Cognitive Robotics

Behavior Control

Hans-Dieter Burkhard, Marija Brkic Bakaric

October 2014

Overview

Introduction

Control Architectures

Aspects of Rationality

BDI Architectures

Behavior Based Robotics

Behavior Control needs …

… integration of perception, decision/planning, action
on different complexity levels

All parts depend on the others.
Improving one part may result in worse performance.

- Household much more complicated than car driving.
- Soccer much more complicated than chess.

Programming Environments

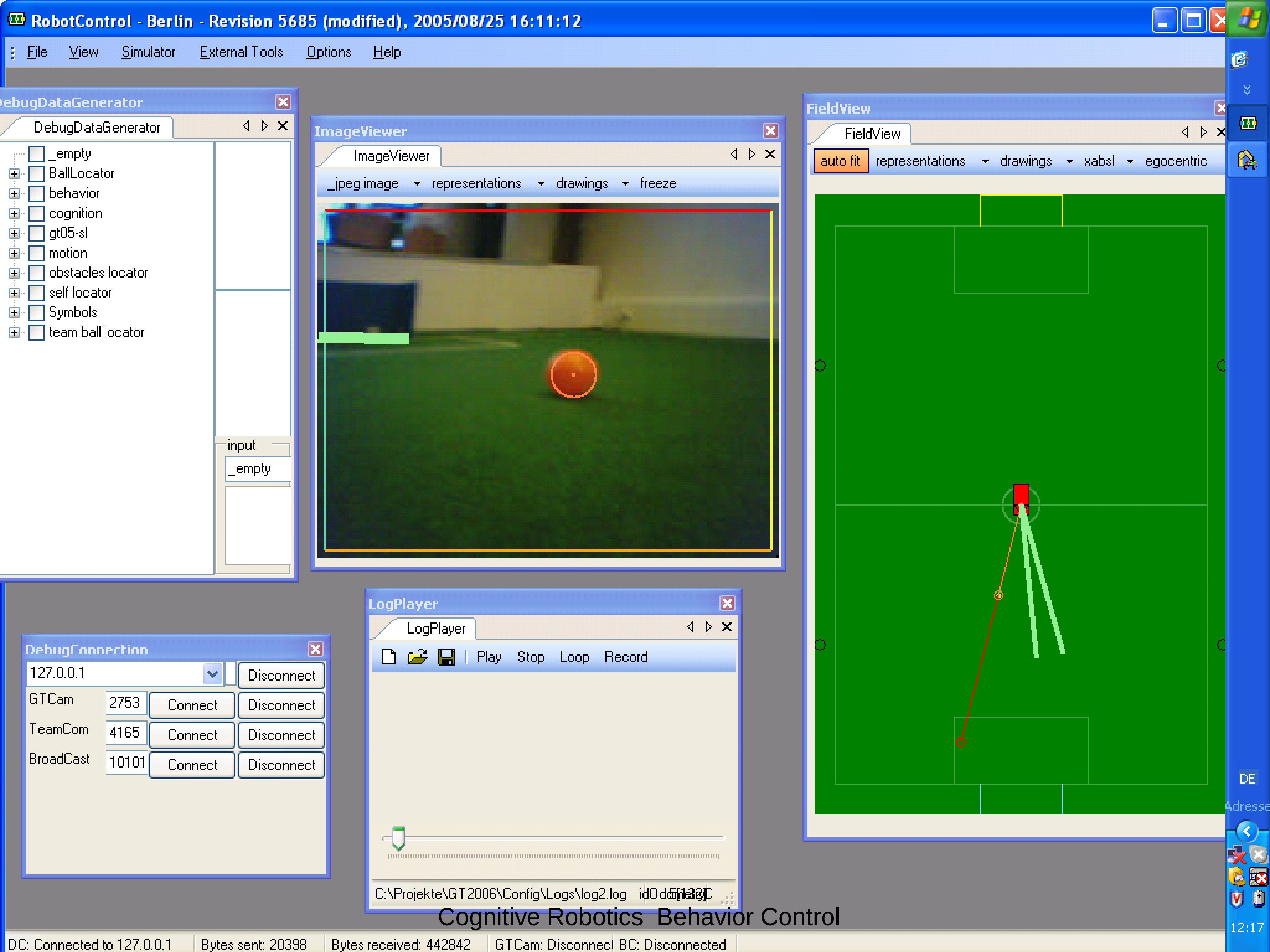
Different tools for

- Development of Programs
- Checking Programs
- Middleware

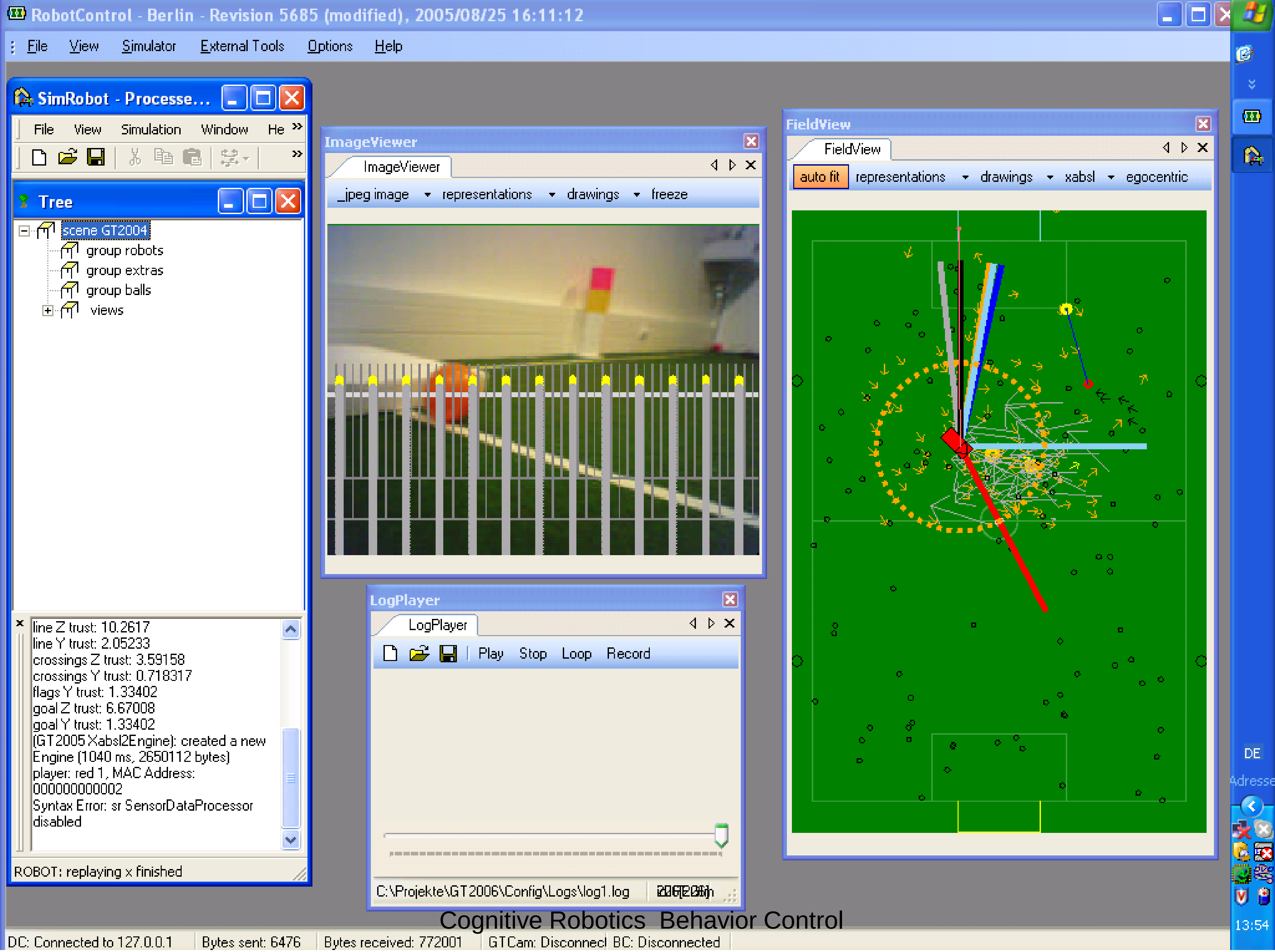
  e.g. ROS (= Robot Operating System) http://wiki.ros.org/

Next Slides:

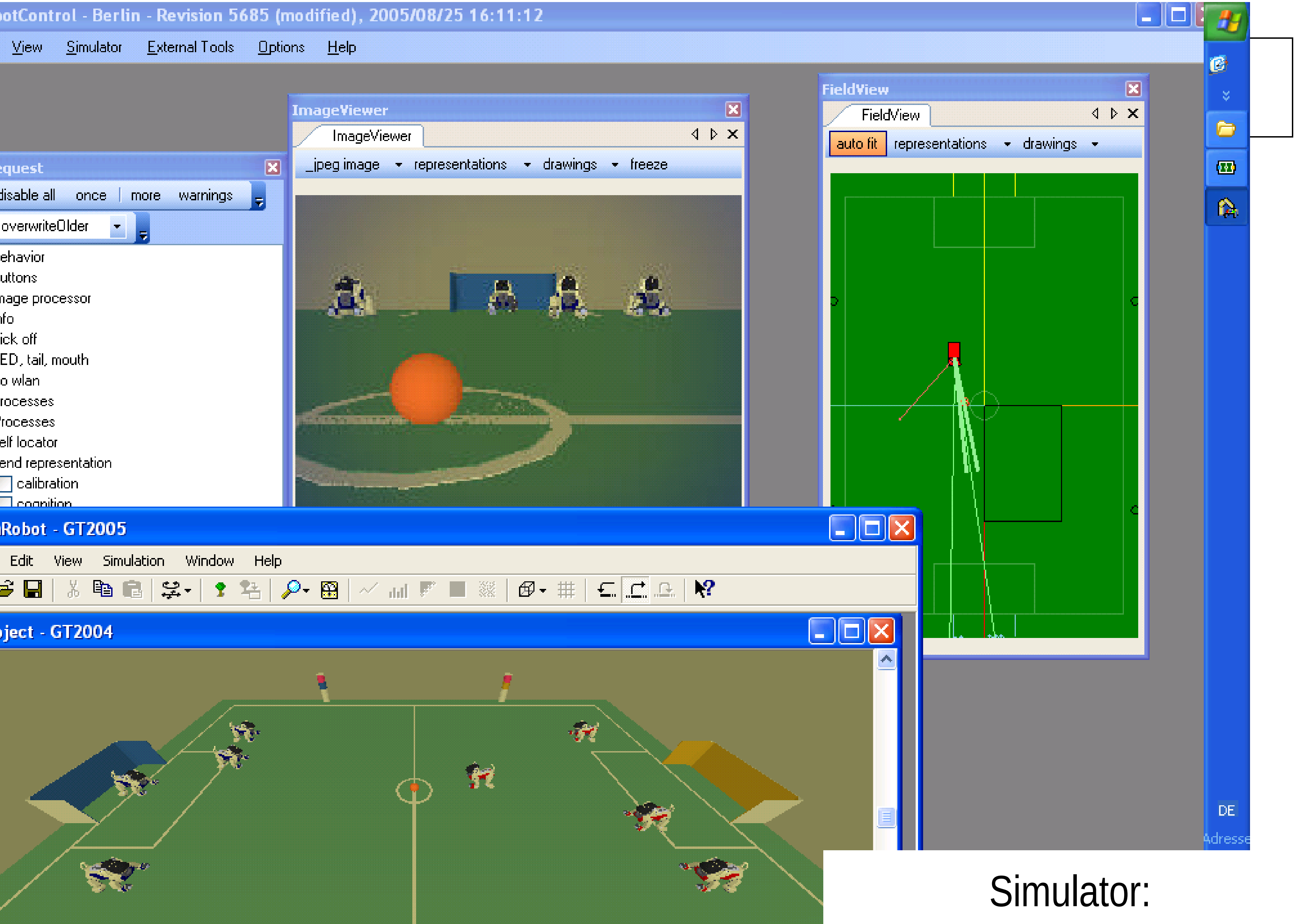
RobotControl developped for GermanTeam (Aibos)



Cognitive Robotics Behavior Control



Cognitive Robotics Behavior Control



Simulator:
Red robot in the middle

Chess like Control for Soccer?

Evaluate options for future success

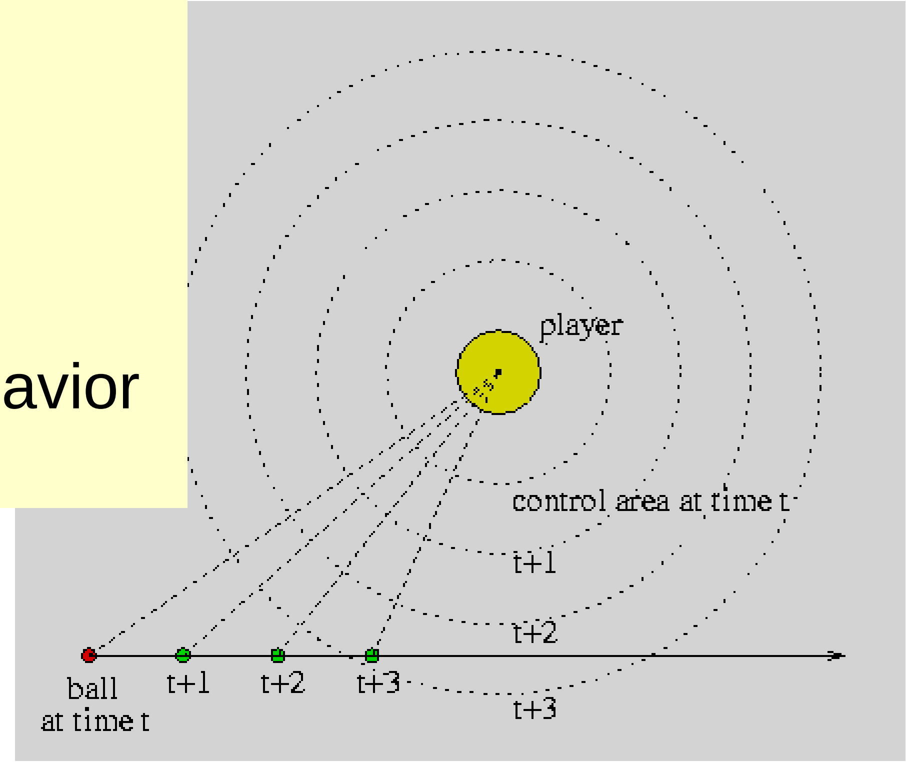
Choose the best alternative

Where to intercept the ball?

By calculation

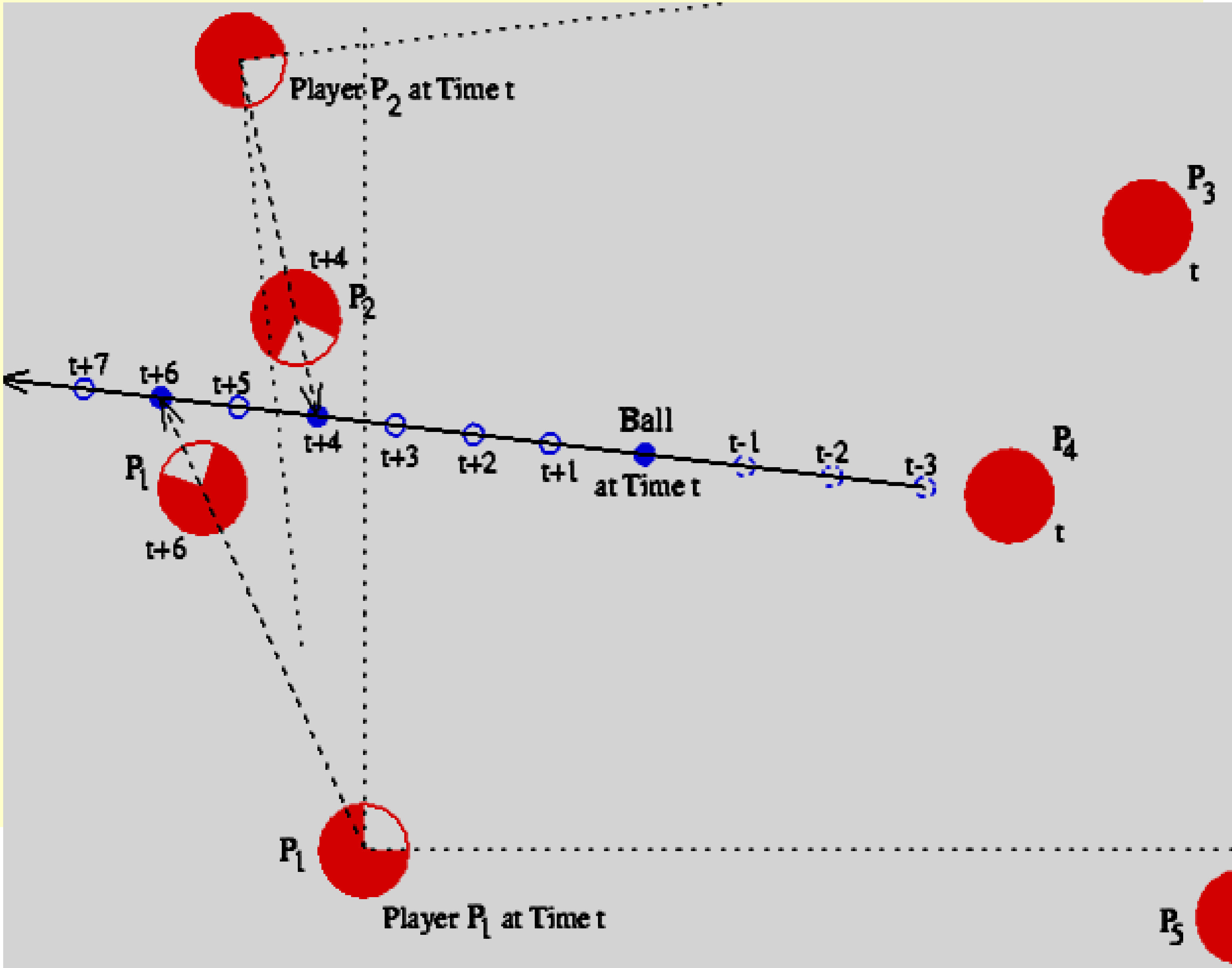
By simulation

By learned behavior



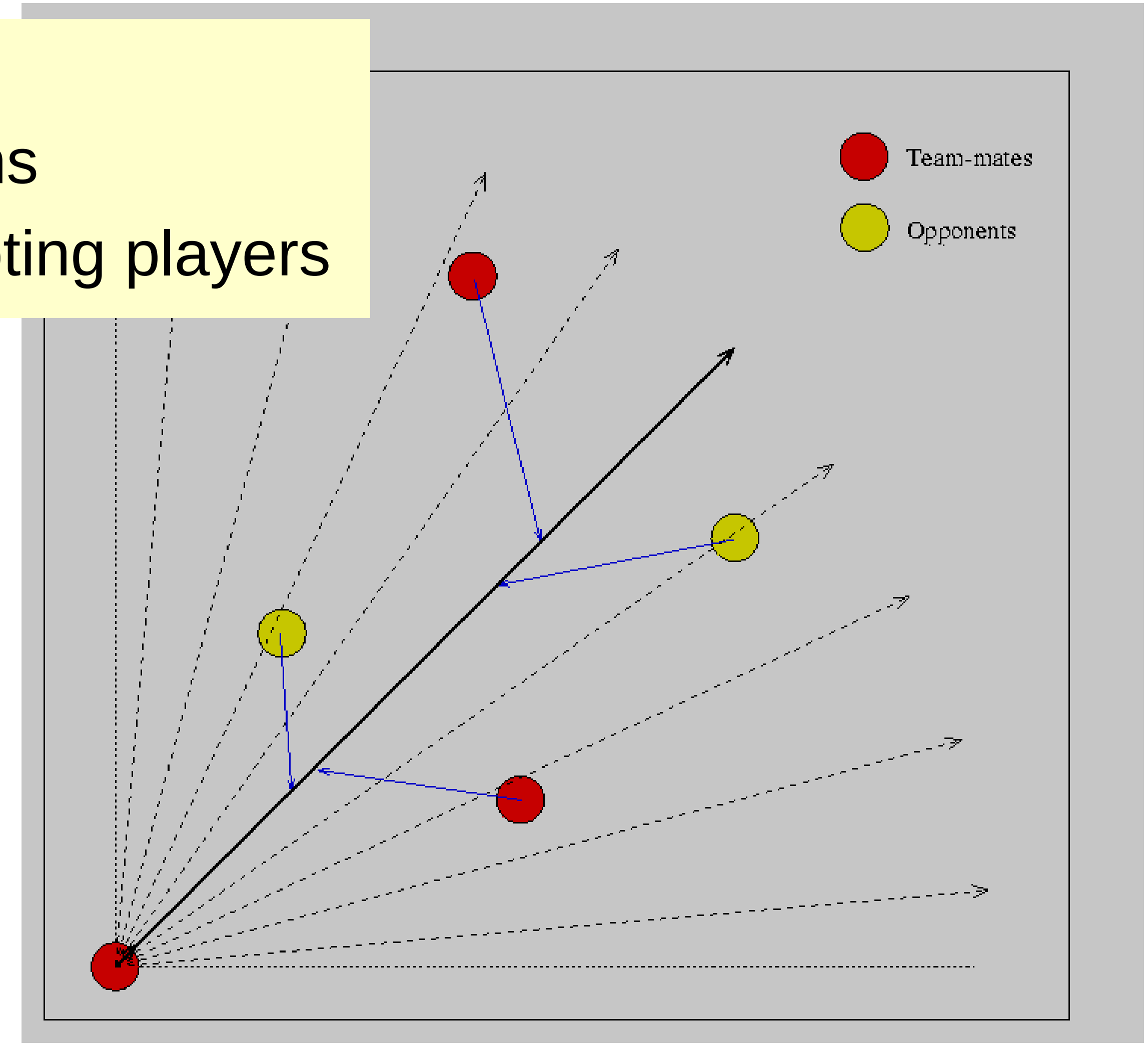
Which player can intercept first?

Based on calculation of intercept



Pass to which team mate?

Based on calculations of intercepting players



Simulation 2D

RoboCup1997 Nagoya Final Match.

AT-Humboldt (Humboldt University of Berlin, Germany) vs andhill (Tokyo Institute of Technology, Japan)

Chess like Control for Soccer?

Evaluate options for future success

Choose the best alternative

Does not work for more
Complicated situations

Overview

Introduction

Control Architectures

Aspects of Rationality

BDI Architectures

Behavior Based Robotics

Classical Types of Agent/Robot Behavior

Reactive Behavior:

like Stimulus-Response: short term

„*simple*“ behavior patterns, simple skills

Deliberative Behavior

Goal directed, plan based behavior: long term

„*complex*“ behavior

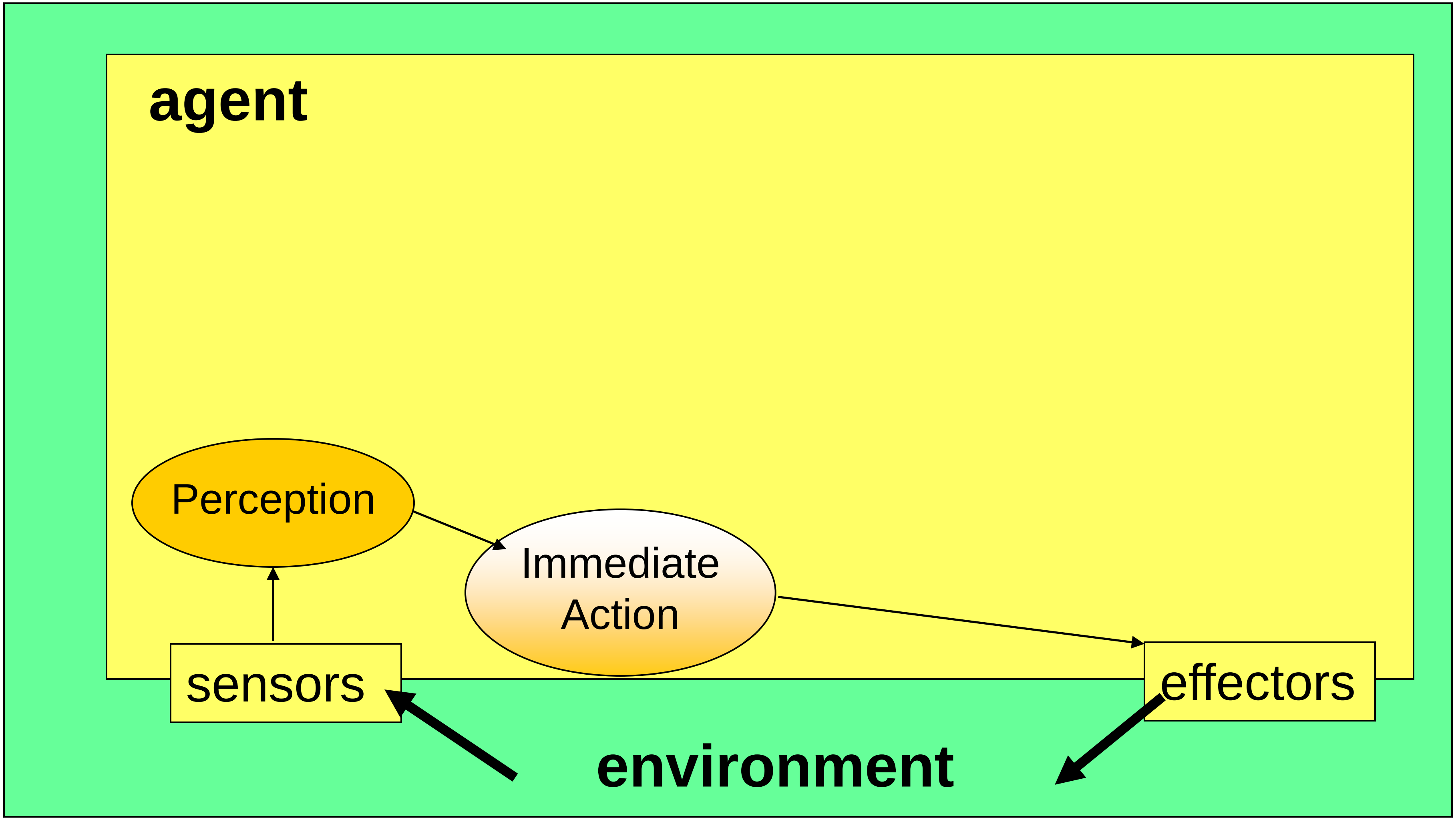
Hybrid:

Combination of reactive and deliberative behavior

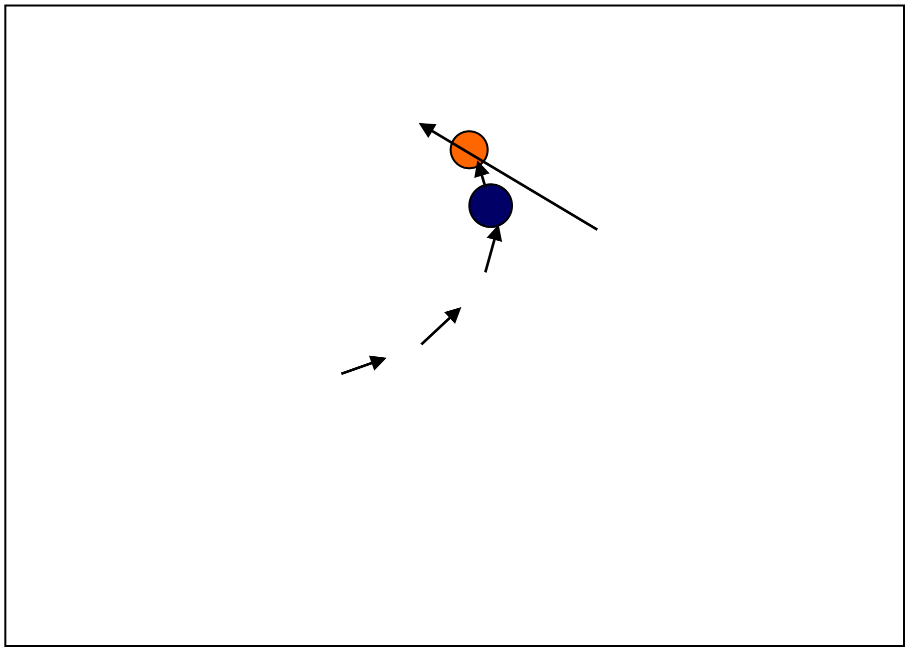
e.g. goal driven usage of reactive skills

In robotics up to now:
More emphasis put to aspects of low level control.
Recently:
Increasing interest in high level control.

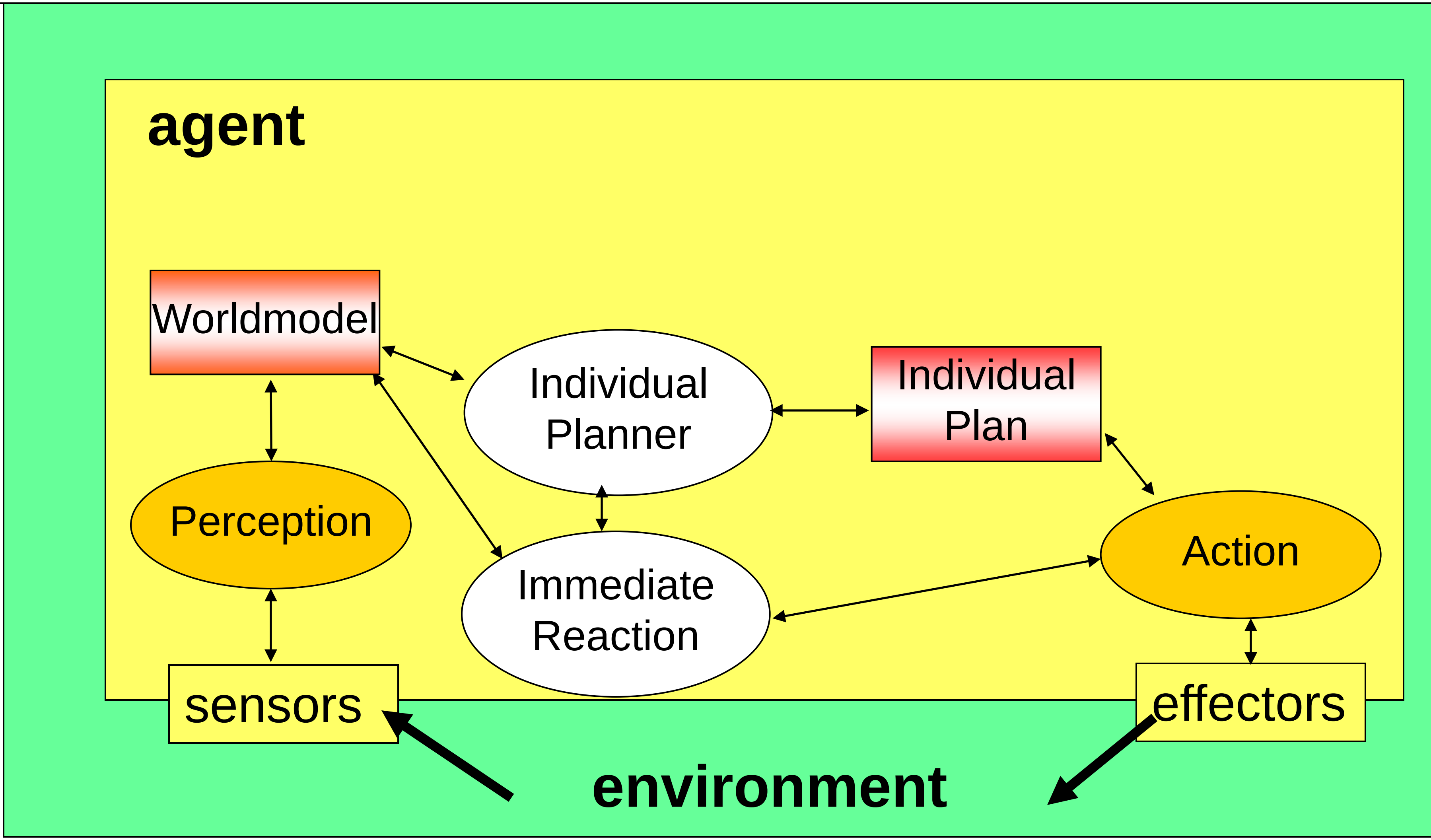
Reactive Behavior



Reactive (“stimulus-response”)

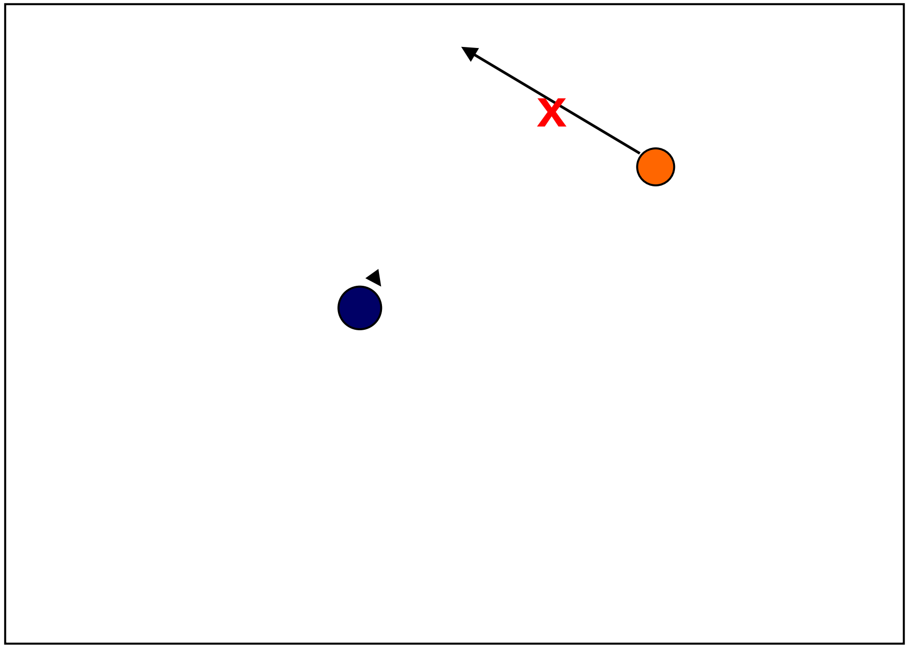


Goal directed behavior



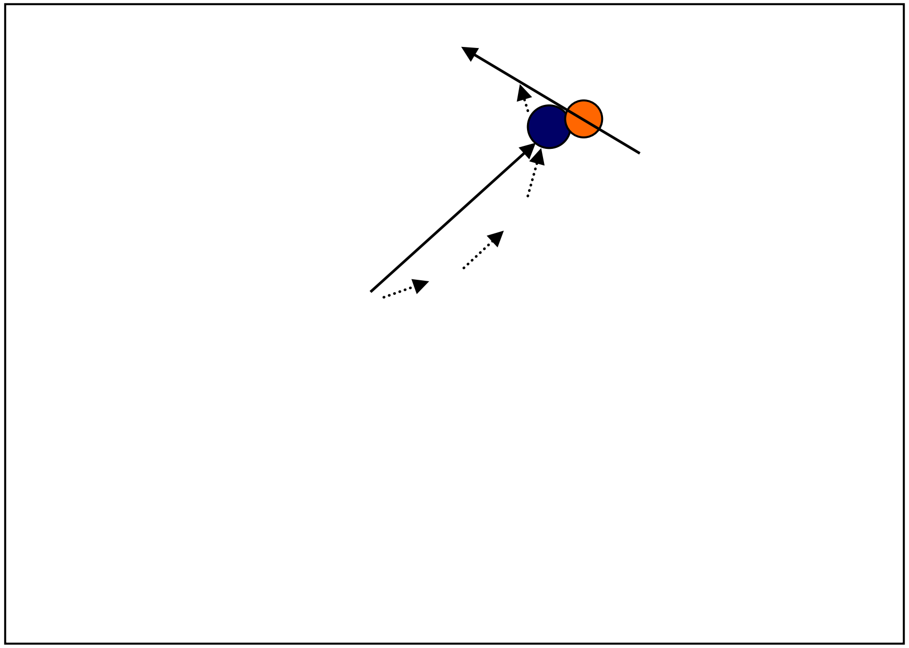
Goal directed behavior

Acting according to a predefined goal

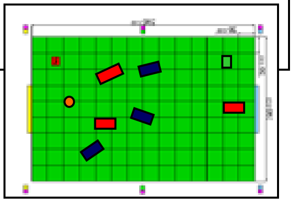


Goal directed behavior

Acting according to a predefined goal

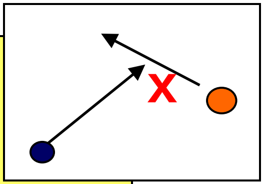


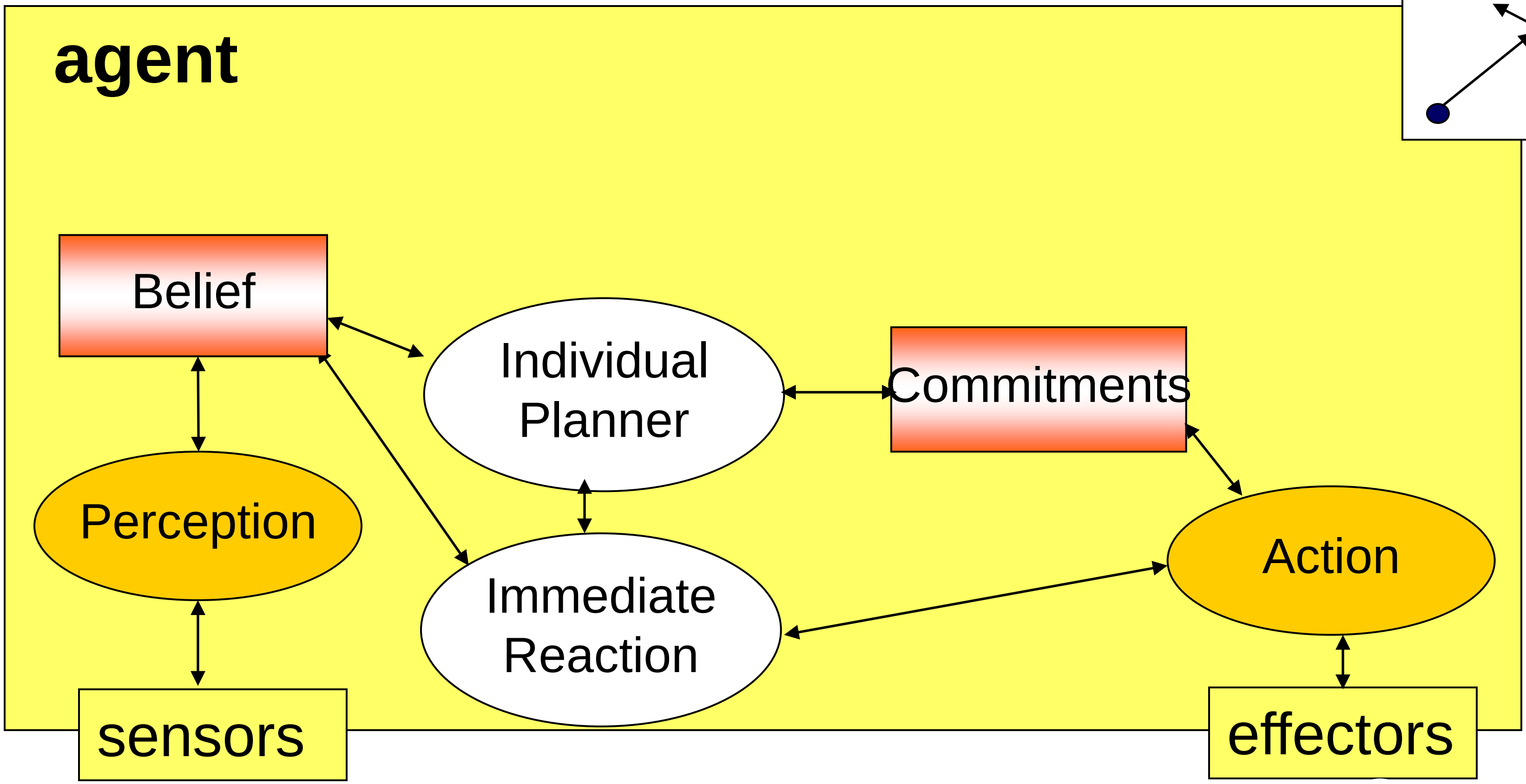
“Mental States”



Past: *Belief* (world model)

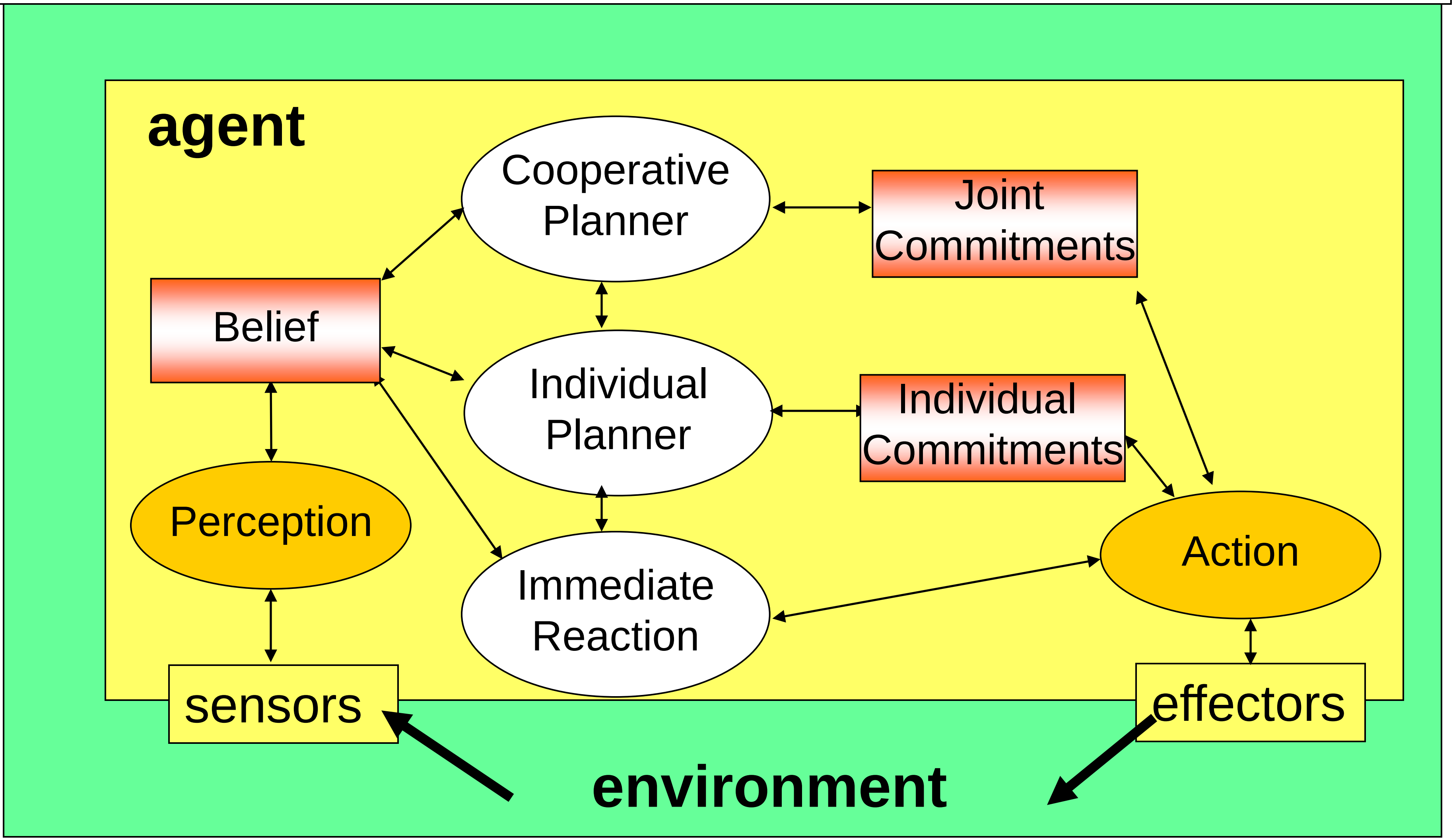
Future: *Commitment* (goal, intention, plan, ...)





60

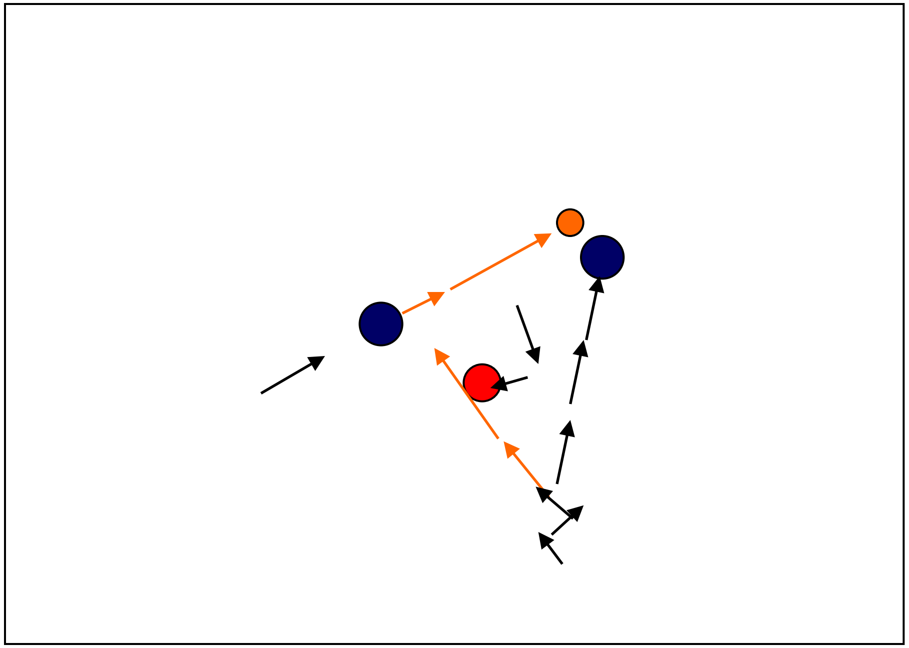
Cooperative Behavior



Cooperative Behavior

Cooperation

Joint intention *(Double pass)*



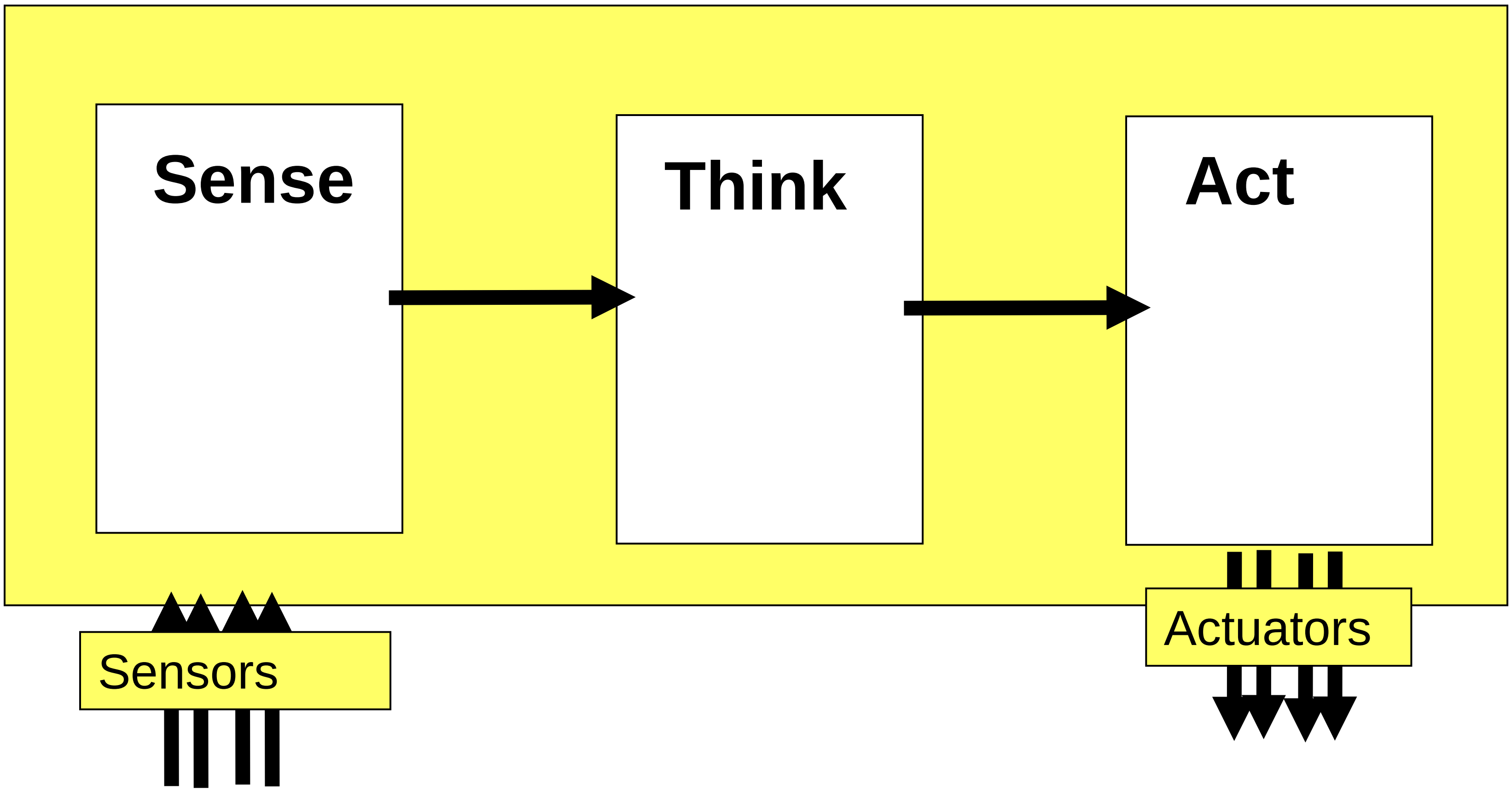
Control Architectures

Distribution

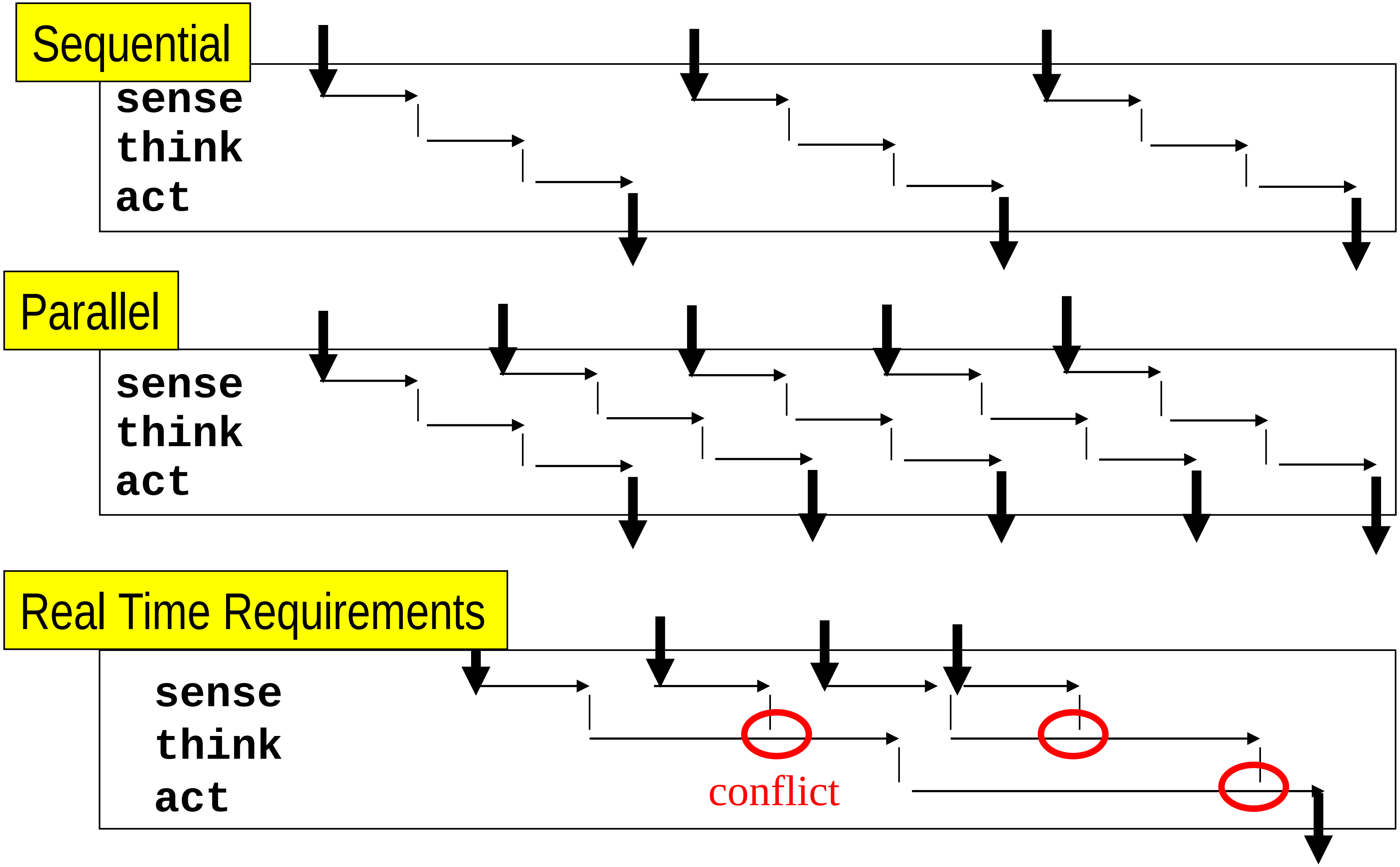
Interfaces

Information flow

„Horizontal“ Structure



Synchronisation

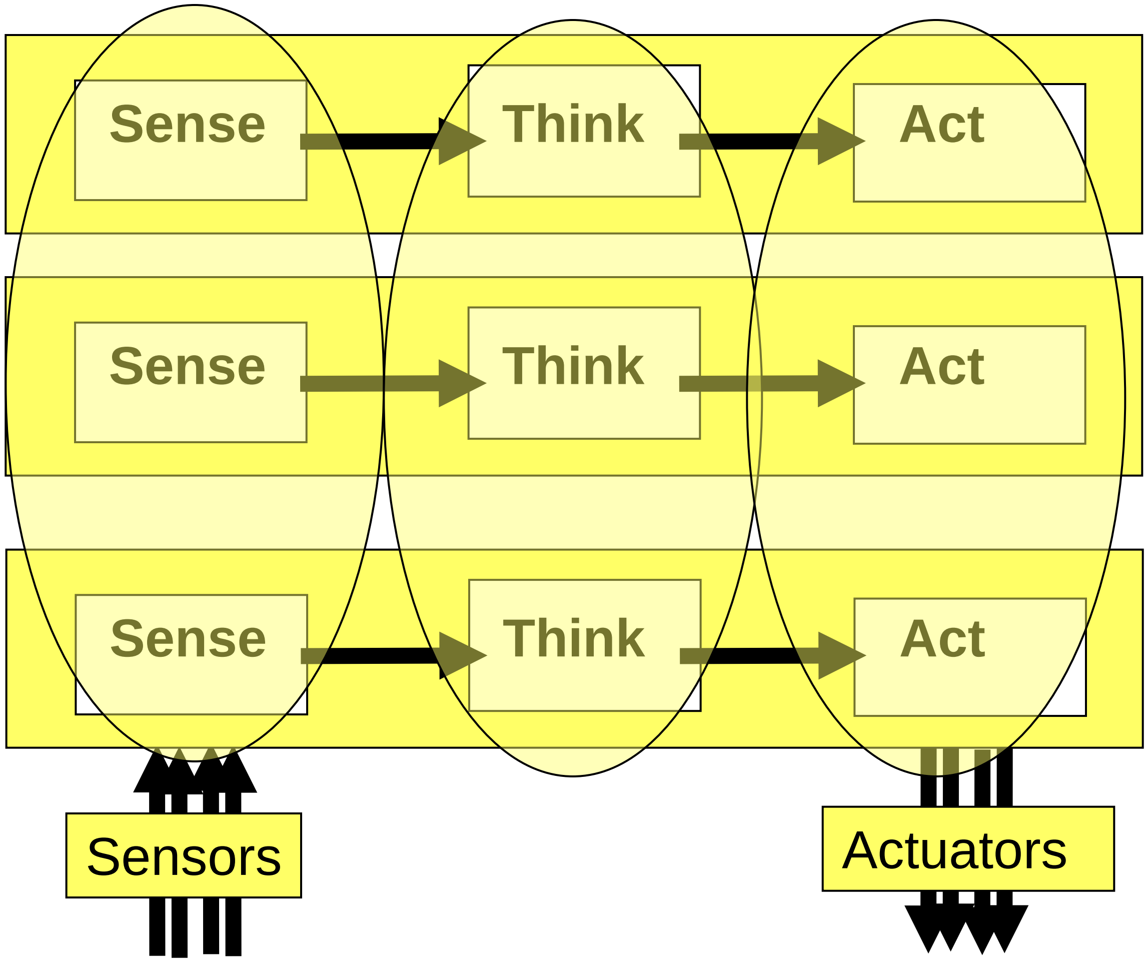


Different Complexities

- World Model
- ...
- Simple Percepts
- Sensor Signals

- Cooperative Planning
- Planning
- ....
- Choice of Skill
- Reaction (Stimulus Response)

Layered Architecture: Example



Deliberative Layer:

- Long Term Planning
- Time consuming

Working Layer:

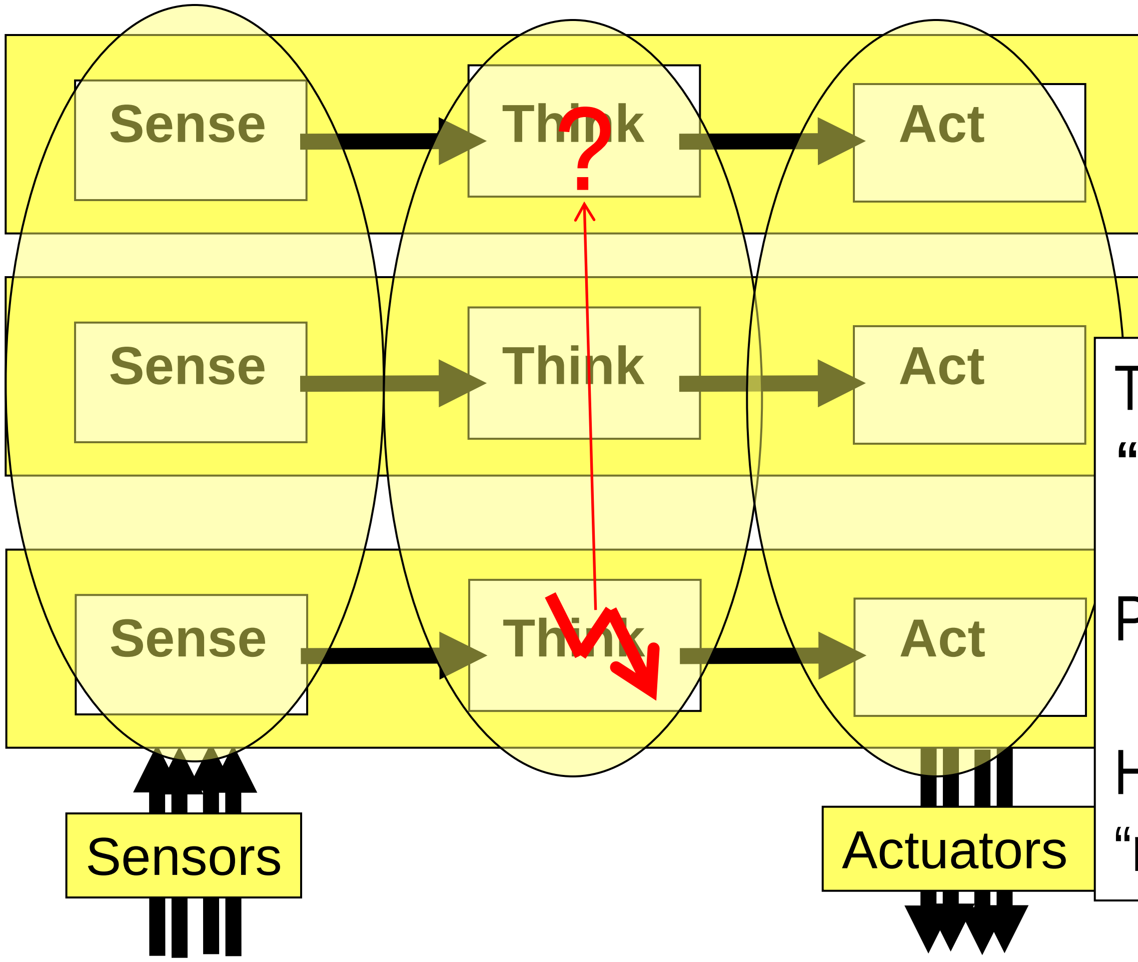
- Scheduling of „skills“
- Needs moderate time

Reactive Layer:

- Immediate reactions

Layered Architecture: Reaction Time

Different cycle times at the layers



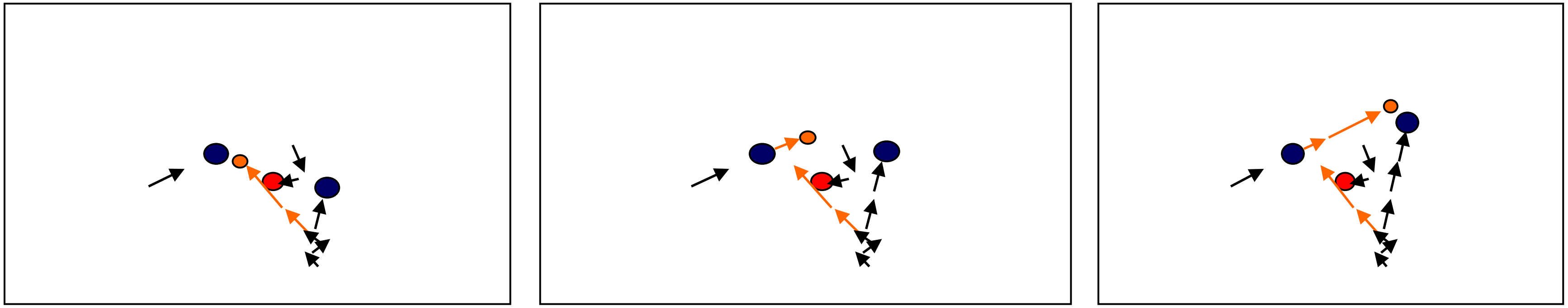
Time problems with "upwards failures":

Problem on low level.

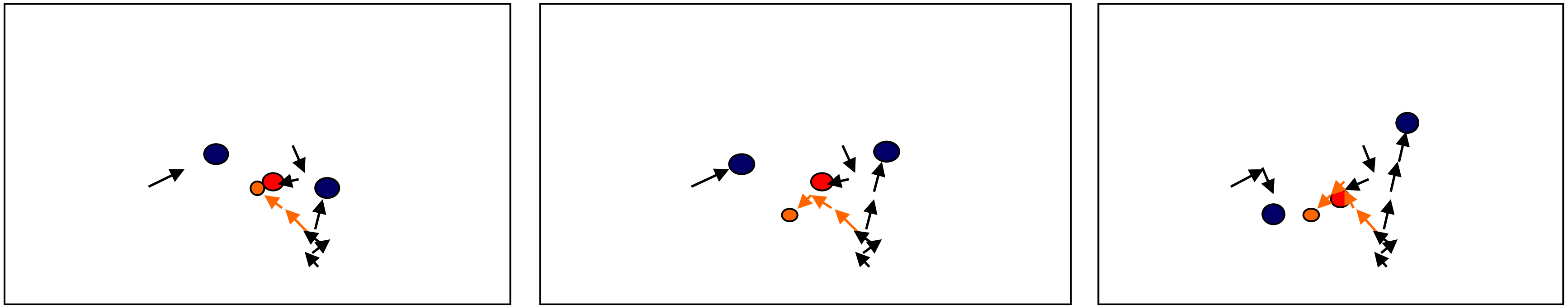
Higher levels react with delay "moment of schock"

Upwards Failure

Planned behavior (double pass)

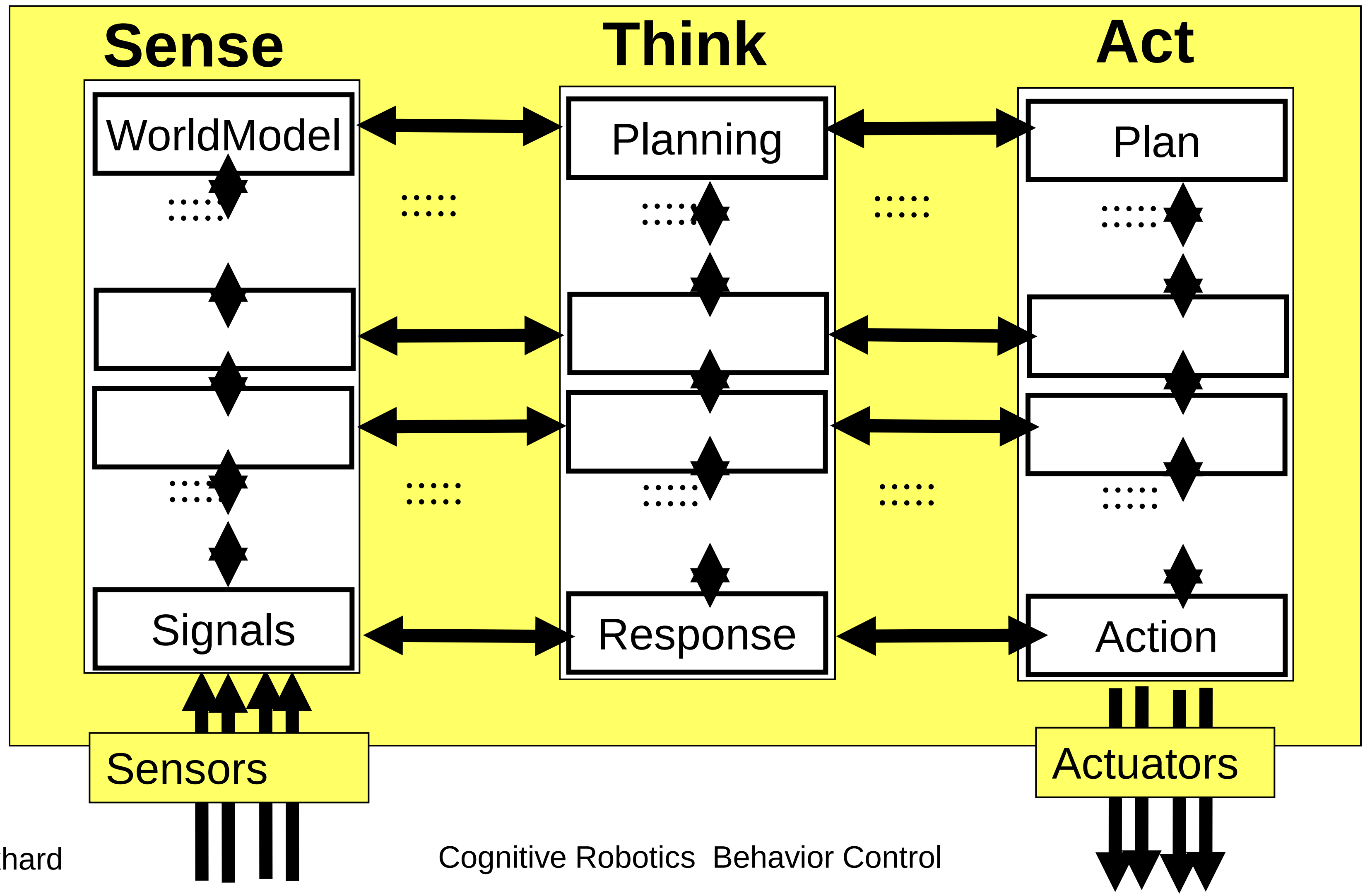


Intercept fails, but other player continues

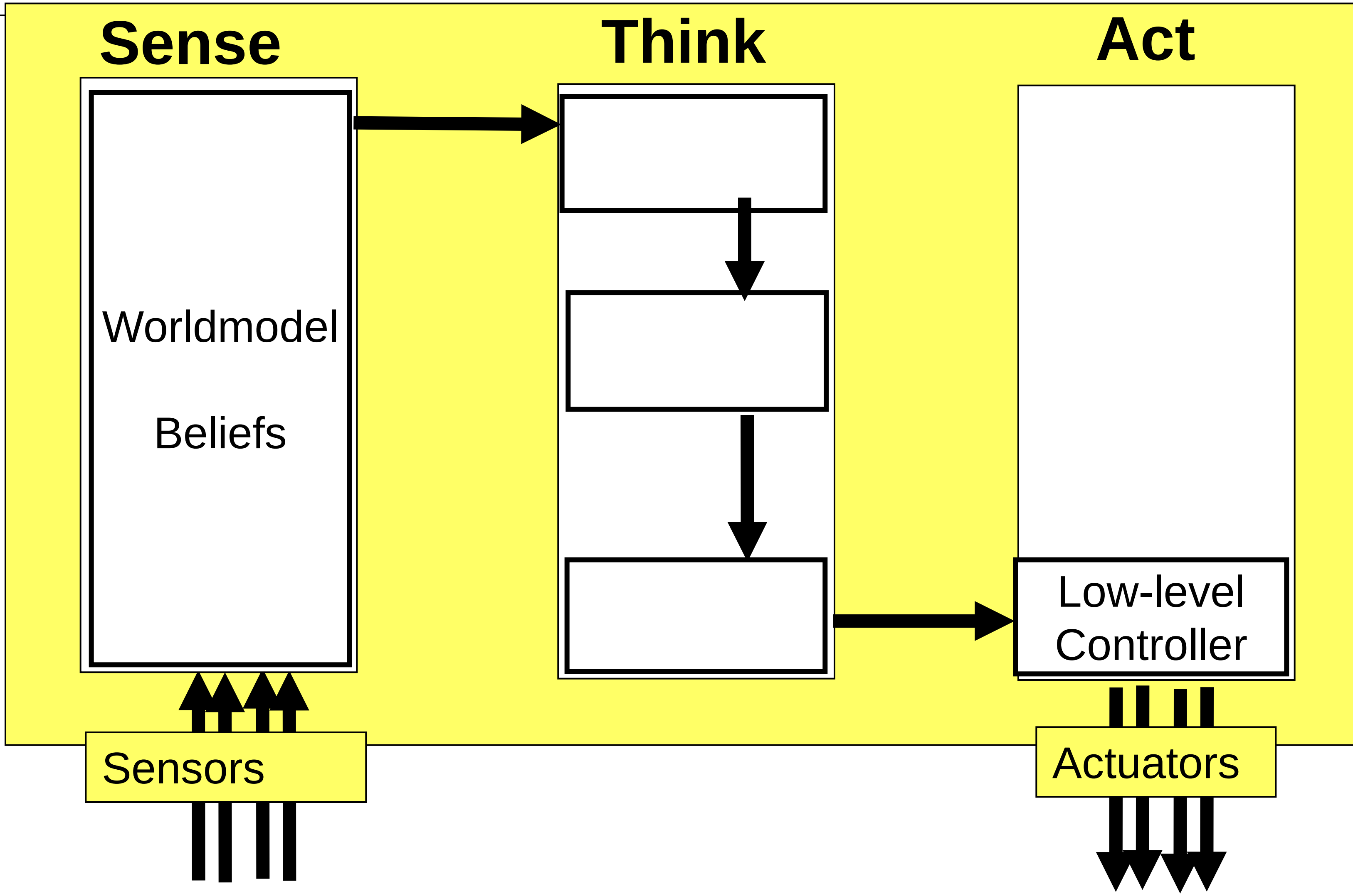


How to Organize Data Flow?

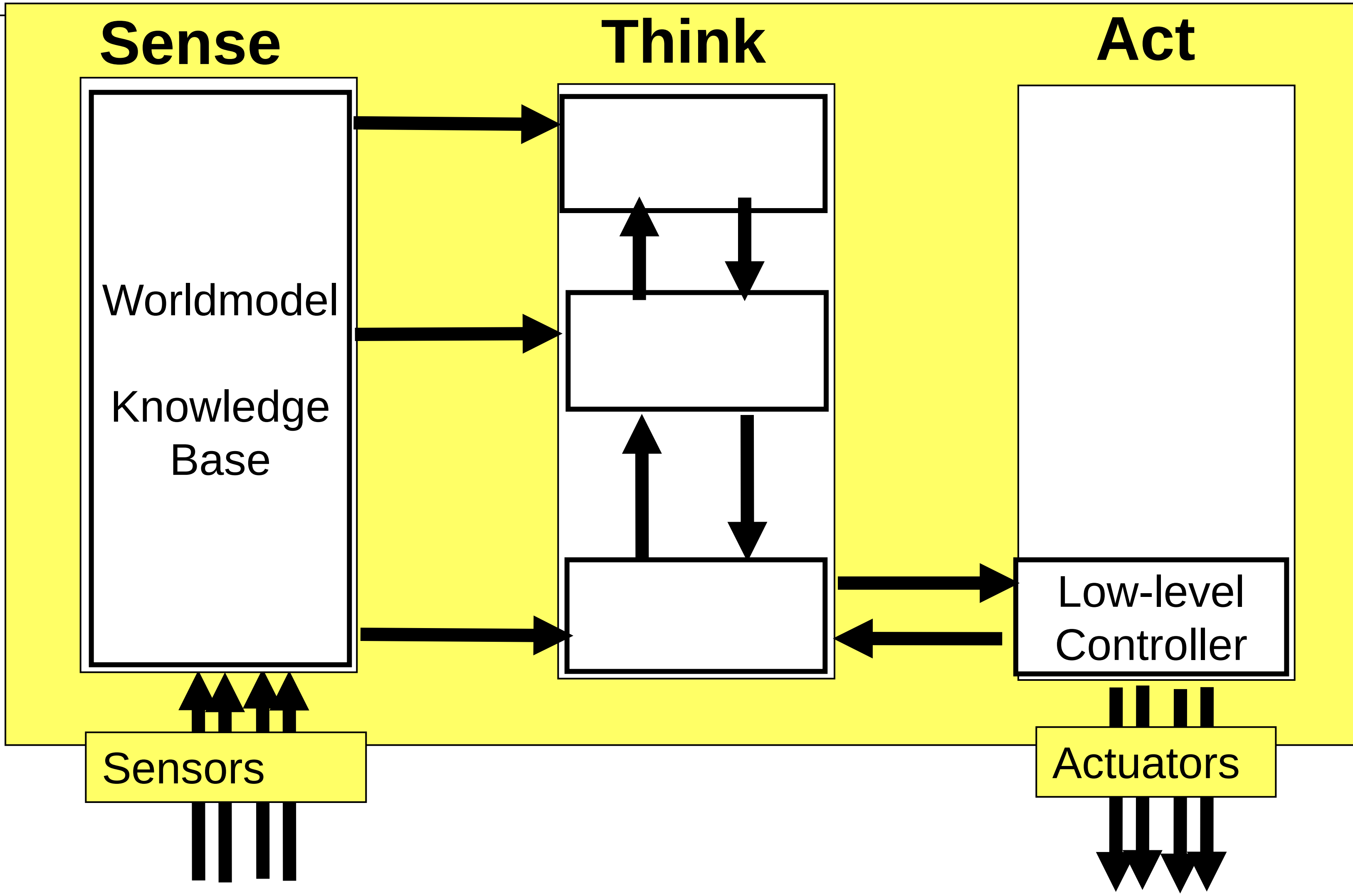
Need for reduction of dependencies



Classical One-Pass-Architecture

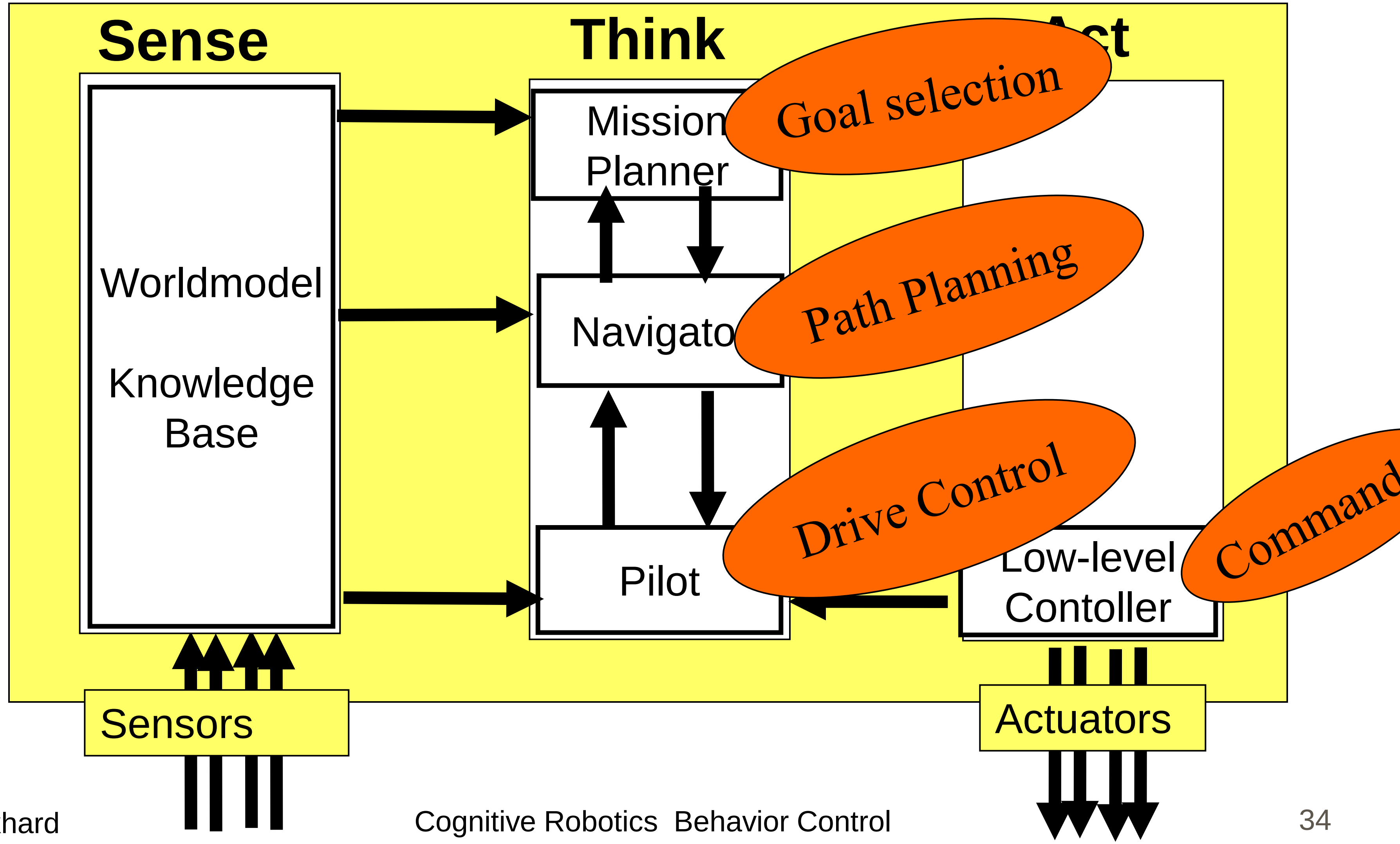


Classical Two-Pass-Architecture



Classical Two-Pass-Architecture

Example: 3-Tiered (3T) Architecture (NASA)



Overview

Introduction

Control Architectures

Aspects of Rationality

BDI Architectures

Behavior Based Robotics

Concept: (Bounded) Rationality

Rational Choice Assumption:

- Agents act as utility maximizers

Needs exact knowledge about future consequences

Critics (Simon): Bounded Rationality

- Only limited knowledge about real world available
- Only limited resources for deliberation

Ideal Rational Agents

Definition by Russell/Norvig:
Artificial Intelligence – A Modern Approach.

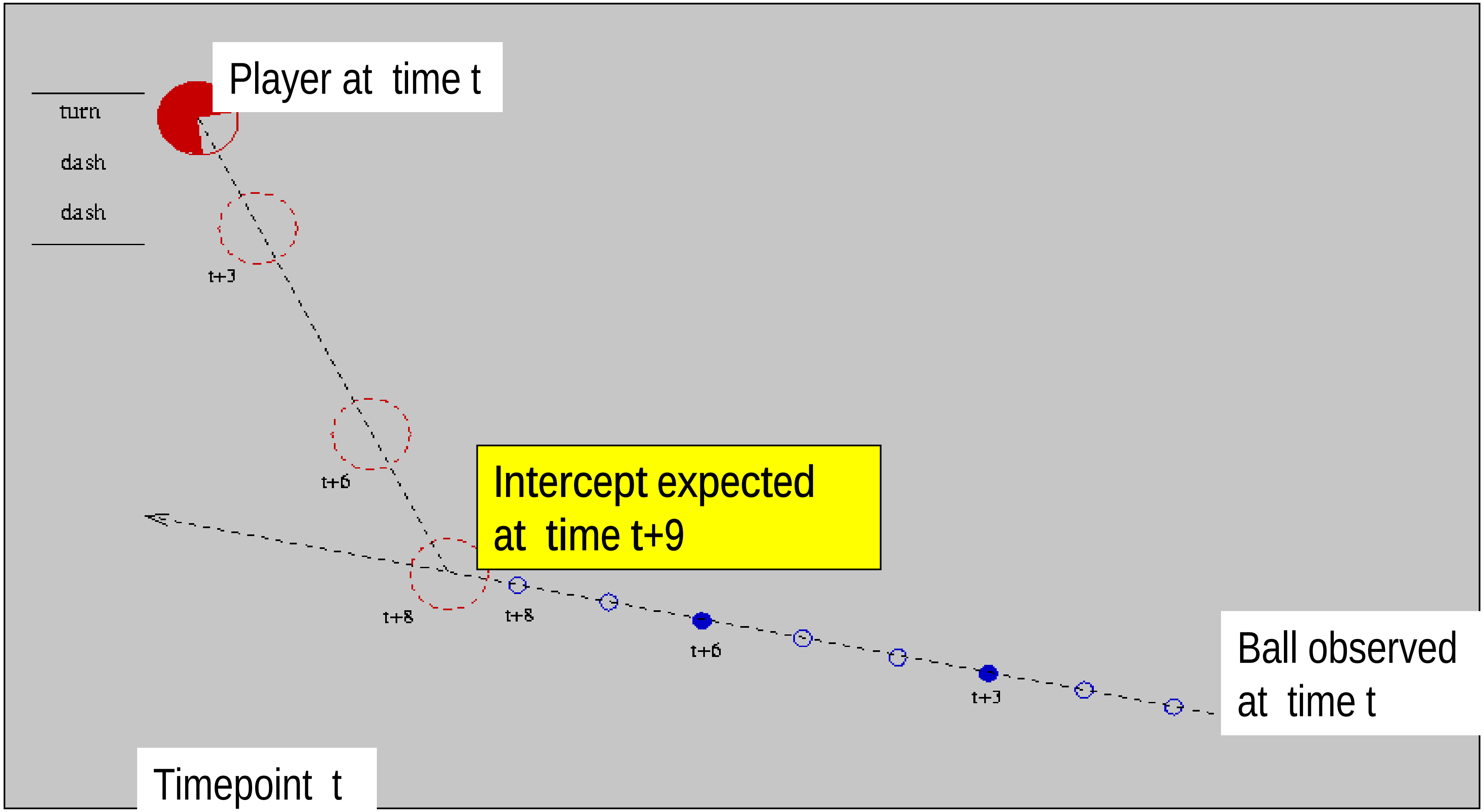
For each possible percept sequence, an ideal rational agent should do whatever action is expected to maximize its performance measure, on the basis of the evidence provided by the percept sequence and whatever built-in knowledge the agent has.

Aspects of the definition:

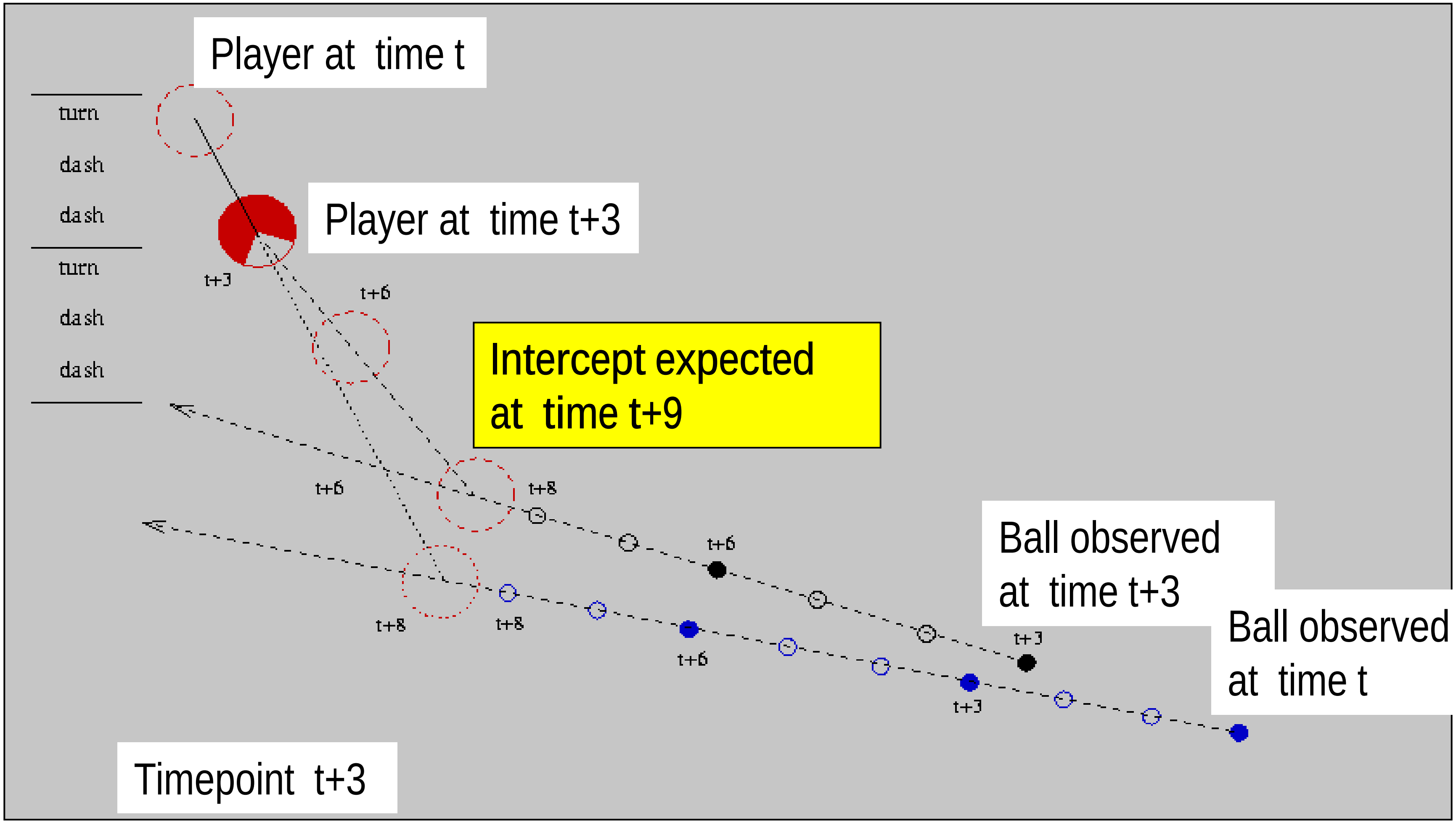
- Performance measure: determines the purposes of agent/robot
- Design problem: designer has to built the necessary means

"Bounded Rationality": Efficiency w.r.t. limited resources

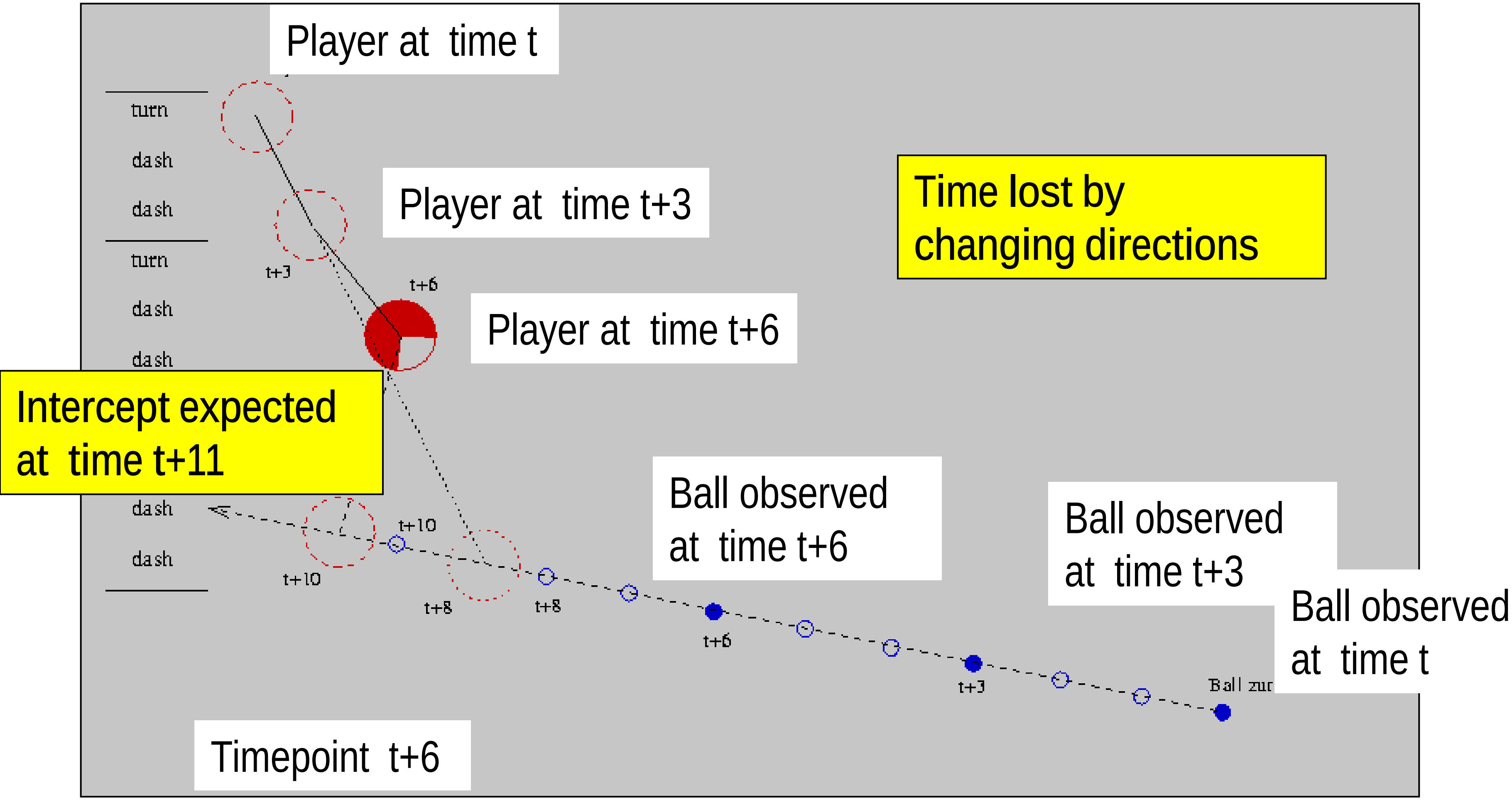
Stability vs. Adaptation



Stability vs. Adaptation



Stability vs. Adaptation



Conflicts between old/new Options

Keep old option

+ Stability
+ Reliabilty (cooperation!)
- Fanatism (misses better options)

Change for new option

+ Adapt to better options
- May lead to oscillations

Treatment of conflict is up to choice by designer
(architecture may even cause an implicit design decision)

Protocols for Coordination

Communication
- needs time
- can be disrupted
- can be inconsistent/conflicting

Coordination possible withoutcommunication …

… if all robots have the same world model

and follow the same protocol.

Otherwise communication can help to

- unify world model
- distribute roles/tasks (by protocol or negotiation or leader …)

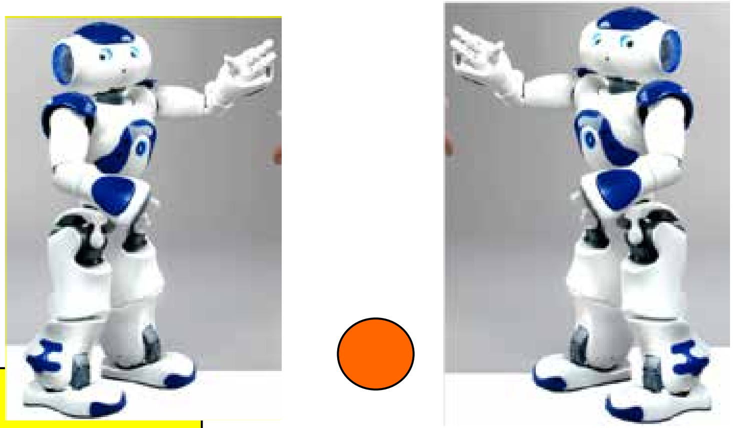
Example: Roles by Protocol

Role	**Task**	**Assignment by**
Goalie	defend goal	fixed
Attacker	ball handling	closest to ball
Supporter	support attacker	close to attacker
Defender	backward support	most back

Example: Stability vs. Adaptation

Role change: Player closest to ball is attacker

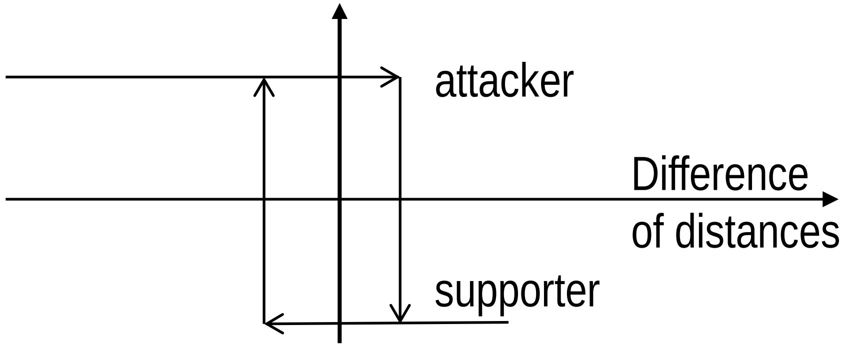
Distance to ball can oscillate by noisy observations



Solution:

Keep role in case of small deviations

("Hysteresis control")



Competing Desires

Robot wants e.g. to

- Change position (for supporting)
- Avoid obstacles
- Look for landmarks (for localization)
- Observe the ball
- Observe other players

Can he pursue all these desires in parallel?

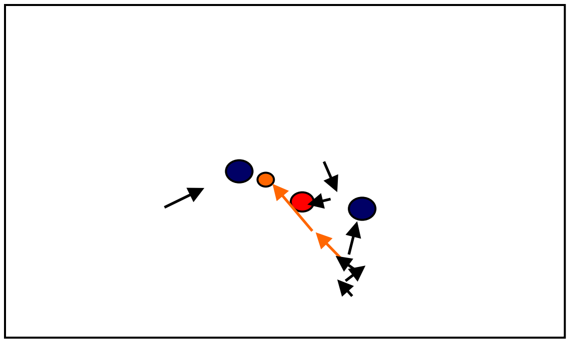
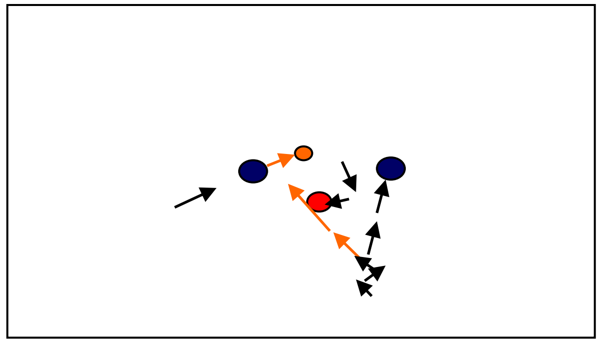
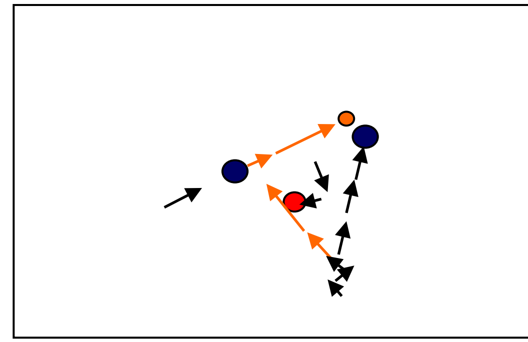
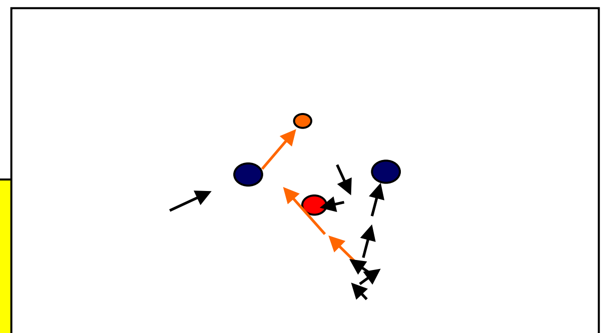
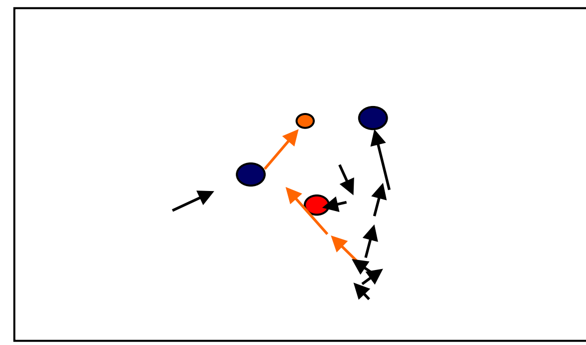
Rational behavior: Commit only to achievable intentions.

Possible solution: “Screen of admissibility”

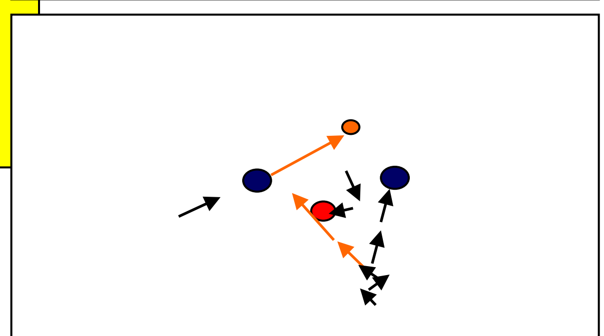
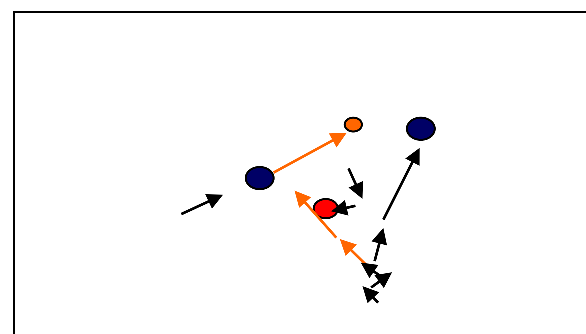
Adapt new intentions only if not in conflict with already adapted intentions. (® gives priority to stability)

Least Commitment

Cannot plan all future details in advance

Solution
"Least commitment":
Postpone decisions
as long as possible.

Overview

Introduction

Control Architectures

Aspects of Rationality

BDI Architectures

Behavior Based Robotics

BDI Agent Architecture

Most popular architecture for reasoning agents.

Originally based on concepts of

Michael E. Bratman: “Intention, Plans, and Practical Reason” , 1987.

The architecture is built on

- Possible facts about the world
- Potential options the agent might achieve

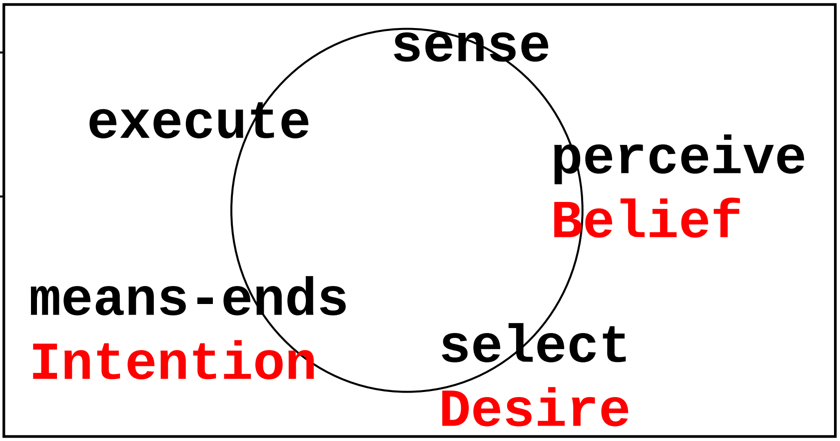
BDI stands for

Beliefs: Information the agent has about the world

Desires: States of affairs the agent *would like* to accomplish

Intentions: States of affairs the agent *is trying* to accomplish

BDI-Modell



Belief (world modell)

Desire (useful options)

Intention (committed options)

new_Belief := *update*(Perception, old_Belief);
new_Desires := *select* (new_Belief,old_Desires);
new_Intentions := *means-ends*(new_Belief, new_Desires, _old_Intentions);

Consistency:

–Desires may be inconsistent

–Intentions must be consistent

Intentions set a screen of admissibility:

Only those desires may be adopted

which are consistent with recent intentions

AgentSpeak and Jason

AgentSpeak

is a logic-based agent-oriented programming language
implementing (some) concepts of BDI-architectures
proposed by Arnand S. Rao 1996 based on experiences with

PRS (Georgeff, Lansky 1987),
dMARS (Kinny 1993),
Agent-0 (Shoham 1993)

Jason

extension of AgentSpeak
with Prolog-like syntactic structures
interpreter in Java, highly customizable
developped by Rafael H. Bordini, Jomi F. Hübner and others

Interpreter

Works with

Plan Library (initially filled)

Belief Base (Memory of actual beliefs)

Event Base (Memory for changes of beliefs and goals)

Intention Base (Stacks of pending goals)

Selection functions to select from the different Bases

Next Slides: Syntax and Informal Semantics.

Cited from Bordini/Hübner:

Jason - A Java-based interpreter for an extended version of AgentSpeak.

Release Version 0.9.5, February 2007.

http://jason.sourceforge.net/Jason.pdf

Simplified Syntax of AgentSpeak

belief::= atomic_formula (*of kind P(t1, . . . , tn)*)

plan ::= triggering event : context <- body .

triggering event ::= + belief | - belief | + goal | - goal

(*belief or a goal , added (+) or deleted (-) before*)

context ::= *conjunction of beliefs (preconditions)*

body ::= *sequence of external actions, goals and belief updates*

goal ::= ! atomic:_formula | ? atomic_formula

achieve goal (!) resp. test goal (?)

belief update ::= + belief | - belief

add(+) resp. delete(-) a belief

Beliefs in Jason

Beliefs: first-order formulae

```
ball(10, 10)
```

agent believes the ball is at position (10, 10)

Beliefs can have annotations

```
ball(10, 10)[source(percept)]
```

information was perceived from environment

Support for strong negation (besides negation by CWA)

```
~near(ball):
```

agent believes it's not near the ball

Belief base can also process (*Prolog*-like) rules

Goals

2 types of goals:

Achievement goals for calling plans

!kick(ball)

might e.g. invoke a plan to bend the knee and kick

Test goals for tests of beliefs:

?see(ball)

succeeds if the agent actually sees the ball

Plans

Plan in rule form

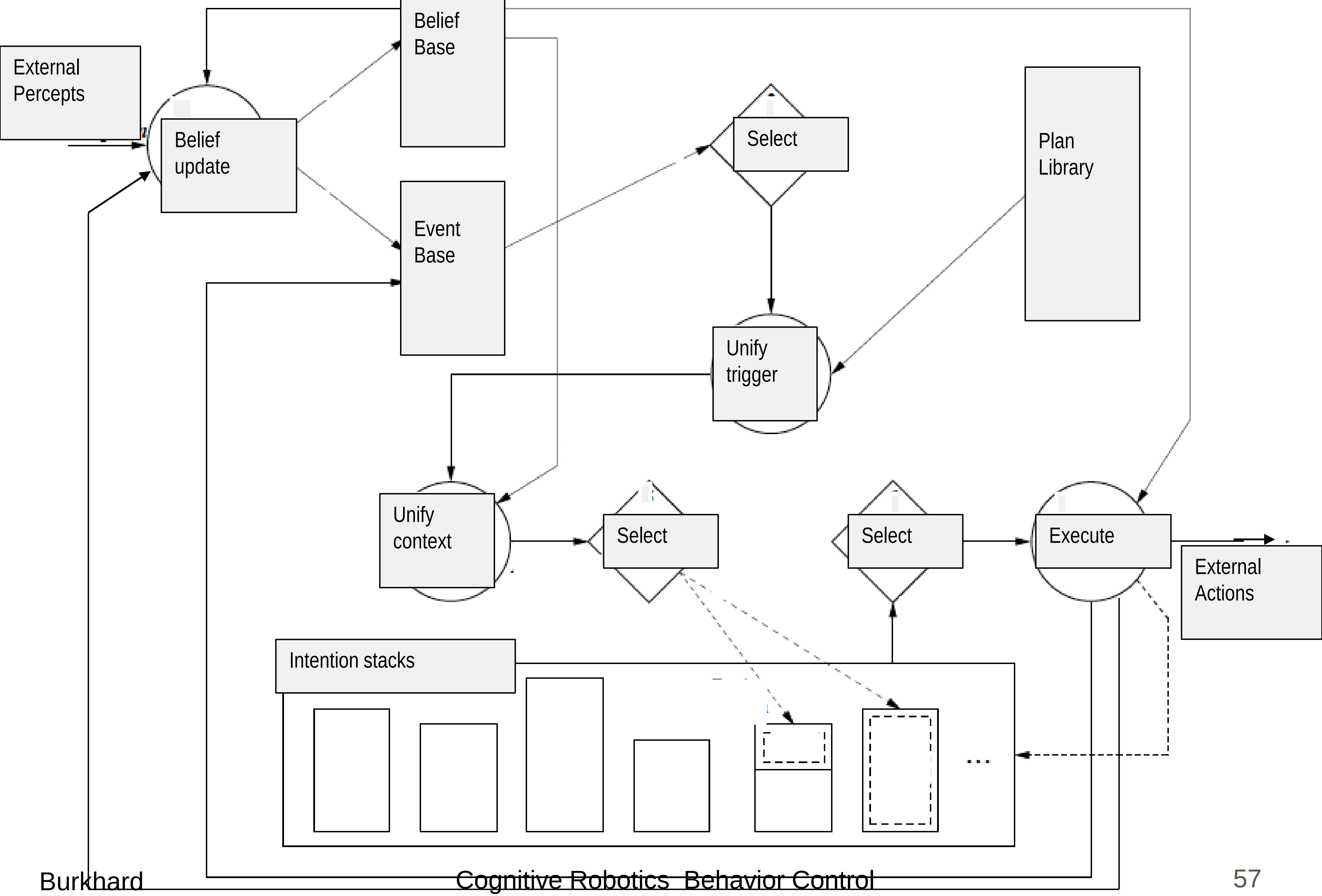
triggeringEvent	*(Change of beliefs, goals …)*
: context	*(Conditions which must hold)*
<- body	*(Sequence of goals and external actions)*

```
+down
  :  isDownOnBack
  <- standUp; -down; +search.
+down
  :  isDownOnFront
  <- rollOver; standUp; -down; +search.
```

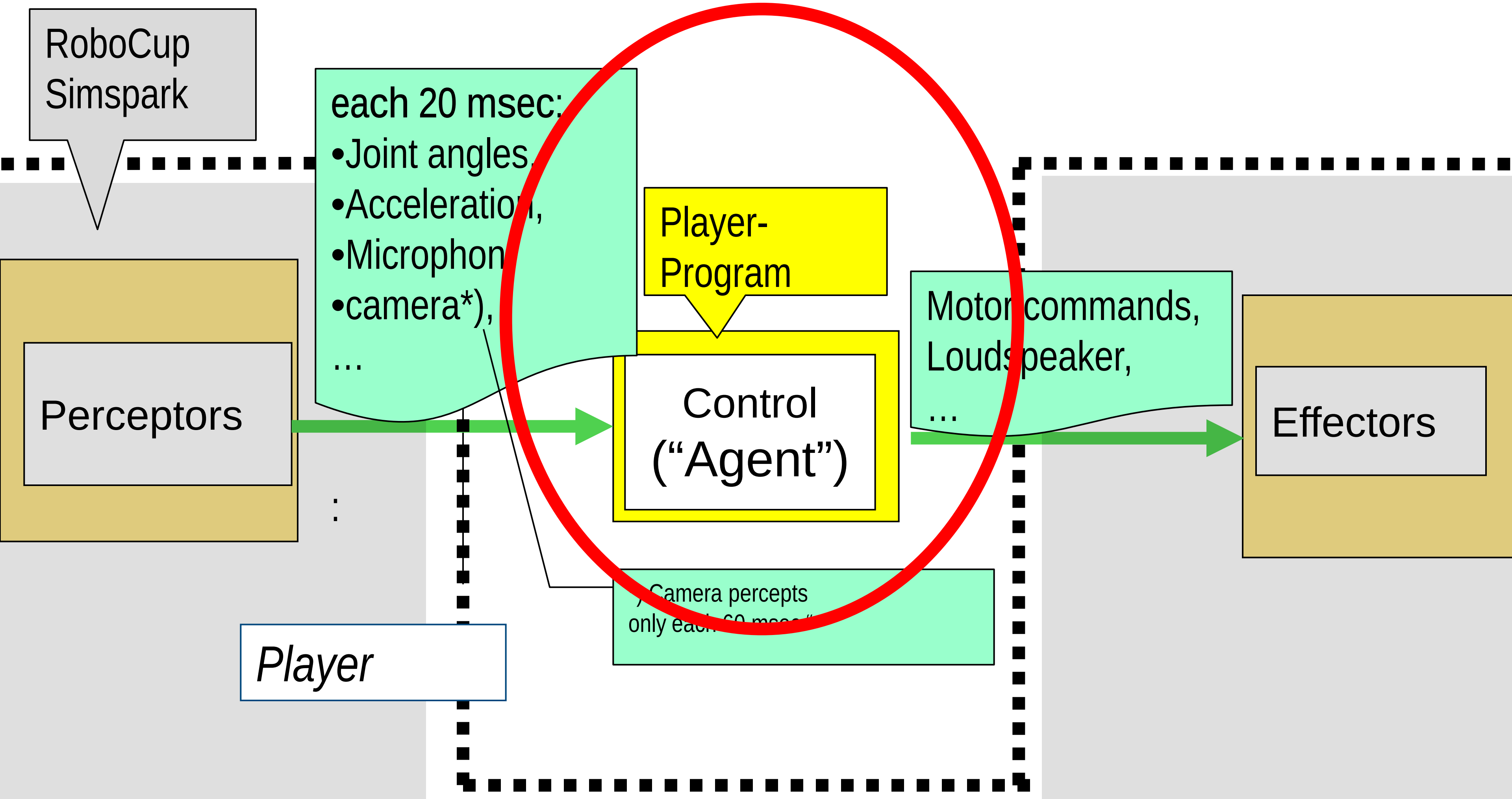
Simplified Reasoning Cycle

1. Update belief base by external percepts
2. Update event base according to previous steps
3. Select actual triggering event e
4. Determine relevant plans by unification with e
5. Determine applicable plans by checking contexts
6. Select a plan p from applicable plans
7. Update actually processed intention according to p
   resp. initialize new intention (for external event)
8. Select intention i for processing
9. Execute next subgoal from top of intention i :
   perform external action or update belief or test belief

Basic Cycle of the Interpreter



Implementation of Soccer Agents



Implementation of Soccer Agents

Sense-think-act-cycle

Experimental Implementation
of Soccer Agents
by Dejan Mitrovic (Novi Sad)

Sense: process perceptor data from SimSpark Simulator
implemented in Java

Think: analyse sitation and specify goals
implemented in Jason

Act: send action commands to SimSpark Simulator
implemented in Java

Implementation of Soccer Agents

Redefined methods from class *AgArch:*

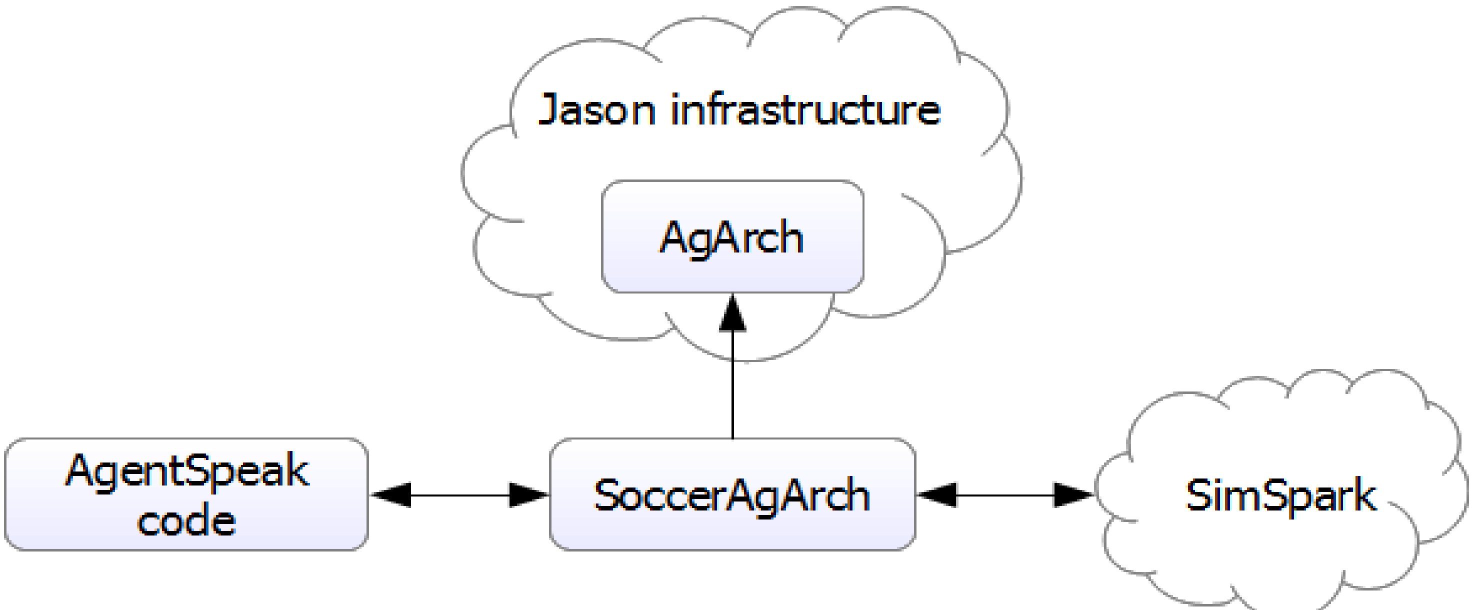
List<Literal> perceive()

merges list of perception with belief base

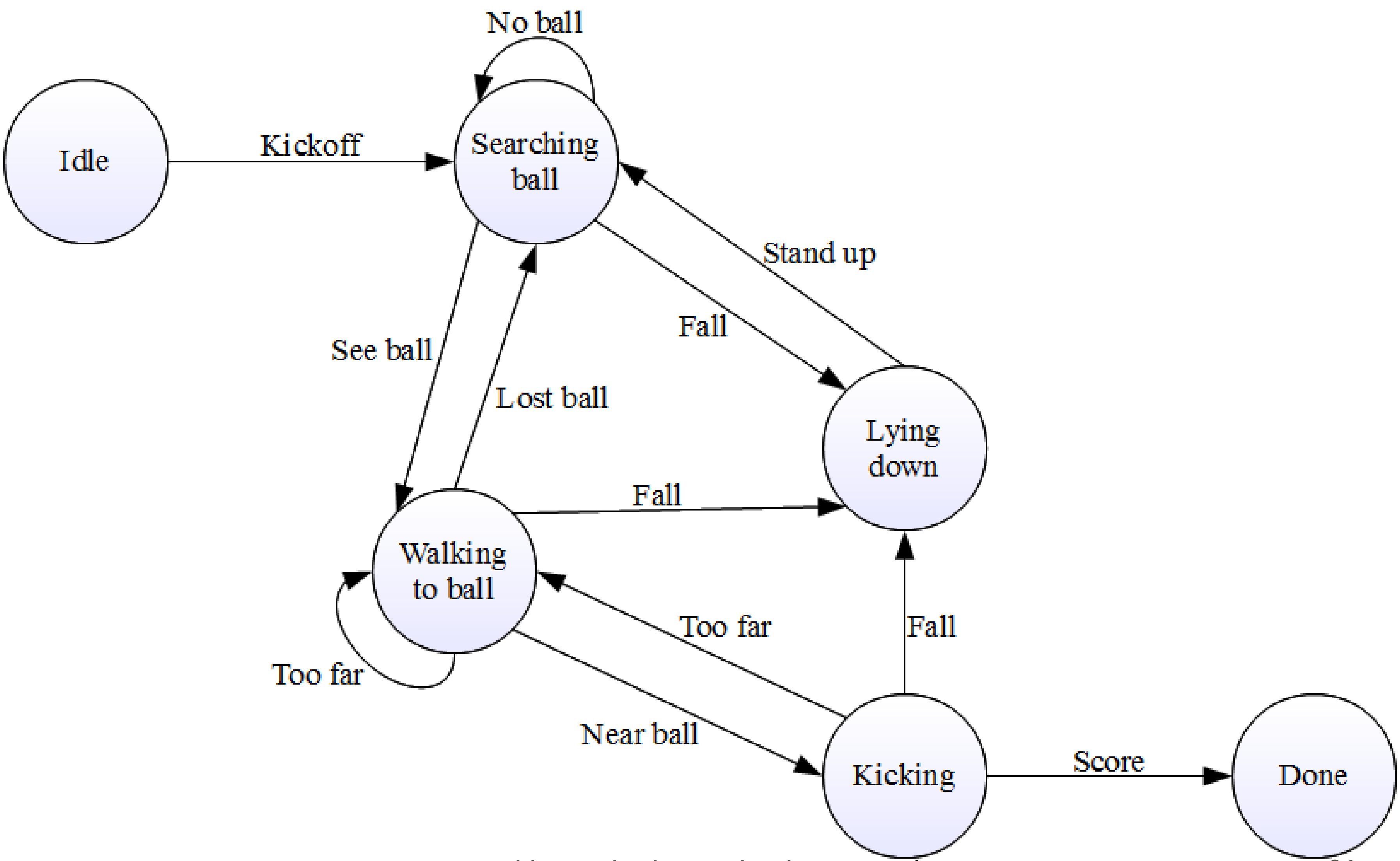
at the beginning of each cycle

void act(ActionExec action, List<ActionExec> feedback

executes action at the end of each cycle



Example of a Simple Agent



(Partial) Implementation of the Example

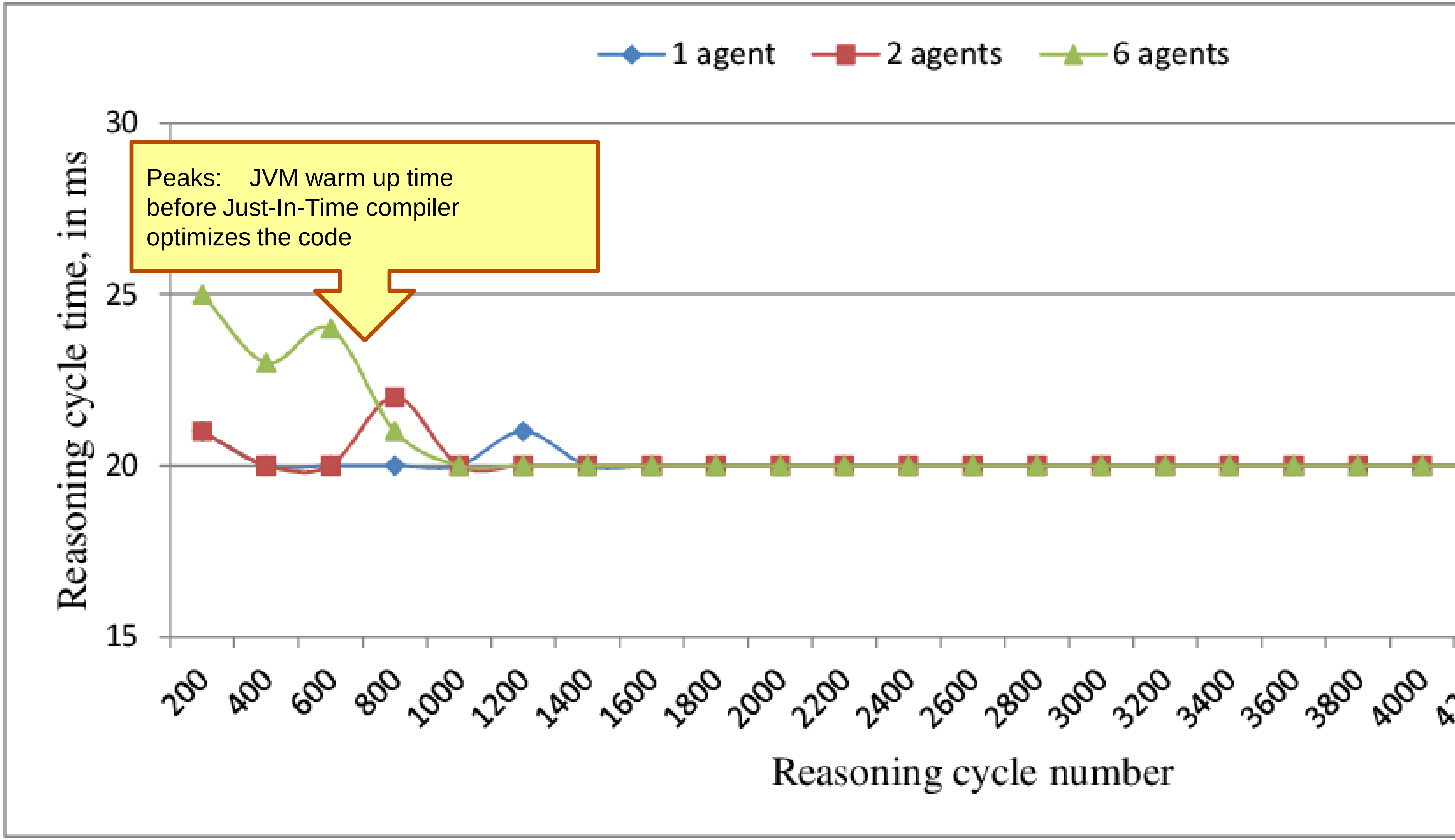
```
isDown :- acc( ,  , Z) & Z < 7.
```

```
+kickOff : true <- +search.

+search : true <- !findBall.

+!findBall : isDown <- -search; +down.
+!findBall : not isDown <- turnLeft;
             ?seeBall(_, _, _); -search; +walk.
-!findBall : true <- !!findBall.
```

```
+down
  :  isDownOnBack
  <- standUp; -down; +search.
+down
  :  isDownOnFront
  <- rollOver; standUp; -down; +search.
```

Performance

Duration of the reasoning cycle with 1,2,6 running agents:



DPA = Double Pass Architecture

… another approach to implement BDI

Diploma Thesis Ralf Berger, used in RoboCup 2D league

How to program a double pass?

1. Trial („Chess-like“):

- Foresight simulation
- Choice of best alternative

Result:
Useful only for
short term decisions

2. Trial („Emergence“)

If every player behaves in some simple optimal way,

then a double pass can emerge without planning.

Result:
Double pass emerges
only from time to time

How to program a double pass?

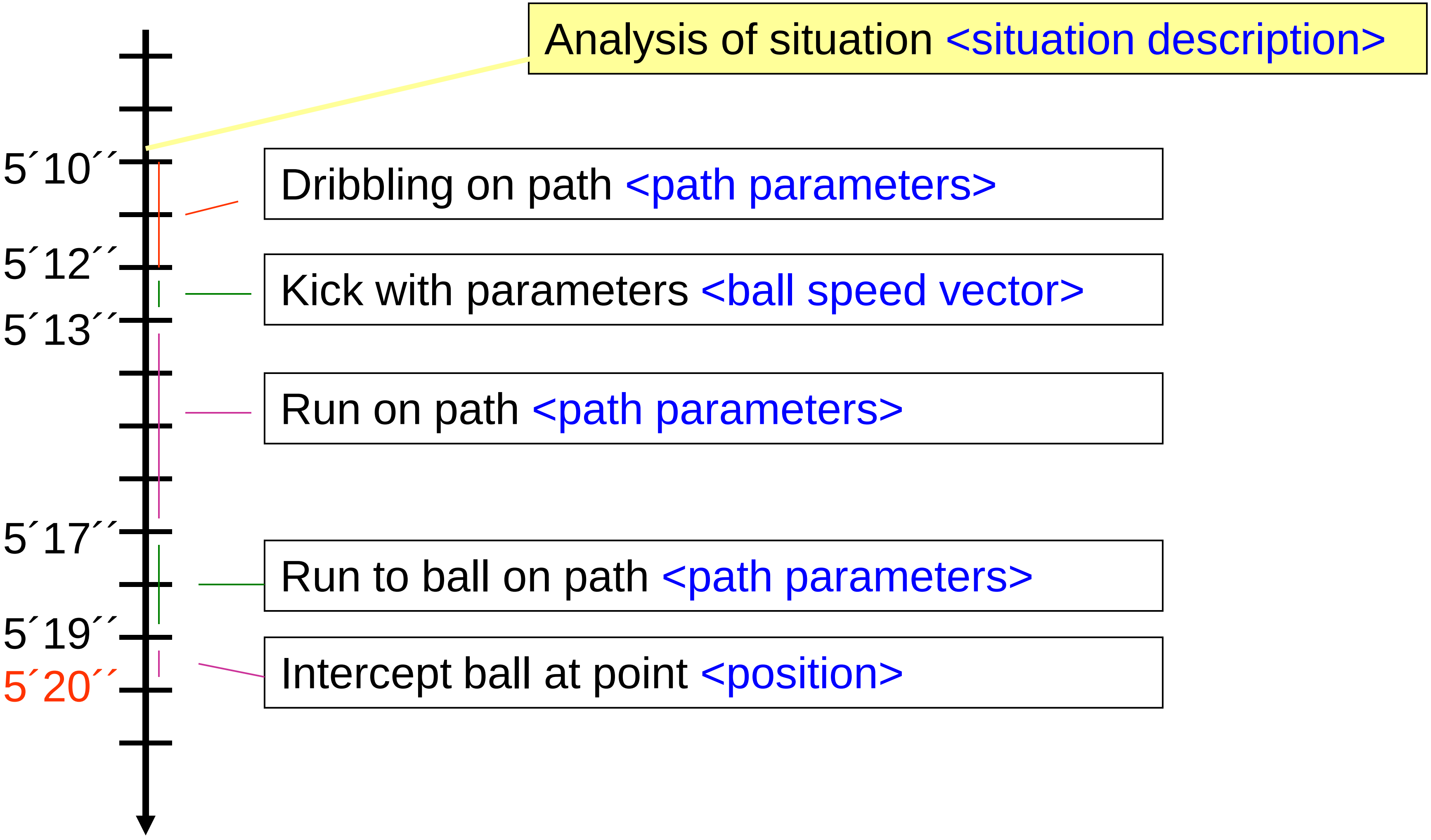
3. Trial:

- Use Bratman´s concept of „bounded rationality“

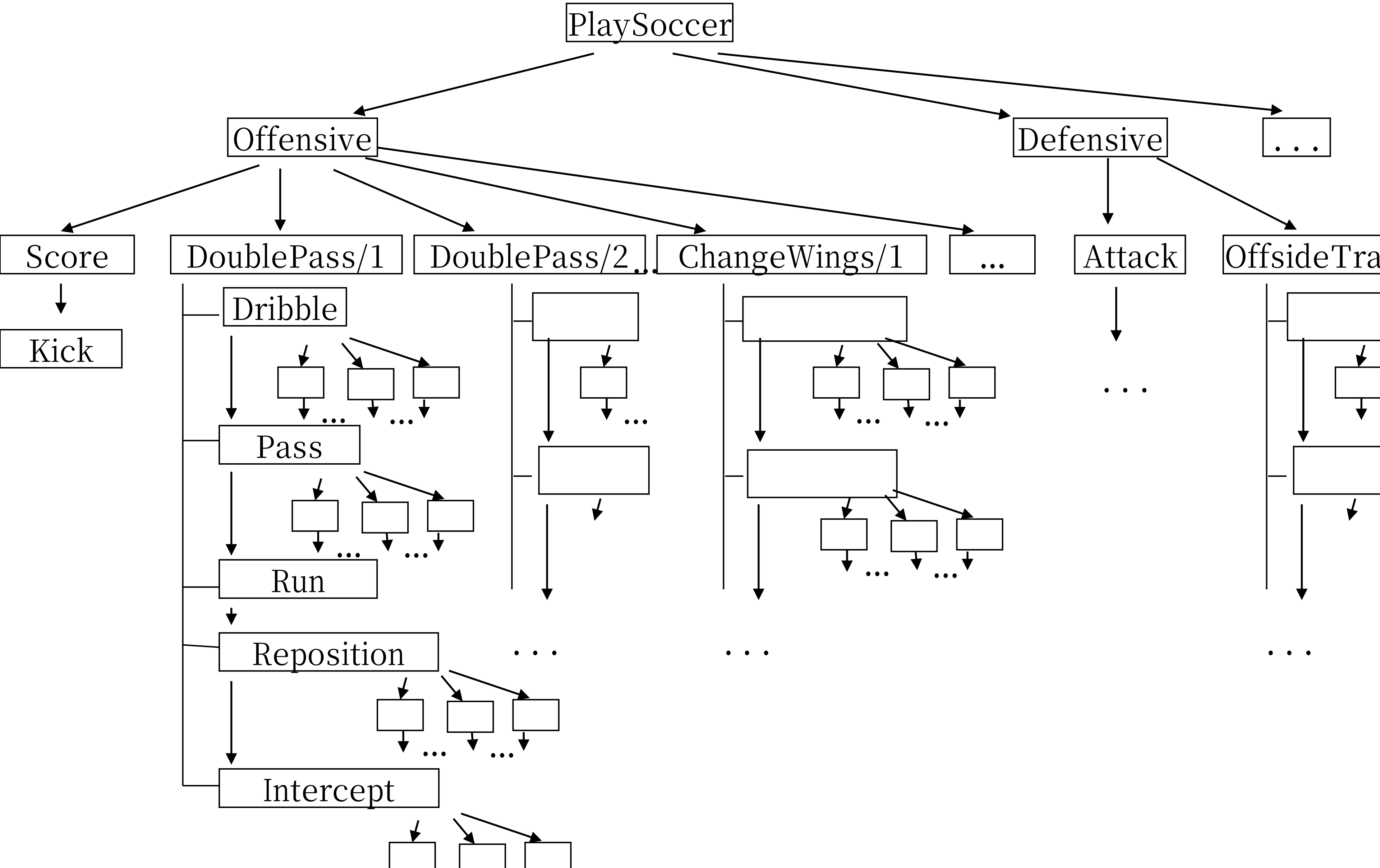
  Belief-Desire-Intention-Architecture (BDI)

- Use Case-Based Reasoning

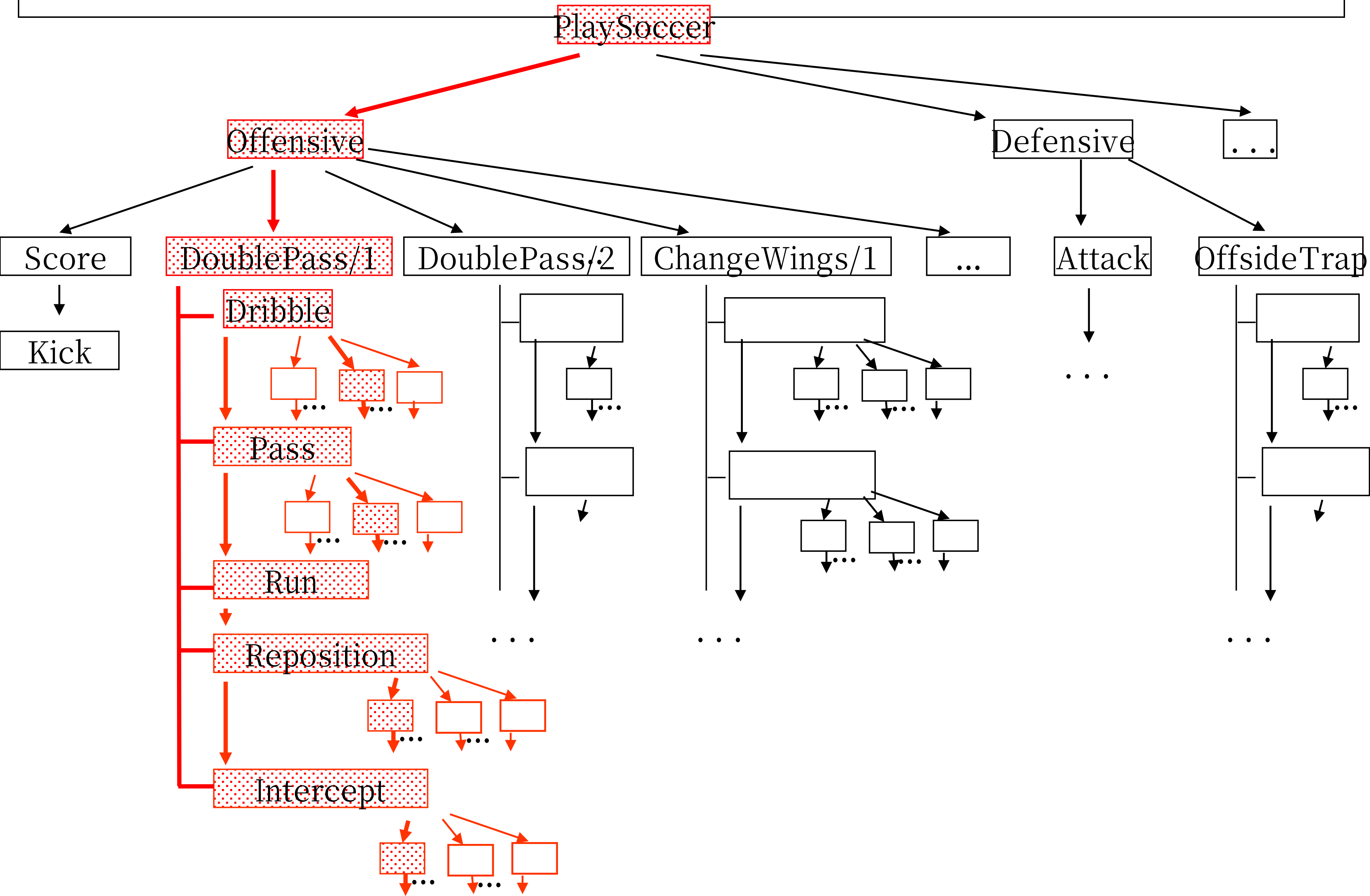
Time 5´20´´ (Least Commitment)



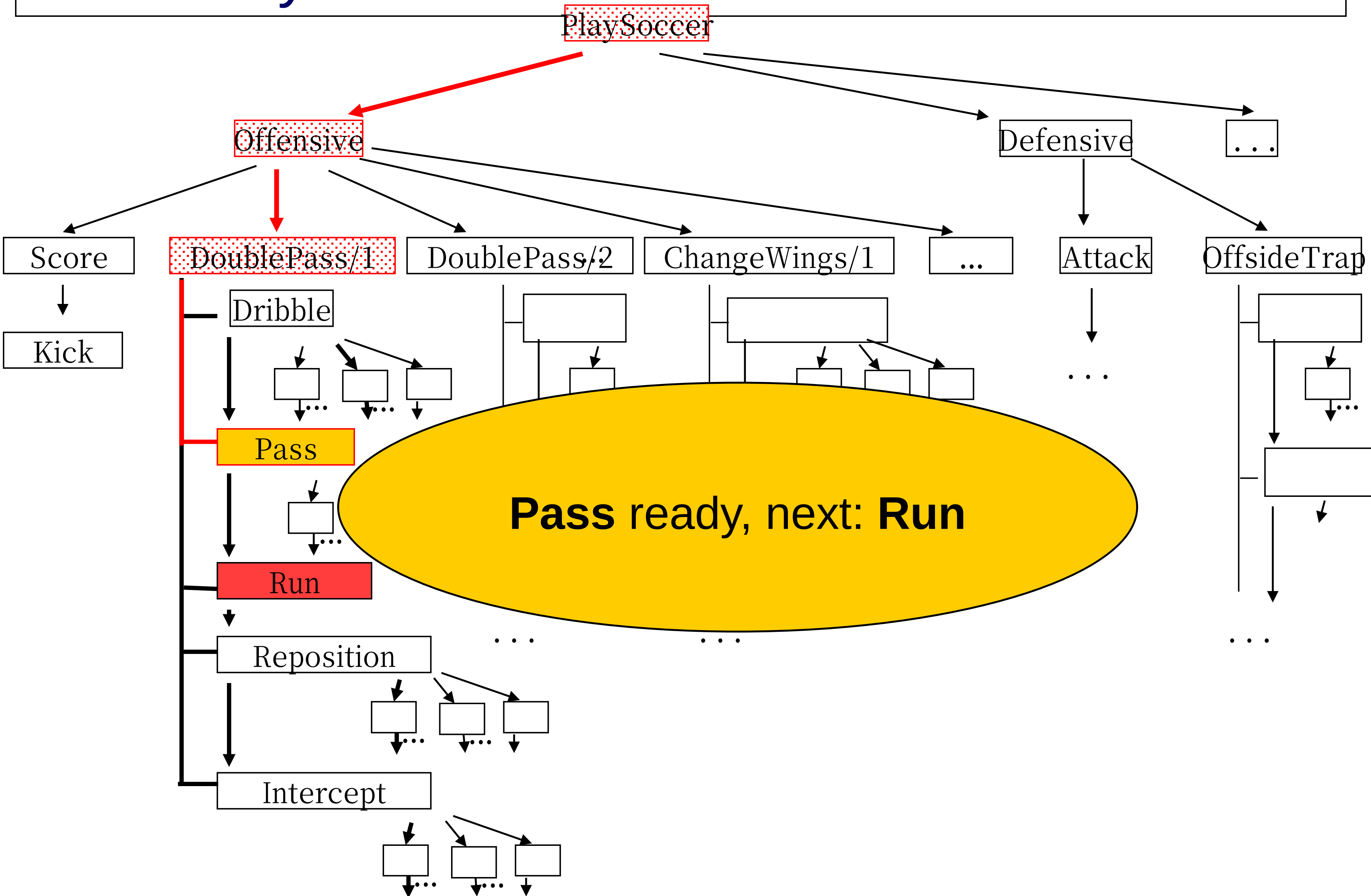
Hierarchy of Options



Result of “Deliberator“: Intention Subtree



Activity Path: Present state of an Intention



Double-Pass Architecture

- Predefined Option Hierarchy
- Deliberator
- Executor

„Doubled“ 1-Pass-Architecture:

1. Pass: Deliberator (goal-oriented: intention subtree)
2. Pass: Executor (stimulus-response: activity path)

- on all levels -

Differences to “classical” Programming

Control flow by Deliberation (“Agent- oriented”)

Runtime organization by 2 Passes through all levels

Overview

Introduction

Control Architectures

Aspects of Rationality

BDI Architectures

Behavior Based Robotics

Behavior Based Robotics

Hypothesis:
Complex behavior emerges by combination of simple behaviors

Simple behavior by e.g.

- Immediate reaction to sensor data (sensor-actor-coupling)
- Simple physical „transformation“ (clever design)

Intelligent action without intelligent thinking:

- No worldmodel
- No symbols
- No deliberation

Emergent behavior:

Complex behavior **emerges** by interaction of **situated** robots with the environment.

„New AI“

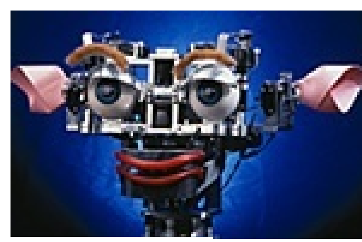
Since middle of 1980s

Papers by Rodney Brooks:
„Elefants don´t play chess“
„Intelligence without reason“
„Intelligence without representation“

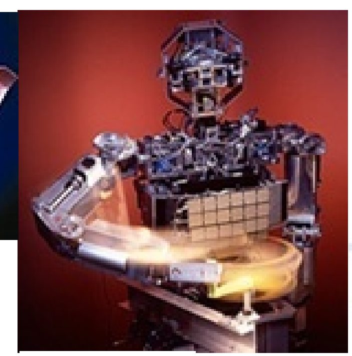
Patty Maes

Orientation on natural principles:

- Emergent behavior
- Situated agents/robots
- No internal representation



Kismet



Roomba

Coq

Critics on Classical AI

GOFAI = “Good Old Fashioned AI”

Problems with

- Closed world assumption: „everything is known“
- Frame problem:„all assumptions/effects are modelled
- Physical systems hypothesis: complete symbolic representation

1966-72: Robot Shakey (Stanford)
with hierarchical planner STRIPS
(„Stanford Research Institute Problem Solver“)

Physical Symbol System Hypothesis

"A physical symbol system has the necessary and sufficient means for intelligent action.“

Newell/Simon: "Computer Science as Empirical Inquiry: Symbols and Search“

GOFAI= „good old fashioned AI“

Needs:

- Complete Descriptions of the Worlds
- Algorithms for actions

Many critics
(Dreyfus, Searle, Penrose, ..., Brooks, Maes, Pfeiffer...)

Physical Grounding Hypothesis

This hypothesis states that to build a system that is intelligent it is necessary to have its representations grounded in the physical world. Our experience with this approach is that once this commitment is made, the need for traditional symbolic representations fades entirely. The key observation is that the world is its own best model. It is always exactly up to date. It always contains every detail there is to be known. The trick is to sense it appropriately and often enough.

To build a system based on the physical grounding hypothesis it is necessary to connect it to the world via a set of sensors and actuators. Typed input and output are no longer of interest. They are not physically grounded.

R.A. Brooks: Elephants Don´t Play Chess

Physical Grounding Hypothesis

This hypot… ystem that is intelligent
it is necess… ons grounded in the
physical w… s approach is that once
this comm… traditional symbolic
representa… observation is that the
world is its… s exactly up to date. It
always contains every detail there is to be known. **The trick is to sense it appropriately and often enough.**

New Problem

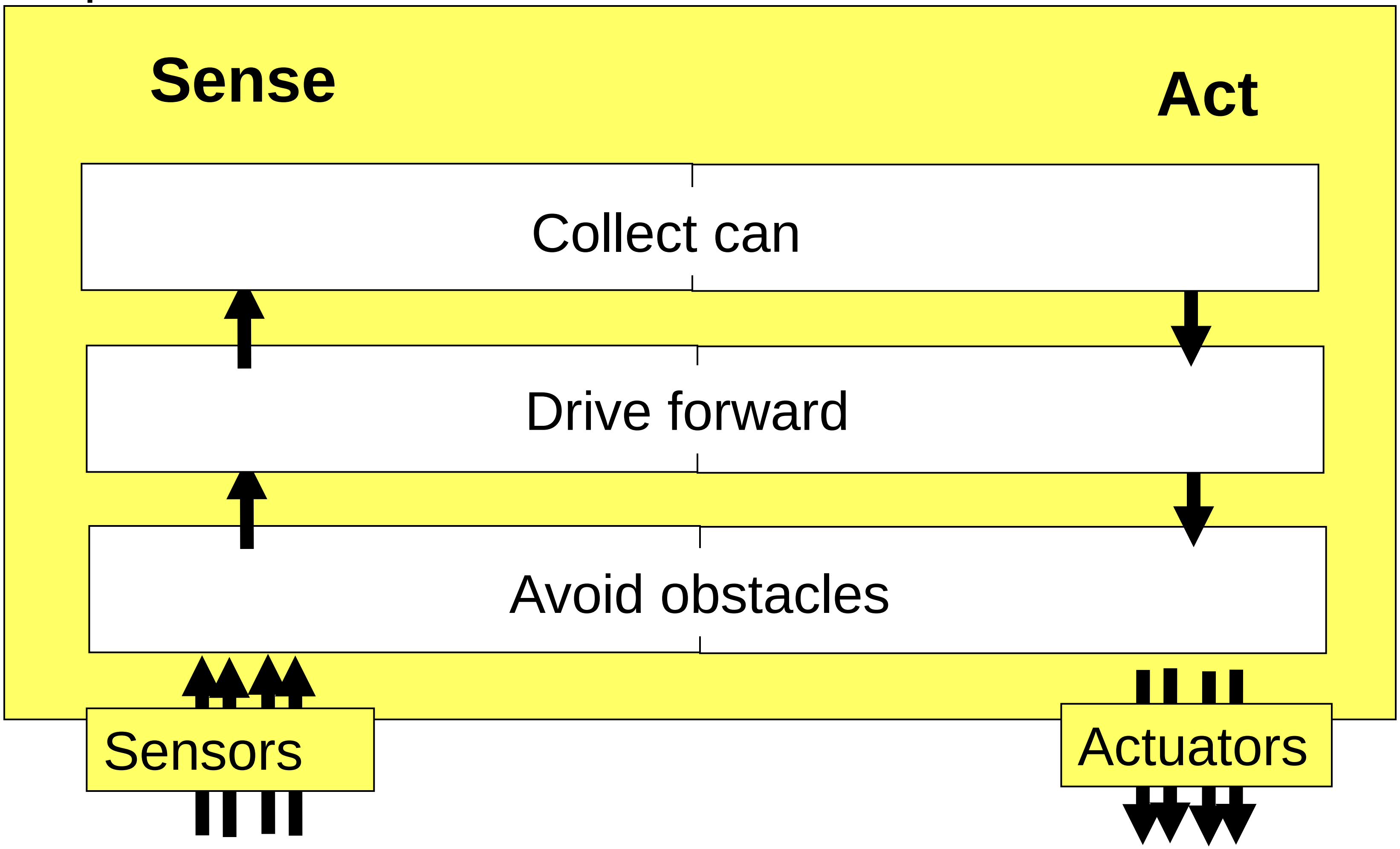
To build a system based on the … s
it is necessary to connect it to t…
and actuators. Typed input and…
interest. They are not physically…

R.A. Brooks: Elephants Don´t Play Chess

But: To bring the Beer from the basement, the robot should have an idea about the location etc...

Subsumption Architecture (Brooks):

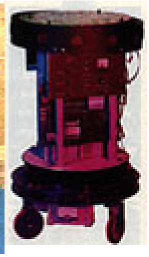
Example:



Subsumption Architecture (Brooks)

- Behaviors realized by simple AFSM (augmented finite state machines)
- No other internal modelling
- Layers: Hierarchical collection of behaviors
- Parallel control by all layers
- In case of conflicts:
  higher layer overwrights („subsumes“) other layers

First successful robot designs for simple tasks.

Problems with too many behaviors:

Design and prediction of resulting behavior?

Consequence: Different Approaches Needed

Reactive Behavior:

like Stimulus-Response: short term

„*simple*“ behavior patterns, simple skills

Deliberative Behavior

Goal directed, plan based behavior: long term

„*complex*“ behavior

Hybrid:

Combination of reactive and deliberative behavior

e.g. goal driven usage of reactive skills

In robotics up to now:
More emphasis put to aspects of low level control.
Recently:
Increasing interest in high level control.