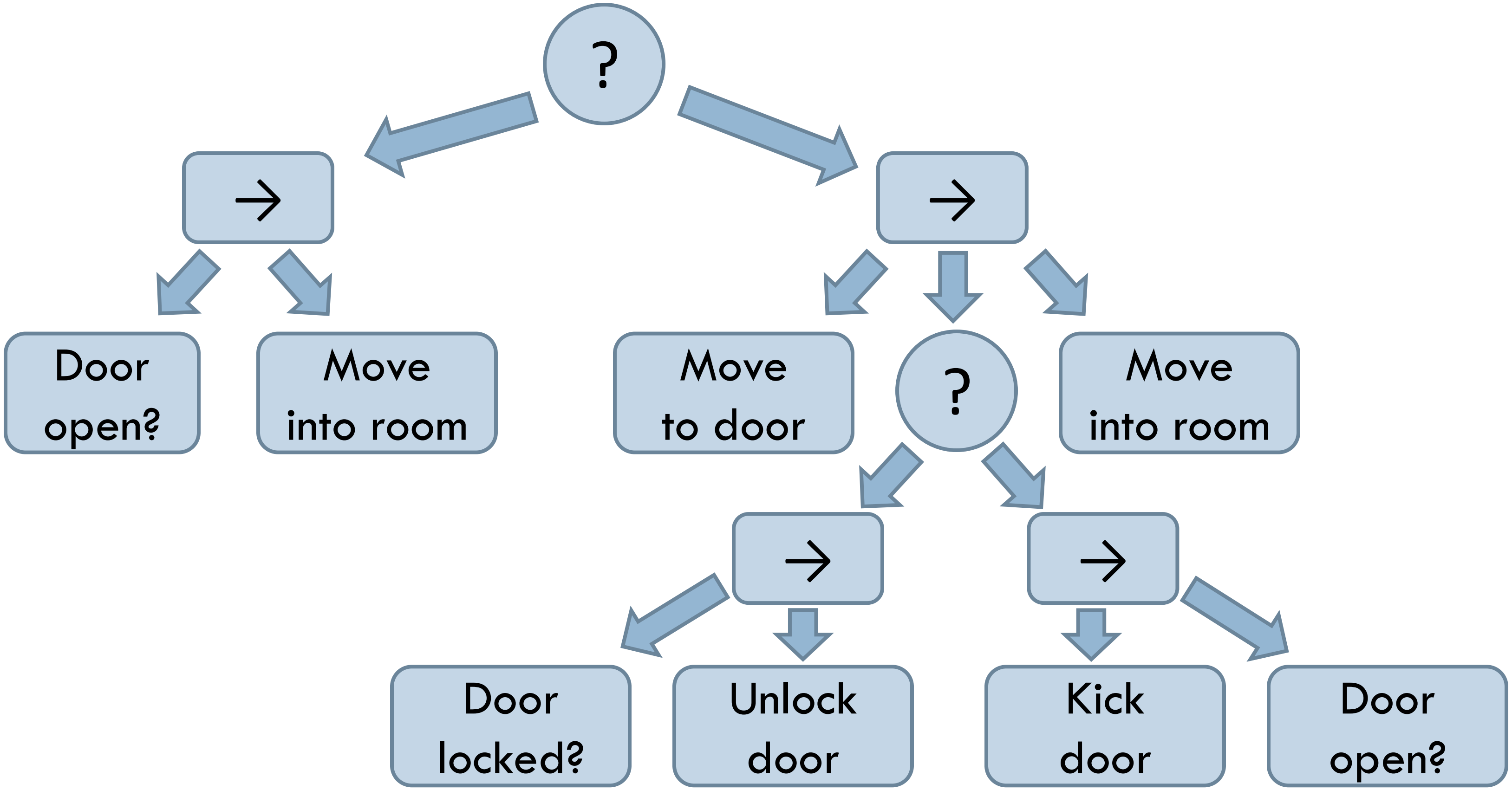
Behavior Trees (BTs)

Note that **no search** is involved in this paradigm: the behavior tree is traversed as a kind of **pre-defined program**



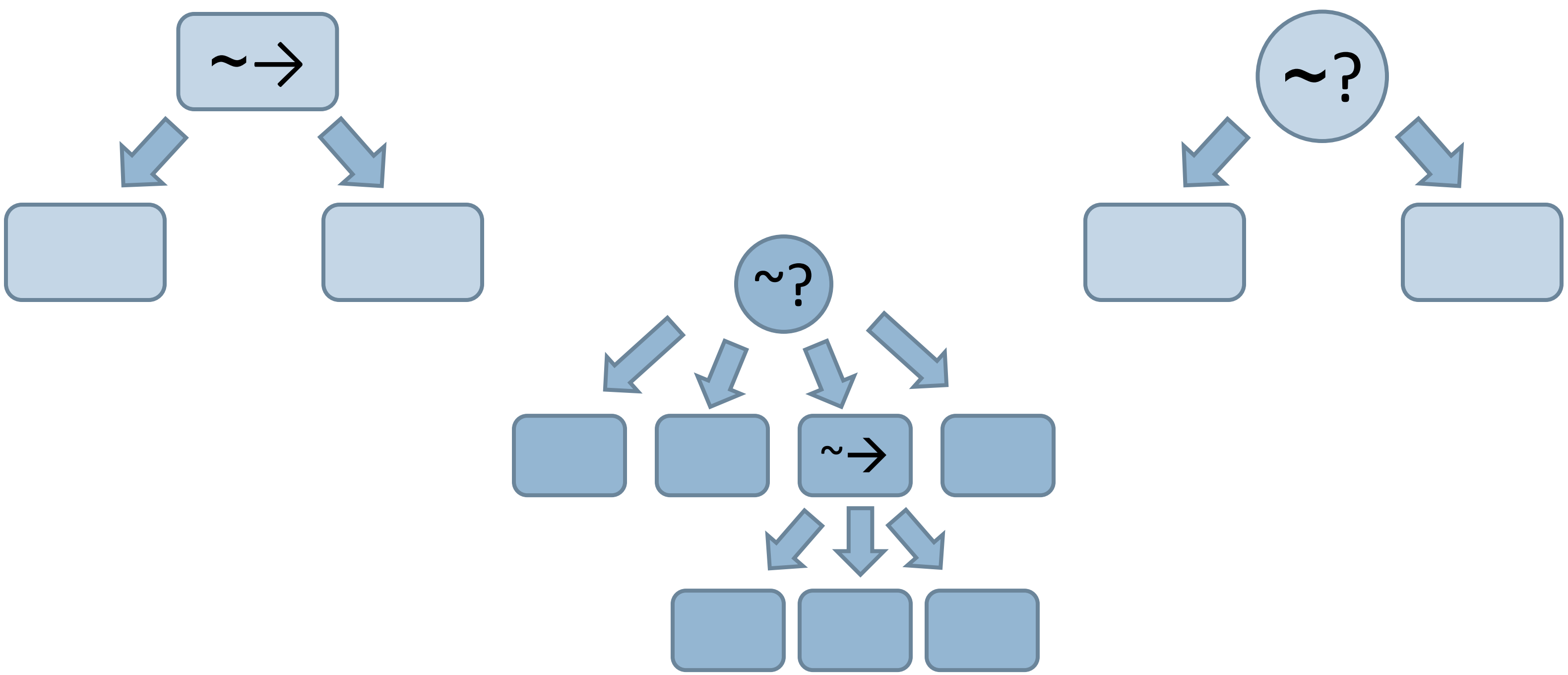
Behavior Trees (BTs)

The way the tree is traversed depends on the implementation, e.g., always start over, keep track of the current node, etc



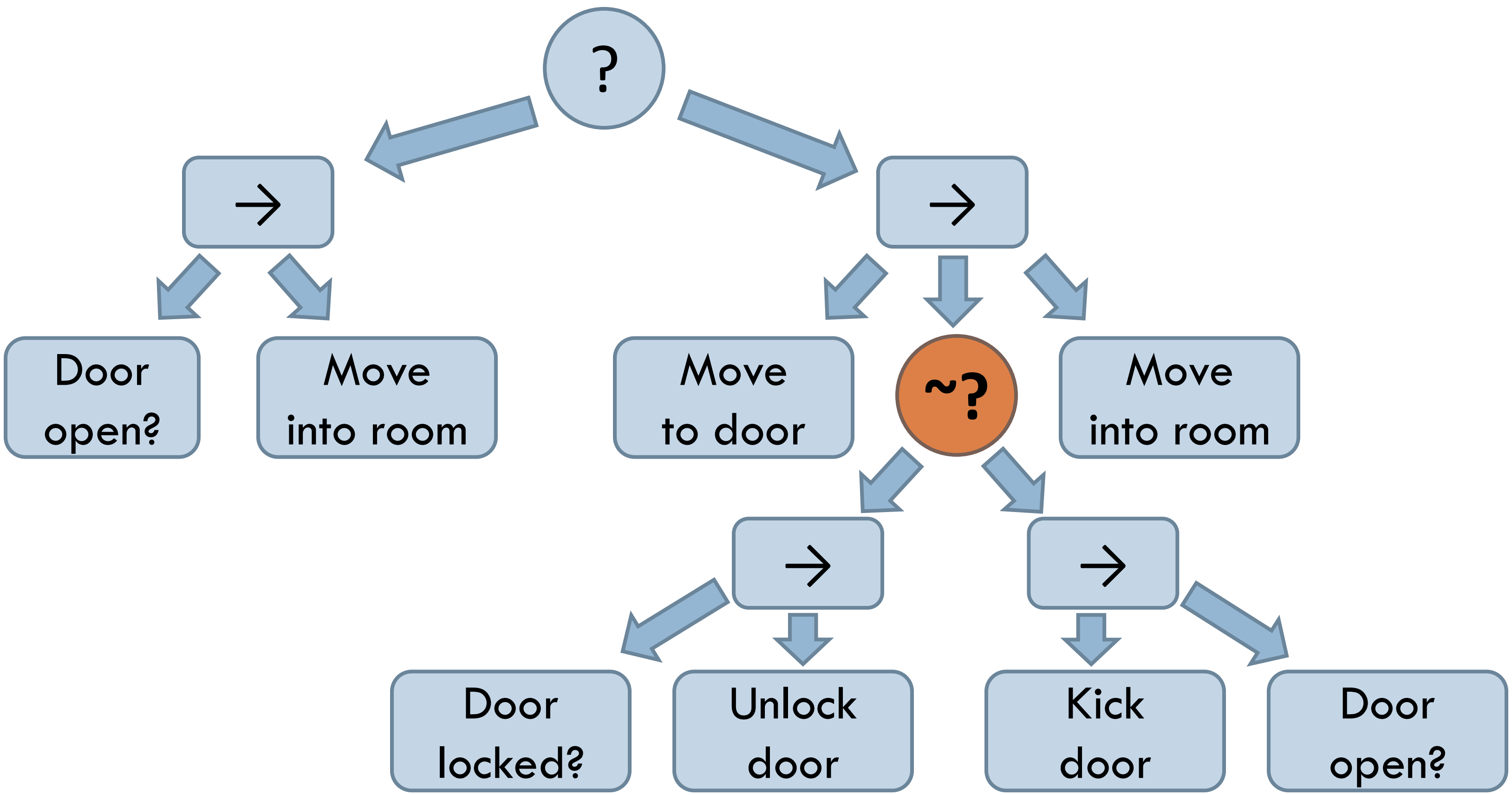
Behavior Trees (BTs)

- NPC behavior based on more refined conditions and strategies
- **Non-deterministic** sequence task and selector task



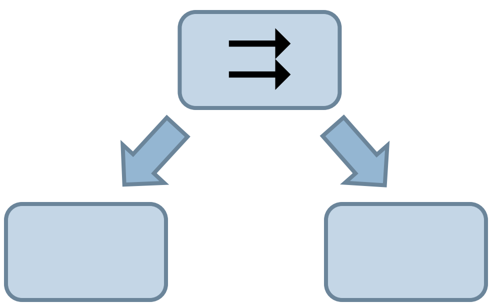
Behavior Trees (BTs)

- NPC behavior based on more refined conditions and strategies



Behavior Trees (BTs)

- NPC behavior based on more refined conditions and strategies
- **Parallel sequence** task (similar to sequence)

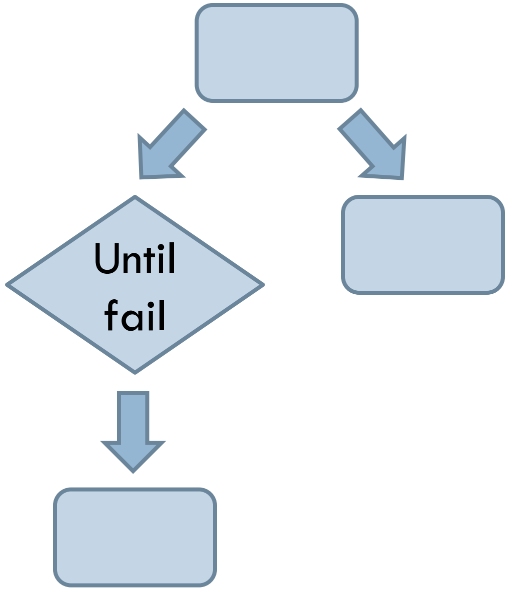


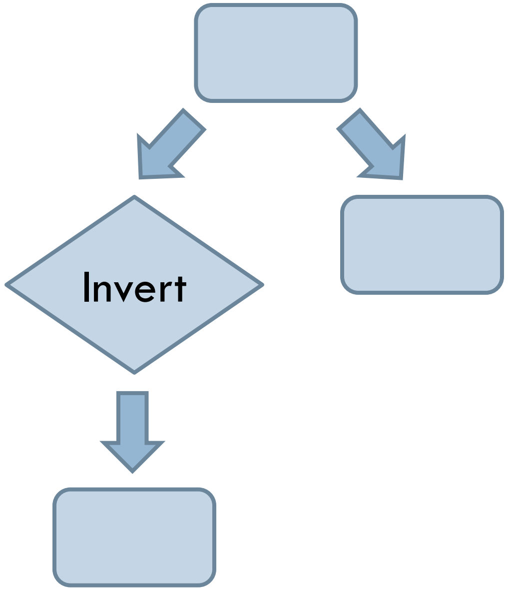
E.g., perform move actions while also shooting at target

Also used to simulate "state-like" behavior by ensuring that a condition holds

Behavior Trees (BTs)

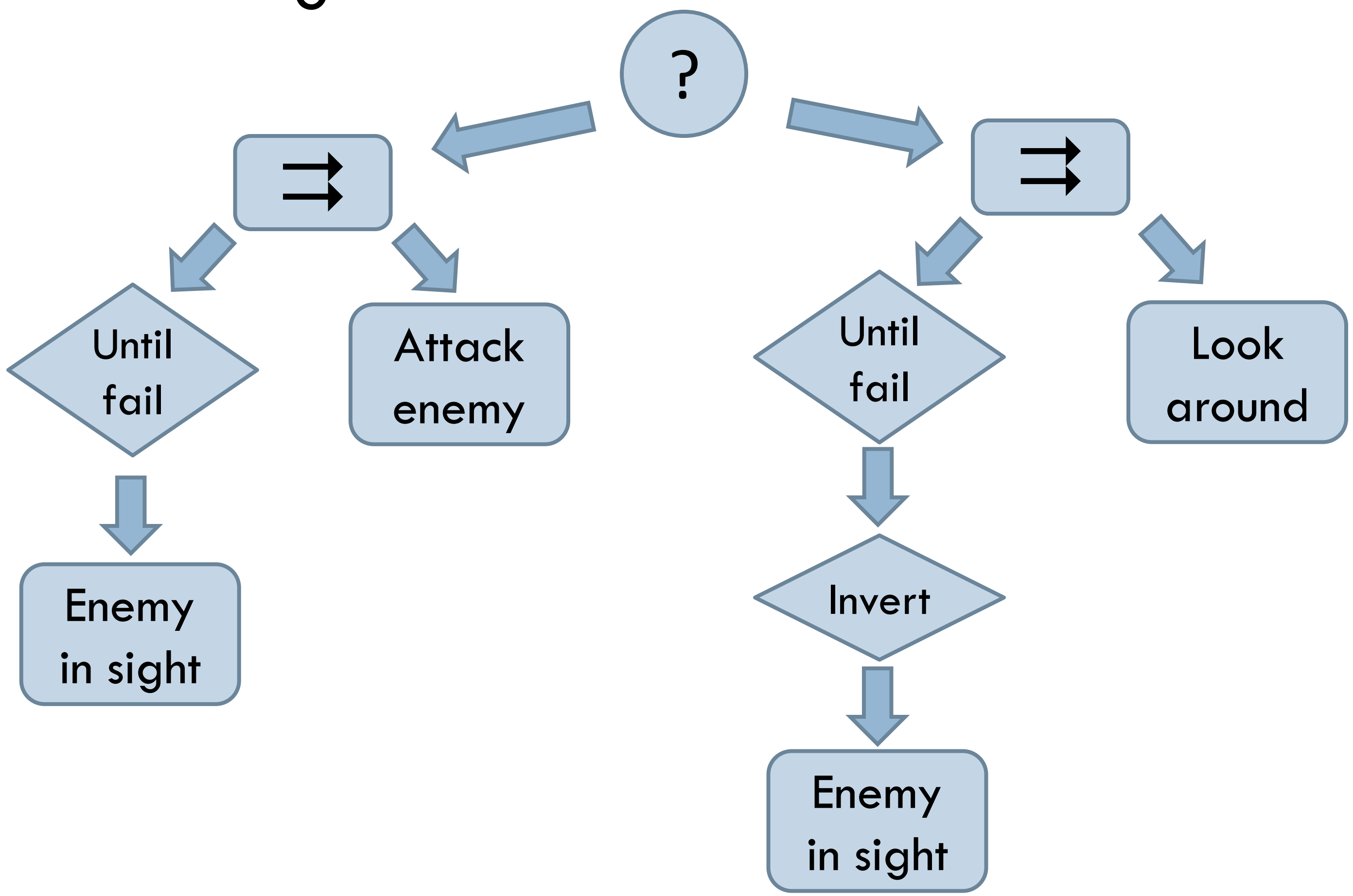
- NPC behavior based on more refined conditions and strategies
- **Decorator** tasks (wrap objects with same interface)





Behavior Trees (BTs)

- NPC behavior based on more refined conditions and strategies



Behavior Trees (BTs)

- One of the first commercial video games that used BTs is Halo2 (2004)
- Simple to understand
- Simple to implement
  - ...

- Separation between the work of the programmer and the game designers
- Offers the specification of fine-grained behaviors

Reactive Behavior

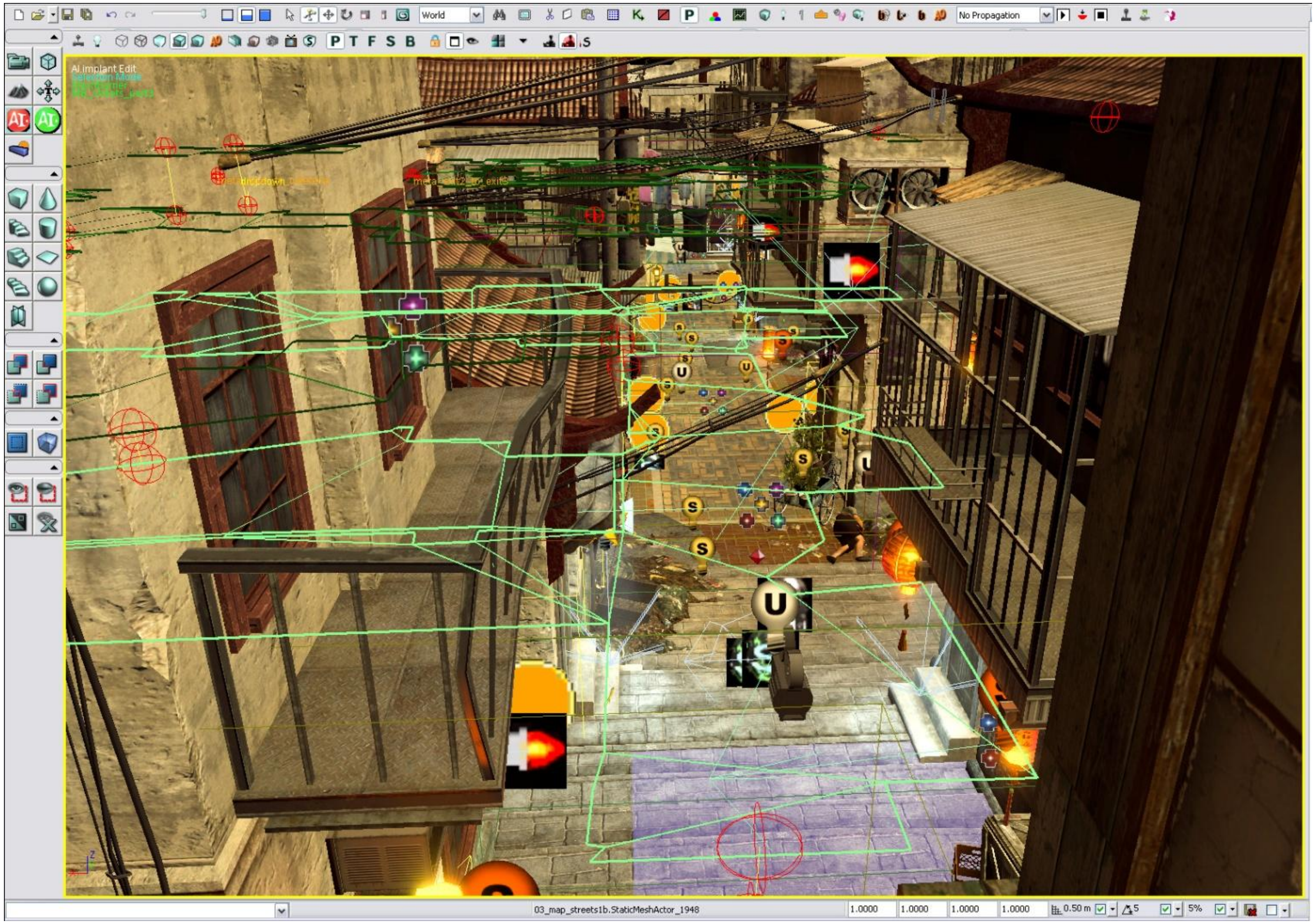
- Both FSMs and BTs are **reactive** techniques
  - The NPC follows a **pre-programmed strategy** that specifies how the NPC should react in the game depending on the **current state/node** and **conditions** that currently hold in the game-world
  - A **sequence of actions** that may be executed in the game, e.g., [move to door, kick door, move into room], need to be **represented explicitly** in the **structure** of the FSMs or BTs

Reactive Behavior

- Historically, the vast amount of video games with NPCs use FSMs and BTs for NPC decision making
  - Simple to understand/implement
  - Separation between programmers and game designers
  - Any extensions needed can be handled effectively using programming tricks
  - The behavior is strengthened by extra information in the game world that is carefully prepared for NPCs

Reactive Behavior

- A game level from the eyes of an NPC



Reactive Behavior

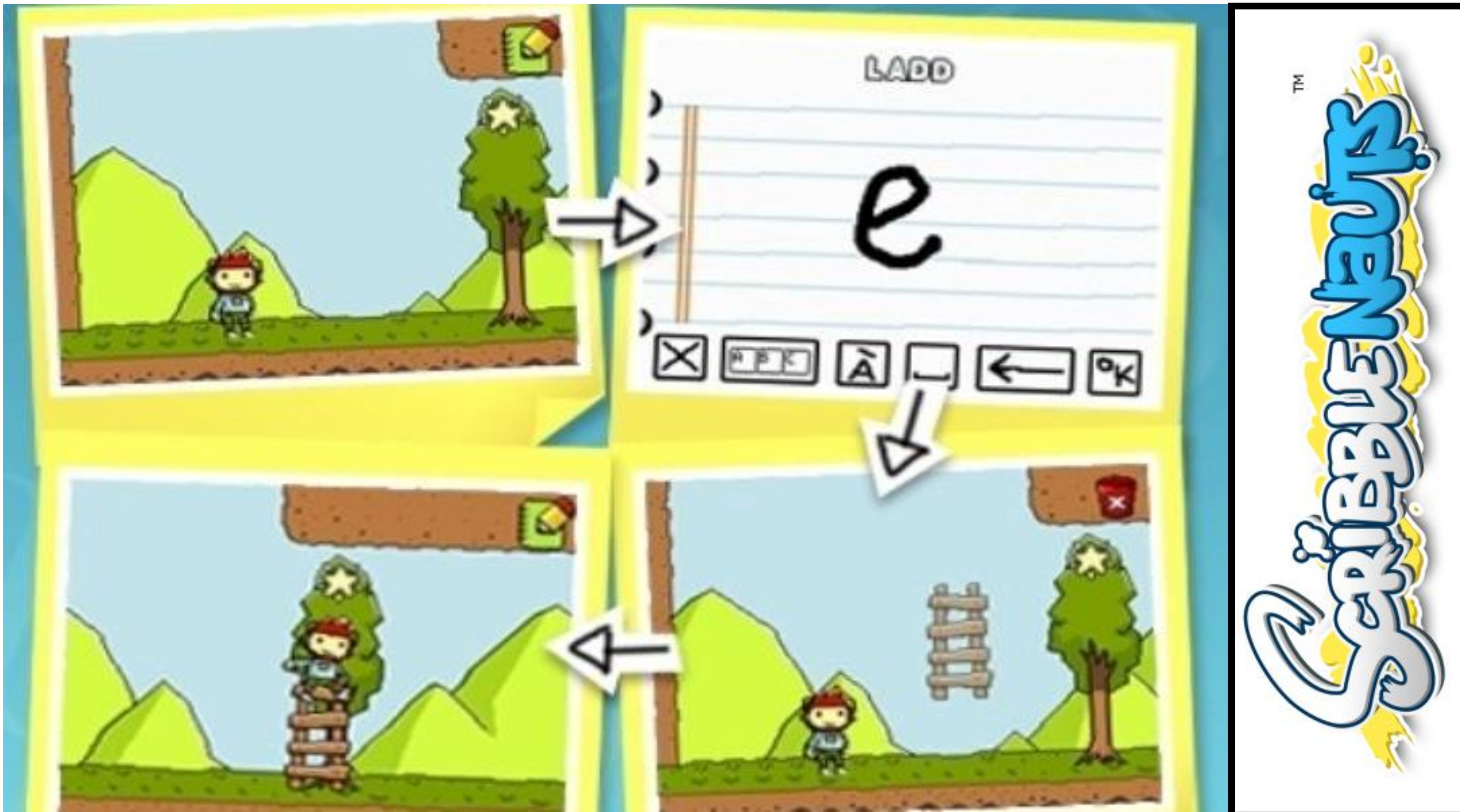
- A game level from the eyes of an NPC

Reactive Behavior

- The situation today
  - Open worlds with **increasing available interactions**
  - NPCs need to be **autonomous**, with their **own agenda**, goals, personality

Reactive Behavior

- The situation today



Reactive Behavior

- The situation today
  - Under these circumstances, maintaining the possible and applicable interactions using reactive techniques becomes **complex** and **difficult**
  - The need for more **flexible** techniques arises

Reactive Behavior

- The situation today

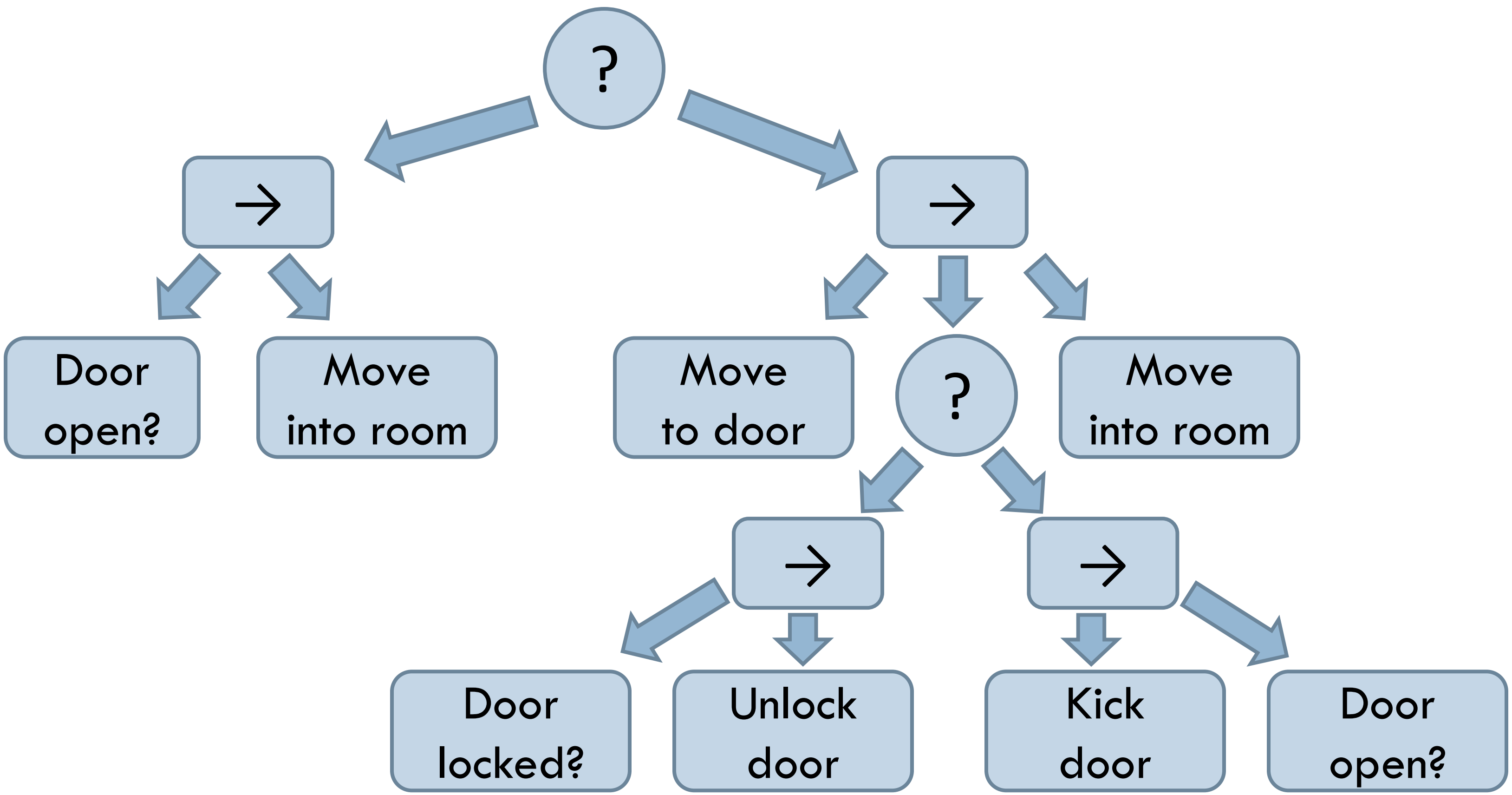
[youtube link]

Goal Oriented Action Planning (GOAP)

- Video Games:
  - Finite State Machines
  - Decision Diagrams
  - Behavior Trees
  - Goal Oriented Action Planning
- Academic AI on agents:
  - Knowledge representation, First-order logic, Classical planning, Planning with preferences, …
  - Belief-Desire-Intention architecture, Agent-based programming, …
  - Probabilistic reasoning, Bayesian networks, Utility theory, Markov Decision Processes, …

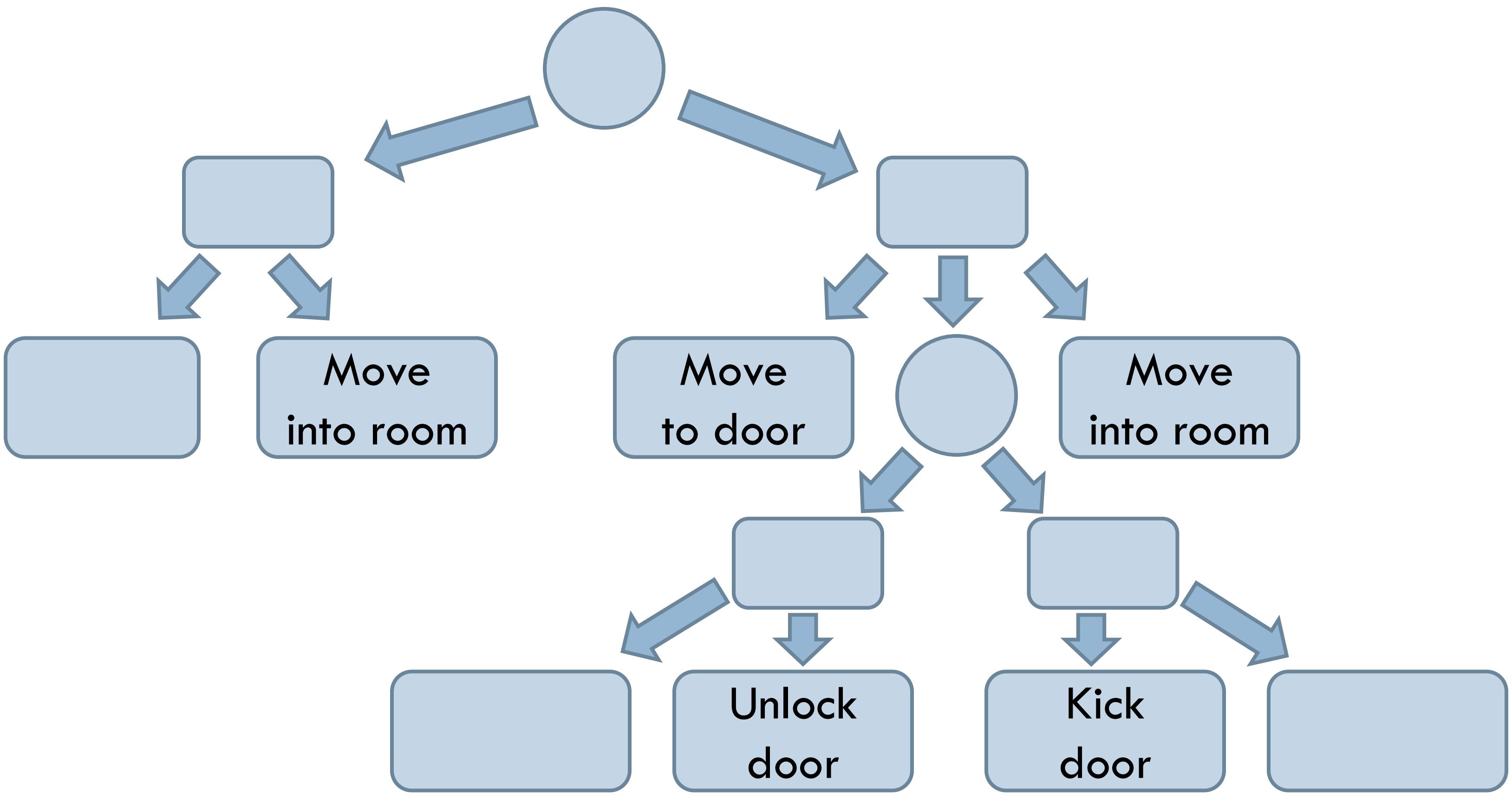
Goal Oriented Action Planning (GOAP)

- Replace the pre-defined strategies with a description of goals and available actions



Goal Oriented Action Planning (GOAP)

- Replace the pre-defined strategies with a description of goals and **available actions**



Goal Oriented Action Planning (GOAP)

- Replace the pre-defined strategies with a description of goals and **available actions**

Move into room	Preconditions: Door open Effects: In room
Move to door	Preconditions: - Effects: At door
Unlock door	Preconditions: Hold key Effects: Door open
Kick door	Preconditions: - Effects: Door open

Goal Oriented Action Planning (GOAP)

- Replace the pre-defined strategies with a description of **goals** and available actions

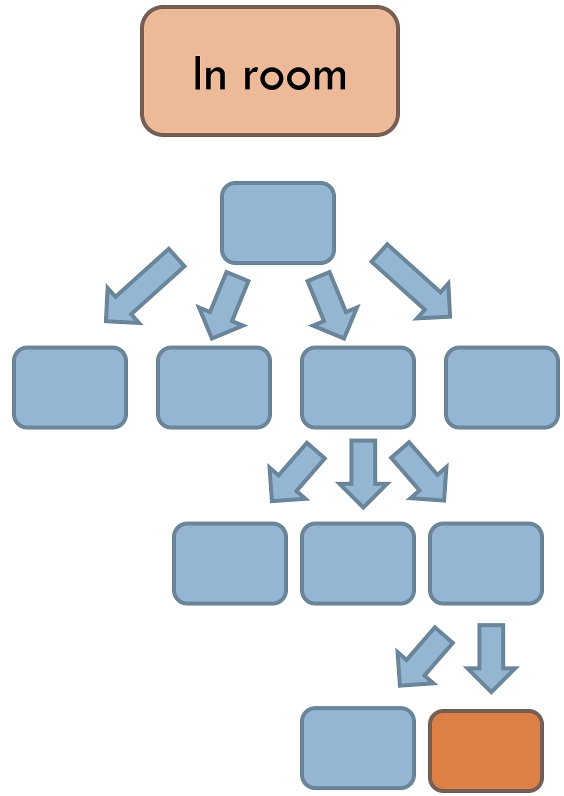
Move into room	Preconditions: Door open Effects: In room
Move to door	Preconditions: - Effects: At door
Unlock door	Preconditions: Hold key Effects: Door open
Kick door	Preconditions: - Effects: Door open

In room

Goal Oriented Action Planning (GOAP)

- **Search** in **real-time** for a **strategy** that achieves the goal in the current state

Action	Preconditions / Effects
Move into room	Preconditions: Door open Effects: In room
Move to door	Preconditions: - Effects: At door
Unlock door	Preconditions: Hold key Effects: Door open
Kick door	Preconditions: - Effects: Door open



Goal Oriented Action Planning (GOAP)

- Advantages
  - Easy to **manage** a **large number** of generated behaviors
  - Able to achieve **different behaviors** that satisfy the given requirements under different conditions **without explicitly listing** the resulting strategies

- But it needs to **solve** planning problems in **a few frames!**

Behavior Trees (BTs)

- One of the first commercial video games that used BTs is Halo2 (2004)
- Simple to understand
- Simple to implement
  - ...

- Separation between the work of the programmer and the game designers
- Offers the specification of fine-grained behaviors

Goal Oriented Action Planning (GOAP)

- One of the first commercial video games that used GOAP is FEAR (2005)
- Not so simple to understand
- Not so simple to implement
  - …
- Not so clear separation between the work of the programmer and the game designers
- The specification of fine-grained behaviors is actually tricky

Goal Oriented Action Planning (GOAP)

- Some details about GOAP in FEAR:
  - One AI programmer responsible for NPC behavior
  - Idea: Different behaviors can be achieved among characters by using GOAP and providing each character with same goals but a different set of available actions

Goal Oriented Action Planning (GOAP)

Soldier

	Action
1	AI/Actions/Attack
2	AI/Actions/AttackCrouch
3	AI/Actions/SuppressionFire
4	AI/Actions/SuppressionFireFromCover
5	AI/Actions/FlushOutWithGrenade
6	AI/Actions/AttackFromCover
7	AI/Actions/BlindFireFromCover
8	AI/Actions/AttackGrenadeFromCover
9	AI/Actions/AttackFromView
10	AI/Actions/DrawWeapon
11	AI/Actions/HolsterWeapon
12	AI/Actions/ReloadCrouch
13	AI/Actions/ReloadCovered
14	AI/Actions/InspectDisturbance
15	AI/Actions/LookAtDisturbance
16	AI/Actions/SurveyArea
17	AI/Actions/DodgeRoll
18	AI/Actions/DodgeShuffle
19	AI/Actions/DodgeCovered
20	AI/Actions/Uncover
21	AI/Actions/AttackMelee

Assassin

	Action
1	AI/Actions/Attack
2	AI/Actions/InspectDisturbance
3	AI/Actions/LookAtDisturbance
4	AI/Actions/SurveyArea
5	AI/Actions/AttackMeleeUncloaked
6	AI/Actions/TraverseBlockedDoor
7	AI/Actions/UseSmartObjectNodeMounted
8	AI/Actions/MountNodeUncloaked
9	AI/Actions/DismountNodeUncloaked
10	AI/Actions/TraverseLinkUncloaked
11	AI/Actions/AttackFromAmbush
12	AI/Actions/DodgeRollParanoid
13	AI/Actions/AttackLungeUncloaked
14	AI/Actions/LopeToTargetUncloaked

Rat

	Action
1	AI/Actions/Animate
2	AI/Actions/Idle
3	AI/Actions/GotoNode
4	AI/Actions/UseSmartObjectNode

Goal Oriented Action Planning (GOAP)

- Simplifying STRIPS planning:
  - Literals are stored as variables (essentially having one argument)
  - The state is stored as an array of a fixed size
  - Search goes up to depth …3

- A* for path finding…
- A* also for planning!

Reactive Planning Vs. Classical Planning

- HALO2 (2004)
- Since then BTs have become a standard for NPC behavior

- FEAR (2005)
- Since then GOAP has not picked up much speed

Reactive Planning Vs. Classical Planning

Behavior Trees

Goal Oriented
Action Planning

Reactive Planning Vs. Classical Planning

Behavior Trees

Goal Oriented Action Planning

- A combination of these techniques?
  - BTs for reactive decision making
  - GOAP for tactical decision making

Artificial Intelligence and Video Games

- Amazing tools available for (indie) game developers!

Artificial Intelligence and Video Games

- Source available!

Bibliography

- Material
  - Artificial Intelligence for Games, Second Edition. Ian Millington, John Funge. Morgan Kaufmann Publishers Inc., 2009.
  - Sections 5.3, 5.4