A Dual Interaction Perspective for Robot Cognition: Grasping as a "Rosetta Stone"

H. Ritter, R. Haschke, and J. J. Steil

Neuroinformatics Group, Bielefeld University, Germany
{helge,rhaschke,jsteil}@techfak.uni-bielefeld.de

Summary. One of the major milestones to higher cognition may be the ability to shape movements that involve very complex interactions with the environment. Based on this hypothesis we argue that the study and technical replication of manual intelligence may serve as a "Rosetta Stone" for designing cognitive robot architectures. The development of such architectures will strongly benefit if improvements to their interaction capabilities in the task domain become paired with efficient modes of interaction in the domain of configuring and restructuring such systems. We find that this "dual interaction perspective" is closely connected with the challenge of integrating holistic and symbolic aspects of representations. In the case of grasping, this requires a very tight marriage between continuous control and more discrete, symbol-like representations of contact and object states. As a concrete implementation, we propose a two layered architecture, where the lower, subsymbolic layer offers a repository of elementary dynamical processes implemented as specialised controllers for sensori-motor primitives. These controllers are represented and coordinated in the upper, symbolic layer, which employs a hierarchy of state machines. Their states represent semantically related classes of dynamic sensori-motor interaction patterns. We report on the application of the proposed architecture to a robot system comprising a 7-DOF redundant arm and a five-fingered, 20-DOF anthropomorphous manipulator. Applying the dual interaction approach, we have endowed the robot with a variety of grasping behaviours, ranging from simple grasping reflexes over visually instructed "imitation grasping" to grasping actions initiated in response to spoken commands. We conclude with a brief sketch of cognitive abilities that we now feel within close reach for the described architecture.

1 Introduction

One of the most remarkable feats of nature is the evolution of the cognitive abilities of the human brain. Despite the use of neurons which are – compared to technical transistor devices – slow, of low accuracy and of high tolerance, evolution has found architectures that operate on highly complex perception and control tasks in real time. These by far outperform our most sophisticated technical solutions not only in speed, but also in robustness and adaptability.

What can we learn from these systems in order to replicate at least partially some of their enticing properties, bringing robots (or other artificial systems) closer to a level that might deserve the term "cognitive"?

Brains seem always intimately connected with action and change: as soon as evolution started to endow organisms with modestly rich capabilities of motion, we also see the advent of nerve cells that are invoked in the organisation of such motion. Ordering organisms according to the sophistication of their self-controlled movements suggests a strong correlation with the complexity of their brains: "simple" central pattern generators of a handful of cells for generating periodic movements adequate for a homogeneous environment such as water, added sensorimotor-complexity in the form of a cerebellum for coping with the more varying demands of walking on a solid ground, the advent of cortex for even higher flexibility, enabling exploration and context-specific assembly of complex motor programs that later can become automated by learning. Finally there is generalisation of manipulation of physical objects and tool use to the domain of "mental objects" in one's own brain or in the brain of a conspecific: communication and thinking.

We may look at this evolution from at least two perspectives: the first perspective is that of neurophysiology, trying to understand the evolution of the physical structure of the brain. Since single neurons are the elements that can be studied best, this approach faces a strong bottom-up bias, confronting us with a task that is at least as monumental as reverse-engineering the architecture of the operating system of a computer when we have access to a small subset of its transistors. The increasing availability of more global windows into the brain begins to enrich our picture also from a top-down perspective to the extent that we can devise clever experiments that tell us more than just where a brain function seems to be located.

A totally different approach results when we step back from the brain's hardware and instead take the perspective that our main concern is the *functional architecture* needed for a cognitive system. Then we look at the issue of implementation only to the extent that it may pose significant constraints of what architectures are realisable at all. If this route can be made productive, the outcome will be functional invariants that characterise cognitive systems per se. These invariants possibly have very different mappings onto actual hardware, depending on whether the target system employs neural tissue, silicon processors or even some still undiscovered technology of the future.

2 Grasping as a "Rosetta Stone" for Cognitive Robots

Following this route, an obvious question to ask is: can we find a *key problem* that is more manageable than the whole, but sufficiently rich to provide essential insights into the architectural principles enabling natural cognition? We believe that there is an affirmative answer to this question: *The study*

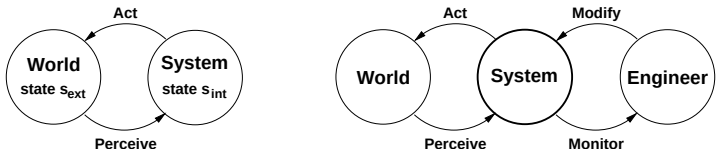
and technical replication of manual intelligence can be regarded as a good "Rosetta Stone" for designing an architecture for cognitive robots.

Using our own hands, we hardly notice the complexity of the involved processes: depending on the presence of other objects and our intended action, we select an appropriate approach direction and pre-shaping of our hand, initiate finger contact and create new contacts while closing our fingers. We adjust our finger forces and positions according to the object properties perceived during our grasp, and we break existing and create new contacts for controlled re-grasping.

From an abstract perspective, manipulation involves the structuring of a complex physical interaction between a highly redundant, articulated manipulator and a physical object, which can be highly variable. Dextrous manipulation is a form of mechanical control that is pervaded by frequent discontinuous changes of the contact topology between the actuator and the object. At the same time, these contact switches are accompanied by significant uncertainties in geometrical and physical parameters of the object (and, in robot reality, of the actuator too!). As a result, dextrous manipulation calls for an unusually tight marriage between continuous control and more discrete, symbol-like representations that can deal with the issues of topology-switching and encapsulation of parameter uncertainty. Viewed from the perspective of architecture research, it is this tight coupling of continuous, sensorimotor control and discrete, symbol-like representations that seems to be a major preparatory step for realising system structures which can finally lead to cognition with its remarkable integration of holistic and symbolic processing.

Once the necessary control architecture capable of controlling complex *external physical states* was in place, it did not really matter whether the controlled state referred to physical objects or to more abstract entities. One may speculate, that this control machinery was first applied to the organisation of the brain itself, finally enabling self-awareness [1]. From an architectural point of view, this would have required the creation of some "internal interface", making a subset of the brain's subsymbolic activity available in a similar more "tangible" fashion as physical objects outside. Once such an interface had become available, the control capabilities could become "elevated" with relatively small effort from the physical to the *mental domain*, leading to thinking and reasoning, and a new, consciously accessible "symbolic" representation of oneself. From this point, it appears no longer as a distant step to extend this representation also to other individuals, ultimately refining the manipulation of mental objects to enable coordination of "mental state objects" across individuals: communication.

Although speculative, from a functional point of view the drawn picture offers a very natural road map for the generalisation of movement control sophistication. In this picture, hands and manipulation have the pivotal role of preparing the transition from the organisation of more rigid behaviour to mental operations and thinking. Therefore, the synthesis of dextrous manip-



(a) Interaction (b) Dual Interaction

Fig. 1. Interaction perspective on architectures. Left (a): Classical interaction scheme, focusing on task-interaction between world and system only. Right (b): The dual interaction perspective takes a second branch of "config-interaction" between engineer and system into account.

ulation skills for robots appears to have great promise for driving us towards "the right architecture" to finally master the step to higher levels of cognition.

3 A Dual Interaction Perspective on Artificial Systems

The grasping scenario focuses our efforts on a key task: to organise a usually complex *interaction pattern* with the environment. Here, the notion of interaction pattern refers to the *overall behaviour* of the system due to different situations in the world. This requires to select adequate sensors and motor actions from a broad repertoire of elementary behaviour patterns or skills in order to realise a highly structured, situation-dependent coupling between selected regions of the world and the sensors and actuators of the robot or organism.

From a formal point of view, any architecture for the interaction between a robot (or any other technical or biological system) and the world implements a system of the form depicted in fig. 1(a), where the state of the world and its change due to actions are at best partly known. The aim of an architecture then is two-fold: first, to implement the system as the result of interactions between more elementary subsystems, and second, to shape the *overall behaviour* of the system in such a manner that an adequate, task- and situation-dependent interaction pattern results. The subsystems of the architecture should be organised in a modular and hierarchical manner, and they should obey a parsimonious set of carefully selected design principles, requiring the recurring application of a restricted set of design patterns at different levels, leading to a partial self-similarity of the architecture design.

So far, our discussion was focused on the "task-interaction" between the system and the world in its application domain. However, as an artefact, any technical system inevitably is involved in a second sphere of interaction, namely the interaction with its human developer(s). We may distinguish this second type of interaction with the term *config-interaction*. Current archi-

tectures of cognitive systems are largely blueprinted by humans. Engineers have to interact with the architecture during development, tuning and maintenance which is heavily facilitated by a well-structured and easily accessible config-interaction.

As seen from fig. 1(b), perception and action flow is reversed for config-interaction: Actions are inwards directed and have the goal to alter the structure of the system, perception is outwards directed and has the aim to provide system monitoring. Both interaction types strive for contrary representations of the internal system state. While config-interaction has a strong bias to well-structured, symbolic representations which are easily comprehensible by humans, task-interaction is designed towards successful system behaviour in its application context and hence may prefer a distributed, less structured coding.

Despite these differences, task- and config-interaction share an important commonality: Both of them support a form of human-machine cooperation towards certain goals using a range of actions in which the system has to participate. (While we take this active view for granted in the case of task-interaction, it is not yet wide-spread when considering config-interaction. However, we anticipate that "cooperative" monitoring and reconfiguration interfaces will gain importance as systems become more and more complex.)

Embracing config-interaction as an important factor that may affect architecture design as much as task-interaction leads us to a *dual interaction perspective* on architectures for artificial cognitive systems. It emphasises the need to explicitly consider the development of a proper architecture as a coupled optimisation problem in which the constraints resulting from the dual goals of successful task-interaction and easy config-interaction compete and may exert partly opposing forces on potential solutions.

A first area of "opposing forces" concerns system interconnectivity. Experience teaches us that most architectures – due to their fixed modular structure – lack a flexible interconnection between the low- and high-level cognitive processes. A strong interconnection would allow to employ dynamic and context-specific representations of symbolic and subsymbolic information between different involved processes as well as a fine-grained top-down control of low-level information processing. Both capabilities are necessary for a cognitive system to flexibly adapt itself to changing task requirements. Contrary, a strong interconnection complicates the view for the engineer. The example of artificial neural networks shows, that good human readability and maintenance (serving good config-interaction) does not by itself follow from functional efficiency (serving good task-interaction), while some early approaches of classical AI provide examples that the opposite conclusion does not work either.

Another typical drawback of classical system approaches is their inflexible organisation of process flow which is mostly hand-coded statically into the system. This typically causes system deadlocks due to unforeseen system dynamics and at the same time prohibits dynamical extension through on-

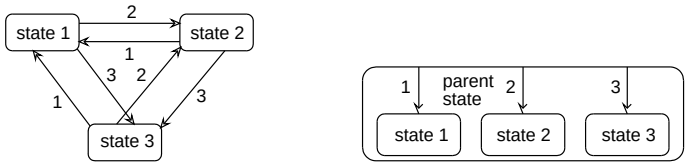
line acquisition of new skills and control patterns. Nevertheless, we think that this capability of online learning on a high system level is a major prerequisite for autonomous cognitive systems which interact with humans in their environments. On the other hand, we foresee that autonomous learning may serve "task-interaction", but erode "config-interaction" by changing the inner working of the system in a way that is not fully comprehensible for the human engineer.

As we see, with a significant number of specialised processing modules at hand, topics of communication and coordination between components, as well as integration and fusion of information become major issues in the design of complex cognitive systems. To date, use cases of highly integrated cognitive systems running in a real-world scenario and close interaction with the human while integrating a large number of different system behaviours in a unified framework are only sparsely reported. This reflects the difficulty to simultaneously serve the demanding goals of task- and config-interaction. A few examples of such systems acting in a robotics or pure vision domain can be found in [2, 3, 4, 5]. Some of our own earlier work along these lines is summarised in [6, 7]. Experience from this work led to the dual interaction perspective described above and to ideas for its implementation in a compact and highly manageable architecture employing hierarchical state machines (HSMs). Before describing our approach, therefore, it seems appropriate to insert a short technical section motivating and explaining some HSM basics. Readers familiar with HSMs can directly proceed to the next section.

4 Hierarchical State Machines

Typically the correct action in response to an event depends not only on the event itself but also on the internal state of various submodules which in turn evolve depending on the input history of the whole system. It is one major task of large reactive systems to monitor this current state and finally choose the right decisions and actions if a new event has to be handled. Actually, this decision process establishes the behaviour of the system.

From a programming perspective, the dependence on the event history very often leads to deeply nested if-else or switch-case constructs. Most reactive programs start out fairly simple and well structured, but as more features and behaviours are added, more flags and variables have to be introduced to capture the relevant state and event history such that the logical expressions formed from the various variables become increasingly complex. Therefore it is much more elegant to express the behaviour using more abstract programming tools, e.g. finite state machines [8] or petri-nets [9]. Their deployment can drastically reduce the number of execution paths through the code, simplify the conditions tested at each branching point, and simplify transitions between different modes of execution. Most important, its states and transitions can be visualised concisely.



(a) classical FSM (b) hierarchical FSM

Fig. 2. Due to their hierarchically nested states and inheritance of state transitions from parent to child states, simple finite state machines (a) can be expressed much more concisely using hierarchical state machines (b).

One major drawback of both finite state machines and petri-nets is the quadratical increase of state transitions w.r.t. the number of states. Take for instance a default "error" state of a component, which is targeted from every possible state in case of an error. In this case the transition with its associated action(s) has to be duplicated for every *source*-state, although essentially all transitions are identical. Clearly, the structure and behaviour of such FSMs is poorly represented in their state transition diagrams and the large number of transitions tends to obscure any visualisation and makes their implementation an error-prone process.

Additionally, classical FSMs suffer from another major and closely related problem: To express an infinite, or at least large number of states, an appropriate number of states have to be introduced, e.g. a vending machine needs a particular state for every possible amount of money already inserted. Nevertheless, the logical state "waiting for money" is the same for all these states. To simplify the state diagram it is useful to augment the state machine with so called extended state variables, which can be used in arithmetic and conditional expressions just like the state flags before.

However, the borderline between real states and extended state variables is blurred, because any extended state variables can be expressed as (a possibly large) number of real states and vice versa. Hence, a tradeoff must be found keeping conditional expressions in state transitions simple and using extended state variables to combine similar behaviour. Practical experience shows, that in a complex system extended variables are not sufficient to avoid a "state explosion", e.g. the introduction of a new state for each semantically different hand movement in the robotics grasping scenario introduced later.

In practice, many states are similar and differ only by a few transitions. Therefore, what is needed is an *inheritance mechanism for transitions*, analogous to the inheritance of attributes and methods that is familiar from object oriented programming. The formalism of statecharts, proposed in [10] and extended in [8], provides exactly that: a way of capturing the common behaviour in order to share it among many states. Its most important innovation is the

introduction of hierarchically nested states, which is why statecharts are also called hierarchical state machines (HSMs). As an example consider the hierarchical state diagram shown in fig. 2b. Here all states of the simple FSM of fig. 2a are embedded in a grouping parent state, which defines the common state transitions inherited by all sub-states. Consider for example state 1 getting event 3 which should trigger a transition to state 3 regardless of the current state. Because state 1 doesn't handle this event explicitly, the transition rules of the super-state apply, which in turn provoke the intended transition to state 3. This transition inheritance mechanism tremendously facilitates the proper structuring of transitions, since now sub-states need only define the differences from their super-states. By the same token, *state differentiation* (splitting an existing state into several slightly different states if additional information becomes available to the system) is facilitated, making incremental system development an easier process. In the context of robot grasping, for example all grasp postures can be grouped together and inherit from its super-state a transition to an "error" state, which also triggers the actuation of a relax posture.

5 A HSM-based Architecture for Grasping

In the following, we describe our approach to an architecture for grasping which allows both, a tight interconnection and coordination between subprocesses as well as a highly structured and modular design of the system and its state representation. We employ dynamically configurable hierarchical state machines, which reflect the high-level, symbolic system state and coordinate several behaviour controllers directly interfacing to the world on a subsymbolic level. Hence the approach tries to consider the opposing constraints of the task- and config-interaction.

Due to their symbolic nature, several HSMs can be flexibly connected by a suitable communication framework. Here, we make use of the excellent extensibility and human-readability of XML-based representations and employ a very flexible communication layer (AMI, cf. Sec. 7) that has been developed previously to support distributed processing for interactive systems. Continuing to take grasping as our "Rosetta stone", we will explain the main ideas in this context.

Once the external conditions (object location and shape, possible obstacles around) have been determined, e.g. by a suitable vision front-end, the core part of the grasping process involves a coordinated movement of arm and hand in order to grasp the object. Our architecture distinguishes two separate layers of processing (see fig. 3):

Controllers. Both the reaching movement of the arm and the actuation of the hand can be regarded as the evolution of dynamical systems constituted by controllers which implement a specific low-level behaviour. Each controller binds to a specific subset of sensory inputs and motor outputs. For example,

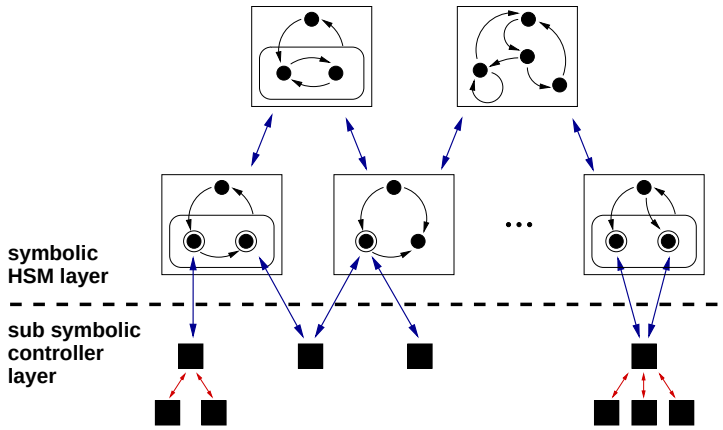


Fig. 3. Sketch of the system architecture comprising a high-level, symbolic HSM layer and a low-level, subsymbolic controller layer. Controller states within the HSM (indicated as double circles) are bound to a particular set of controllers. When entered, the state configures its associated controllers through parameterised events and then awaits termination of the process.

a controller to actuate given joint angles would bind to angle sensors and joint motors. To actuate postures specified in Cartesian coordinates another controller would bind to the end-effector position instead. To implement a visual servoing controller, one would bind the Cartesian target positions to the output of a vision-based tracker of the end-effector. Each controller might offer a parameterisation interface, e.g. to change the default execution velocity, smoothness or accuracy of a motion. However, the most typical parameterisation is the actually targeted posture of the robot.

HSMs. While controllers provide elementary behavioural skills on a subsymbolic level, hierarchical state machines provide a symbolic abstraction layer which allows a clearly structured, *semantic* view of the interaction realised by the controllers. States represent different behavioural modes of the system. In particular, states can be bound to low-level controllers, meaning that the associated interaction pattern is currently being executed and the HSM is waiting for its termination. Each of these *controller states* maintains a fixed selection of active controllers bound in a particular fashion to relevant sensory inputs and motor actuators.

Transitions between states are triggered by internal or external events which may carry optional parameters to transmit additional information, e.g. the object type or required hand posture for a `GRASP` event. Besides the state change itself, each transition can evoke *actions*, e.g. the generation of *parameterised events* to trigger transitions in concurrent HSM processes or to parameterise and (de)activate involved controller(s).

All actions associated to a transition are executed and finished before the next event is processed by an HSM. This avoids race conditions in a simple and

efficient manner, because actions cannot be interrupted by concurrent process activity. To ensure high responsiveness of the system to human instructions or urgent events, all actions should have a short execution period. Anyway, to allow long-running remote procedure calls, we generate a parameterised event to transmit the procedure request and enter a state waiting for the termination event of the triggered activity. Simultaneously we can respond to urgent events arriving in the meantime.

The proposed two-layer architecture provides a clear separation between the *semantic structure* of the interaction dynamics and its subsymbolic parts encapsulated at the level of controller(s). On the one hand, this permits to easily upgrade controllers by more sophisticated ones, e.g. replacing an initial PID controller by a hierarchical controller employing adaptive neural networks.

On the other hand, state nesting and transition inheritance permit a flexible top-down and incremental shaping of the task-interaction at a semantic level: Whenever a task requires to refine an existing task-interaction pattern into a number of more specialised patterns, we can express this at the state level by introducing a corresponding number of (mutually exclusive) sub-states. As explained in the previous section, each sub-state can inherit all properties of its parent state, so that we only need to specify the "delta part" of its more specialised interaction dynamics.

Connecting a HSM in the described way to a layer of low-level controllers, the HSM performs the usually highly non-trivial task of "glueing together" a set of (possibly complex) dynamical systems. While the HSM representation alone gives no theoretical foundation of how to achieve a desired combination behaviour, *it does* provide a very powerful and structured interface for the config-interaction with the human designer. This largely facilitates to explore and tune different alternatives and, in this way, to arrive at a robust solution in an "evolutionary" manner.

6 Application in an Imitation Grasping Scenario

In the concrete example of the coupled hand-arm system, the hand subsystem is realised by the HSM depicted in fig. 4. The `handPosture` state corresponds to an actuation of the hand using a particular controller. It is differentiated into three sub-states for attaining a pre-grasp posture (`pre-grasping`), performing finger closure from a pre-grasp to a target grasp posture (`grasping`), and for opening the hand (`stretching`). Transitions into any of these sub-states share the emission of a parameterised `TARGET` event to the hand controller, which provides the targeted hand posture and suitable joint stiffness values.

The distinction between different hand postures within these sub-states is achieved by appropriately parameterised events, i.e. the `PRE-GRASP` and `GRASP` events carry information about the concrete grasp type to be used. This information is finally forwarded to the controller as the targeted posture. Instead

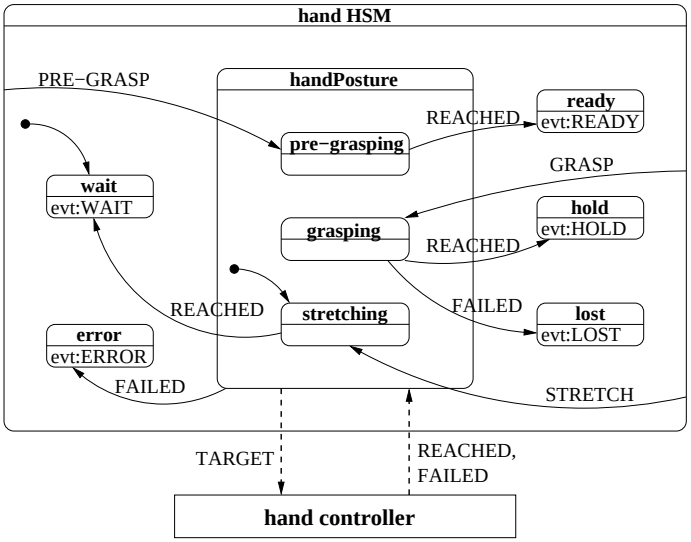


Fig. 4. HSM-implementation of the hand control layer: entry actions marked as "evt: ..." generate appropriate events within the overall system, which in this case trigger state transitions in the arm control layer (see fig. 5).

of introducing new sub-states for different postures (which would complicate the state diagram) the three employed states only capture the *semantic state* of the grasping process. Arcs emanating from a black dots, indicate initial transitions to a more detailed sub-state, if the parent state is entered directly.

A second, similar HSM is used to shape the high-level structure of the arm subsystem. Here, sub-states of an embracing `moving` state represent semantically different movement strategies of the arm, i.e. movements w.r.t. different coordinate systems, and a planned arm motion avoiding collisions. Again, different arm postures are specified through parameterised events to the HSM states as well as the associated controllers.

Once each dynamical subsystem has been packaged as a HSM box, we arrive at the task of coupling a number of such processes. Here again, we can use event communication between the HSM-subsystems in order to coordinate their concurrent activity at the next higher integration level. Inevitably, at this level we are no longer spared from the enormous complexity of coordinating complex concurrent processes. However, having cleverly deployed the power of HSMs to flexibly coordinate a small number of (controller) processes in each HSM box already and providing the resulting subsystems with judiciously designed interfaces, the task of their coordination at the next higher level becomes much more manageable.

As a concrete example, fig. 5 shows the organisation of the grasping sequence based on two cooperating HSMs for the hand and arm control system. Our implementation of the grasping sequence starts with an initial manip-

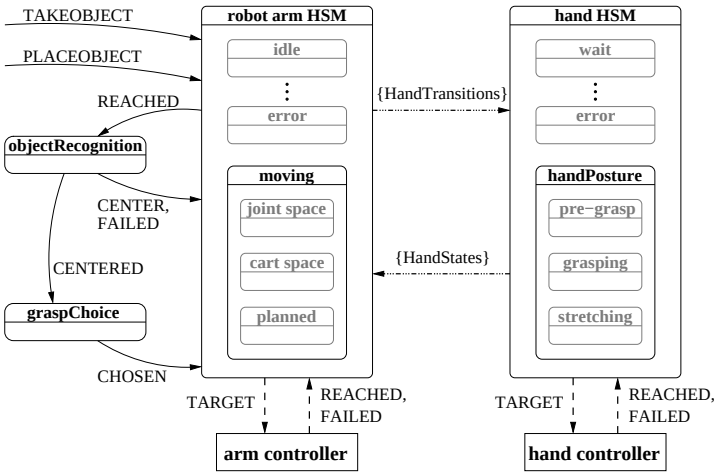


Fig. 5. Implementation of the grasping behaviour employing two HSMs running as separate processes and communicating via events.

ulator movement to a coarsely determined offset position above the object. The according parameterisation event, called `TAKEOBJECT`, is generated by a coordination instance if a grasping command was articulated and the 3D visual object detection has recognised an appropriate object. To disambiguate between similar objects on the table, the grasping region might be restricted further by a pointing gesture. Due to the limited spatial resolution in object detection, a subsequent fine-positioning above the grasping target is required. Hence, the manipulator is centred above the object employing a local, visually driven feedback loop alternating between the states `objectRecognition` and `armMovement` until the visual centre of gravity assumes a predefined position.

Next an appropriate grasp prototype has to be selected. Following Cutkosky's taxonomy of human prehensile grasps [11] we implemented four basic grasp prototypes: "two finger pinch", "two finger precision", "all finger precision", and "power grasp". The appropriate prototype is selected either based on previously acquired knowledge "How to grasp a specific class of objects" or according to a visual observation of an instructing human hand posture, which will be associated to the given object for later reuse. Depending on the selected grasp prototype, the robot hand has to be aligned along some particular axis – typically the main axis of the object – and an initial hand posture is achieved.

When this pre-grasp posture has been reached, we initiate a purely haptically driven finger closure phase. Empirically we found, that by careful fine-tuning of the four employed grasp prototypes, the described grasping architecture can capture the relevant "reactive grasp knowledge" sufficiently well to allow successful grasps for a considerable range of everyday objects [6]. Many alternative approaches to grasping put a high emphasis on rather elaborate

grasp planning [12, 13, 14], achieved with sophisticated optimisation methods to find optimal contact points [15, 16]), and with very precise joint position control. In this view, we find it remarkable that our approach is quite reliable for a large variety of objects and parametrically robust against uncertainties and small errors in the localisation of the object, the actuation of a particular hand posture, and the controller parameters.

Due to its flexibility, the present grasping architecture could be enhanced in many ways. One promising direction will be the discrimination of a larger number of different contact states between fingers and object, and modifying the closure synergy accordingly. Such control switches, but also initiation and termination of the whole process, as well as diagnosing failure and error conditions, and, possibly, triggering of suitable response actions, can all be rather easily adopted by adding the required interaction patterns in the form of additional states to the HSM depicted in fig. 5.

7 Managing Interconnectivity

So far, we have focused only on the hand-arm component of our robot system in order to illustrate the basic ideas behind our architecture approach. Evidently, the core hand-arm behaviours have to be embedded into a network of additional skills in order to coordinate grasping actions with vision and spoken commands. At this point, it may have become clear how the described HSM-based interaction architecture can be extended to include additional skill modules, each consisting of a cluster of related dynamical low-level controllers encapsulated as a HSM box which offers a semantic interface to the engineer for tailoring the overall behaviour of the system.

However, when tying an increasing number of HSM subsystems together, a second key issue for a complex cognitive system comes into view: how to realise and manage *a high degree of interconnectivity* between a large number of functional elements that must operate concurrently and in a largely asynchronous manner. (This seems to have also been a significant challenge for the brain: The most part of its volume is taken by the *connections between* neurons, not the neurons themselves.)

Having encapsulated the subsymbolic layer of our robot system within HSM boxes with a flexible coarse-grained interface, we may hope to escape the necessity to implement a similarly fine-grained interconnectivity as encountered in the brain. Instead, we can hope to exploit the convenience of communication via parametrised events in order to manage the communication between even a large number of modules.

Working with a technical system, we again face “evolutionary pressures” arising from the config-interaction. A major issue is system maintainability, which has several facets in its own. First of all, the evolution of a large system progresses slowly and design decisions on data representations made in the early development process might turn out to be not powerful enough for

```
<OBJECT>
  <CLASS reliability="0.87">
    cup
  </CLASS>
  <CENTER x="32" y="44"/>
  <REGION>
    <RECT x="12" y="30"
          w="40" h="80"/>
  </REGION>
  <IMAGE uri="xcf://img/123"/>
</OBJECT>
```

```
<EVENT name="RobotCommands">
  <SELECT robot="LeftArm"/>
  <MAXSPEED xyz="200" ypr="50"/>
  <POSTURE> home </POSTURE>
  <MOVETO mode="relToTool">
    <POS> 10 0 0 </POS>
    <EULER> 90 35 0 </EULER>
  </MOVETO>
  <NOTIFY id="GraspModule"/>
</EVENT>
```

Fig. 6. Two examples of XML messages used within the robotics system. Their semantics is easily recognised. While the left message shows the output of an object classifier, the right message shows a commands script accepted by robots.

subsequently added applications. As a consequence, data formats have to be extended, which in classical approaches using binary data formats requires modifications of all affected processes in turn, thus causing a considerable amount of programming effort. Instead, a data format is desirable which can be easily extended on demand without the need to modify the overall system. Second, for debugging purposes system developers should be able to easily inspect the data flow within the system in an online fashion during system operation, or to record these data for an offline evaluation.

These requirements call for an embedding of the "atomic" event format into a coherent communication framework based on a human-readable data format which should be easily extensible and simultaneously can ensure type safety in a scalable manner. Additionally, we anticipate the need to accumulate and interact with significant databases for storing episodes or general semantic world knowledge about objects and actions.

Looking for a suitable framework fulfilling these needs, we found the *Active Memory Infrastructure*[1] (AMI) [17] ideally suited for the task. The AMI framework is based on the *XML based Communication Framework* (XCF) [18] which provides an easy-to-use middleware supporting synchronous and asynchronous remote method invocation (RMI) as well as publisher-subscriber communication semantics. Exposed methods of server processes are bound and invoked dynamically on client side, with XML schemas optionally providing runtime type safety of exchanged parameters. In the same manner messages published via streams can be checked for correctness.

On top of the communication framework, an *Active XML Memory* provides a flexible basis for coordination and shared data management. First of all, the Active XML Memory provides the memory needed by cognitive systems to track a complex interaction context in terms of perceived objects, commands and episodes as well as to persistently store long-term knowledge

[1] available for download at `http://xcf.sourceforge.net`

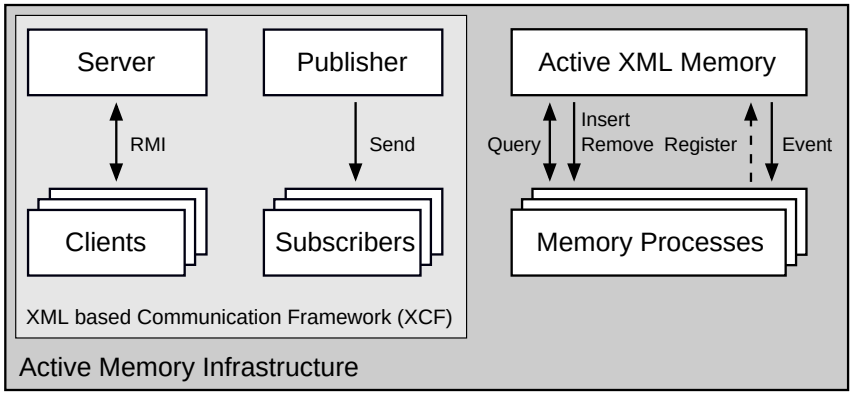


Fig. 7. The XML-based Communication Framework (XCF) provides distributed process communication including 1:n stream semantics as well as n:1 remote method invocations (RMI). The Active XML Memory build upon XCF supplements persistent (and transient) XML memory as well as event-triggered notification.

acquired during learning. Processes can insert XML documents together with attached binary data into the database and thus provide information for subsequent use by other processes. To selectively retrieve documents by their semantic structure or content the framework allows to specify XPath expressions, e.g. of the form "`/OBJECT[@reliability>0.8]`".

Basic coordination between components is provided by a flexible event-notification mechanism extending the mere storage functionality of the database. Processes interested in particular documents inserted into (or removed from) the database can subscribe to an insertion (or removal) event. Again, interesting documents are selected by specifying a restricting XPath expression. If a matching XML document is inserted (or removed), the Memory *actively* transmits this document to the subscriber.

The framework supports all the necessary processes and primitives to run an *active blackboard* where system components can subscribe to XML messages inserted into the database, or pasted on the blackboard to keep this picture. If any other component inserts a matching XML message, the subscriber is automatically (and asynchronously) notified about this event. Coordination thus matches ideally with the event format of the HSM layer and is not bound to explicit links between a fixed set of components present in the system.

8 Towards Higher Cognition

The AMI-framework can be easily employed to address issues such as information fusion from different modalities or active forgetting of information that is no longer needed. For example, Wrede et al [17, 19] employ a dedicated information fusion process to integrate unreliable 3D position information over

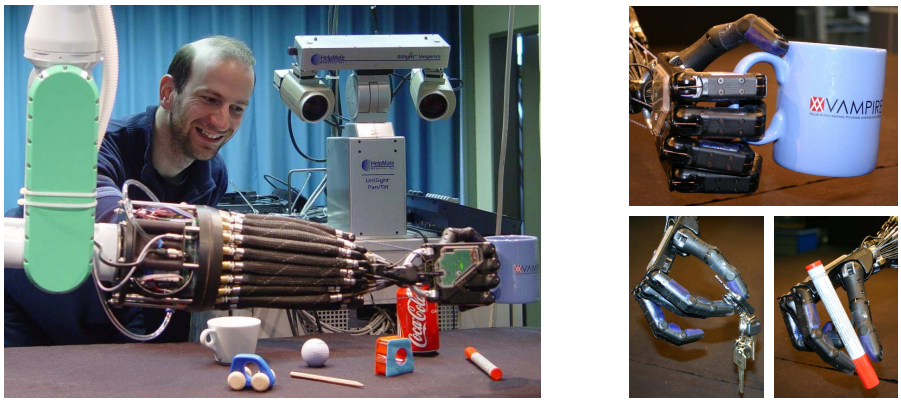


Fig. 8. Considered scenario for imitation grasping integrating a 7-DOF Mitsubishi robot arm, a 20-DOF dextrous robot hand and a 4-DOF stereo camera head.

time and from different low-level recognition processes to arrive at more robust hypotheses about positions of visually recognised objects.

Starting from a previous robotics setup developed in the course of the special research unit "SFB 360" and providing a large number of specialised processing modules [20], we have implemented a robot system (fig. 8) whose grasping abilities connect sensori-motor control with vision and language understanding. In reaction to a spoken command, the robot can grasp a variety of every-day objects from a table [21]. By integrating a special vision module tailored specifically to the classification of human hand postures, a human instructor can teach the robot to grasp new objects by showing an appropriate grasp posture with his own hand. This posture is recognised and classified as one of the four grasp prototypes by the visual system. Currently, we integrate other previously developed processing modules, which allow the robot to pay attention to objects on the table and to interprete pointing gestures to disambiguate between similar objects by further narrowing the region of interest.

The implemented system can be seen as a rather large-scale exercise in coordinating the modality- and context-specific interaction patterns of a significant number of low-level subsymbolic processes packaged into HSM boxes with a semantic interface at a symbolic level. To coordinate these boxes, as well as data from a number of additional processes directly publishing to the AMI blackboard, once again the technique of a coordinating HSM is adopted (see fig. 9).

In view of this coordinating HSM instance, all subprocesses – as well implemented as HSMs – can be treated like a "pseudo-controller" providing some specialised functionality: moving an arm, grasping, recognising objects, finding or tracking a hand, looking for salient points, listening to user commands. Hence, they can be represented in the coordinating state machine as

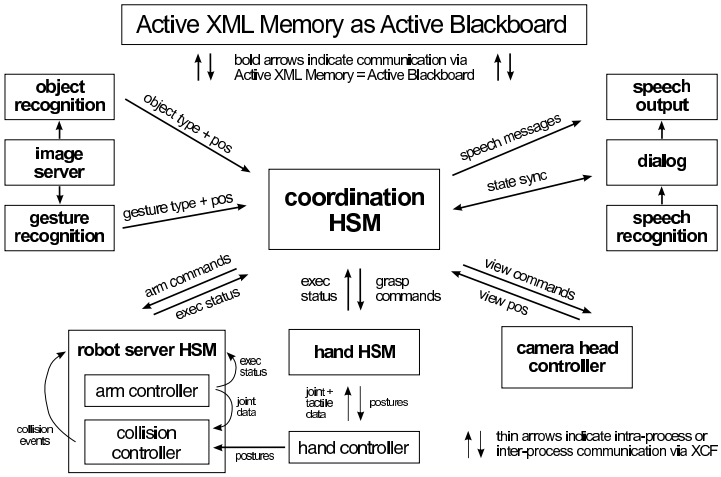


Fig. 9. Sketch of the overall system architecture of our cognitive robot.

a kind of "skill states", which have to be orchestrated to achieve a particular behaviour. From this point of view, an action sequence is represented as a sequence of state transitions within the coordinating HSM, each triggered by a set of predefined events from the subprocesses. Simultaneously each transition can generate events to trigger the activation or deactivation of some skill states within the subprocesses.

The flexible config-interaction achieved with the HSM layer offers us an interesting and potentially very powerful way of making system changes: Designing the coordinating state machine such that its structure can be loaded or extended on demand from a (textual) XML description, we can easily instantiate different (or partially modify existing) coordinating patterns from a "memory" of such patterns. From a functional perspective, this resembles a first step towards a capability of flexible and very high-level "self-reprogramming", utilising memories of previously successful coordination strategies. This makes it possible to implement very high-level learning techniques, e.g. based on imitation learning or heuristic forms of "introspection" and connect them with lower level approaches, such as the application of "datamining" techniques to analyse occuring data streams in an on-line fashion and synthesise new or modify existing behavioural skills.

To restrict the overall complexity of the coordinating state machine, one can also introduce several functional building blocks – each based on a HSM –, which coordinate specific aspects of the overall system only. Again, on the next higher level a state machine would coordinate these functional building blocks. Proceeding in this recursive manner, the system remains much easier configurable and provides a natural mechanism for encapsulation of increasingly complex behaviour at all levels while avoiding complex synchronisation and scheduling schemes.

9 Summary and Discussion

Taking an evolutionary perspective on cognitive architectures, it appears that the control of hands might be a good "Rosetta Stone" problem to master the step from automatically executed low-level behaviour patterns to highly flexible and adaptive behaviour, ultimately leading to mental cognition.

Arguing that the shaping of interaction is the core task for any cognitive system, we have presented a *dual interaction* view on the development of architectures for artificial cognitive systems. A novel element of this view is the emphasis that the shaping of interactions has to occur in two complementary spheres: the task-directed interaction with the world (*task-interaction*), and the interaction with the engineer that develops and maintains the system (*config-interaction*). As a suitable approach to meet the contrary objectives of this dual optimisation problem, we have proposed a two-layer architecture.

The first layer accommodates the necessary building blocks from which we can compose the required task-interaction patterns. Each building block supports a particular elementary dynamical process, implemented in the form of a suitable controller connecting (a subset of the) sensors with (a subset of the) actuators. To establish a desired task-interaction pattern requires to combine and suitably parametrise a subset of such controllers.

The aim of the second layer is to provide a good config-interaction interface for the engineer to achieve the necessary shaping of task-interaction primitives into a coherent task-interaction pattern. For this symbolic layer, we propose to use hierarchical state machines (HSMs) whose states represent semantically related classes of task-interaction patterns. Transitions between such states are coupled via an emission of parametrised events to the controller layer.

As an extensible technical framework to implement the required asynchronous communication as well as to maintain the required basic world knowledge and context-specific data, we employ a global message blackboard built on the XML-based Active Memory Infrastructure described in [17].

From a more general architecture perspective, this approach offers a principled way for the coordination of *continuous dynamical systems (controllers)* by *discrete dynamical systems* implemented as HSMs that act as "discrete dynamical glue". HSMs offer powerful structuring principles (state encapsulation, transition inheritance) to forge complex continuous sensorimotor processes at a level that is much better suited for semantic labelling than the to-be-coordinated continuous processes themselves.

Applying the described scheme to the case of robot grasping with a five-fingered anthropomorphous manipulator, we are able to realise surprisingly robust grasping capabilities for a wide range of common household objects. Based on these very encouraging results, we are currently porting the capabilities of a previously developed system with a more limited robot manipulator, but coupled with speech-understanding and binocular vision capabilities [2, 6, 7], to the new architecture. Developed over a time horizon of several years, this previous system had reached the limits of its extensibility and

maintainability. We view the current porting process as a rather stringent test of our architecture concept, anticipating significant gains in the ability to restructure the system towards even higher cognitive skills, in particular imitation learning.

While our porting project has been triggered by the pressures of evolving a system in the technical domain over the years, it is tempting to speculate that somewhat similar pressures may have been acting on the development of our own "cognitive machinery": as the potentially available skills reached a certain level of complexity, it may have become necessary to introduce something like config-interaction also within the brain. Language in combination with explicit reasoning and planning seems to resemble a config-interaction *into our own brain* that allows us to configure our "pre-rational" skills in a very flexible and rapid way. We are fully aware that the gap between the neural control circuits and the semantic networks in the brain on the one hand, and the artificial controllers and the HSM boxes in our system on the other hand is enormous, but we believe that pursuing the road of dual interaction is promising to bring us a bit closer to artificial systems that can act in a more "brain-like" manner.

Acknowledgements

Among many people who contributed to the robot system, we would like to thank particularly F. Röthling who developed the grasping strategy for our robot hands, S. Wrede who coordinated and implemented many parts of the *Active Memory Infrastructure*, and R. Kõiva whose technical input was invaluable for attaining a smooth and reliable operation of many subsystems. This work was supported by Deutsche Forschungsgemeinschaft DFG.

References

1. M. A. Arbib, "From monkey-like action recognition to human language: An evolutionary framework for neurolinguistics," *Behavioral and Brain Sciences*, vol. 28, no. 2, pp. 105–124, 2005.
2. G.A. Fink, J. Fritsch, N. Leßmann, H. Ritter, G. Sagerer, J.J. Steil, and I. Wachsmuth, *Situated communication*, chapter Architectures of situated communicators: From perception to cognition to learning, pp. p.357–376, Trends in linguistics. Mouton de Gruyter, Berlin, 2006.
3. C. Bauckhage, S. Wachsmuth, M. Hanheide, S. Wrede, G. Sagerer, G. Heidemann, and H. Ritter, "The visual active memory perspective on integrated recognition systems," *Image and Vision Computing*, vol. In Press, 2006.
4. S. Li, M. Kleinehagenbrock, J. Fritsch, B. Wrede, and G. Sagerer, "'BIRON, let me show you something': Evaluating the interaction with a robot companion," in *Proc. IEEE Int. Conf. on Systems, Man, and Cybernetics, Special Session on Human-Robot Interaction*, W. Thissen, P. Wieringa, M. Pantic, and M. Ludema, Eds., 2004, pp. 2827–2834.

5. H. Asoh, Y. Motomura, F. Asano, I. Hara, S. Hayamizu, K. Itou, T. Kurita, T. Matsui, N. Vlassis, R. Bunschoten, and B. Krose, "Jijo-2: An office robot that communicates and learns," *IEEE Intelligent Systems*, vol. 16, no. 5, pp. 46–55, 2001.
6. J.J. Steil, F. Röthling, R. Haschke, and H. Ritter, "Situated robot learning for multi-modal instruction and imitation of grasping," *Robotics and Autonomous Systems*, vol. Special Issue on "Robot Learning by Demonstration", no. 47, pp. 129–141, 2004.
7. H. Ritter, J.J. Steil, C. Nölker, F. Röthling, and P. McGuire, "Neural architectures for robotic intelligence," *Reviews in the Neurosciences*, vol. 14, no. 1-2, pp. 121–143, 2003.
8. M. Samek, *Practical Statecharts in C/C++: Quantum Programming for Embedded Systems*, CMP Books, 2002.
9. J. L. Peterson, *Petri Net Theory and The Modeling of Systems*, Prentice Hall, 1981.
10. D. Harel, "Statecharts: A visual formalism for complex systems," *Science of Computer Programming*, , no. 8, pp. 231–274, 1987.
11. M. Cutkosky and R. D. Howe, "Human grasp choice and robotic grasp analysis," in *Dextrous Robot Hands*, S. T. Venkataraman and T. Iberall, Eds. Springer, 1990.
12. Xiangyang Zhu, Han Ding, and Jun Wang, "Grasp analysis and synthesis based on a new quantitative measure," *IEEE Trans. Robotics and Automation*, vol. 19, no. 6, pp. 942–953, 2003.
13. J.J. Xu, G.F. Liu, X.Wang, and Z.X. Li, "A study on quality functions for grasp synthesis and fixture planning," in *Proc. IROS*, Las Vegas, 2003, vol. 4, pp. 3429–3434.
14. M. Fischer Ch. Borst and G. Hirzinger, "Grasping the dice by dicing the grasp," in *Proc. IROS*, Las Vegas, 2003, vol. 3, pp. 3692–3697.
15. U. Helmke, K. Hüper, and J. B. Moore, "Quadratically convergent algorithms for optimal dextrous hand grasping," *IEEE Trans. Robotics and Automation*, vol. 18, no. 2, pp. 138–146, 2002.
16. Li Han, Jeff C. Trinkle, and Zexiang Li, "Grasp analysis as linear matrix inequality problems," *IEEE Trans. Robotics and Automation*, vol. 16, no. 6, pp. 663–674, 2000.
17. S. Wrede, M. Hanheide, C. Bauckhage, and G. Sagerer, "An Active Memory as a Model for Information Fusion," in *Proc. 7th Int. Conf. on Information Fusion*, 2004, number 1, pp. 198–205.
18. S. Wrede, J. Fritsch, C. Bauckhage, and G. Sagerer, "An XML Based Framework for Cognitive Vision Architectures," in *Proc. Int. Conf. on Pattern Recognition*, 2004, number 1, pp. 757–760.
19. S. Wrede, M. Hanheide, S. Wachsmuth, and G. Sagerer, "Integration and coordination in a cognitive vision system," in *Proc. Int. Conf. on Computer Vision Systems*, St. Johns University, Manhattan, New York City, USA, 2006, IEEE.
20. P. McGuire, J. Fritsch, H. Ritter, J.J. Steil, F. Röthling, G. A. Fink, S. Wachsmut, and G. Sagerer, "Multi-modal human-machine communication for instructing robot grasping tasks," in *Proc. IROS*, 2002, pp. 1082–1089.
21. J.J. Steil, F. Röthling, R. Haschke, and H. Ritter, "Learning issues in a multi-modal robot-instruction scenario," in *Workshop on Imitation Learning, Proc. IROS*, 2003.