Synthesis of Asynchronous Systems

Christine Rizkallah

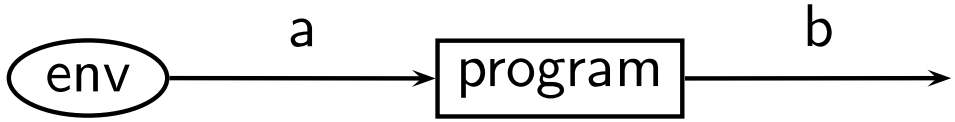
July 3, 2008

Motivation

- Why are asynchronous systems important?
- Natural way to model important problems
- Example:
  - distributed software modules
  - processes running at different speeds

Asynchronisity

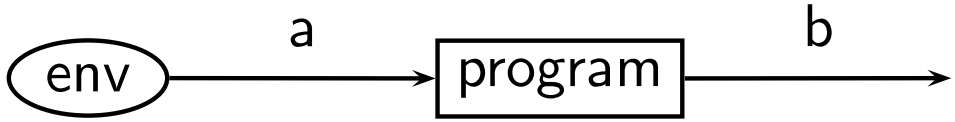
- Programs take inputs from the environment
- Those inputs are invisible to the program until it is scheduled
- Output of the program may vary according to input



- $(\neg a, \neg b, \neg s)$, $(a, \neg b, \neg s)$, (a, b, s), $(a, b, \neg s)$ ...

Asynchronicity

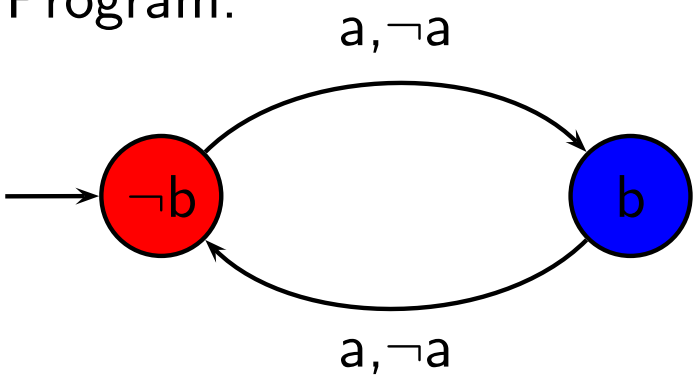
- Programs take inputs from the environment
- Those inputs are invisible to the program until it is scheduled
- Output of the program may vary according to input



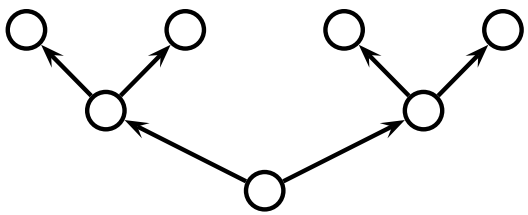
- $(\neg a, \neg b, \neg s)$, $(a, \neg b, \neg s)$, (a, b, s), $(a, b, \neg s)$ ...

Toggling Example

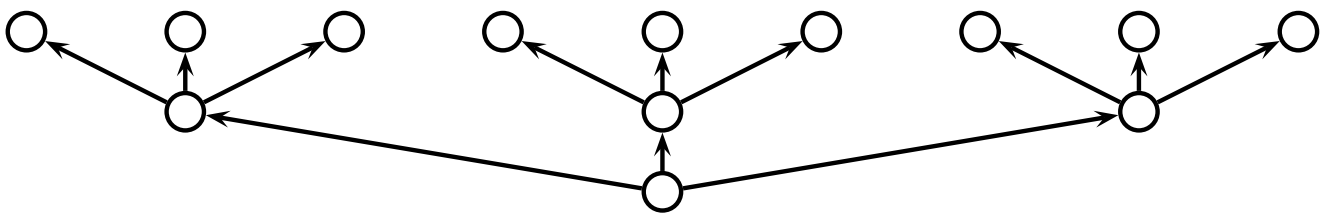
Program:



Behaviour Tree:

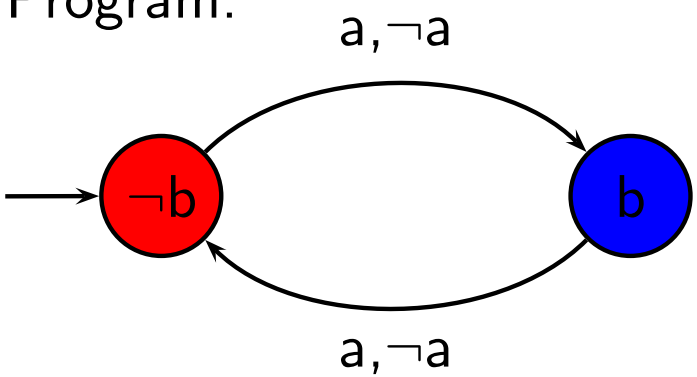


Computation Tree:

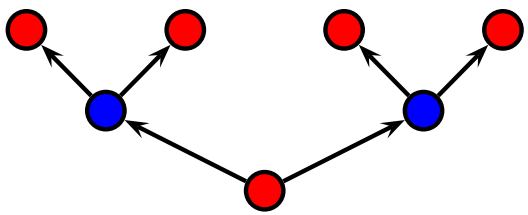


Toggling Example

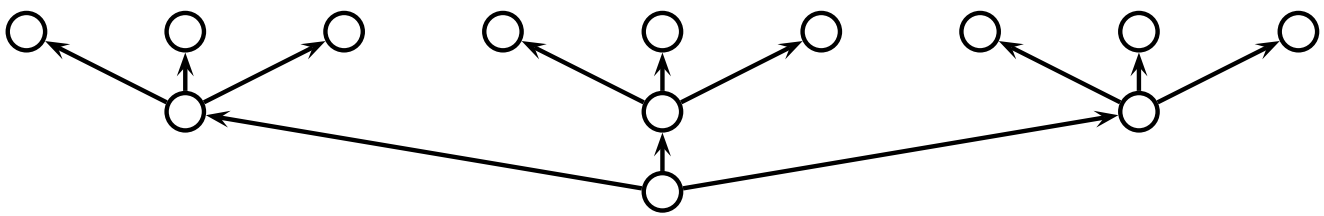
Program:



Behaviour Tree:

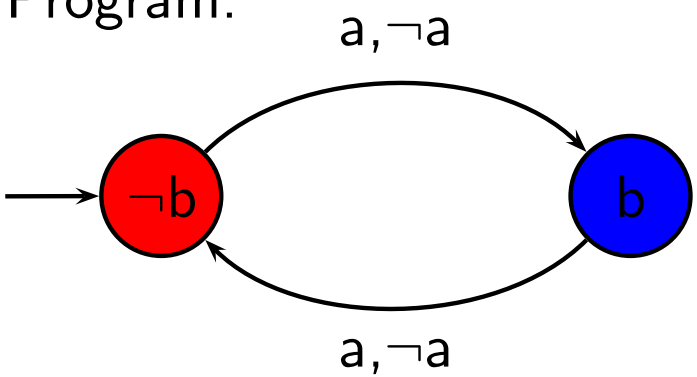


Computation Tree:

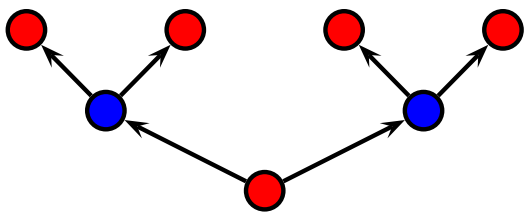


Toggling Example

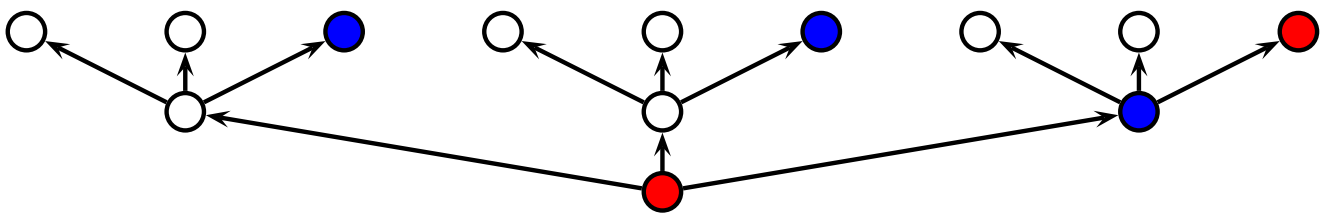
Program:



Behaviour Tree:

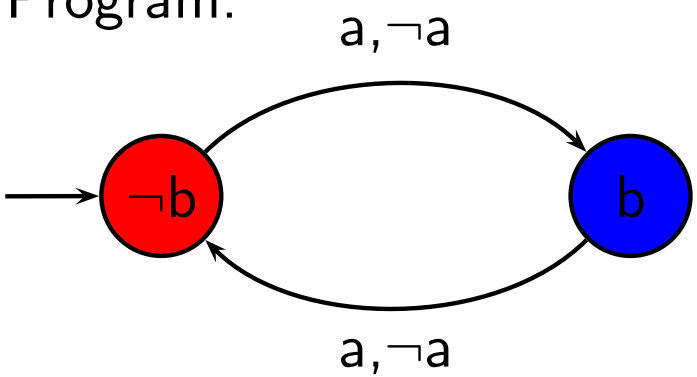


Computation Tree:

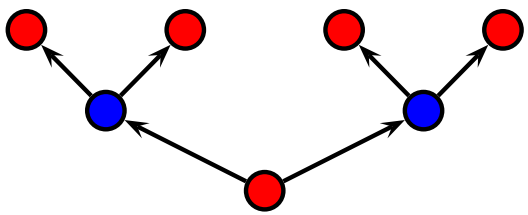


Toggling Example

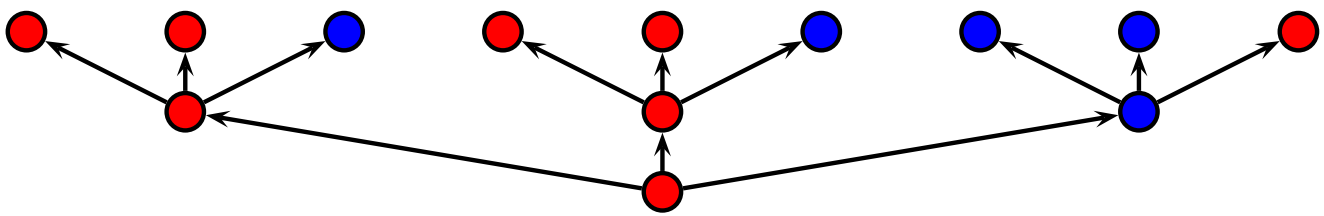
Program:



Behaviour Tree:

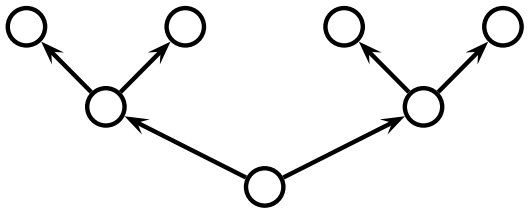


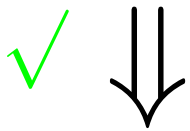
Computation Tree:



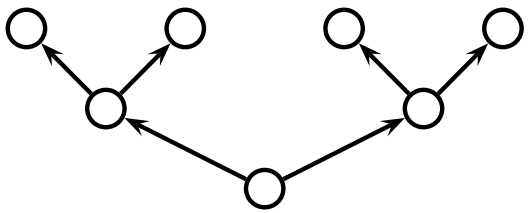
Synchronous Setting

Behaviour Tree:



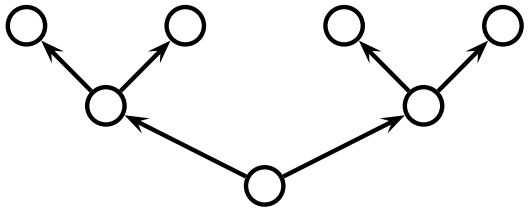


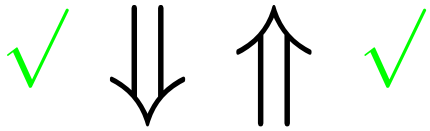
Computation Tree:



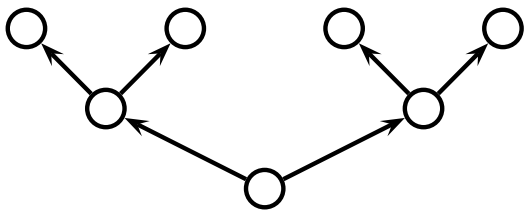
Synchronous Setting

Behaviour Tree:





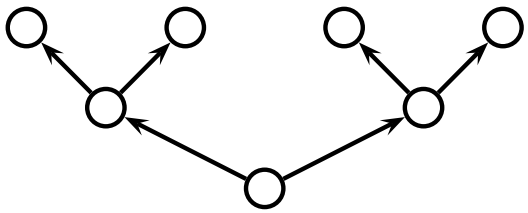
Computation Tree:

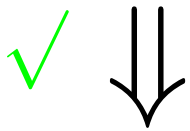


Computation tree and behaviour tree are essentially the same

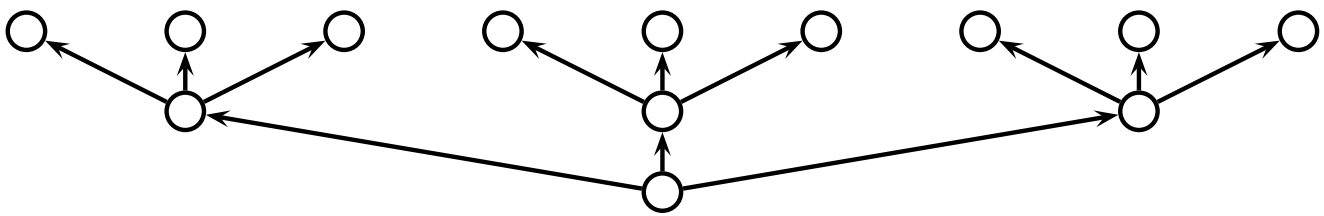
Behavour Tree to Computation Tree

Behaviour Tree:



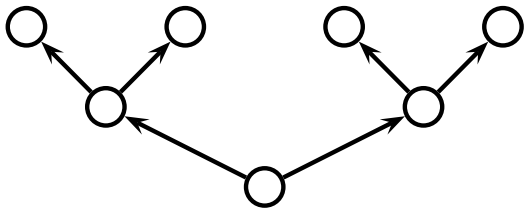


Computation Tree:



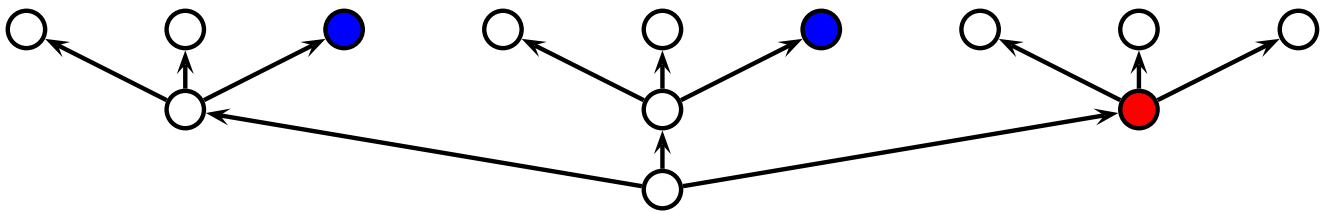
Computation Tree to Behaviour Tree?

Behaviour Tree:



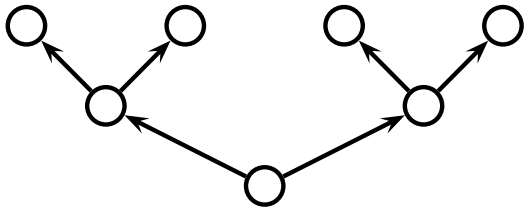
√ ⇓

Computation Tree:



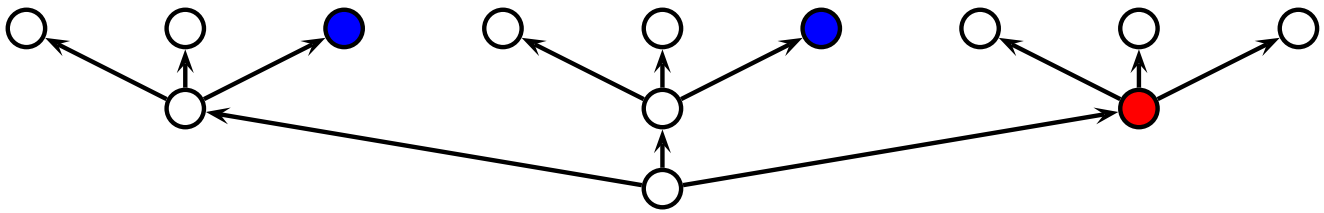
Computation Tree to Behaviour Tree?

Behaviour Tree:



✓ ⇓

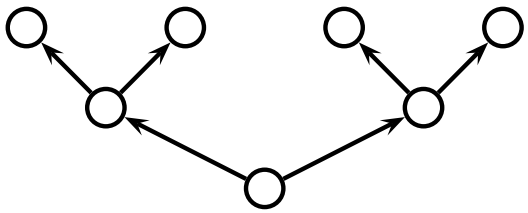
Computation Tree:



No corresponding behaviour tree

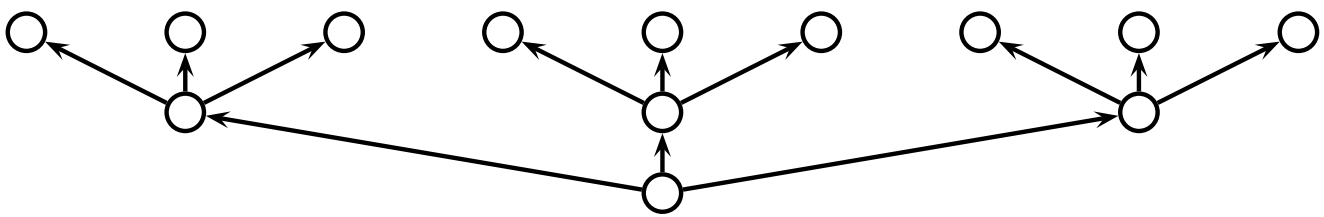
Computation Tree to Behaviour Tree? "No"

Behaviour Tree:



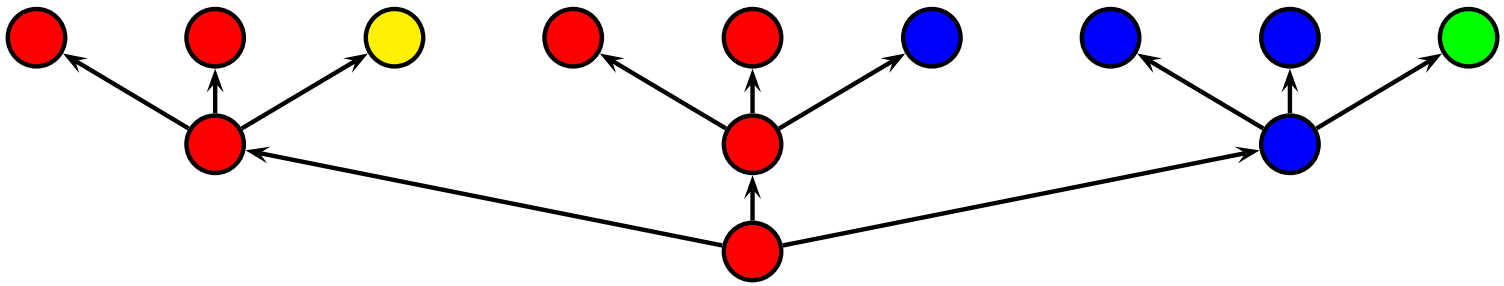
✓ ⇓ ⇑ ×

Computation Tree:

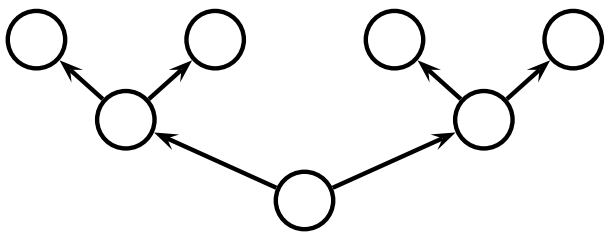


Another Example

Computation Tree:

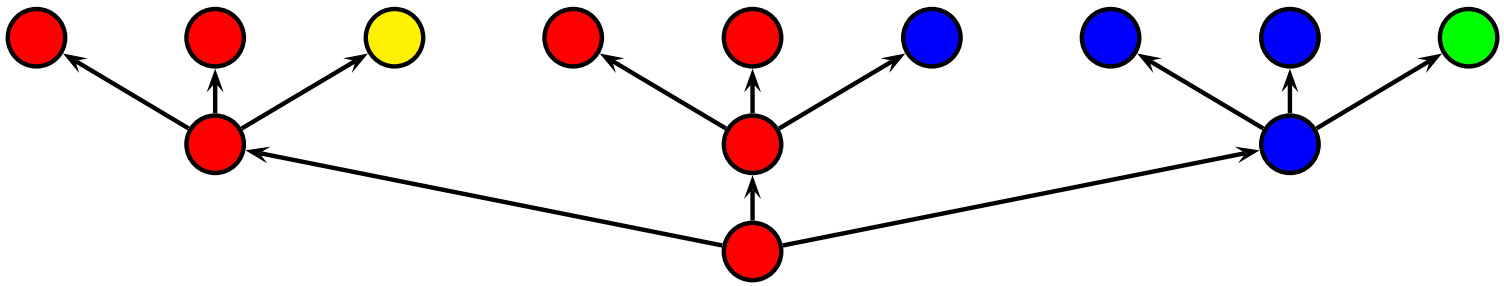


Behaviour Tree:

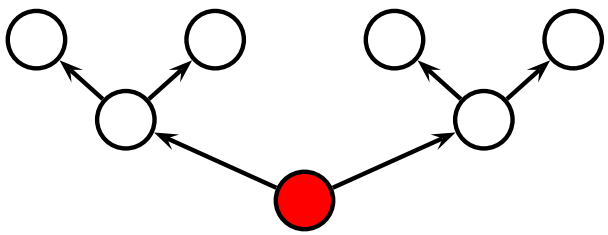


Another Example

Computation Tree:

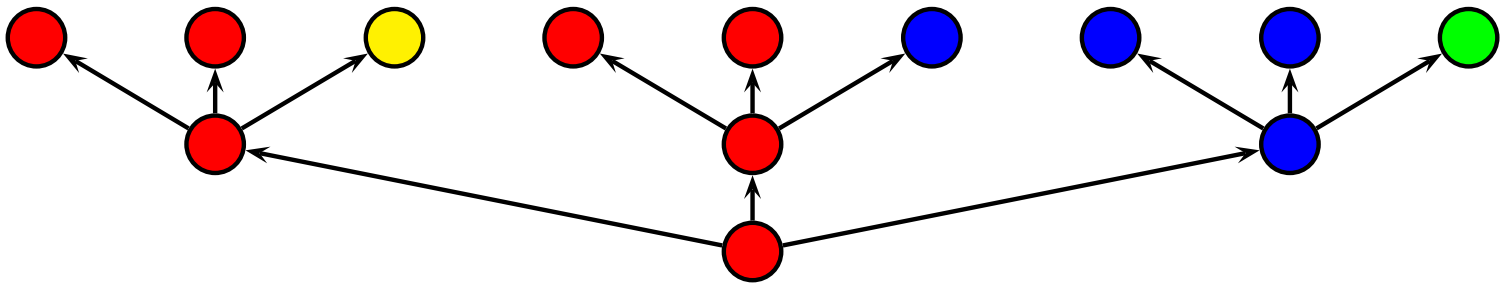


Behaviour Tree:

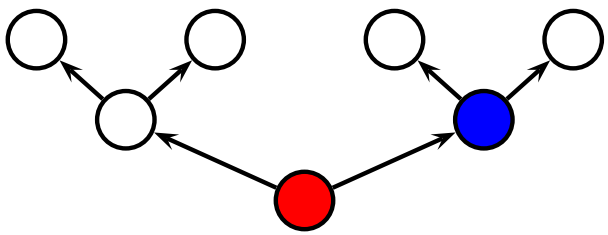


Another Example

Computation Tree:

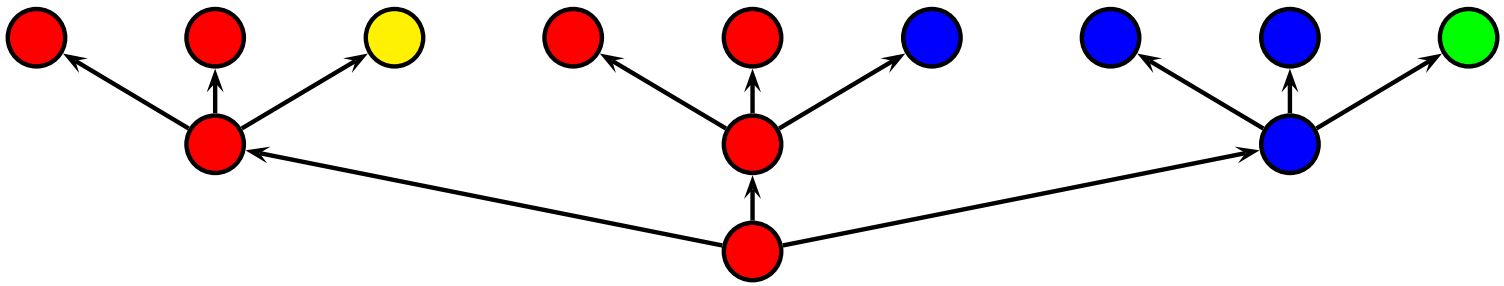


Behaviour Tree:

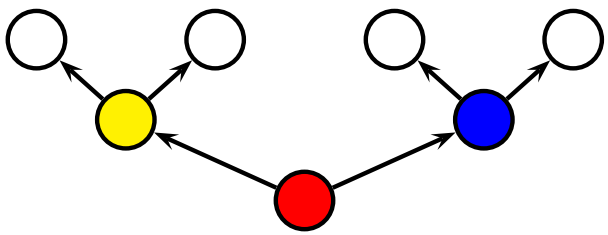


Another Example

Computation Tree:

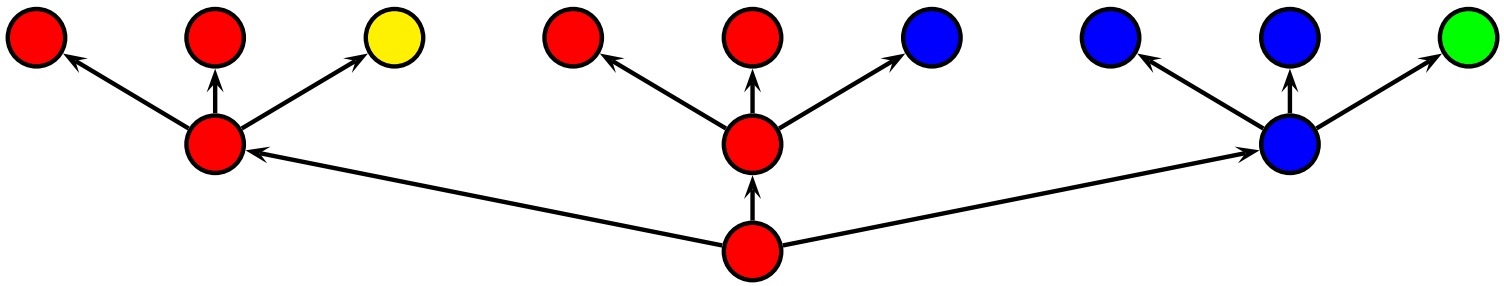


Behaviour Tree:

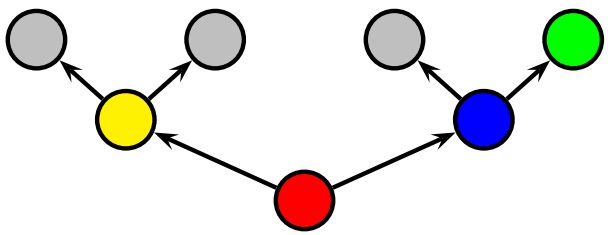


Another Example

Computation Tree:



Behaviour Tree:



Deterministic Safety Tree Automata (DSTA)

- Acceptance game on CT
- Acceptance game on BT
- Automata Transformation
- Extensions

Automata Transformation

- $A_{CT} = (2^I x 2^S x 2^O, Q^{dir}, q_0, \delta)$
- δ is a partial function
- $\delta(q, i, s, o) = (q_1, false, env) \wedge (q_2, true, env) \wedge (q_3, i, prog)$

- $A_{BT} = (Q x 2^I x 2^S x 2^O, dir, q_0, \delta')$
- δ' is a partial function
- $\delta'(q, i, s, o) =$
  $(q_1, false, env, \varepsilon) \wedge (q_2, true, env, \varepsilon) \wedge (q_3, i, prog, i)$

Getting Rid of ε Transitions

- Fixed Point Expansion
  - $\delta'(q, i, s, o) =$
    $(q_1, \mathit{false}, \mathit{env}, o) \wedge (q_2, \mathit{true}, \mathit{env}, o) \wedge (q_3, i, \mathit{prog}, i)$

Overview

- DSTA $A^{CT} \Rightarrow$ UεSTA $A^{BT} \Rightarrow$ USTA $A^{BT} \Rightarrow$ NET $\Rightarrow$ Prog

Extensions

- Universal Safety Tree Automata
  - we get it for free
- Universal Co-Buchi Automata
  - search for ε cycles on rejecting states
  - search for chains with infinitely many rejecting states
- LTL formulas could be represented as Universal Co-Buchi Automata

Thanks For Listening

Questions?