Coordination in multi-agent autonomous cognitive robotic systems

Corso di Dottorato in Robotica Cognitiva

Daniele Nardi

Dipartimento di Informatica e Sistemistica,
Università di Roma "La Sapienza"

Maggio 2002

work with: Claudio Castelpietra, Giuseppe De Giacomo, Luca Iocchi, Riccardo Rosati

Programma delle lezioni

1. Cognitive Soccer Players (SPQR)

2. Coordination in Multi-Robot Systems

SPQR research goal

Experiment the **Cognitive Robotics** approach in the design of Cognitive Football players in the RoboCup environment.

Summary

1. Action Theory
2. Plan Generation
3. Architecture
4. Plan Execution
5. Implementation

Dynamic System Representation

- A *state* of the world is an element of the interpretation domain.

- Each *state* of the world is labeled by a set of concepts, correponding to the properties that hold in that state.

- The execution of actions is modeled through *action-roles*, i.e., binary relations between states.

Propositional Dynamic Logics

PDLs (Propositional dynamic logics) are modal logics originally developed for describing and reasoning about program schemas.

Examples:

$$C \Rightarrow [R]D$$

$$\langle R \rangle \top$$

$$\langle R^* \rangle G$$

Rosenschein 81 gives the basis for representing actions and deductive planning in PDLs.

Relation with Situation Calculus

Situation Calculus	PDLs	
$Poss(a, s)$	$\langle a \rangle \top$	
$Poss(a, s) \Rightarrow \Phi(do(a, s))$	$[a]\Phi$	
$Poss(a, s) \equiv \Phi$	$\langle a \rangle \top \equiv \Phi$	(preconditions)
$Poss(a, s) \land \Phi_1(s) \Rightarrow \Phi(do(a, s))$	$\Phi_1 \Rightarrow [a]\Phi_2$	(effects)

- actions in PDL are **deterministic**

- **propositional fluents** of Situation Calculus

The $\mathcal{ALCK}_{\mathcal{NF}}$ Representation Language

- PDL-DL correspondence [*De Giacomo*, 94],
- Epistemic Description Logics [*Donini et al.*, 97]

Concepts: denote the properties of a set of *individuals*
Roles: model relationships between *individuals*

$$C ::= \top \mid \bot \mid A \mid C_1 \sqcap C_2 \mid C_1 \sqcup C_2 \mid \neg C \mid \exists R.C \mid \forall R.C \mid \mathbf{K}C \mid \mathbf{A}C$$
$$R ::= P \mid P_1 \sqcap \ldots \sqcap P_n \mid \mathbf{K}R \mid \mathbf{A}R$$

State	Individual	$init$
State Description	Concept	C
Action	Role	R
Action Description	Axioms	...
State Constraints	Axioms	...

Transition Graphs

The interpretation structures of PDLs (DLs) are **transition graphs**:

- Each node represents a state, and is labeled by the properties of that state.
- Each arc represents a transition and is labeled by the action the causes the transition.

Each transition graph is a *complete* representation of the behavior of the system. A model of the KB representing the action theory can thus be viewed as a transition graph.

Semantics of $\mathcal{ALCK_{NF}}$

Constructs	PDLs	$\mathcal{ALCK_{NF}}$	$\mathcal{ALCK_{NF}}$ Semantics
Atomic concept/proposition	A	A	$A^{\mathcal{I}} \subseteq \Delta$
Atomic role/action	R	R	$R^{\mathcal{I}} \subseteq \Delta \times \Delta$
Named individual	—	s	$s^{\mathcal{I}} \in \Delta$
True	`tt`	$\top$	Δ
False	`ff`	$\bot$	$\emptyset$
Conjunction	$C \wedge D$	$C \sqcap D$	$C^{\mathcal{I}} \cap D^{\mathcal{I}}$
Disjunction	$C \vee D$	$C \sqcup D$	$C^{\mathcal{I}} \cup D^{\mathcal{I}}$
Negation	$\neg C$	$\neg C$	$\Delta \setminus C^{\mathcal{I}}$
Universal quantification	$[R]C$	$\forall R.C$	$\{d \in \Delta \mid \forall d'.\,(d,d') \in R^{\mathcal{I}} \Rightarrow d' \in C^{\mathcal{I}}\}$
Existential quantification	$\langle R \rangle C$	$\exists R.C$	$\{d \in \Delta \mid \exists d'.\,(d,d') \in R^{\mathcal{I}} \wedge d' \in C^{\mathcal{I}}\}$
Inclusion assertion	$C \Rightarrow D$	$C \sqsubseteq D$	$C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ for every $\mathcal{I}$
Instance assertion	—	$C(s)$ $R(s_1, s_2)$	$s^{\mathcal{I}} \in C^{\mathcal{I}}$ $(s_1^{\mathcal{I}}, s_2^{\mathcal{I}}) \in R^{\mathcal{I}}$ for every $\mathcal{I}$

Dynamic System Representation

- Initial situation assertions (Γ_I): *"C holds in the initial situation"*.

$$C(init)$$

- State constraints (Γ_S): *"C implies D in every situation"*.

$$C \sqsubseteq D$$

- Action preconditions (Γ_P): *"If C is known to be true in the current situation, then it is possible to perform action R_M"*.

$$\mathbf{K}C \sqsubseteq \exists \mathbf{K} R_M . \top$$

- Effect axioms (Γ_E): *"If C is known to be true in the current situation, then in the resulting situation D is known to be true"*.

$$\mathbf{K}C \sqsubseteq \forall R_M . D$$

Features of the $\mathcal{ALCK}_{\mathcal{NF}}$ approach

- **Epistemic abilities**: explicit representation of the robot's epistemic state and introduction of sensing (knowledge-producing) actions.

- **Concurrency of primitive actions**: reasoning about concurrent execution of actions without an explicit introduction of new actions.

- **Persistence and exogenous events**: formalization of both deterministic frame axioms and default frame axioms.

- **State and epistemic constraints**: describing relationships between dynamic properties that enforce ramification.

Sensing actions

- Action preconditions (Γ_P): *"If C is known to be true in the current situation, then it is possible to perform the sensing action R_S"*.

$$\mathbf{K}C \sqsubseteq \exists \mathbf{K}R_S.\top$$

- Effect axioms (Γ_E): *"The sensing action R_S leads to a new situation in which either S or $\neg S$ is known"*.

$$\top \sqsubseteq \mathbf{K}(\forall R_S.S) \sqcup \mathbf{K}(\forall R_S.\neg S),$$

Concurrent actions

- Precondition axiom schema (Γ_P):

$$\exists\mathbf{K}R_1.\top \sqcap \exists\mathbf{K}R_2.\top \sqcap \neg\mathbf{A}(\forall R_1 \sqcap R_2.\bot) \sqsubseteq \exists\mathbf{K}(R_1 \sqcap R_2).\top$$

Two actions R_1, R_2 can be concurrently executed in a state s if and only if the following conditions hold:

1. both R_1 and R_2 can be executed in s;
2. the effects of R_1 and R_2 are mutually consistent.

Frame axioms

- Default frame axioms (Γ_{DFR}): *"If in the current state the property C holds, then, after the execution of the action R, the property C holds, if it is consistent with the effects of R."*.

$$\mathbf{K}C \sqsubseteq \forall \mathbf{K}R.\mathbf{A}\neg C \sqcup \mathbf{K}C$$

- Epistemic frame axioms (Γ_{EFR}): *"The property C is propagated only if property D holds in the successor state"*.

$$\mathbf{K}C \sqsubseteq \forall \mathbf{K}R.\neg \mathbf{K}D \sqcup \mathbf{K}C$$

Example KB for defensive actions

$\mathbf{K}(BallClose \sqcap OpponentOnBall) \sqsubseteq \exists\mathbf{K}tackle.\top$
$\mathbf{K}(\neg BallClose \sqcap OpponentOnBall) \sqsubseteq \exists\mathbf{K}intercept.\top$
$\mathbf{K}(BallClose \sqcap \neg OpponentOnBall) \sqsubseteq \exists\mathbf{K}kick.\top$
$\mathbf{K}(\neg BallClose \sqcap \neg OpponentOnBall) \sqsubseteq \exists\mathbf{K}goToBall.\top$

$\mathbf{K}\top \sqsubseteq \forall tackle.GoalProtected$
$\mathbf{K}\top \sqsubseteq \forall intercept.GoalProtected$
$\mathbf{K}\top \sqsubseteq \forall kick.(GoalProtected \sqcap \neg BallClose)$
$\mathbf{K}\top \sqsubseteq \forall goToBall.(GoalProtected \sqcap BallClose)$

$\mathbf{K}BallInLPS \sqsubseteq \exists\mathbf{K}senseBallClose.\top$
$\mathbf{K}\top \sqsubseteq \mathbf{K}(\forall senseBallClose.BallClose) \sqcup$
$\quad \mathbf{K}(\forall senseBallClose.\neg BallClose)$
$\mathbf{K}BallInLPS \sqsubseteq \exists\mathbf{K}senseOpponentOnBall.\top$
$\mathbf{K}\top \sqsubseteq \mathbf{K}(\forall senseOpponentOnBall.OpponentOnBall) \sqcup$
$\quad \mathbf{K}(\forall senseOpponentOnBall.\neg OpponentOnBall)$

$BallInLPS(init)$

Deductive Planning

The reasoning of interest is the following **logical implication**:

$$\Gamma \models S \Rightarrow \langle \alpha^* \rangle G$$

- Γ are axioms that represent the system.
- S is a formula denoting a partial description of the initial state.
- $\langle \alpha^* \rangle G$ is a formula expressing the reachability of a state where the goal G holds.

From a constructive proof, one can extract the **plan**.

Planning Problem in $\mathcal{ALCK}_{\mathcal{NF}}$

Deductive planning in $\mathcal{ALCK}_{\mathcal{NF}}$ is phrased as:

$$\Sigma \models C_G(init).$$

(i) Σ is the action theory including static, dynamic and frame axioms.

(ii) $C_G(init)$ is any concept belonging to the set $\mathcal{P}_C$ defined inductively as:

1. $\mathbf{K}G \in \mathcal{P}_\mathcal{C}$;
2. if $C_1, \ldots, C_m \in \mathcal{P}_C$, then $\exists\mathbf{K}(R_{M_1} \parallel \ldots \parallel R_{M_k} \parallel R_{S_1} \parallel \ldots \parallel R_{S_l}).(\mathbf{K}S_1 \sqcap \ldots \sqcap \mathbf{K}S_l \sqcap C_1) \sqcup \ldots \sqcup (\mathbf{K}\neg S_1 \sqcap \ldots \sqcap \mathbf{K}\neg S_l \sqcap C_m) \in \mathcal{P}_\mathcal{C}$ for each $0 \leq k, l \leq n$, where $m = 2^{k+l}$, each R_{M_i} is an ordinary action and each R_{S_i} is a sensing action for the property S_i.

Examples of plans

$$\exists\mathbf{K}(R_{M_1} \parallel R_{M_2}).\mathbf{K}G$$

$$\exists\mathbf{K}(R_{S_1} \parallel R_{S_2}).(\mathbf{K}S_1 \sqcap \mathbf{K}S_2 \sqcap \exists\mathbf{K}R_{M_1}.\mathbf{K}G) \sqcup (\mathbf{K}S_1 \sqcap \mathbf{K}\neg S_2 \sqcap \exists\mathbf{K}R_{M_2}.\mathbf{K}G) \sqcup (\mathbf{K}\neg S_1 \sqcap \mathbf{K}S_2 \sqcap \exists\mathbf{K}R_{M_3}.\mathbf{K}G) \sqcup (\mathbf{K}\neg S_1 \sqcap \mathbf{K}\neg S_2 \sqcap \exists\mathbf{K}R_{M_4}.\mathbf{K}G)$$

Partial Transition Graphs

A **partial transition graph** is a transition graph in which

- Only part of the possible states are represented.
- The represented states are in fact only partially represented (their properties are only partially known).
- Only part of the possible transitions are represented.

A **partial transition graph** summarizes all common features of all possible transition graphs satisfying the axioms.

First Order Extension

The **FOE** of an epistemic knowledge base is a *partial* transition graph built by **applying** dynamic axioms through a forward propagation algorithm.

The FOE provides a unique characterization of all knowledge shared by all the models of Σ in a **non-epistemic knowledge base**.

Note: During FOE Calculus each sensing action R_S is replaced by two special actions R_S^+ and R_S^-, with the effect axiom for R_S is replaced by the following effect axioms:

$$\top \sqsubseteq \forall R_S^+ . S \qquad \top \sqsubseteq \forall R_S^- . \neg S$$

Reasoning tasks

- Given a dynamic system specification in $\mathcal{ALCK}_{\mathcal{NF}}$ and a goal expressed in terms of a set of concepts, we are able to express the **plan generation problem** in terms of a reasoning problem in $\mathcal{ALCK}_{\mathcal{NF}}$.

- Given a plan we are able to reduce **verification** of such a plan (i.e., the problem of establishing whether such a plan allows the robot to reach a state where the goal is satisfied) to a deduction problem in $\mathcal{ALCK}_{\mathcal{NF}}$.

Plan generation

- **Build** the FOE (*first-order extension*).

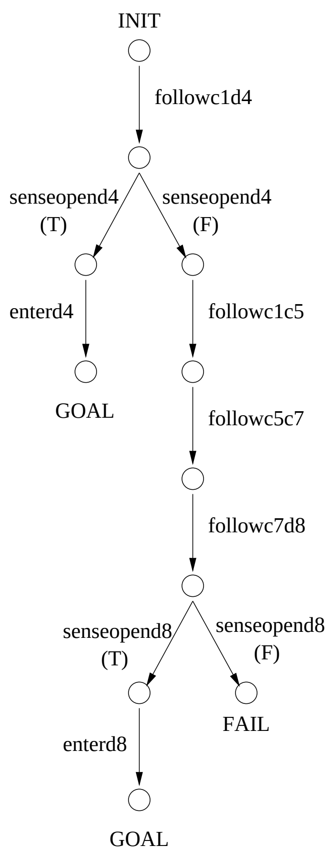
- **Visit** the FOE, searching for a path from the initial state to a set of nodes where G holds.

In the implementation:

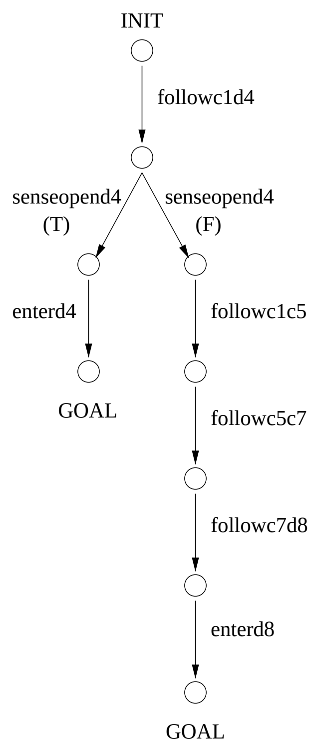
- The construction of the whole FOE is not necessary

- The FOE can be constructed off-line for a specific situation

Extending the notion of plan



a)



b)

Planning defensive actions

$\mathbf{K}(BallClose \sqcap OpponentOnBall) \sqsubseteq \exists \mathbf{K} tackle.\top$
$\mathbf{K}(\neg BallClose \sqcap OpponentOnBall) \sqsubseteq \exists \mathbf{K} intercept.\top$
$\mathbf{K}(BallClose \sqcap \neg OpponentOnBall) \sqsubseteq \exists \mathbf{K} kick.\top$
$\mathbf{K}(\neg BallClose \sqcap \neg OpponentOnBall) \sqsubseteq \exists \mathbf{K} goToBall.\top$

$\mathbf{K}\top \sqsubseteq \forall tackle.GoalProtected$
$\mathbf{K}\top \sqsubseteq \forall intercept.GoalProtected$
$\mathbf{K}\top \sqsubseteq \forall kick.(GoalProtected \sqcap \neg BallClose)$
$\mathbf{K}\top \sqsubseteq \forall goToBall.(GoalProtected \sqcap BallClose)$

$\mathbf{K} BallInLPS \sqsubseteq \exists \mathbf{K} senseBallClose.\top$
$\mathbf{K}\top \sqsubseteq \mathbf{K}(\forall senseBallClose.BallClose) \sqcup$
$\quad \mathbf{K}(\forall senseBallClose.\neg BallClose)$
$\mathbf{K} BallInLPS \sqsubseteq \exists \mathbf{K} senseOpponentOnBall.\top$
$\mathbf{K}\top \sqsubseteq \mathbf{K}(\forall senseOpponentOnBall.OpponentOnBall) \sqcup$
$\quad \mathbf{K}(\forall senseOpponentOnBall.\neg OpponentOnBall)$

$BallInLPS(init)$

Example 1

Goal: $GoalProtected$

Plan generated:

```
senseBallClose || senseOpponentOnBall;
if (BallClose and not OpponentOnBall)
  { kick; }
  else { if (BallClose and OpponentOnBall)
       { tackle; }
       else { if (not BallClose and OpponentOnBall)
            { intercept; }
            else { goToBall; }
```

Planning the pass

$\mathbf{K}(BallPoss_i \sqcap FreeAhead_i) \sqsubseteq \exists\mathbf{K} fwdKeepingBall_i.\top$
$\mathbf{K}(BallPoss_i \sqcap ShootPsn_i \sqcap FreeAhead_i) \sqsubseteq \exists\mathbf{K} kick_i.\top$
$\mathbf{K}(BallPoss_i \sqcap ShootPsn_j \sqcap \neg FreeAhead_i) \sqsubseteq \exists\mathbf{K} pass_i^j.\top$
$\mathbf{K}(BallClose_j \sqcap ShootPsn_j) \sqsubseteq \exists\mathbf{K} receiveAndKick_j.\top$
$\mathbf{K} BallPoss_i \sqsubseteq \exists\mathbf{K} positionForPass_j.\top$

$\mathbf{K}\top \sqsubseteq \forall fwdKeepingBall_i.(BallPoss_i \sqcap ShootPsn_i)$
$\mathbf{K}\top \sqsubseteq \forall kick_i.BallKicked$
$\mathbf{K}\top \sqsubseteq \forall pass_i^j.BallClose_j$
$\mathbf{K}\top \sqsubseteq \forall receiveAndKick_i.BallKicked$
$\mathbf{K}\top \sqsubseteq \forall positionForPass_i.ShootPsn_i$

$\mathbf{K}\top \sqsubseteq \exists\mathbf{K} senseFreeAhead_i.\top$
$\mathbf{K}\top \sqsubseteq \mathbf{K}(\forall senseFreeAhead_i.FreeAhead_i) \sqcup$
$\quad \mathbf{K}(\forall senseFreeAhead_i.\neg FreeAhead)$

$\mathbf{K} ShootPsn_i \sqsubseteq \forall\mathbf{K} pass_j^i.\mathbf{A}\neg ShootPsn_i \sqcup \mathbf{K} ShootPsn_i$
$\ldots$

$(BallPoss_1 \sqcap FreeAhead_1)(init)$

Example 2

Goal: $BallKicked$

Plan generated:

```
senseFreeAhead1 || fwdKeepingBall1 || positionForPass2;
if (FreeAhead1)
   { kick1 }
   else { pass12; receiveAndKick2;}
```

Limitations of Plans

1. plan structure is a tree

2. actions duration is not considered

We can define a generalization of the plan structure, which allows to capture a simple form of cycles.

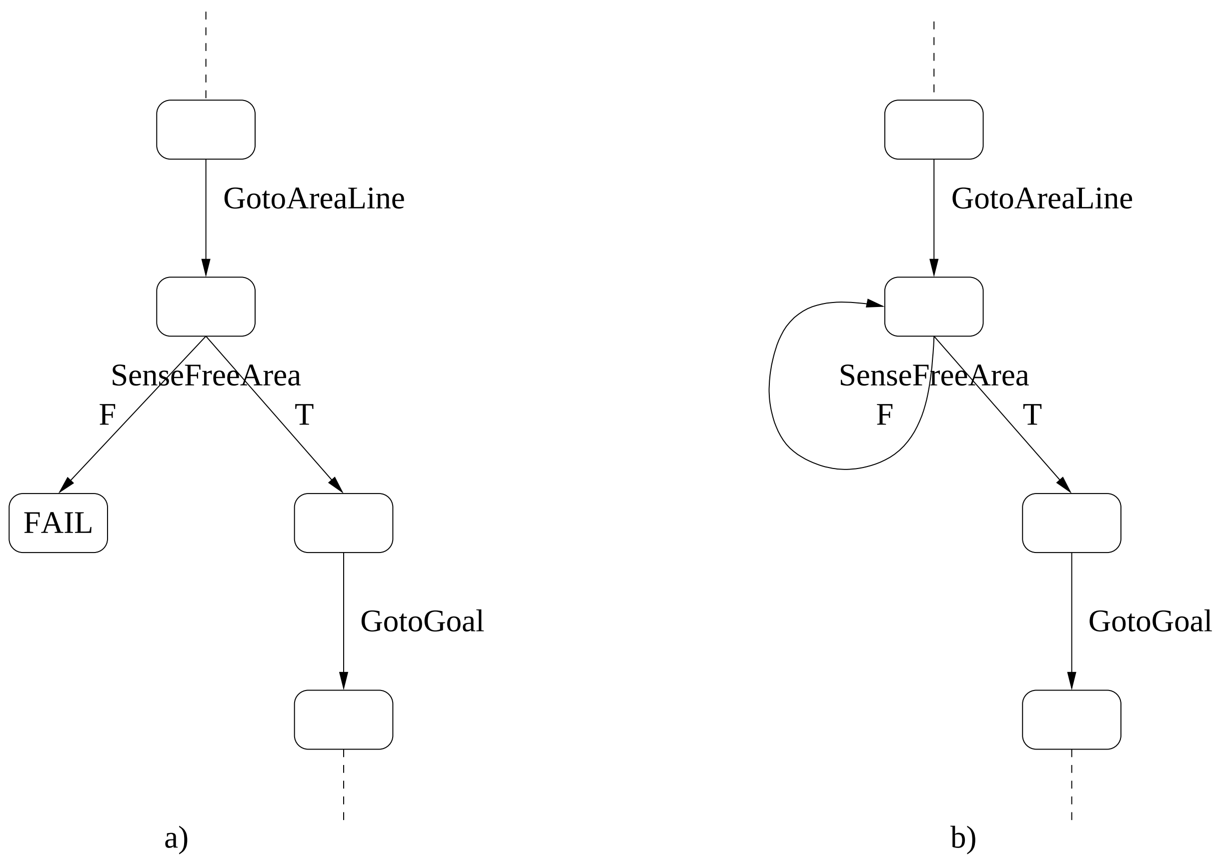
The issue of non instantaneous actions is solved by the executor (outside the action theory), by providing additional specification for the execution of actions.

Generalization of plan structure

- A *strong plan for* G is a conditional plan $\mathcal{S}_G$ such that the execution of $\mathcal{S}_G$ leads to a state in which G is known to hold *for each possible outcome of the sensing actions in* S_G.
- A *weak plan for* G is a conditional plan $\mathcal{S}_G$ such that *there exists an outcome of the sensing actions in* $\mathcal{S}_G$ for which the execution of $\mathcal{S}_G$ leads to a state in which G is known to hold.

We call *partially strong plan for* G a plan $\mathcal{S}_G$ such that, *for each possible outcome of the sensing actions in* $\mathcal{S}_G$, if the execution of $\mathcal{S}_G$ terminates, then it leads to a state in which G is known to hold.

Example of partially strong plan



Plan Execution

A plan is a transition graph, where each node denotes a state, and is labeled with the properties that characterize the state, and each arc denotes a state transition and is labeled with the action that causes the transition.

A *Plan Execution Monitor* is in charge of the correct execution of the actions composing the plans.

The monitor's task is that of visiting the graph, calling and suspending the actions as necessary.

Executable Plans

Actions are not instantaneous, thus preconditions and effects can be interpreted in different ways by the plan's executor.

The actual execution of the plan requires an additional (extra-logical) specification on how to execute the actions.

Checking preconditions

- Preconditions that must be constantly verified during the execution of the action (*NearBall* in a *PushBall* action)

- Preconditions that need to be checked only for the action's activation (*NearBall* in a *Kicking* action)

If the condition becomes false during the execution of the action, an exception occurs and the action fails.

Checking effects

- Effects that determine the action's termination and the state transition (*NearBall* in a *GoToBall* action)

- Effects that are side effects of the action but do not cause the state transition (*NearGoalArea* in a *GoDefense* action)

Executable plan

An executable plan is the result of a two-step procedure.

1. Designing a plan: this plan can be the result of the automatic generation process.

2. Modifying the plan resulting from the first step in order to have it executed by the monitor.

Plan selection

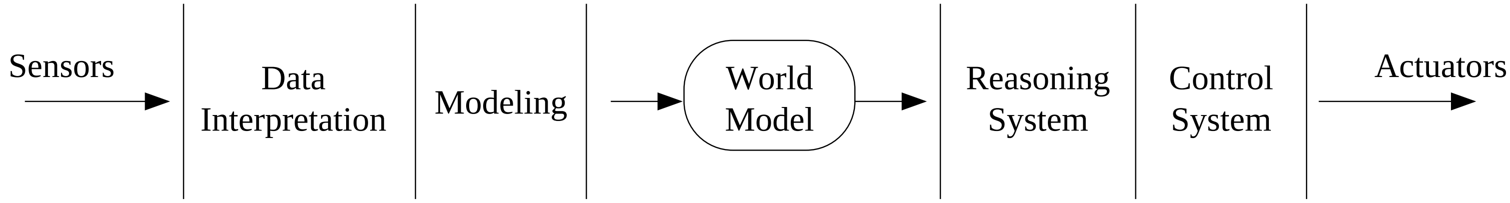
Given a set of initial situations one can generate a *library* of plans.

A plan selector allows the monitor to choose the current plan to be executed

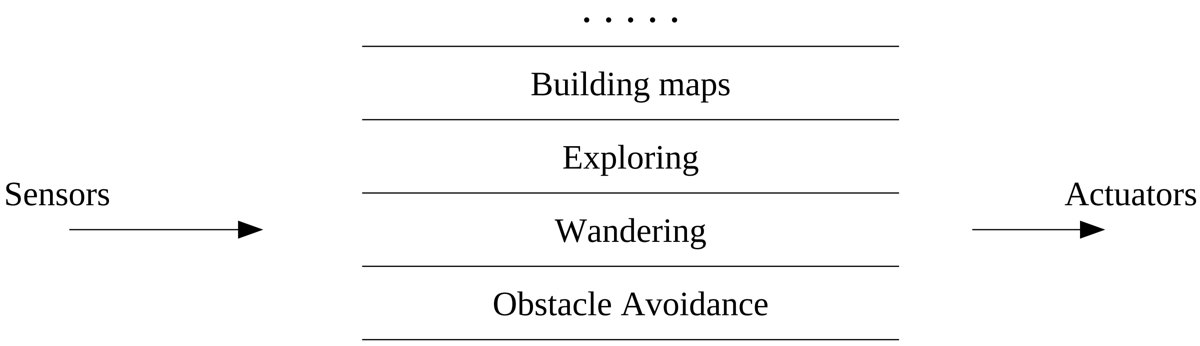
Examples:

- penalty kick
- rising from a fall

Architecture review

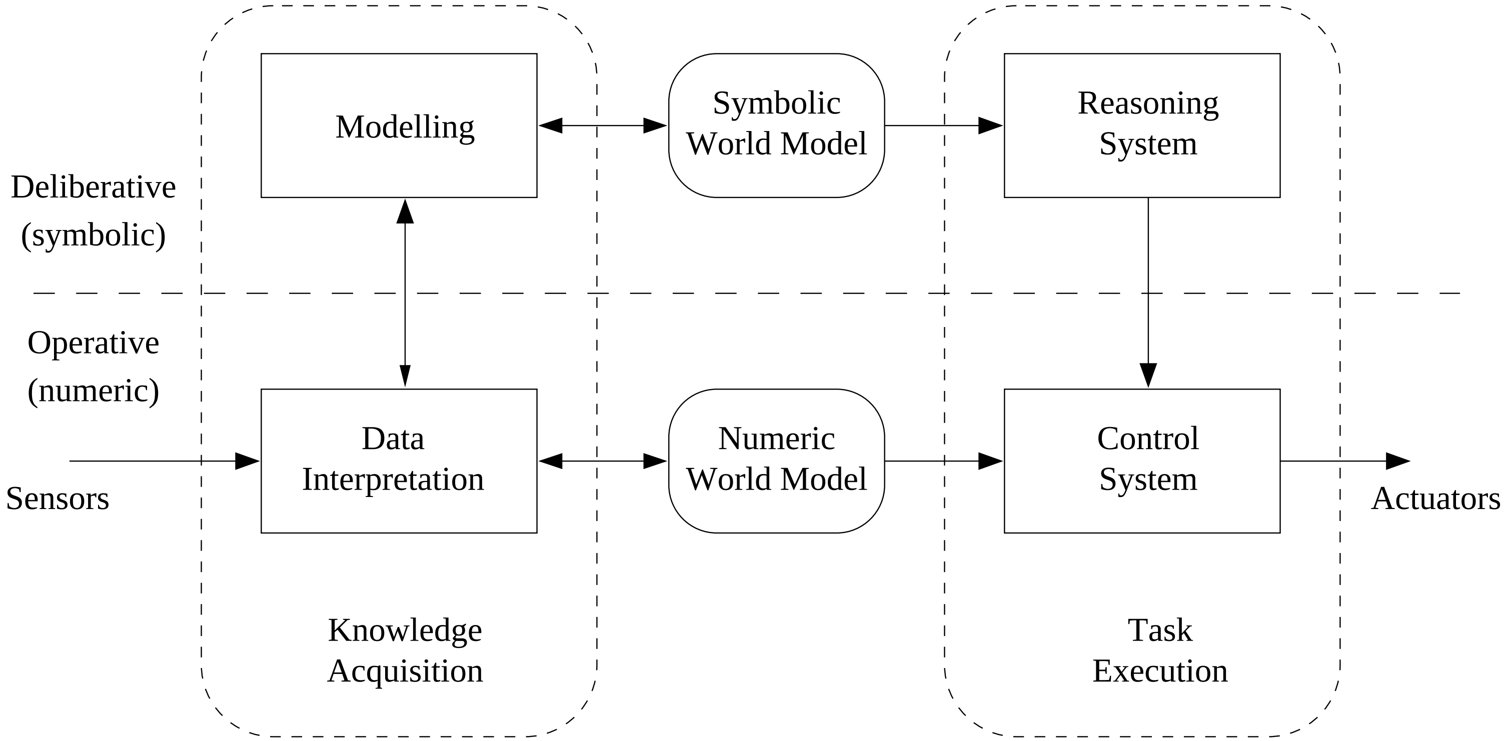


a)



b)

Hybrid Architectures



Requirements for the SPQR Architecture

Integration of reasoning and reactivity through:

- *heterogeneity*
- *asynchronism*

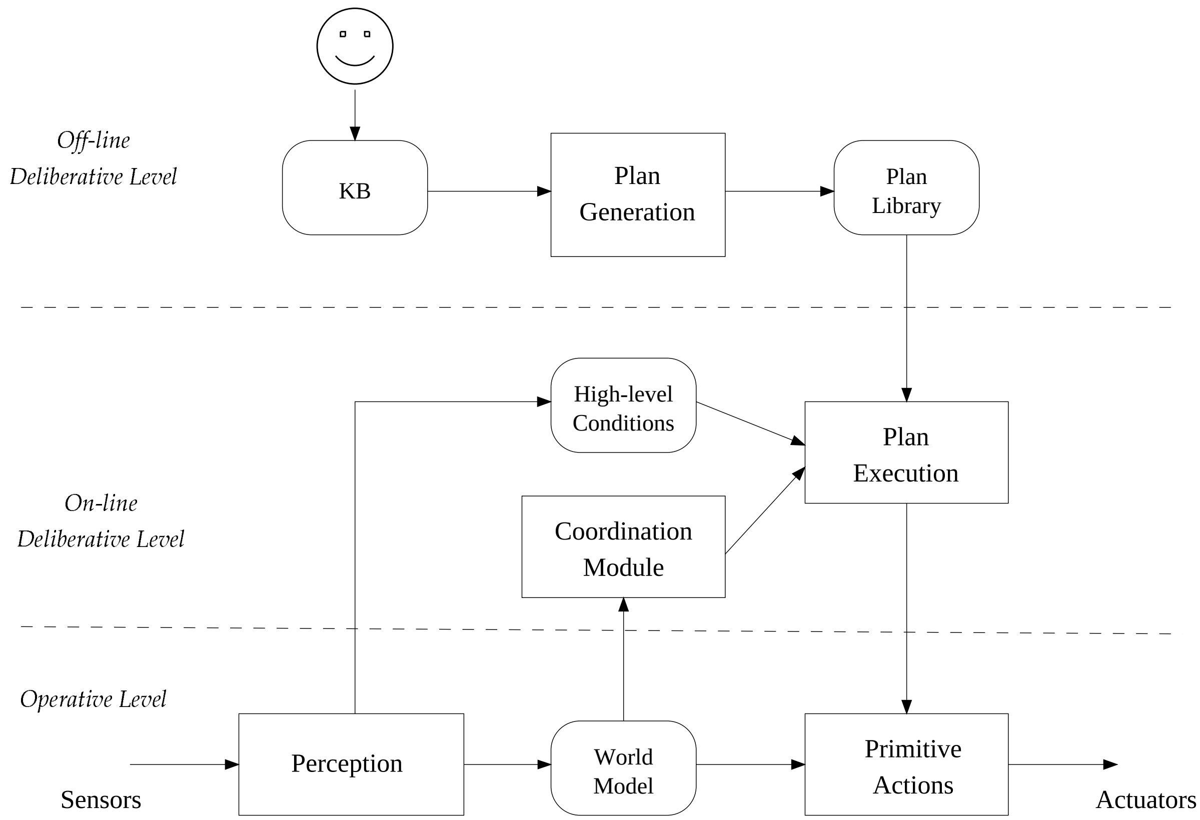
The deliberative level is the same for all our robotic platforms, while the operative level depends on the robotic platform.

Deliberative level

- A plan execution module running *on-line* during the accomplishment of the robot's task and is responsible for coordinating the primitive actions of a single robot;

- A reasoning module, running *off-line* before the beginning of the robot's mission, and generates a set of plans to be used to deal with some specific situations.

SPQR Architecture

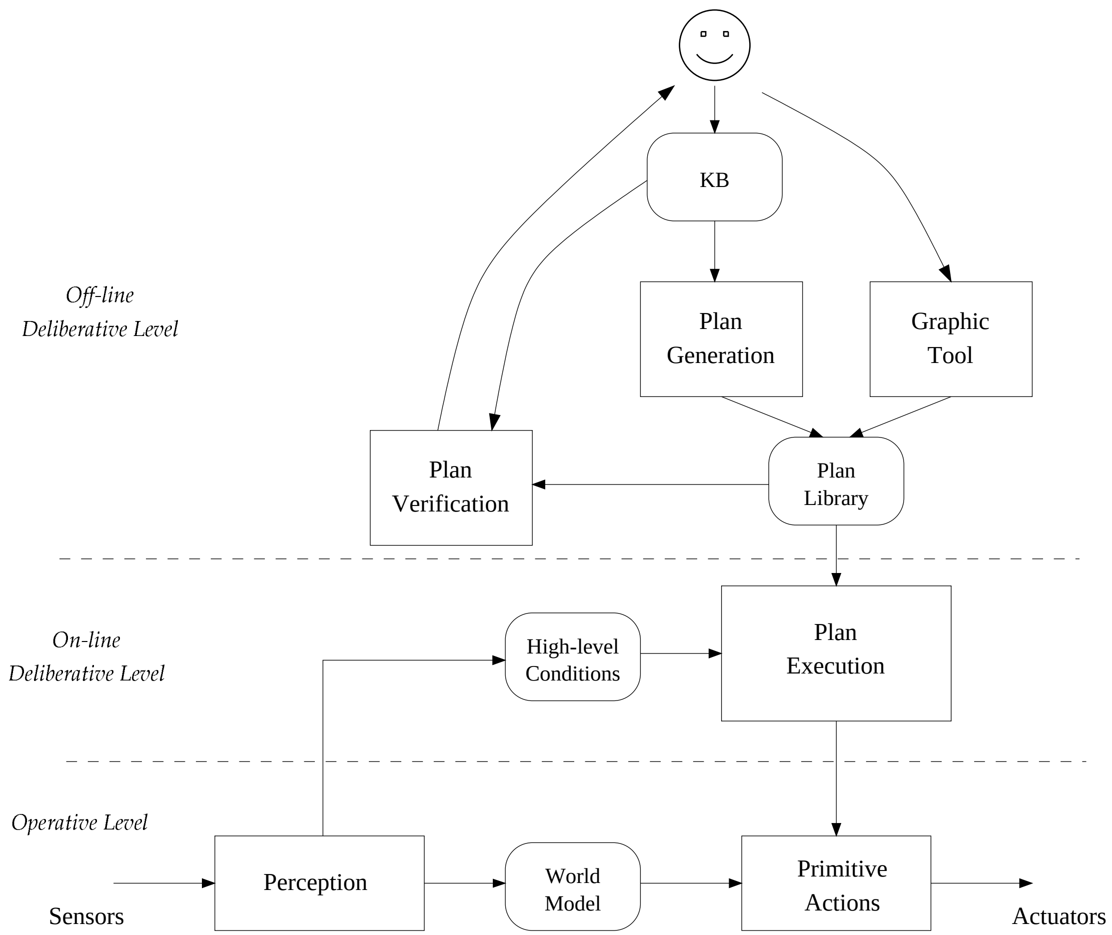


Integration of plans in the robot control system

- High-level robot control is driven by a plan execution program and coordination among robots is obtained by explicit communication.

- Plans are obtained *off-line* by the planner from a declarative specification of the environment.

SPQR Mixed Architecture



Communication-based Coordination

The effects of the *communication* among agents are embedded into the state and thus verified through high-level conditions.

1. *Dynamic role assignment* can be viewed as a goal selection process

2. *Constraints on shared resources* correspond to additional pre-conditions on the actions to be executed.

Design steps

1. Define the primitive control actions

2. Define the high level-conditions

3. Specify the Knowledge Base

4. Build the plan library by analysing various situations

5. Specify the execution of actions

Conclusion

- Implementation of plan generation procedure in a rich formal framework in RoboCup

- Same approach in Legged and Middle-size

- Fast software development from specification