Layered and Resource-Adapting Agents in the RoboCup Simulation

Christoph G. Jung*

GK Kogwiss. & MAS Group, FB Inform., Univ. des Saarlandes & DFKI GmbH
Im Stadtwald, D-66123 Saarbrücken, Germany
jung@dfki.de

Abstract. Layered agent architectures are particularly successful in implementing a broad spectrum of (sub-)cognitive abilities, such as reactive feedback, deliberative problem solving, and social coordination. They can be seen as special instances of *boundedly rational* systems, i.e., systems that trade off the quality of a decision versus the cost of invested resources. For sophisticated domains, such as the soccer simulation of RoboCup, we argue that a generalised framework that combines a layered design with explicit, resource-adapting mechanisms is reasonable. Based on the InteRRaP model, we describe a prototypical setting that is to guide and to evaluate the development of reasoning about *abstract resources*. These are representations of general interdependencies between computational processes. The realised soccer team, CosmOz Saarbrücken, participated successfully in the RoboCup-98 competition and confirmed that abstract resources are an appropriate modelling device in layered and resource-adapting agents.

1 Introduction

In the nineties, the complementary AI paradigms of deliberative, perfect rationality and of reactive, myopic emergence have found their reconciliation in the more and more prominent principle of *bounded rationality* [20]. Boundedly rational systems trade off the quality of a solution versus the cost of invested computation and interaction. On the one hand, this implies turning away from purely complex[1] decision making into a more tractable, thus situated form of intelligence. On the other hand, bounded rationality still demands optimality with respect to given domain constraints.

We adopt the term *resource* to describe these mostly quantitative constraints that are imposed onto the agent either by its environment (external resources, such as tools, fuel, workspace, etc.) or by its own computation device (internal resources, such as time and memory). Along Zilberstein [22], there are three options to realise bounded rationality based on this notion. *Resource-adapted* systems, e.g., [18], are built with

* supported by a grant from the "Deutsche Forschungsgemeinschaft" (DFG).

[1] We speak of *complex* decisions as long-term intentions that are composed of several primitive system operations in order to produce an optimal answer. Generally, the corresponding decision procedures turn out to be complex, too: Their computational needs increase exponentially with problem size. Complementary, *simple* decisions denote primitive measures that maximise the system's performance just for a single step in time. Often, they can be computed using fairly undemanding, therefore simple procedures.

a pre-designed off-line reflection of the resource characteristics of a specific domain. Secondly, *resource-adaptive* systems are “somehow” able to react on-line to changes in domain-specific resources. However, generic agent models should be applicable to a range of demanding and dynamic environments. They should cope with a great variety of resources. This renders the construction of adapted or adaptive mechanisms highly difficult. Thus, domain-independent agent architectures favour the third option of *resource-adapting* designs, e.g., [19], that incorporate explicit resource representations and reasoning[2].

Hybrid agents, especially the three-*layered* InteRRaP [16] design, are resource-adaptive systems since integrating the different computational expenses and subsequently the different decision qualities of reactive feedback, deliberative problem solving, and even social reasoning. Compared to monolithic agent models incorporating only a single form of inference, hybrid agents provide advantages in domains in which a whole spectrum of (sub-)cognitive abilities is required.

Along with recent developments to define a formal methodology for hybrid systems [4, 12, 10], we have proposed a more elaborate notion of *layering*. Our investigations identify a meta-object relationship in InteRRaP, i.e., the deliberative *Local Planning Layer* (LPL) monitors and configures the computations inside the reactive *Behaviour-Based Layer* (BBL). Similarly, the *Social Planning Layer* (SPL) negotiates about LPL-goals and LPL-intentions, commits to change them, and adjusts the LPL accordingly.

This “layering as meta-reasoning” perspective is very close to the resource-adapting framework of Russell & Wefald [19] and leads to a generalised InteRRaP architecture [7, 8] in which resources of a lower layer are explicitly represented and reasoned about by its upper companion. The proposed representation is called *abstract resource* and denotes both internal, computational as well as external, environmental interdependencies between computational processes.

To guide the refinement of such boundedly rational models, we regard the definition of challenging domains, such as the soccer simulation (Figure 1) of the RoboCup initiative [13], to deliver an appropriate experimental and empirical basis. Simulated robot soccer offers a controllably continuous, dynamic, inaccessible, and noisy environment. It provides for a concise success criterion and introduces a competitive, realistic background for cross-evaluation.

The explicit management of internal as well as external resources is an important topic in RoboCup. As exemplified in Figure 1, a wide range of rapidly — during the fraction of a second — and unpredictably changing situations occur. Therefore, an immediate and flexible motivational shift of the computational engine is essential, e.g., for implementing a sudden retreat if your team instantly looses possession of the ball. Furthermore, the simulated physics of the soccer agents is also imposed some far reaching constraints. Especially the recently revised model of *stamina* severely penalises constant, exhaustive dashing.

[2] Russell & Subramanian argue that any such reasoning is itself subject of resource consumption and thus prevents optimality [18]. We rather define the task of a generic, resource-adapting system to *approximate* optimality — as mentioned in [19], experiences of explicit resource reasoning can be compiled into simpler, implicit control.

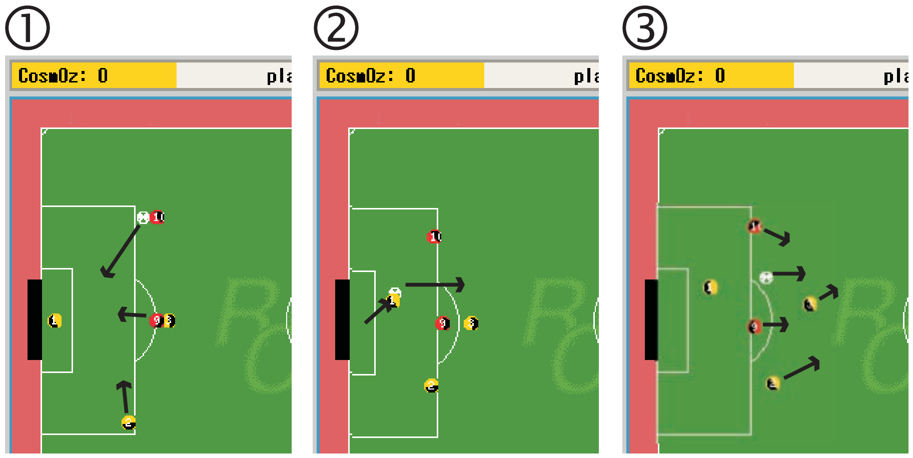

Fig. 1. The Rapid Shift of Motivations in the RoboCup Simulation

We also recognise the aspect of layering inside soccer agents. Positioning, orientation, ball handling, aiming, and tracking are reasonably defined as reactive patterns of behaviour. The strategic composition of "moves" (defending in your own half, dribbling in the middle-field, attacking from the right flank) can be assigned to deliberative planning. Hereby, each move can be realised by configuring the reactive subsystem accordingly, in particular by influencing the resource allocation to the behaviour patterns. Finally, coherent team play is the ultimate requirement to score goals and to prevent your opponent from doing so. Social reasoning including negotiation has to coordinate the deliberative planning in order to obtain combined moves (double-pass, offside-trap), assign roles (attacker, defender, goalie), and implement tactics (offensive, defensive).

Contribution. To gain experience with the generalised InteRRaP model (Section 2), in particular with abstract resources and respective decision algorithms, the present paper specifies prototypical layered and resource-adapting agents (Section 3) for the demanding RoboCup simulation. Our team, CosmOz Saarbrücken, has successfully participated the RoboCup-98 competition (quarter final; 9th rank) and will guide future research on the complete architecture (Section 4).

Opposed to, e.g., Burkhard et al. [3], our aim is to study mainly domain-independent mechanisms for intelligent real-time systems which addresses the RoboCup *Synthetic Teamwork Challenge* [14]. Abstract resources provide advantages with respect to both typical control-of-search and typical control-of-behaviour techniques. Russell & Wefald [19], for example, focus on a single and *sequential* time resource. Controlling real-time systems, but, requires a more flexible treatment of general interdependencies between *concurrent* processes. Using *a priori* estimations in conflict resolution, our approach is able to prevent the redundant computations found in *a posteriori* arbitration used by, e.g., Riekkie & Röning [17]. Furthermore, our meta-control is smoothly integrated with three important styles of inference (reactive, deliberative, social) only found separately in current RoboCup systems.

2 From Resource-Adaptive to Resource-Adapting Agents

2.1 InteRRaP: A Layered Model

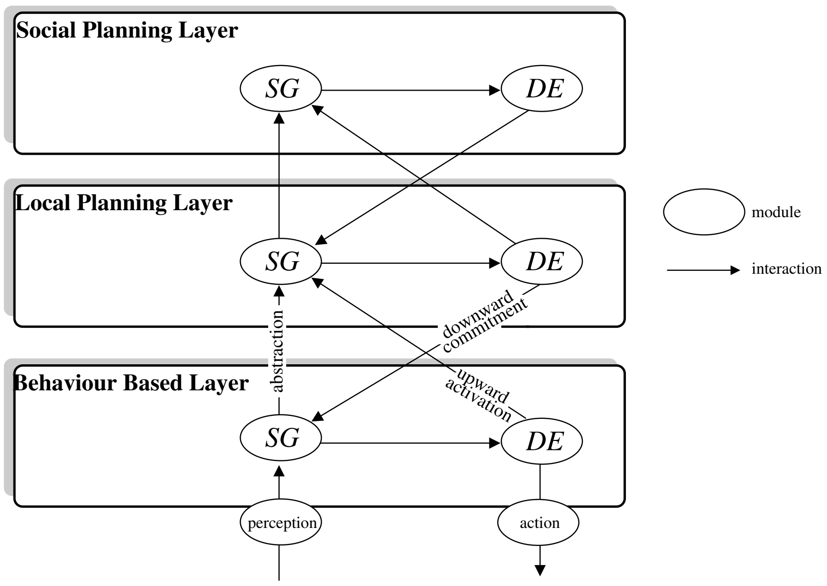

Fig. 2. The Layered InteRRaP Architecture

Hybrid architectures, such as InteRRaP [16] (Figure 2), integrate the functionality of separate *modules* by determining their interactions in a rather pragmatic manner. InteRRaP models the smooth transition from sub-symbolic reactivity to symbolic deliberation and even social capabilities by realising three different *layers*. Each layer internally follows a common flow of control: the *Situation Recognition and Goal Activation* module (SG) spawns new goals out of the maintenance of belief. The *Decision Making and Execution* module (DE) decides about how to meet these goals and thus obtains plans or intentions to execute.

The most concrete *Behaviour-Based Layer* (BBL) provides a short feedback loop with the environment by applying procedural routines, so-called *patterns of behaviour*. BBL decision making is quite fast and simple, because behaviour patterns are reactively triggered (*reflex*) by recognised situations. Stacked on top, the *Local Planning Layer* (LPL) reasons (plans) about how to meet long-term, abstract goals. Similarly, social decisions (at the *Social Planning Layer* — SPL) that involve negotiating and coordinating with other agents are also expressed as a planning problem.

These informal architectural considerations have been formalised by means of a detailed formal specification [12, 10] that bridges the gap to verifiable implementations and also to theorical issues. Our *computational model* of InteRRaP applies the principle of fine-grained concurrency between (sub-)cognitive *processes* located **inside** the

SG and DE modules, such as perception, reflexes, patterns of behaviour, or planning. Processes encapsulate logical inferences and compute continuously in an independent manner. Communication between processes happens (explicitly) via *signals* and (implicitly) via logical data structures in a *shared memory*. The Oz language [21, 9] turns out to be a highly suitable implementation platform for both the inference engines and the process model, because Oz combines a logical background with modern programming features, such as concurrency, object-orientation, and transparent distribution[3].

The formalisation of a computational model also requires the conscientious reinvestigation of layering. Originally ([16]), layers interact via upward activation, e.g., a behaviour pattern "calls" the planner as a subroutine, and downward commitment, e.g., the planner activates additional patterns of behaviour (see Figure 2). Thus each layer represents an optional, possibly more useful, but expensive path of computation. Decisions of any layer have basically the same status and are arbitrated in between. This resembles many other layered designs, such as the Subsumption architecture [2].

In the extended specification [12, 10], the layer itself is realised as a designated *control process* that supervises the communication of encapsulated (sub-)cognitive processes, such as the BBL desires, reflexes, and patterns of behaviour. The inter-layer interaction is now installed as a special form of meta-object relationship. Hereby, a lower layer, such as the BBL, does indeed implement **all** the functionality of the agent. To be guided towards exhibiting a specific rational function, it is monitored, reasoned about, and reconfigured by its super-layer by means of the control process, e.g., for approaching objects, the LPL demands to suppress an avoid-collision reflex.

The BBL is thus no more subsumed, but supported by its super-layer LPL. Layers do no more stand in competition, but in a structured, cooperative relation with their super-layers. This perspective decouples the higher-level reasoning from the critical timing constraints of dynamic environments, especially the social reasoning in the SPL has the state of the LPL (current goals, future intentions) as its topic and its commitments non-monotonically affect the goals (adding or removing goals) and intentions (adopt or drop intentions) in the LPL.

2.2 A Layered and Resource-Adapting Model

By the different computational needs of its layers, InteRRaP so far represents an instance of resource-adaptive rationality. Changing resources in the environment and in the computation device influence the quality of actually executed decisions: If the environment becomes more calm, it is more likely that the deliberative module can timely influence the fast decisions of the reactive module. If the environment becomes more dynamic, the reactive module will constantly act without the planner being able to intervene. This adaption is implicitly encoded into the model.

Demanding environments are full of changing constraints and require to make such relevant design-time conventions rather a part of the run-time decisions of the agent. Russell & Wefald [19], for example, develop a resource-adapting architecture where

[3] In CosmOz, the distribution facilities are used to neatly manage a computer pool for running agents in parallel. In concordance with the official simulation rules, inter-player communication uses the soccer server as the only medium.

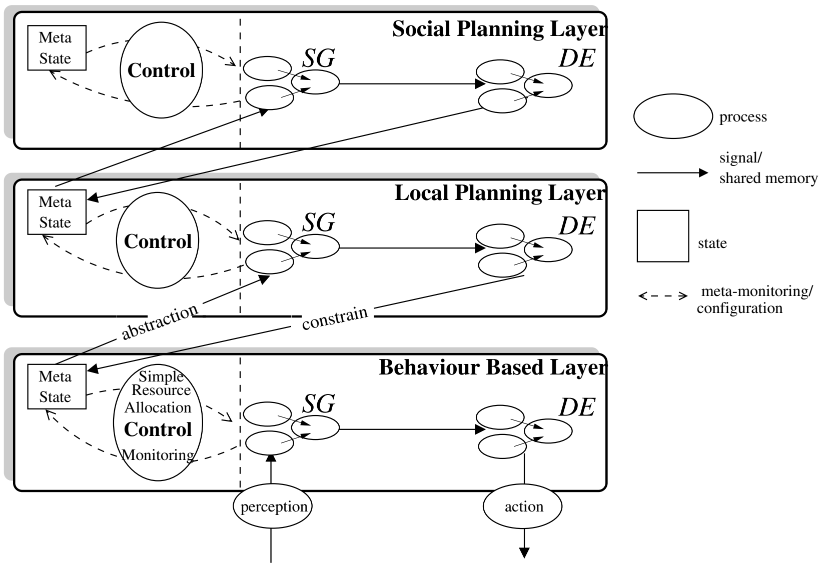

Fig. 3. Extending InteRRaP to a Resource-Adapting Scheme

complex object-level reasoning about external resources is guided by simple meta-level allocation of a single, internal resource (time). They report its successful application to non-trivial game-playing. In [7, 8], we have argued that this architecture is not immediately applicable to real-world, multi-agent domains such as RoboCup. One reason is the complexity of the single object-level. Another problem relates to the short-sight of the meta-level caused by the inherent object-level dependencies which go beyond the pure consumption of time. Generally, a clear separation of internal and external resources is not possible since they are substitutable, i.e., by withdrawing computation time from a particular decision process, taking actions can be prevented.

Consequently, a generalisation of both the layered InteRRaP model and the design of Russell & Wefald [19] into a multi-staged resource management is envisaged (Figure 3). Each layer hereby realises a complete meta-functionality by guiding (constraining) the resource allocation to the computations on its subordinate layer. LPL and SPL planning is thus established as a form of explicit and complex reasoning about resources. To uphold tractability, the control processes of any layer (see Section 2.1) are additionally equipped with a simple and situation-oriented mechanism to refine higher-level guidelines into a concrete resource assignment to the supervised processes. The control process and its simple resource allocation are thus already a part of the meta-interface; the complex higher-level reasoning can focus on rather abstract, long-term conflict resolution. In both forms of (resource) decision making, the representation device of *abstract resources* is employed. It denotes general interdependencies between (or constraints on) the supervised processes, be they of internal or external nature.

3 A Prototypical Soccer Agent

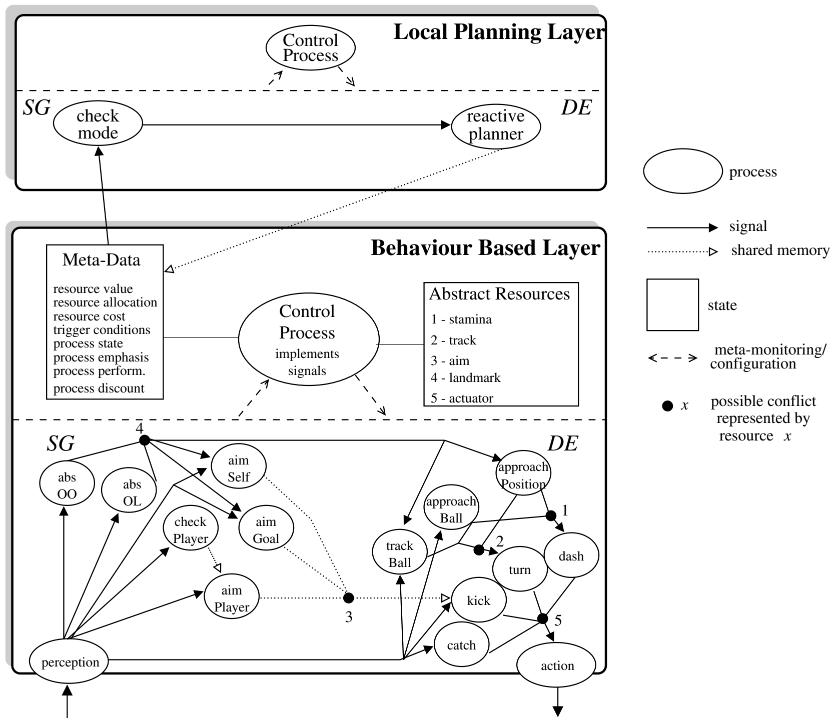

Fig. 4. The Lean Model of Resource-Adapting InteRRaP Agents in the RoboCup

Making the ideas of the preceding Section 2.2 more concrete is not trivial since building on an already matured agent framework which incorporates many design decisions. Therefore, we first step back to a leaner agent model consisting of BBL and simplified LPL instantiated to the already motivated RoboCup simulation (Figure 4). This model is used to gain experience with the important representational and algorithmic issues of abstract resources. Our detailed presentation in the following sections focuses on the interface between deliberative and reactive facilities, thus the control process of the BBL and its parameterisation by LPL plan operators. We discuss our presumptions, such as synchronous concurrency, process-built-in self-evaluation, and discrete resource values which are statically, i.e., in fixed amounts, assigned to active processes. As Section 4 concludes, the lean model has proven successful in RoboCup-98 and appears to be incrementally extendible to the whole layered and resource-adapting design.

3.1 Reactive, Sub-Cognitive Processes

The computational model of InteRRaP [12, 10] extends the ideas of reactive control systems [2, 15, 5] to a dynamic network of concurrent computation spread over the whole layered agent. Concurrency in the model hereby ensures the responsiveness[4] of any inference within the agent. This is of course particularly important for the reactive BBL processes of RoboCup agents depicted in Figure 4.

The `perception` process senses perceptual data from a UDP datagram connection to the simulation server. Its activity is further distributed by outgoing signals triggering other processes, such as `catch` or `kick`. Signals are emitted upon specific trigger conditions in a process state (see Section 3.2), e.g., if `perception` indicates that the ball appears to be near the player, `kick` will be invoked. Because `kick` additionally uses defaults for determining where and how hard to kick, this signal path thus implements a highly reactive reflex which does not involve much computation.

The direction and power defaults inside `kick` are accessible via shared memory to the `aimGoal`, `aimSelf`, and `aimPlayer` processes in order to allow more sophisticated goal-kicks, dribbling, and passes. These computations determine their decisions both from the relative data in `perception` and from other preprocessing steps, such as the derivation of absolute coordinates via triangulation (`absOO`: use two landmarks, `absOL`: use a landmark and an adjacent line) and such as checking team mates with respect to the usefulness of their position (`checkPlayer`).

Navigation is encoded into the `approachPosition`, the `approachBall`, and the `trackBall` patterns of behaviour. They position and orientate the player with respect to landmarks and to the ball by triggering *virtual* `turn` or `dash` actions. Approaching and tracking mainly depend on relative perception, but additionally require absolute data in the case that the envisaged objects are not visible. *Virtual actions* (including the aforementioned `kick` and `catch`) signal activity to the `action` process that finally sends primitive, external actions (*commands*) over the datagram connection to the simulation server.

Indeed, this reactive network already incorporates the whole functionality of soccer agents, e.g., the complete motivational basis for all different situations and for all different player roles in Figure 1, **at the same time**. This eventually raises conflicts because processes are unaware of their side-effects. For example, simultaneously haunting the ball and keeping a certain position results in a "paranoid" floundering of the soccer agent. This is a conflict with external grounding in the simulated "body" of the player. Similarly, commands could fail due to such restrictions (`catch` is useless for non-goalies; limited number of commands per cycle; exhausted stamina). Conflicts also have internal grounding, such as redundancy (`absOO` and `absOL`) or mutual overwriting of output (the `aim` processes). In Figure 4, we have marked five such sources of interdependencies between the reactive processes in our soccer agents.

[4] This is not to say that other models could not exhibit interactivity. To obtain a similar degree of responsiveness from a sequential model, but, the programmer has to put additional interaction and scheduling knowledge into the domain-dependent part of the agent. A good model, however, already integrates those facilities required in most domains, thus eases the programmer's task. Therefore, we regard both *Turing Machines* as well as models with hidden concurrency as bad agent models.

3.2 The Control Process, Synchronous Concurrency, and Abstract Resources

The BBL control process (Figure 4) is responsible for a reasonable mid-term interaction of the supervised behaviour patterns. The control process influences their computation by implementing their communication. This amounts to a synchronous form of concurrency, because each reactive process is sliced into subsequent computation *chunks* of flexible, but limited size. For example, such a chunk once takes aim (in `aimGoal`) or once produces a limited trajectory by firing a number of turn and dash actions (in `approachPosition`).

To trigger and determine the next, ready-to-run chunk of a process, the control process generates *signals* in each of its cycles. This is guided by *trigger conditions* that have to be satisfied by the state of the signal-emitting process, e.g., the relative position to the opponent's goal has to change in `perception` to trigger `aimGoal` anew.

In a second step, the set of generated signals is filtered to obtain an approximately optimal set of chunks, i.e., to minimise the conflicts between active computations. Therefore, we introduce the discrete, quantitative representation of *abstract resources* for each of these interdependencies: Natural numbers indicate the available amount of each resource. The dependent processes "consume" this amount by each computation chunk according to its length and its type. Once a resource is exhausted, signals to the "applying" processes are suppressed. In each cycle, resources recreate according to a recreation function.

Interestingly, the interdependencies in Figure 4 fit very well into this generic scheme. For example, the limited `landmark` resource (range $\{0, 1\}$) restricts the concurrent and redundant operation of the two optional methods for obtaining absolute coordinates. Both `absOO` and `absOL` apply and only one is able to get a hold on it. The `aim` resource (range $\{0, 1\}$) introduces a similar restriction with respect to `aimPlayer`, `aimSelf`, and `aimGoal`. So to speak, the agent has only a limited aiming capacity[5].

`landmark` and `aim` denote purely internal interactions inherent in the computational design. We do also find abstract resources with combined internal and external grounding. For example, the `stamina` resource (range $\{0, \ldots, 2000\}$[6]) matches the external model of the soccer agents' power and also represents the internal dependency between the possibly conflicting positioning behaviours. Thus `approachPosition` and `approachBall` apply for it. By assigning larger portions of `stamina`, a respective chunk is able to implement a longer trajectory.

The `actuator` resource (range $\{0, 1\}$) describes the limited amount of commands allowed in each simulation cycle: the virtual actions `kick`, `catch`, `turn`, and `dash` access it. Finally, the `track` resource (range $\{0, 1\}$) mediates between the different needs in orientation of `approachPosition` and `trackBall`.

[5] As discussed in [7], abstract resources provide an attractive device for cognitive science as well. They can be used to emulate cognitive restrictions of the human mind, such as limited short-term memory, focus of attention, etc. Indeed, cognitive architectures, such as ACT-R [1], also take a bounded rationality perspective towards such constraints.

[6] The range depends on the relation of inner-agent and simulator scale. The stamina resource is not purely traced internally, because RoboCup allows for "mental" actions that deliver the exact, external value. We have omitted this fact for reasons of simplicity.

Performance Monitoring and Simple Resource Allocation. We now develop the control process as a simple resource allocation procedure which optimises the choice of admissible signals, thus maximises the expected utility of a particular allocation choice. However, for such a short-term decision, sporadic external performance measures, such as the score in soccer, are not suitable. We therefore propose a form of internal profiling which is surprisingly applicable to many difficult cases: supervised processes incorporate self-evaluation functions. During each chunk, they produce a performance report which is then used as a prediction for the allocation decision.

An example is the `aimGoal` process: In order to adjust the `kick` defaults, this process has to analyse distance and direction to the opponent's goal, anyway. The greater the distance, the less likely the next kick will score a goal and the less useful the current chunk of aiming has been. This way, straightforward performance mappings (also of `aimPlayer` and `aimSelf` chunks; similarly for chunks of all other processes) onto a normalised scale ($[0, 1]$) can be found.

If we additionally assume the possible allocations of abstract resources to processes to be static, the simple cycle of the control process can now be formalised[7] as in Figure 5: After having updated the values for resources with a recreation function (1.), we obtain a set of signals using the trigger conditions (2.). This set is now sorted according to the utility of the destination processes (3.). Hereby, the utility of running a process depends on its expected performance reported by its last chunk minus the cost of the resource allocation. Here, we use independent cost functions for each resource. As long as the utility is greater than zero (4.), the static allocations are tried to be granted to the processes in order. If possible, resource values are updated (4.(a)) and the respective signals are transmitted (4.(b)). Otherwise, if any related resource would get exhausted, the selected set of signals is deleted (6.). A particular problem is posed by inactive processes not being able to adjust their evaluation to the current setting. Therefore, a discount of frequently selected "winners" (3.;4.(c);5.) improves the chance of steadily probing also the supposedly bad candidates. Furthermore, we introduce for each process an emphasis parameter.

3.3 Deliberative Influence: Plan Operators

In the RoboCup simulation, the BBL already delivers a reasonable mid-term behaviour: motivations and decisions are smoothly interpolated and appropriate parameterisations of the control process implement interesting "moves" of the agent: An example for such a move is the right-wing attack of player 10 in Figure 1 which, by the `aim` resource, steadily mediates between shooting to the goal, dribbling, and passing. Mediation hereby depends on an appropriate set of trigger conditions monitoring the distance to the goal, the free space of the agent, and the position of team mates.

Long-term motivations and strategic intentions composed out of particular moves are however not exhibited at the BBL level. This does also exclude projecting the evolution of resources into the future, thus a complex resource optimisation. The `stamina`

[7] In the present paper, we do not want to go into the detailed decision-theoretic assumptions, such as abstract resources describing conflicts in an independent and exhaustive manner. This will be the topic of a separate paper.

```
processes $\mathcal{P} = \{P_i\}$; resources $\mathcal{R} = \{R_j\}$; signals $S$; trigger condition $T : 2^P \times P \to S$;
resource value $V : R \to \mathrm{N}_0$; expected process performance $\pi : P \to [0, 1]$;
allocation $A : P \times R \to \mathcal{N}_0$; resource cost $C : R \times N_0 \times [0, 1]$;
recreation function $\rho : R \times \mathrm{N}_0 \to \mathrm{N}_0$; process emphasis $E : P \to [0, 1]$;
discount $D : P \to [0, 1]$; discount factors $\{d_i\}$.

1. recreate resource values $V(R_j) \leftarrow \rho(R_j, V(R_j))$.
2. out of processes $\mathcal{P} = \{P_i\}$ and trigger conditions $T$,
   generate signals $\mathcal{S} = \{S_i = T(\mathcal{P}, P_i)\}$.
3. choose a $P_i$ with $\mathcal{S} \ni S_i \neq \{\}$ and highest utility

       $U(P_i) = D(P_i) * E(P_i) * \pi(P_i) - \sum_j C(R_j, A(P_i, R_j))$

4. if $U(P_i) > 0$ and for each $R_j$, $V(R_j) - A(P_i, R_j) \geq 0$
   (a) consume resources $V(R_j) \leftarrow V(R_j) - A(P_i, R_j)$.
   (b) start $P_i$ with signals $S_i$ and allocation $A(P_i)$.
   (c) discount $D(P_i) \leftarrow \frac{D(P_i)}{d_i}$.
5. else reset discount $D(P_i) \leftarrow 1$.
6. delete signals $\mathcal{S} \leftarrow \mathcal{S} \setminus \{S_i\}$.
7. if $\mathcal{S} \neq \{\}$ then goto step 3
8. else goto step 1.
```

Fig. 5. Cycle of the Control Process

resource, for example, desperately needs such a projection because it does not allow freely dashing across the field. The required reasoning is appropriately described as a planning task of the LPL. Primitive moves are the *means* to be concatenated into plans that probably install the *ends*, such as to score a goal or to prevent your opponent from doing so. Our setting implies a form of decision-theoretic planning, because the planner additionally has to minimise the (resource) costs of the plan. Yet, these considerations are not in the focus of the lean model in Figure 4 concentrating on the interface between plan operators and the control process of the BBL.

At the moment, a simple situation recognition, `checkMode`, is tied to the state of the BBL control process and in turn triggers the `reactivePlanner`. `checkMode` infers the agent's role depending on the shirt number, derives the possessor of the ball, and determines a coarse play mode. The planner thereafter activates a particular corresponding move, such as what has to be done for a "kickoff", for "defending with the ball in own half", for "passing the ball into the middle field", or for "attacking from the right wing". Section 4 comes back to strategic, goal-oriented planning upon these moves.

As already anticipated, moves are implemented by appropriate configurations of the BBL control process. The configurable meta-data (Figure 4) comprises the trigger conditions of signals, the possible allocation of resources, emphasis of processes, and discount factors.

Let us discuss the impact of these parameters at hand of the defensive behaviour of the goal keeper (shirt number 1) in Figure 1 which is invoked once the opponent possesses the ball (one of the opponent players is recognised to have the least distance to it) and once the ball is residing within the goalie's half of the field. Herein, the emphasis parameters are set to prioritise catching before kicking before turning before dashing. This allows the goalie to quickly intercept the movement of the ball (the installed trigger condition of `catch` monitors the ball to fall short off a velocity-dependent distance), afterwards shooting it away, possibly to a team mate nearby. Catching and kicking are vital; their discount is chosen to be small ("no experiments in the emergency case"). Since `aimGoal` and `aimSelf` do not make sense for the goalie in this mode, they are completely deemphasised ($E(P_i) = 0$).

Besides, it is important for the goalie to frequently adjust its central position while tracking the ball (`approachPosition` triggers on leaving the goal). Hereby, the `stamina` resource is only available in minimal amounts while the sudden interception of `approachBall` triggered upon the ball entering the penalty area should be a continuous trajectory, thus a greater amount of `stamina` is granted if activated.

4 Conclusion and Outlook

Although we have yet implemented a lean version of the generalised framework for hybrid, resource-adapting systems, our CosmOz team already went into the quarter finals of the RoboCup-98 competition. Its performance demonstrated reactive, rational, and even implicitly cooperative behaviour. This supports our claim that the connection between layered and resource-adapting mechanisms can be made and that the representation of abstract resources applies well. The current prototype will be used to empirically study the influence of the resource-adapting mechanisms and its parameterisation onto the efficiency of a soccer team. Besides the close investigation of the decision-theoretic assumptions within our model (partially discussed in [7]), open issues for the future are:

Reactive Processes. The modularisation of reactive soccer facilities into processes has to be improved. For example, `aimPlayer` and `aimSelf` were only poorly implemented at RoboCup-98 not taking the trajectory of opponent players into account. Additional processes are needed to leave bad positions for actually receiving a pass or avoiding off-side. We also think of how to pursue particular opponent players in one-to-one defensive strategies. These changes will eventually introduce new conflicts whose management is straightforwardly supported by our generic resource representation.

Resource Allocations and On-line Learning. In the current model, possible allocations of resources and the priority of processes are just guided by LPL operators and otherwise remain static. Our experiences in RoboCup-98 have shown that the overall team behaviour is extremely sensitive to minimal changes in those parameters. Since this furthermore depends on the actual simulator settings, it poses a major design problem. [6] proposes to use local search which frequently probes allocation variants at runtime in order to improve the overall sum of process utilities to a satisficing level.

We also plan to adjust process priorities on the fly by a combination of reinforcement learning and memory-based reasoning.

Long-Term Deliberation and Social Abstract Resources. The goal-oriented, partial-order planner of InteRRaP is described in [11]. Its latest version includes hierarchical planning and on-line facilities in a logical setting. We are keen to integrate this planner into the lean model in order to obtain reasonable strategies out of complex intentions of the agent. We are exploring the role of BBL resources in representing the effects of LPL moves. Furthermore, the LPL is also subject of adaption due to its control process and the SPL. It is not clear up to now, how abstract resources can be used to describe the computational interdependencies within the LPL. We expect that this will change the discrete representation of resources from numbers to sets of items, such `roles` to incarnate, or different motivational `foci` of planning in order to coordinate moves (double pass, off-side trap) or even constrain the behaviour of the whole team (tactics, role assignment, etc.).

Acknowledgements

The author is grateful for the fruitful discussions with his colleagues of the MAS group. In particular, the concept of abstract resources has been developed in cooperation with Christian Gerber. The author would like to thank Jörg Müller for his path-setting thesis on InteRRaP. CosmOz owes much to the wonderful Oz programming language developed at the PS Lab at the Universität des Saarlandes and the Swedish Institute for Computer Science. Finally, thanks to the RoboCup initiative for designing a highly motivating and entertaining domain.

References

1. J. R. Anderson. *Rules of the mind.* Lawrence Erlbaum Associates, Hilldsdale, NJ, 1993.
2. R. A. Brooks. A robust layered control system for a mobile robot. In *IEEE Journal of Robotics and Automation*, volume RA-2 (1), pages 14–23, April 1986.
3. H. D. Burkhard, M. Hannebauer, and J. Wendler. AT humboldt — development, practice and theory. In *RoboCup-97: Robot Soccer World Cup I*, volume 1395 of *Lecture Notes in Artificial Intelligence*, pages 357–372. Springer, 1998.
4. B. Dunin-Keplicz and J. Treur. Compositional formal specification of multi-agent systems. In *Intelligent Agents*, volume 890 of *Lecture Notes in Artificial Intelligence*, pages 102–117. Springer, 1994.
5. R. James Firby. Task networks for controlling continuous processes. In *Proc. of the 2nd International Conference on Artifical Intelligence Planning Systems*, 1994.
6. C. Gerber. An Artificial Agent Society is more than a Collection of “Social” Agents. In *Socially Intelligent Agents - Papers from the 1997 AAAI Fall Symposium.* Technical Report FS-97-02, AAAI, 1997.
7. C. Gerber and C. G. Jung. Towards the bounded optimal agent society. In C. G. Jung, K. Fischer, and S. Schacht, editors, *Distributed Cognitive Systems*, number D-97-8 in DFKI Document, Saarbrücken, 1997. DFKI GmbH.

8. C. Gerber and C. G. Jung. Resource management for boundedly optimal agent societies. In *Proceedings of the ECAI´98 Workshop on Monitoring and Control of Real-Time Intelligent Systems*, pages 23–28, 1998.
9. S. Haridi, P. Van Roy, P. Brand, and C. Schulte. Programming languages for distributed applications. *New Generation Computing*, 1998. To appear.
10. C. G. Jung. On the Role of Computational Models for Specificying Hybrid Agents. In *Cybernetics And Systems'98 — Proceedings of the 14th European Meeting on Cybernetics and System Research*, pages 749–754, Vienna, 1998. Austrian Society for Cybernetic Studies.
11. C. G. Jung. Situated abstraction planning by abductive temporal reasoning. In *Proc. of the 13th European Conference on Artificial Intelligence ECAI'98*, pages 383–387. Wiley, 1998.
12. C. G. Jung and K. Fischer. A Layered Agent Calculus with Concurrent, Continuous Processes. In *Intelligent Agents IV*, volume 1365 of *Lecture Notes in Artificial Intelligence*, pages 245–258. Springer, 1998.
13. H. Kitano, M. Asada, Y. Kuniyoshi, I. Noda, and E. Osawa. Robocup: The robot world cup initiative. In *Proc. of The First International Conference on Autonomous Agent (Agents-97)*, Marina del Ray, 1997. The ACM Press.
14. H. Kitano, M. Tambe, P. Stone, M. Veloso, S. Coradeschi, E. Osawa, H. Matsubara, I. Noda, and M. Asada. The robocup synthetic agent challenge. In *RoboCup-97: Robot Soccer World Cup I*, volume 1395 of *Lecture Notes in Artificial Intelligence*, pages 62–73. Springer, 1998.
15. D.M. Lyons and A.J. Hendricks. A Practical Approach to Integrating Reaction and Deliberation. In *Proceedings of the 1st International Conference on Artifical Intelligence Planning Systems*, 1992.
16. J. P. Müller. *The Design of Intelligent Agents: A Layered Approach*, volume 1177 of *Lecture Notes in Artificial Intelligence*. Springer-Verlag, December 1996.
17. J. Riekki and J. Röning. Playing soccer by modifying and combining primitive reactions. In *RoboCup-97: Robot Soccer World Cup I*, volume 1395 of *Lecture Notes in Artificial Intelligence*, pages 74–87. Springer, 1998.
18. S J. Russell and D. Subramanian. Provably Bounded Optimal Agents. *Journal of Artificial Intelligence Research*, 2, 1995.
19. S J. Russell and E. Wefald. *Do the Right Thing*. MIT Press, Cambridge Mass, 1991.
20. H. A. Simon. *Models of Bounded Rationality*. MIT Press, Cambridge, 1982.
21. G. Smolka. The Oz Programming Model. In Jan van Leeuwen, editor, *Computer Science Today*, Lecture Notes in Computer Science, vol. 1000, pages 324–343. Springer-Verlag, Berlin, 1995.
22. S. Zilberstein. Models of Bounded Rationality. In *AAAI Fall Symposium on Rational Agency*, Cambridge, Massachusetts, November 1995.