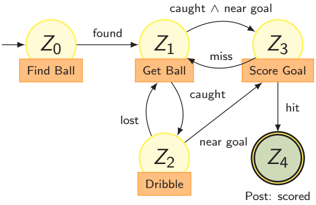
Mutual Goal Agreement for Multi-Agent Systems

Under Adverse Conditions

Andreas Witsch

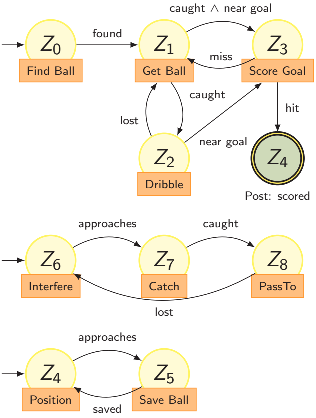
Distributed Systems Group
Kassel University

Team Behavior Modelling



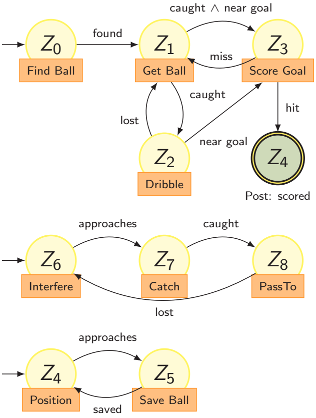
- A plan for a single agent can easily be descriped as finite state machine.
- Each state requires the Agent to execute a certain behaviour.

Team Behavior Modelling



- A plan for a single agent can easily be descriped as finite state machine.
- Each state requires the Agent to execute a certain behaviour.
- Multiple agents ⇒ mutiple state machines.

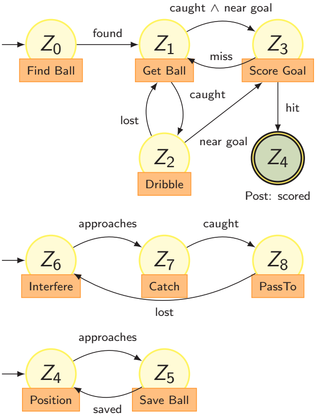
Team Behavior Modelling



Overall Questions:

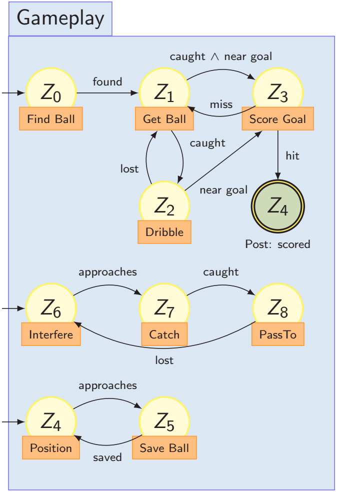
- How to coordinate transitions?
- How to assign agents to the state machines?

Assignment



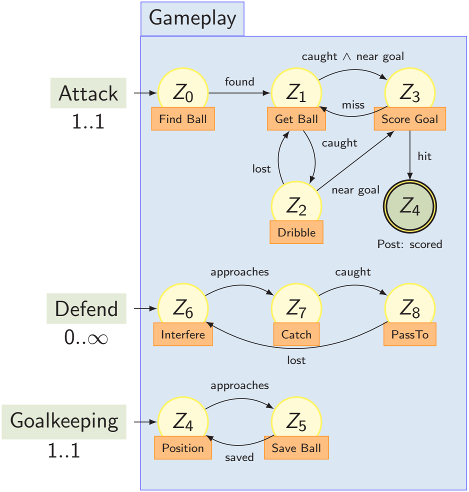
Assignment of robots to the statemachines (SM)

Assignment



Assignment of robots to the statemachines (SM)

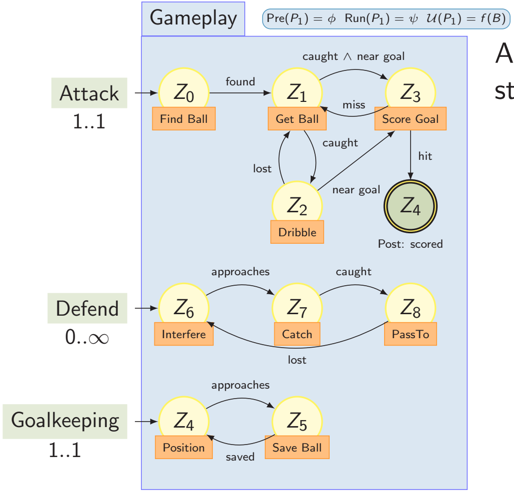
Assignment



Assignment of robots to the statemachines (SM)

- Annotation of each SM with a *Task* and a *Cardinality*,

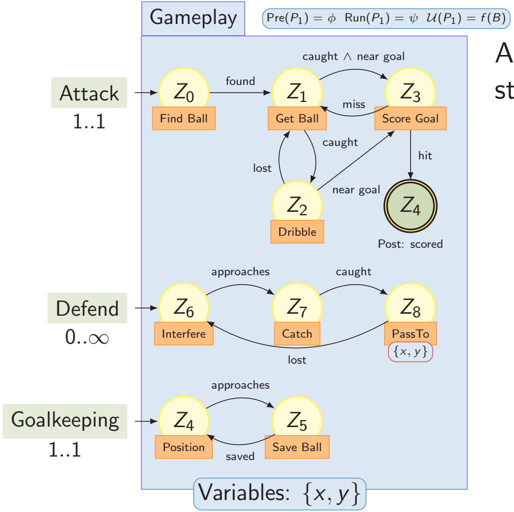
Assignment



Assignment of robots to the statemachines (SM)

- Annotation of each SM with a *Task* and a *Cardinality*,
- Plans have a *Precondition*, *Runtimecondition*, and a *Utility Function*.

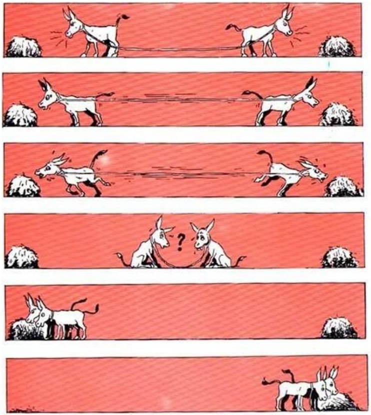
Assignment



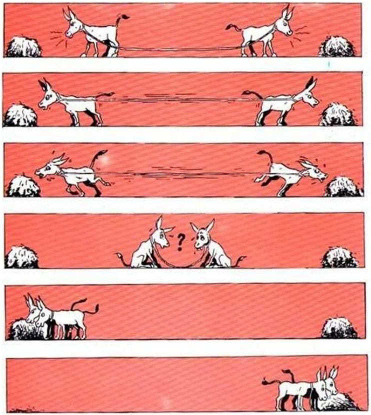
Assignment of robots to the statemachines (SM)

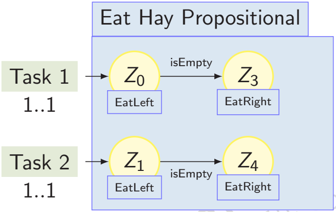
- Annotation of each SM with a *Task* and a *Cardinality*,
- Plans have a *Precondition*, *Runtimecondition*, and a *Utility Function*.
- Plan variables can be queried.

Why do we need agreement on mutual goals?

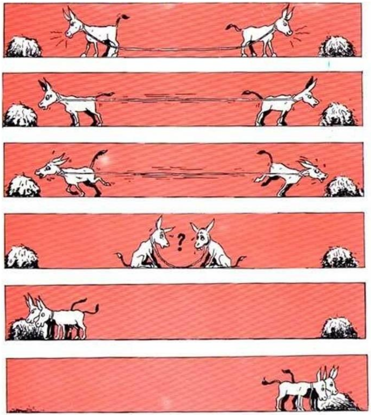


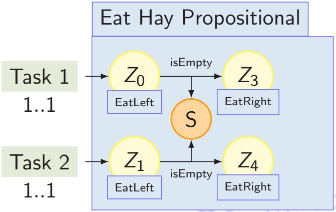
Why do we need agreement on mutual goals?



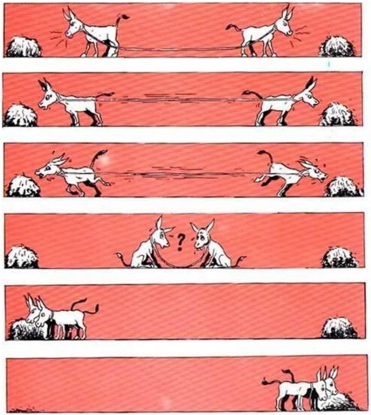


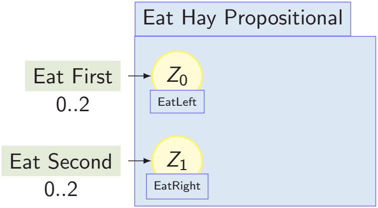
Why do we need agreement on mutual goals?



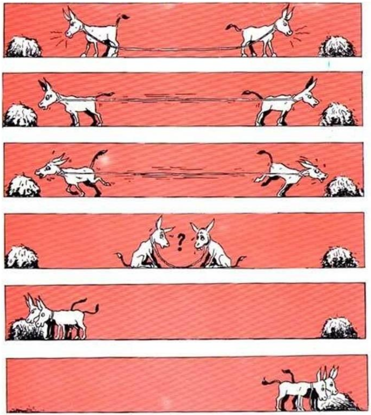


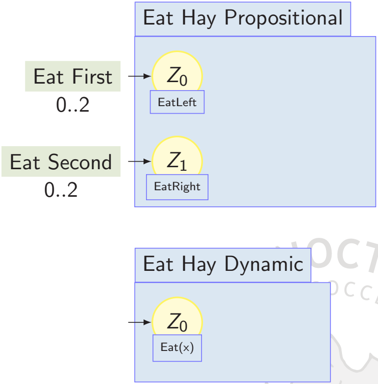
Why do we need agreement on mutual goals?





Why do we need agreement on mutual goals?





Problems with Plan Variables

- Sensor data can cause conflicts
  - $\Rightarrow$ No conflict resolution

Problems with Plan Variables

- Sensor data can cause conflicts
  - $\Rightarrow$ No conflict resolution
- Whole team decides about values
  - $\Rightarrow$ How can a single agent make decisions?

Problems with Plan Variables

- Sensor data can cause conflicts
  - ⇒ No conflict resolution
- Whole team decides about values
  - ⇒ How can a single agent make decisions?
- Only numeric datatypes are supported
  - ⇒ No support for complex datatypes e.g. lists

Problems with Plan Variables

- Sensor data can cause conflicts
  - ⇒ No conflict resolution
- Whole team decides about values
  - ⇒ How can a single agent make decisions?
- Only numeric datatypes are supported
  - ⇒ No support for complex datatypes e.g. lists
- Values are determined each iteration
  - ⇒ Persistency is not possible

Problems with Plan Variables

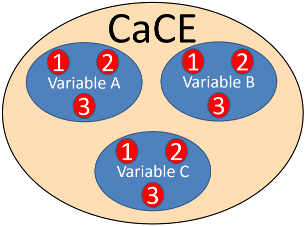
- Sensor data can cause conflicts
  - ⇒ No conflict resolution
- Whole team decides about values
  - ⇒ How can a single agent make decisions?
- Only numeric datatypes are supported
  - ⇒ No support for complex datatypes e.g. lists
- Values are determined each iteration
  - ⇒ Persistency is not possible

We need a agreement for variable values.

Carpe Noctem Consensus Engine

CaCE

CaCE Overview



What is CaCE?

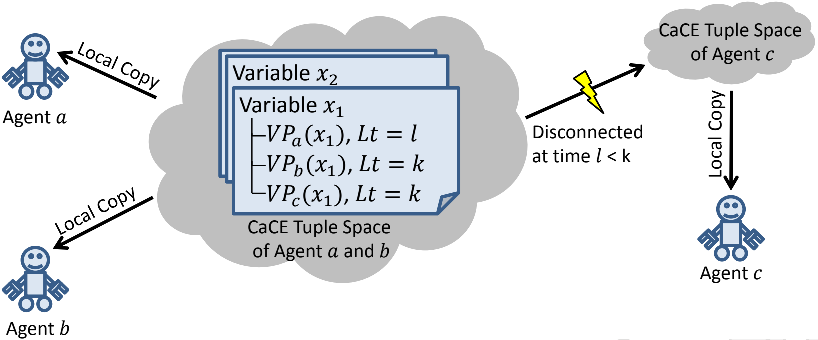
- A distributed tuple space with replication
- An approach to achieve coordination for ALICA variables

Requirements:

- Communication is transient and unreliable
- Conflicts can occur, but we have to deal with them
- Support of persistency and complex datatypes

DISTRIBUTED SYSTEMS

Data Distribution in CaCE



Steps to Achieve Agreement

- Consistency Strategy
  - ⇒ How to distribute value proposals

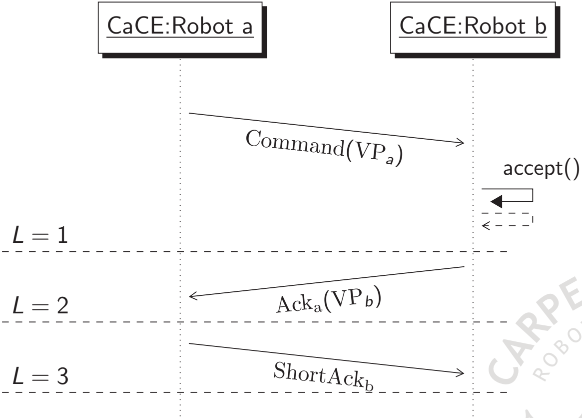
Steps to Achieve Agreement

- Consistency Strategy
  - ⇒ How to distribute value proposals
- Acceptance Strategy
  - ⇒ When to accept a proposal as own proposal

Steps to Achieve Agreement

- Consistency Strategy
  - ⇒ How to distribute value proposals
- Acceptance Strategy
  - ⇒ When to accept a proposal as own proposal
- Value Decission Strategy
  - ⇒ Determine the current value from the known proposals

CaCE Consistency Levels



Acceptance Strategy: Believe Ordering

A conflict resolution strategy is a function C, which decides for the highest ranked input belief for a Variable x:

$$\begin{aligned} \mathrm{VP}_a(x) &= C(\mathrm{VP}_a(x), \mathrm{VP}_b(x), ..., \mathrm{VP}_n(x)) \\ &= \max(\mathrm{VP}(x) \in <_o) \end{aligned}$$

$\Rightarrow$ C choses the maximum w.r.t. a strict ordering relation $<_o$ with:

$$<_o \subseteq \mathrm{VP}(x) \times \mathrm{VP}(x)$$

Acceptance Strategy: Believe Ordering

A conflict resolution strategy is a function C, which decides for the highest ranked input belief for a Variable x:

$$\begin{aligned} \mathrm{VP}_a(x) &= C(\mathrm{VP}_a(x), \mathrm{VP}_b(x), ..., \mathrm{VP}_n(x)) \\ &= \max(\mathrm{VP}(x) \in <_o) \end{aligned}$$

$\Rightarrow$ C choses the maximum w.r.t. a strict ordering relation $<_o$ with:

$$<_o \subseteq \mathrm{VP}(x) \times \mathrm{VP}(x)$$

Example

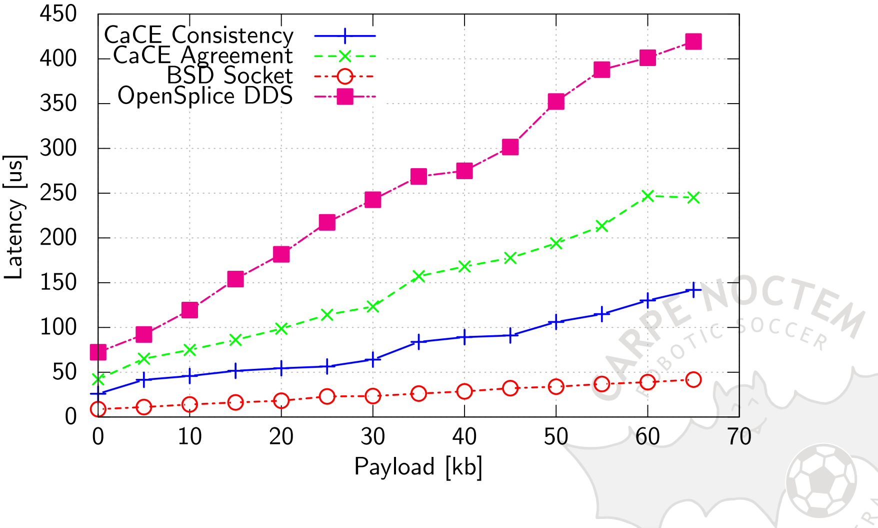
The default conflict resolution order beliefs by the Lamport time $<_{Lt} = \{(\mathrm{VP}_1(x), \mathrm{VP}_2(x)) \mid Lt(\mathrm{VP}_1(x)) < Lt(\mathrm{VP}_2(x))\}$

Value Decission

Most Common Decission Strategies

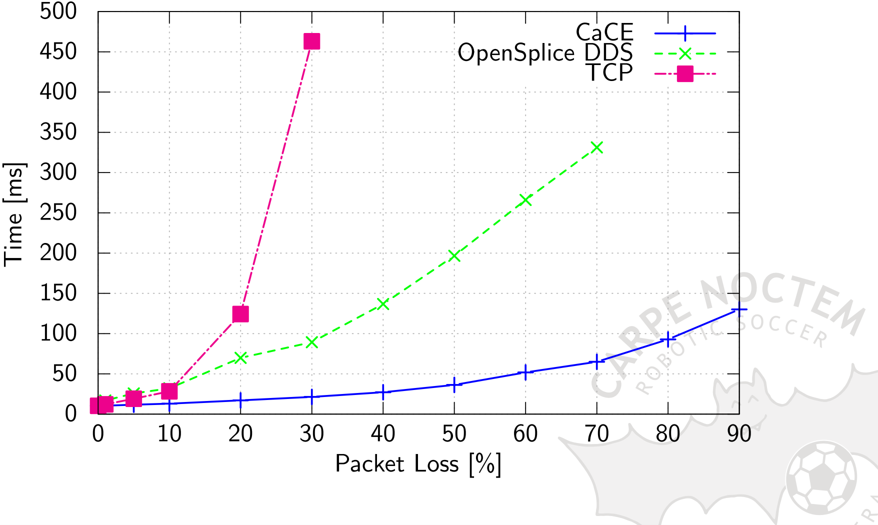
- Most recent proposal
- Majority voting
- Full agreement

Latency


CaCE Consistency
CaCE Agreement
BSD Socket
OpenSplice DDS
Latency [us]
Payload [kb]
0
50
100
150
200
250
300
350
400
450
0
10
20
30
40
50
60
70
CARPE NOCTEM
ROBOTIC SOCCER

DISTRIBUTED SYSTEMS

Transmission Time



Thank you for your attention