Model driven Robot behavior

FALCONS

supported by ASML

Separate robot behavior from SW code

June 2016

World Championship Leipzig

FALCONS

supported by ASML

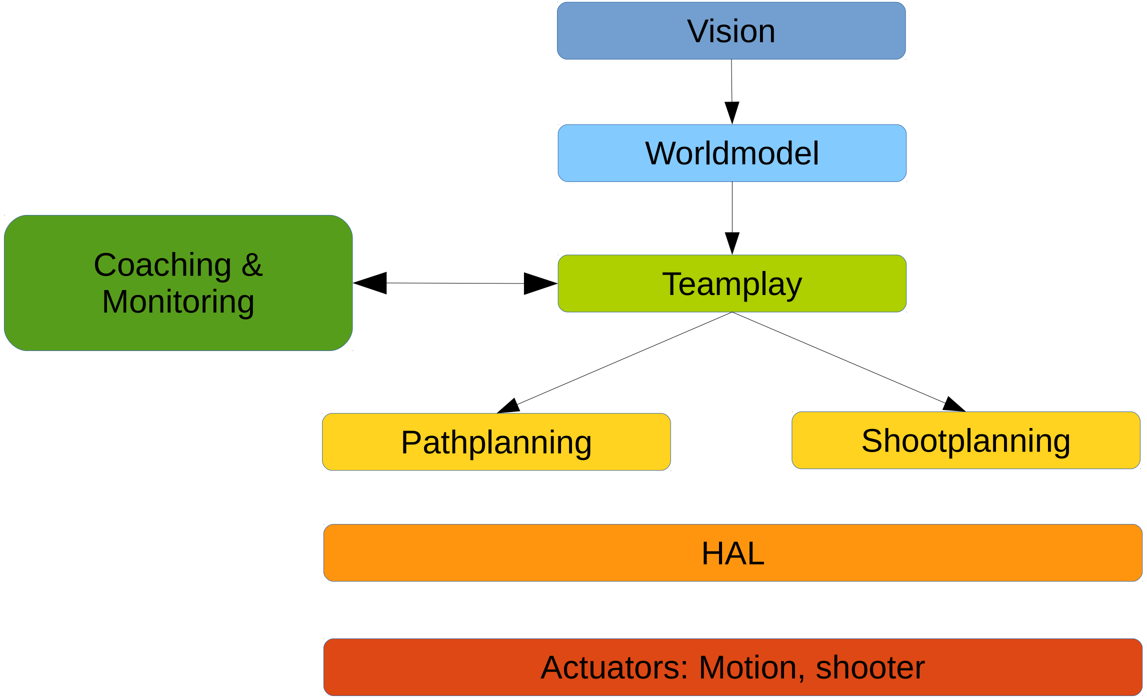
Contents

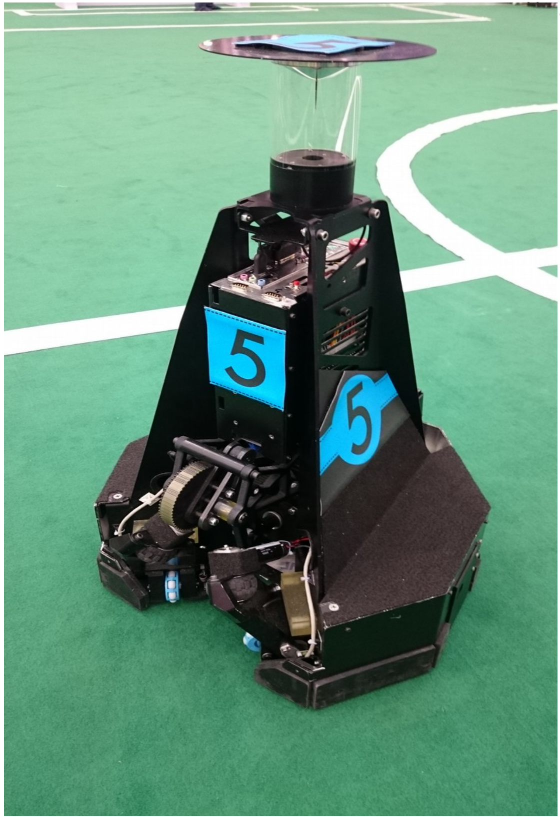
- Introduction
  - Falcons SW architecture
  - Teamplay functionality

- Separating logic from data

- Questions

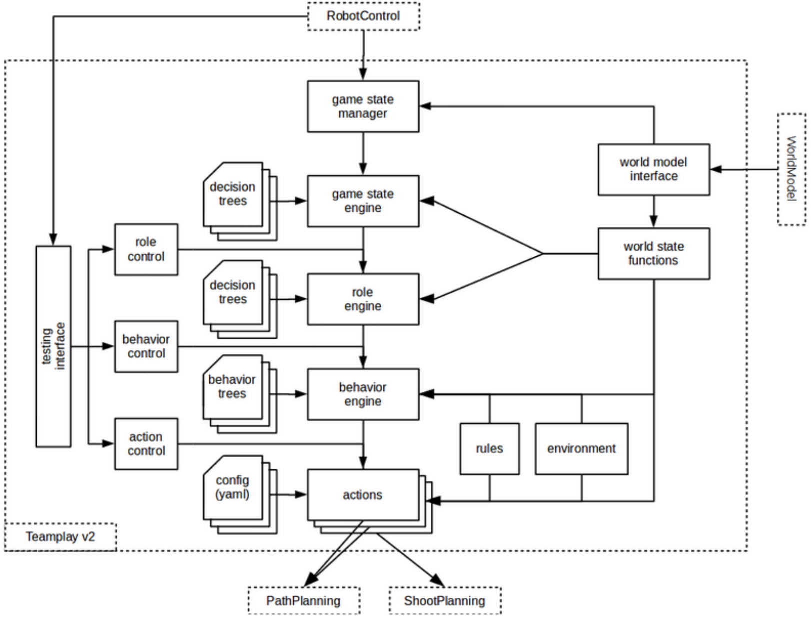
Falcons Software Architecture





Teamplay: functionality and architecture

- Based on synchronized worldmodel determine which action to take
  - Inputs from Worldmodel:
    - Where is the ball
    - Where are my teammates etc.
  - Actions: Pass, shoot, move etc.
  - Game states, roles, behaviors, actions



Autonomous robots: Smart behavior makes the difference

- Smart autonomous behavior key differentiator in MSL
- To allow rapid prototyping and enable non-core software engineers to improve autonomous behavior, separate behavior (data) from code (logic):
    - “Behavior trees” to determine robot's activity
    - Graphical editor to create and update behavior trees
    - Framework to read in and execute behavior trees
        - One time programming effort
        - Load data at SW start → no need to re-build/deploy the software when behavior is updated

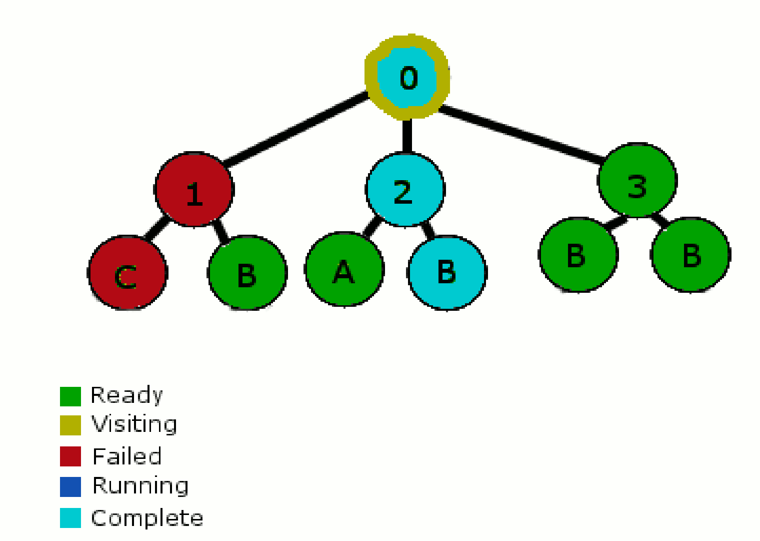
Logic: Behavior trees

Trees consist of nodes and leafs

- Nodes are decision points: “Do I have the ball, yes or no”
- Leafs are actions: shoot/move etc.
- Flow control: sequence with/without memory etc.

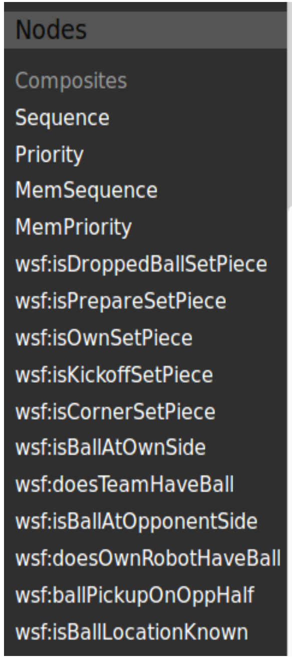
Role determines a behavior; evaluate behavior tree:

- Evaluate each child (from left to right)
- When leaf is reached, action is started
- Each leaf returns passed/running/failed
    - Running: return to leaf in next iteration
    - Failed: Return to parent, evaluate next child
    - Passed: Parent is completed and returns



Editing / creating behavior: graphical editor

- Easy to use; once decision points (nodes) are available, creating complex behavior is quick and painless



Nodes

Composites

Sequence

Priority

MemSequence

MemPriority

wsf:isDroppedBallSetPiece

wsf:isPrepareSetPiece

wsf:isOwnSetPiece

wsf:isKickoffSetPiece

wsf:isCornerSetPiece

wsf:isBallAtOwnSide

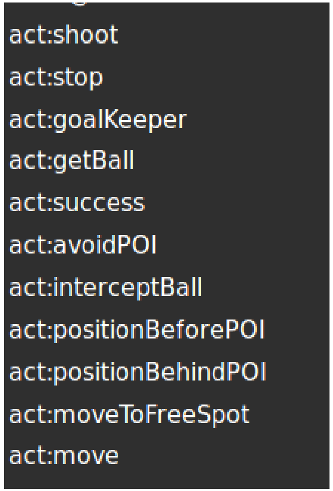
wsf:doesTeamHaveBall

wsf:isBallAtOpponentSide

wsf:doesOwnRobotHaveBall

wsf:ballPickupOnOppHalf

wsf:isBallLocationKnown



act:shoot

act:stop

act:goalKeeper

act:getBall

act:success

act:avoidPOI

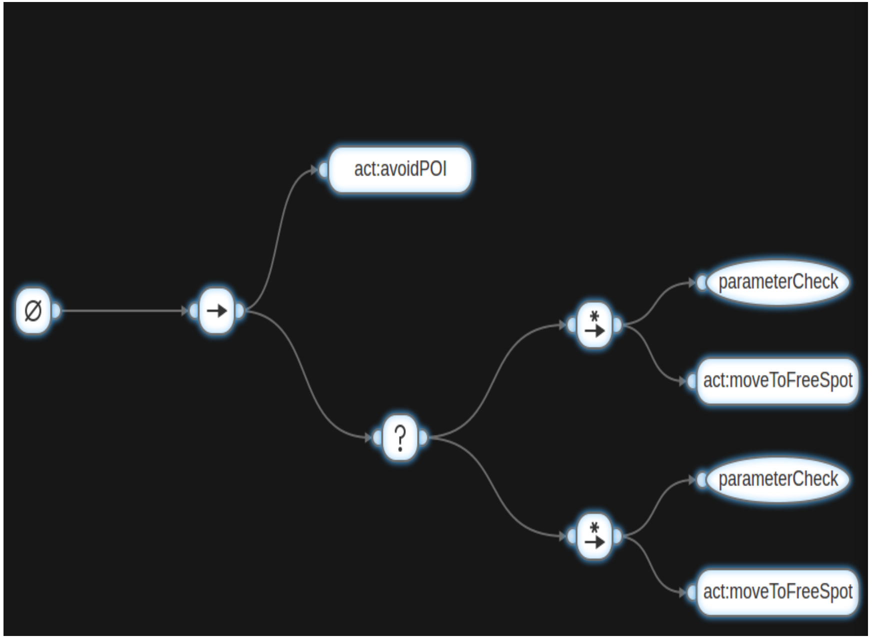
act:interceptBall

act:positionBeforePOI

act:positionBehindPOI

act:moveToFreeSpot

act:move



Questions

Feel free to drop by our team corner!

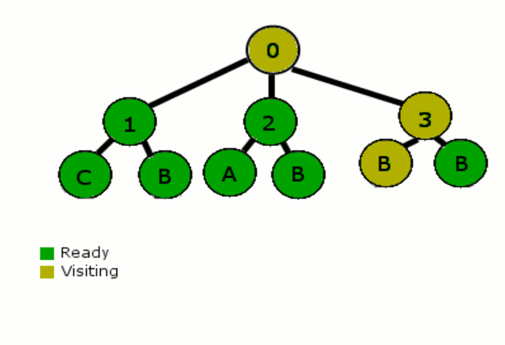
Behavior trees versus Decision trees

Decision trees:

- Always evaluated from root to leaf

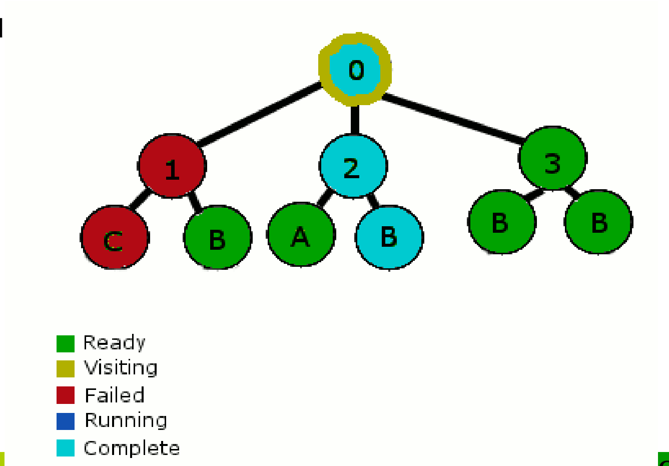
- Traverse down until a leaf is reached

- Ideal for yes/no decisions



Behavior trees:

- Evaluate each child (from left to right)
- When leaf is reached, behavior is started
- Each leaf returns passed/running/failed
  - Running: return to leaf in next iteration
  - Failed: Return to parent, evaluate next child
  - Passed: Parent is completed and returns
- Node has memory
- Can create sequence of actions
- Useful for complex behaviors!



FALCONS
supported by ASML