Principles of Computer Game Design and Implementation

Lecture 24

We already learned

- Decision Tree
- Finite State Machine

FSM Problems: Reminder

- Explosion of states
- Too predictable
- Often created with ad hoc structure
- Mixture of different level concepts:
  - Game engine developer
    - “Atomic” actions and tests linking AI to the game world
  - AI developer
    - Complex behaviours
  - **FSM States** combine both
    - What to do with more than one action per state?

Outline for today

- Behaviour tree

Behaviour Trees

- Inspired by a number of techniques
  - Hierarchical FSMs
  - Scheduling / planning
  - Planning

- First (famously) used in **Halo 2**
  - Picked up by other developers

- Clear separation between AI and Game Engine

Tasks

AI agent runs a ***task***. A task can **succeed** or **fail**

- Simple tasks
  - Conditions
  - Actions

Game engine developers

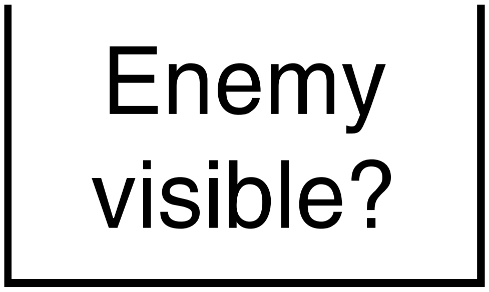
- Complex tasks
  - Built hierarchically from other tasks using
    - Composites
    - Decorators

AI developers

Conditions

- Test some properties of the game.
  - Proximity
  - Line of sight
  - Character properties (has ammo etc)

- Succeed or fail
  - Like **if-then** test
- Typically execute fast



Enemy visible?

Actions

- **Alter** the state of the game
  - Animation, audio
  - Play a dialog
  - Movements
  - Change the character internal state (cure)
- Can take time
- **Typically succeed**
  - Failing is like an exception

Attack

Task Interface

- Actions and tests are used in other AI techniques

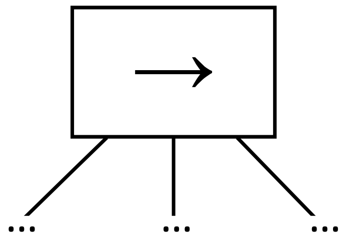
  but…

- In behaviour trees, all tasks have **the same interface**
  - Simple case: return a Boolean value
    - Succeed / fail
  - Can be easily combined together

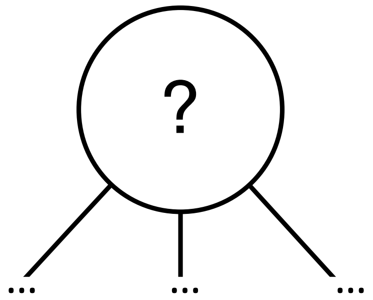
Composites

Composites run their child tasks in turn

- **Sequence**
  - – Terminates immediately with failure if any of child tasks fail
  - – Succeeds if all child tasks succeed

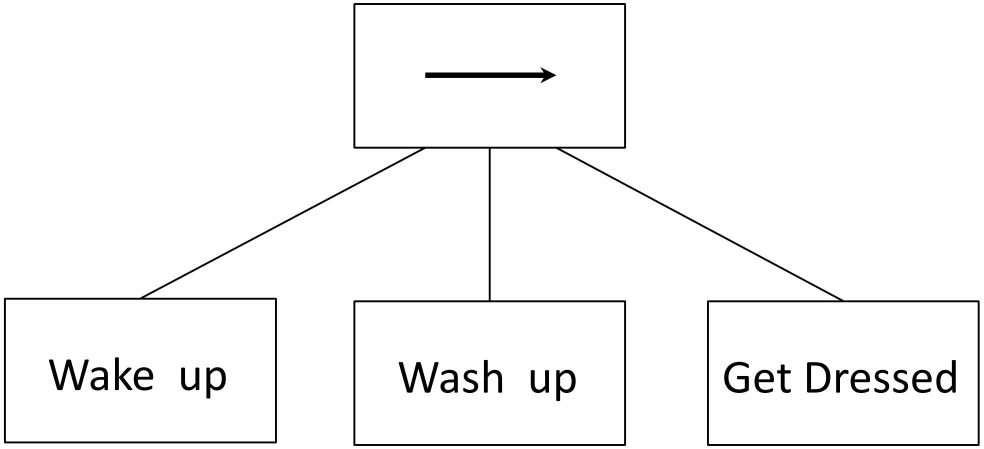


- **Selector**
  - – Terminates immediately with success if any of the child tasks succeed
  - – Fails if all child tasks fail



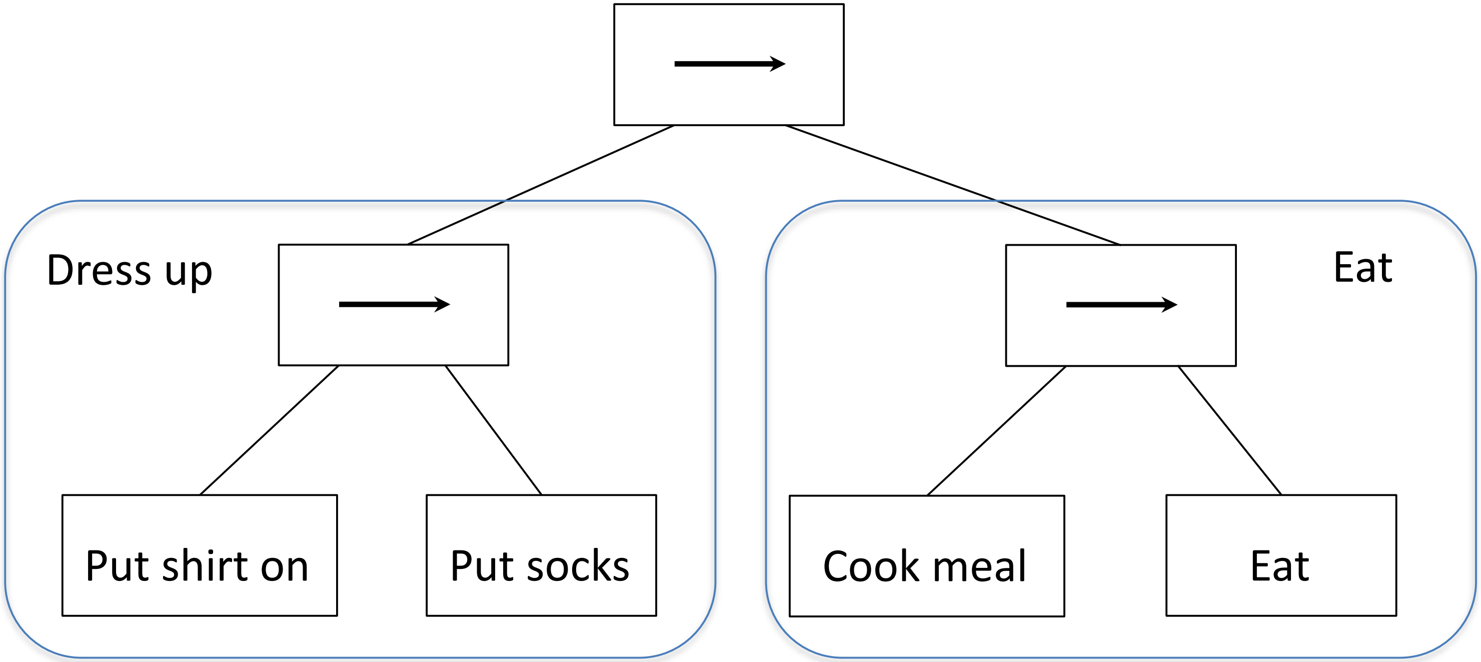
Sequence of Actions

- Sequence of tasks to achieve a goal
  - Get ready for Uni task



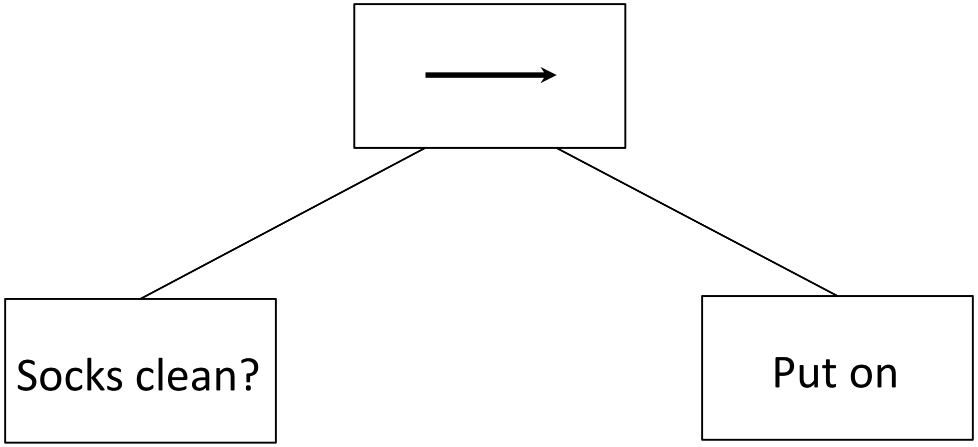
Sequences of Sequences

- Logically, there is no need to have sequences as children of sequences, but…



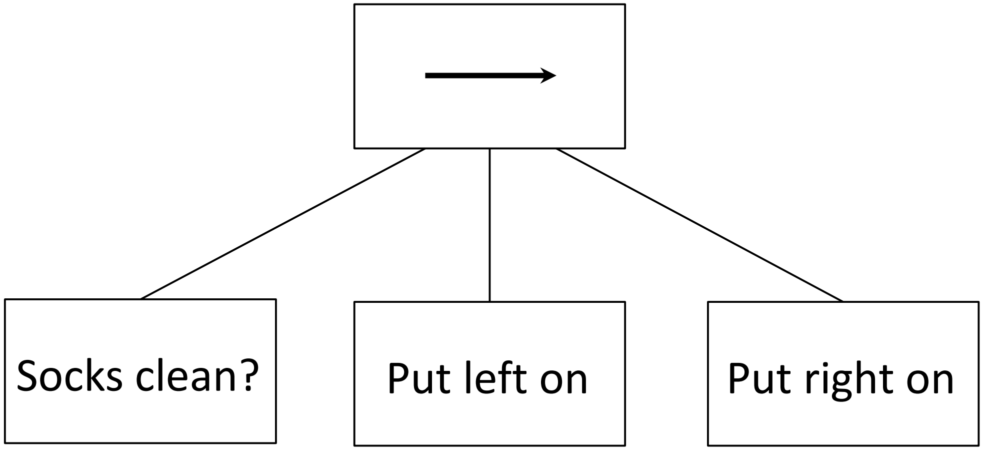
Sequence As Conditions

- Sequence terminates immediately with failure if any of child tasks fail
  - The second task is run ***only*** when first succeeds



Conditions and Actions

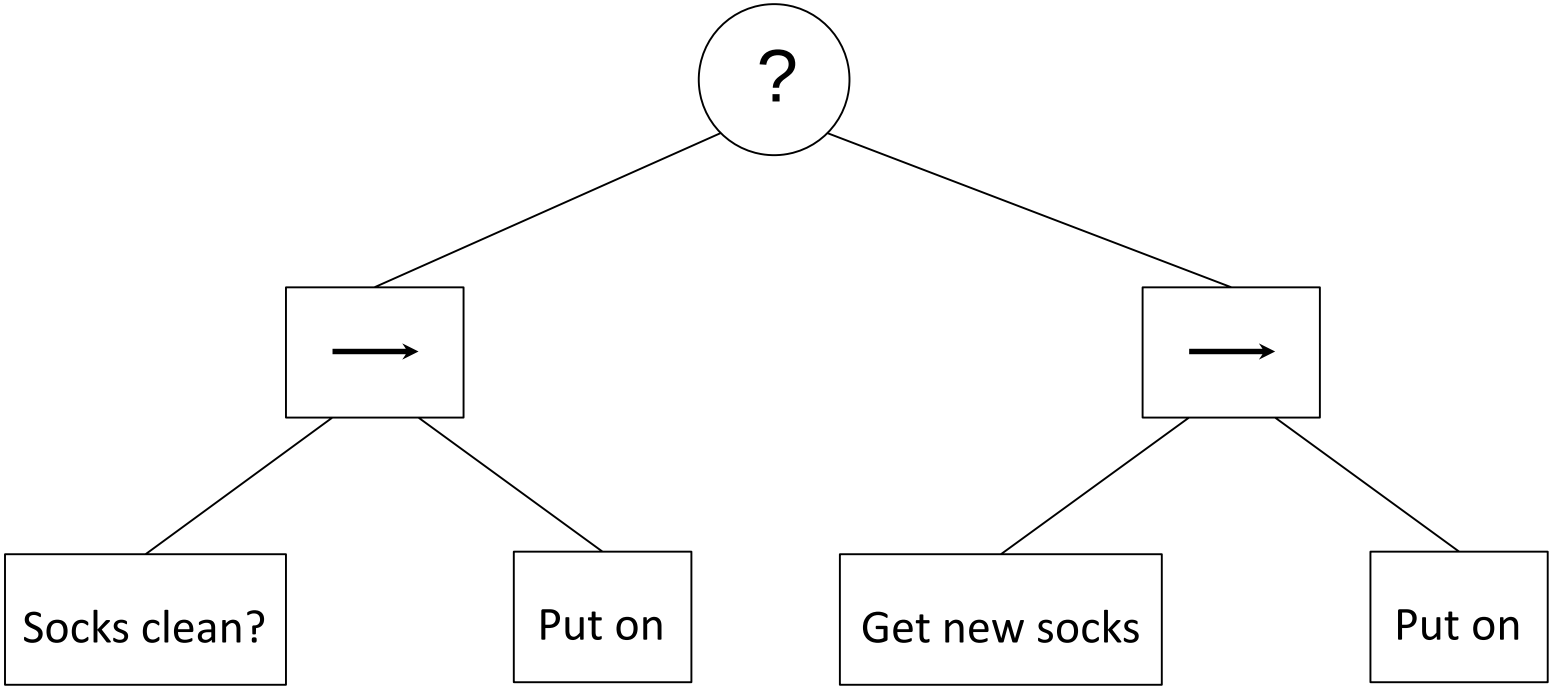
- More than one child



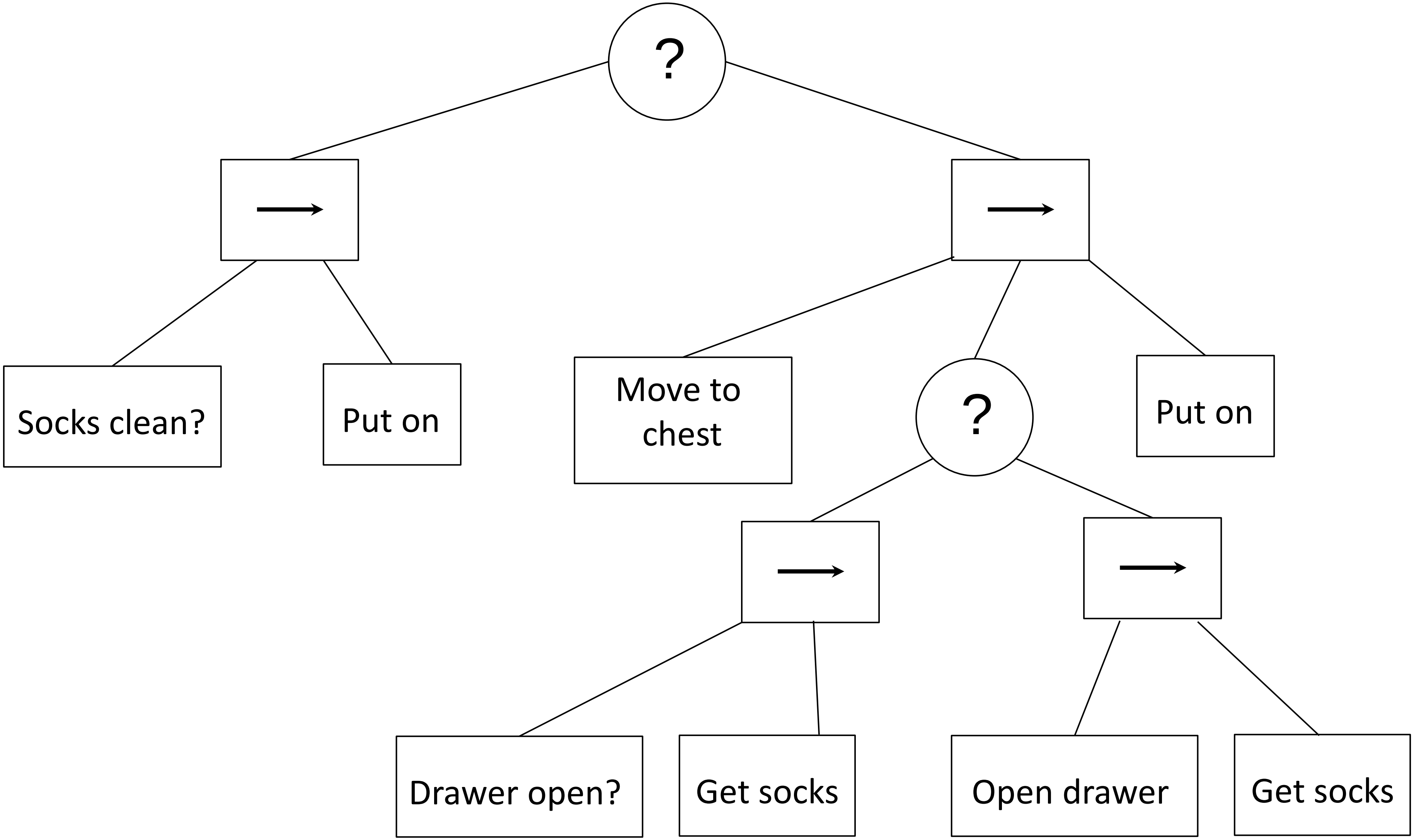
- But what if socks are not clean?

Selectors

Terminate immediately with success if any of the child tasks succeed



More Complicated Behaviour



Conditions Actions and Composites

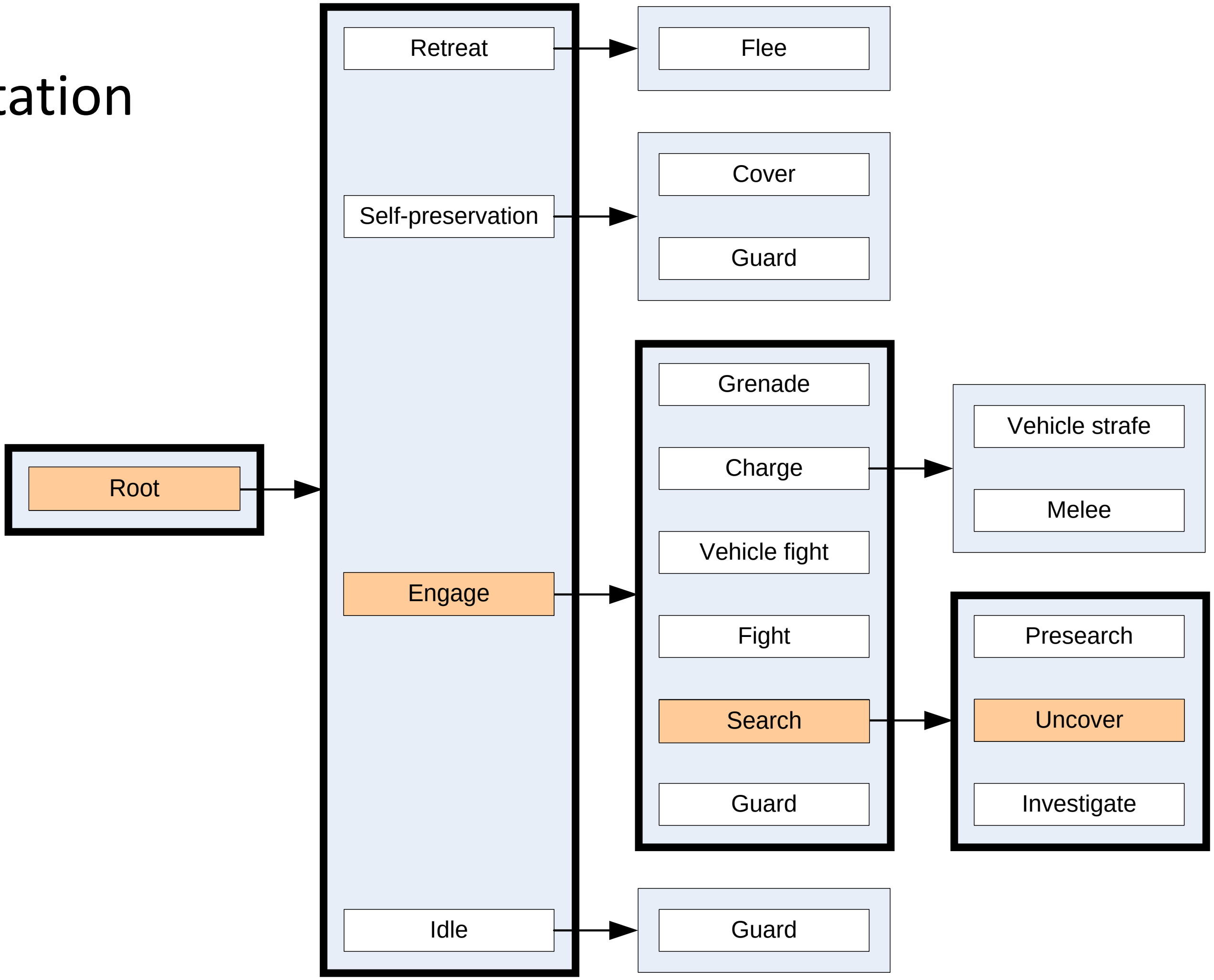
- Conditions and actions combined together with composites allow to express complex behaviours

- Goal-driven scripting

- *Reactive plans*: what if…
  - But not a *planner!*

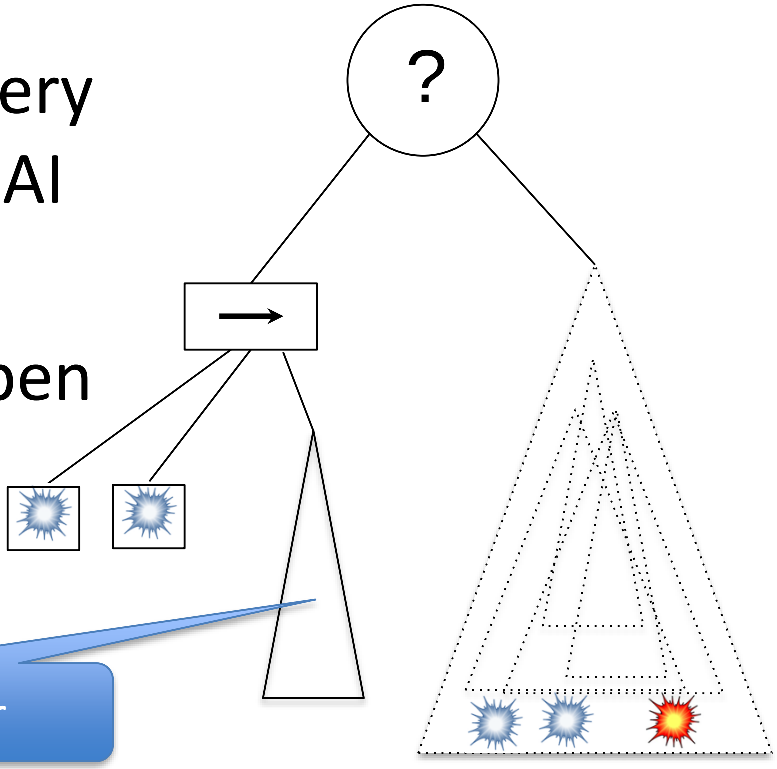
Halo 2 Decision-Making

From Demián Isla's
GDC'05 presentation



Bug Fixes as a Hack

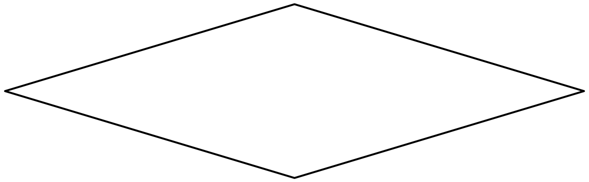
- Behaviour trees are highly adaptable
  - Suppose you discovered a very rare condition under which AI fails
  - You know what should happen
  - But time is pressing



Correct behaviour

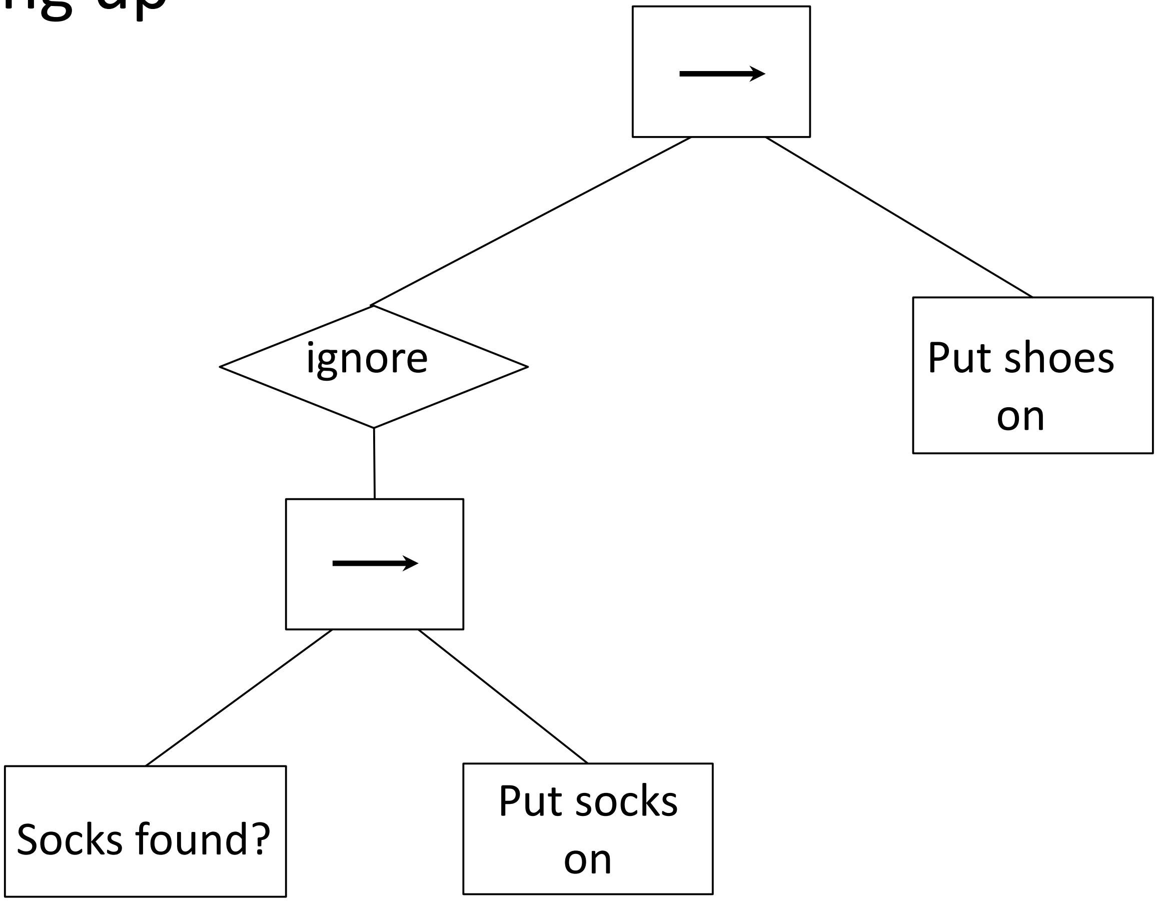
Decorators

- Decorators modify the behaviour of a task
  - Limit (Loop)
    - Time limit / Attempts
  - UntilFail
    - Repeat the task until it fails
  - Inverter
  - Ignorer
    - Runs the task and always reports success



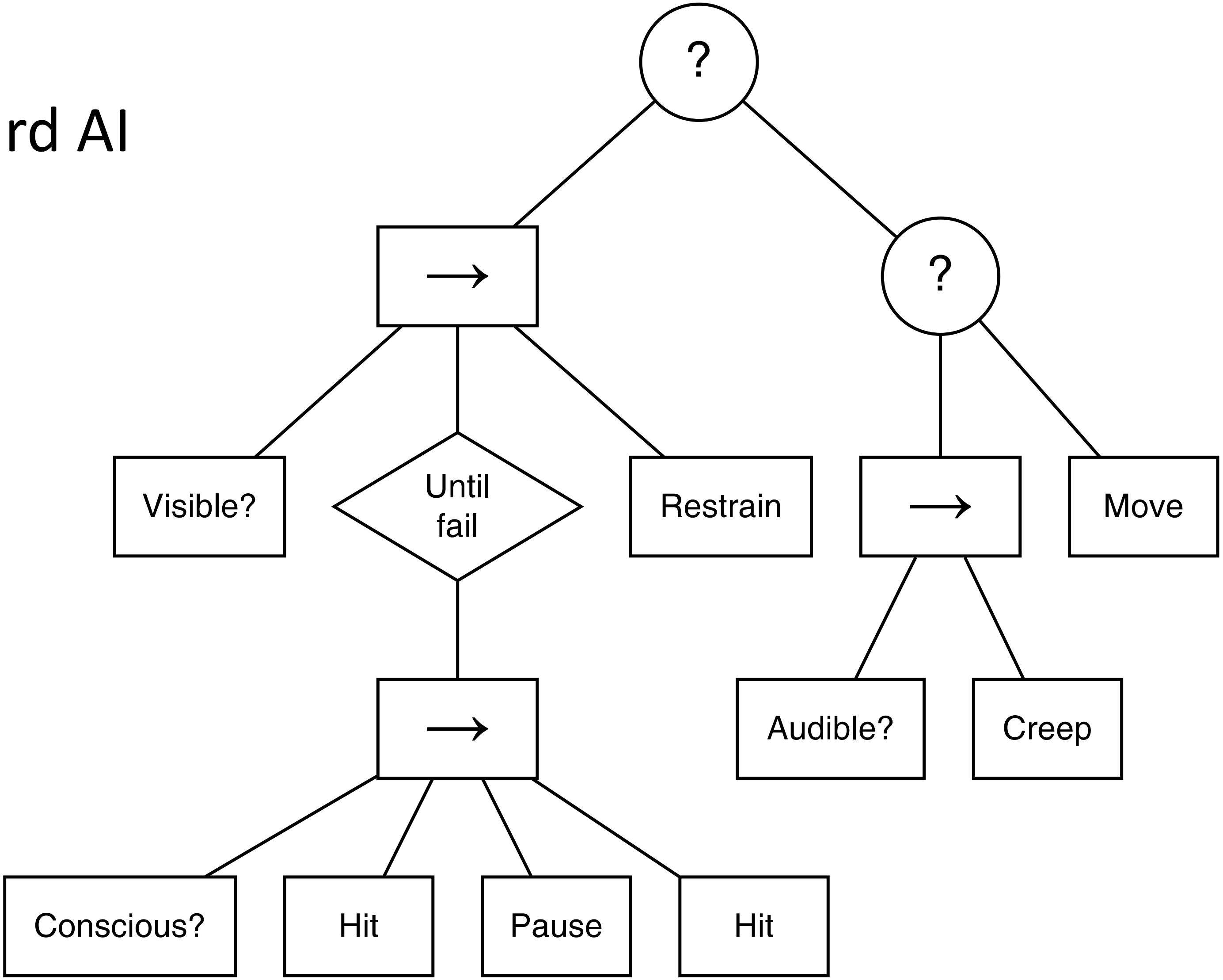
Decorators Example (1)

Dressing up



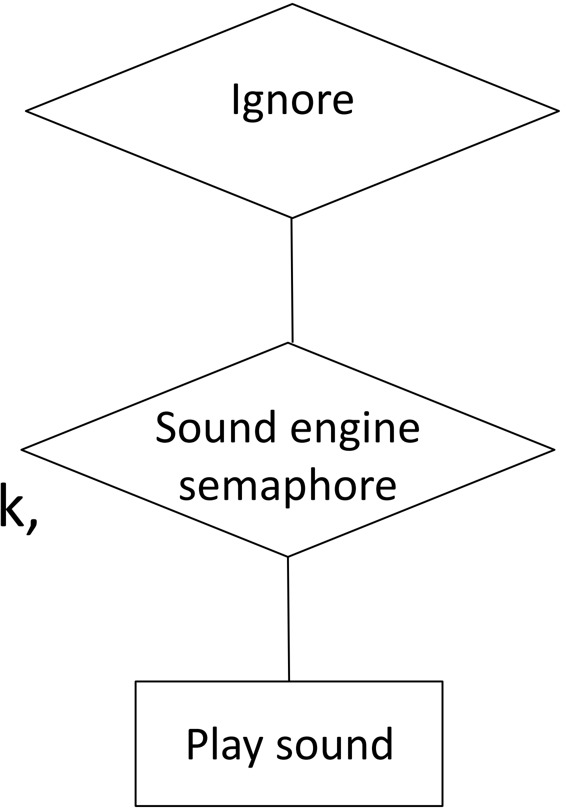
Decorators Example (2)

Guard AI



Guarding Resources with Decorators

- Semaphore decorator
  - Every instance refers to the same flag
  - Whenever an AI entity tries to access resource, checks for the flag
    - If available, set the flag, run the task, unset the flag



Implementation

```
public class Task {
   Boolean run()
}

public class Composite extends Task {
   Composite (Vector<Task> subtasks)
}
```

…

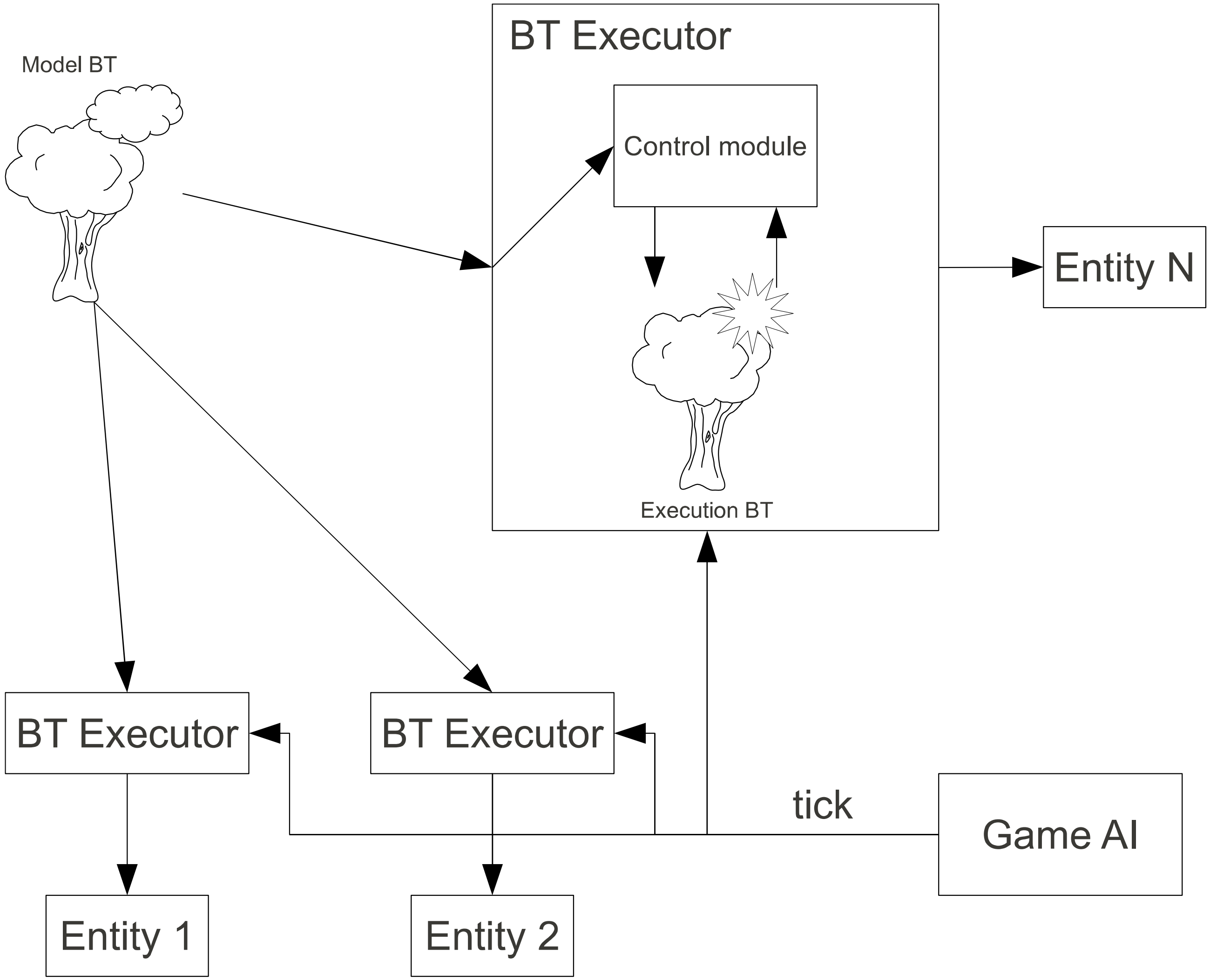
Quite straightforward but…

BTs and Multitasking

- So far we did not consider multitasking
  - Decision trees execute fast
  - FSMs state determines what to do

- In behaviour trees, tasks may span over time
  - Either use multithreading
    - Every tree is being run by a thread
  - Or use *scheduling*

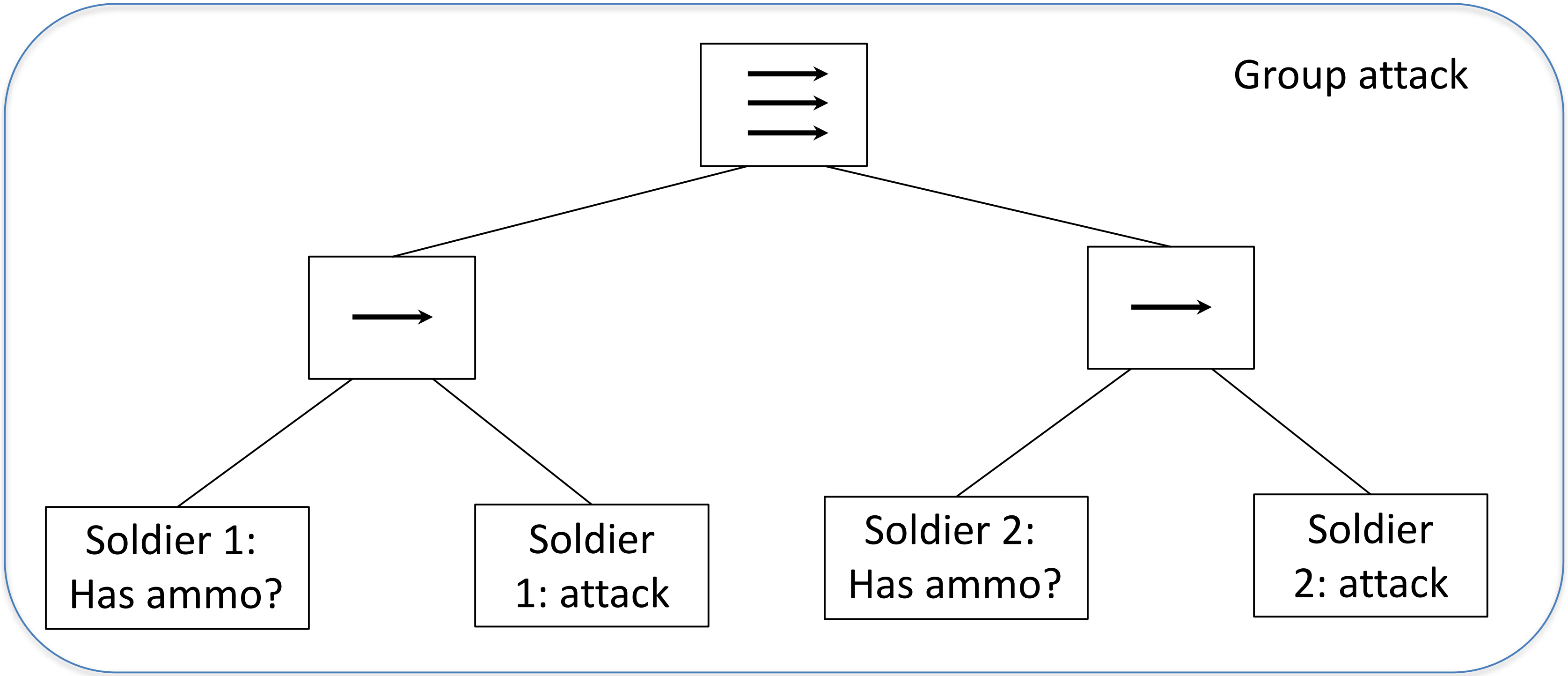
Tick-based model



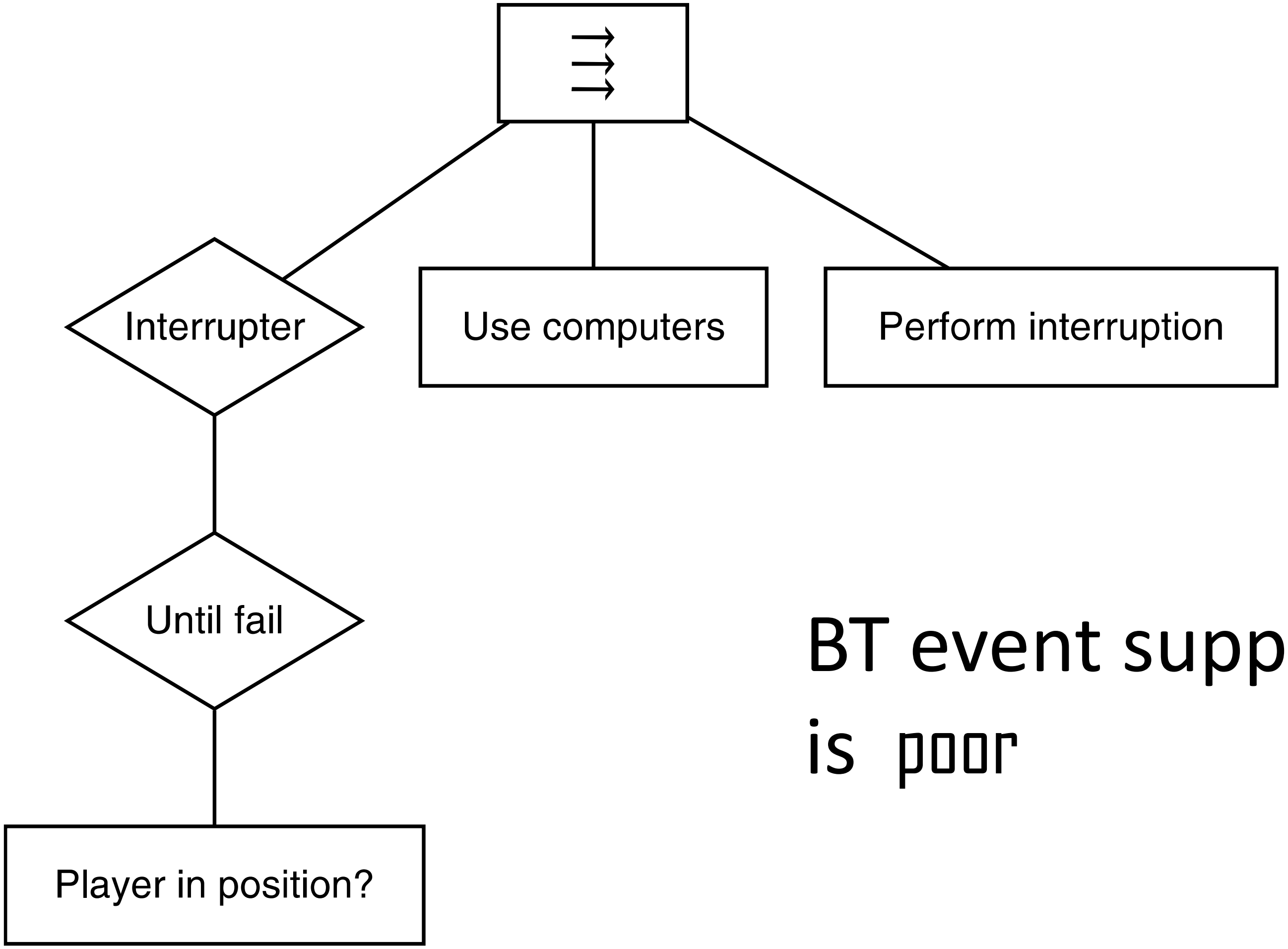
- Tick-based model from http://jbt.sourceforge.net/

Parallel Composites

- In presence of multitasking, one can run tasks in parallel
  - E.g. for group behaviours



Event Handling



BT event support is poor

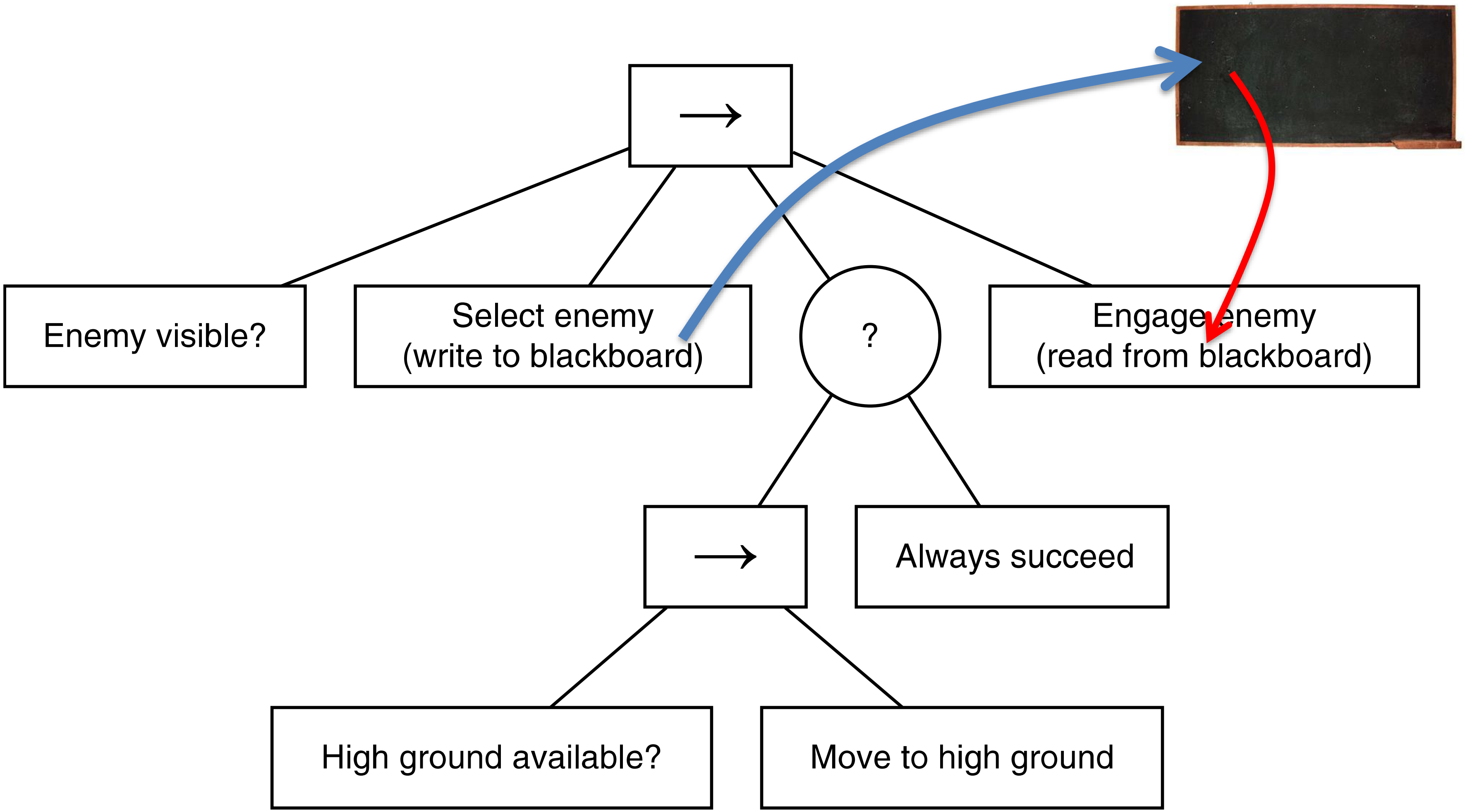
Data in BTs

- One of strong points of BT model is that all tasks have same interface

- Tasks cannot take parameters as input

- Use blackboard AKA notice board for communication (see your COMP213 notes)

Blackboard for Inter-Task Communication



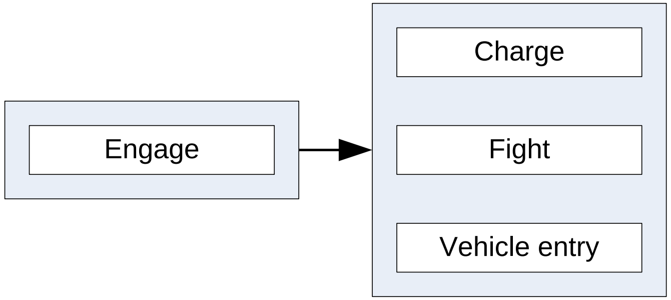
Extensions

- Priority of sub tasks for composites
  - Dynamic priority
    - Low health -> “take cover” gets higher priority
  - kicking out of lower priority behaviour
- Probabilistic
- One-off tasks (random choice but do not repeat)

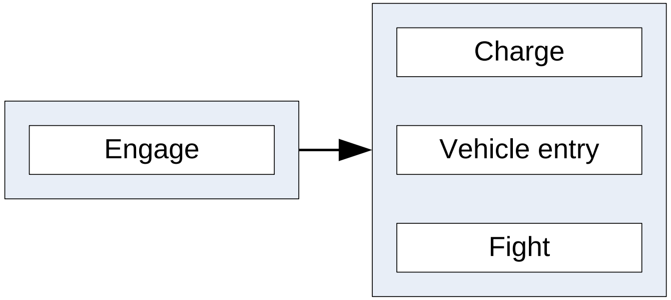
- Interrupting tasks

Halo 2: Impulses (1)

Problem: What happens (with a prioritized list) when the priority is not constant?



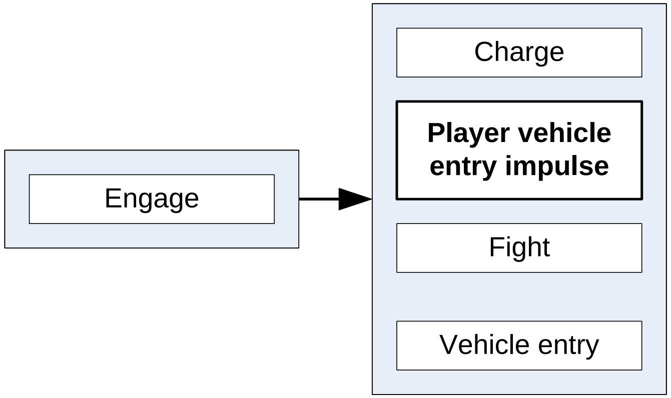
Unless the *player* is in vehicle, in which case...



From Demián Isla's GDC'05 presentation

Halo 2: Impulses (2)

Solution: Separate alternative trigger conditions out into separate **impulse**



Engage

Charge

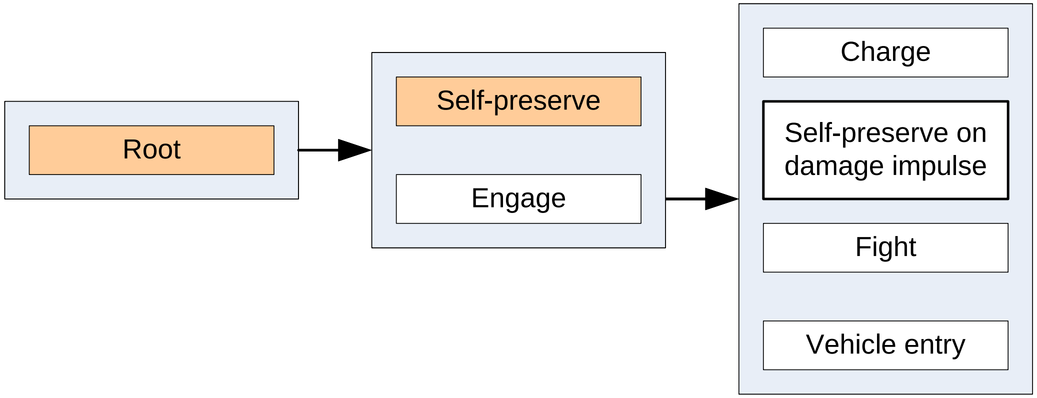
Player vehicle entry impulse

Fight

Vehicle entry

Two execution options

- In-place
- Redirect



Root

Self-preserve

Engage

Charge

Self-preserve on damage impulse

Fight

Vehicle entry

From Demián Isla's GDC'05 presentation

Behaviour Trees: Summary

- Advantages
  - Easy to understand
  - Builds on past experience
  - Executable system specification
  - Support parallelism
- Disadvantages:
  - Reactive and state-based behaviour may be awkward to describe